Towards Systematic Grammar Profiling
Test Suite Technology Ten Years After*

Stephan Oepen
Saarbrücken University
Postfach 151140
66041 Saarbrücken (Germany)
oe@coli.uni-sb.de

Daniel P. Flickinger
CSLI Stanford
Ventura Hall
Stanford, CA 94305 (USA)
dan@csli.stanford.edu

Abstract

An experiment with recent test suite and grammar (engineering) resources is outlined: a critical assessment of the EU-funded TSNLP (Test Suites for Natural Language Processing) package as a diagnostic and benchmarking facility for a distributed (multi-site) large-scale HPSG grammar engineering effort. This paper argues for a generalized, systematic, and fully automated testing and diagnosis facility as an integral part of the linguistic engineering cycle and gives a practical assessment of existing resources; both a flexible methodology and tools for competence and performance profiling are presented. By comparison to earlier evaluation work as reflected in the Hewlett-Packard test suite data, released exactly ten years before TSNLP, it is judged where test-suite-based evaluation has improved (and where not) over time.

1 Motivation

> [...] the study and optimisation of unification-based parsing must rely on empirical data until complexity theory can more accurately predict the practical behaviour of such parsers. [...] It seems likely that implementational decisions and optimisations based on subtle properties of specific grammars can [...] be more important than worst-case complexity. [Carroll (1994)]

Contemporary lexicalized constraint-based grammars (e.g. within the HPSG framework) with wide grammatical and lexical coverage exhibit immense conceptual and computational complexity; as the grammatical framework aims to eliminate redundancy and factor out generalizations, the interaction of lexicon and phrase structure apparatus can be subtle and make it hard to predict how even modest changes to the grammar affect system behaviour. Additionally, in a distributed grammar engineering setup (i.e. for a project where several people or even sites contribute to a single grammatical resource) it becomes necessary to assess the impact of individual contributions, regularly evaluate the quality of the overall grammar, and compare it to previous versions.

Besides concise coverage (i.e. competence) judgments, in most application scenarios efficiency and resource consumption play an increasingly important role; hence, processing components typically provide a (potentially) large inventory of control parameters and preference settings. When tuning the analysis component to improve system performance, grammar writers often rely on introspection, knowledge of the grammar, and personal experience; yet, without systematic profiling and performance analysis, processor optimization amounts to guessing parameter settings and constant experimentation. In general, given the similarity in setup (parsing with large-scale unification grammars), it is to be expected that the observations of Carroll (1994) (see above) hold for HPSG-type systems too.

2 Background: Test Suites in the Context of Multi-Site Grammar Engineering

Despite the growing variety of approaches to NLP, the development of computational grammars still is a prerequisite for NLP systems that rely on (or at least incorporate) what is often called deep (or full) linguistic analysis. To allow developers to achieve the grammatical coverage required for most realistic application contexts, substantial investments into grammar engineering methodology and tools have been made; hence, for most of the contemporary grammatical frameworks in computational linguistics, there is a (sometimes large) number of development environments to choose from.

At the same time, the improved methodology and strict formalization, declarativity, and modularity of grammatical resources facilitate the distribution of development across several grammar writers and sometimes even sites. While in the past (e.g. for purely rule-based systems) grammars often were developed and maintained by a single person over years or decades and ultimately connected in life cycle to that one developer (see Erbach and Uszkoreit (1990)), it is now more fre-

*Part of the research reported presently was funded by the German National Science Foundation (DFG) within the Special Research Divison 378 (*Resource-Adaptive Cognitive Processes*) project B4 (PERFORM) and by the German Federal Ministry of Education, Science, Research, and Technology (BMBF) in the framework of the VerbMobil project under grant FKZ:01IV7024.

Robert Gaizauskas (editor) 1998: Journal of Computer Speech and Language # 12 (4). Special Issue on Evaluation

quently possible to view parts of a grammatical resource as (mostly) independent modules that can be engineered by separate people. This, obviously, not only holds the potential to shorten development time but at the same time can greatly add to the transparency and reusability of the resulting resource.

Yet, it seems, comparatively little effort has been put into systematic, let alone comparable, diagnostic and profiling technology, such that the practical evaluation of processors (for analysis as for generation components) remains an open issue. And although the paradigm shift in grammar development outlined above creates a number of both theoretical and practical challenges for quality assessment, most grammar writing initiatives lack appropriate methodology, test data, and tools to integrate regular, systematic, and in-depth profiling of both grammatical competence and system performance into the engineering cycle.

While corpus-driven efforts along the PARSEVAL lines (see Black et al. (1991)) typically focus on quantitative system comparison and at best give a coarse-grained analysis of system performance, test suites (i.e. systematic collections of artificially constructed and manually annotated reference data) have long been acknowledged as suitable for fine-grained diagnosis, progress evaluation and benchmarking (see e.g. Flickinger et al. (1987), Nerbonne et al. (1993), and Sparck Jones and Galliers (1995)), most of the available data sets follow the traditional design as flat text files listing test sentences annotated with, if at all, grammaticality judgements plus, in some cases, informal section headings grouping sets of sentences according to linguistic phenomena (or sometimes application-specific criteria). Lacking internal structure and annotations, these test suites do not allow developers to (automatically) classify, present, and interprete evaluation results from various perspectives, which, it is argued, is a crucial property of systematic competence and performance profiling.

The following introductory sections (i) outline an ongoing effort on distributed, multi-site HPSG grammar engineering with (ii) its specific desiderata for quality assessment and (iii) summarize a novel test suite package that recently became publicly available from the TSNLP project. Section 3 then reports on a wealth of empirical results (in considerable detail) gained from experimenting with the TSNLP package in a practical evaluation environment over a period of about one year. From this perspective section 4 can be seen as a control experiment that relates the experiences with TSNLP to those obtained from adapting the test data of the ten-year-old Hewlett-Packard (HP) test suite (which had influenced the TSNLP developers in many respects). As technology building was an important aspect within the TSNLP project, section 5 gives a critical assessment of the quality of the tools available and, thus, leads into a concluding discussion that summarizes a number of recommendations to both test suite developers and users building on the experiences reported here.

2.1 The ERG Distributed Grammar Development Effort

The *English Resource Grammar* (ERG) project is a consortium of research groups[1] working within the HPSG (Pollard and Sag (1994)) framework; the consortium was initiated and is coordinated at CSLI Stanford.

Research and implementation both at CSLI and at collaborating sites include work on morphology, lexicon, syntax, and semantics, along with the necessary processing and analysis tools needed in large-scale grammar engineering. The ultimate goal of the consortium is to produce a multipurpose broad-coverage, precise, and reusable computational grammar of English and to enable the exchange of NL software development tools and implementations of English grammar fragments, thereby enhancing the effectiveness of each group in doing its own focused research. Most of the development is carried out within the PAGE (*Platform for Advanced Grammar Engineering*)[2] environment but some consortium members pursue work on proprietary software systems in parallel.

As the actual grammar development is (at least in part) distributed among several sites, the project has established the following integration strategy:

- new versions of the ERG grammar are released (at least) twice a year;
- throughout a six-month development period consortium members submit contributions to the coordinator;
- acting as a benevolent csar, the coordinator approves and incorporates changes;
- the decisions made by the coordinator are reviewed by an advisory board of member institutions;

[1] See '`http://hpsg.stanford.edu/`' for details, online access to the current system, and an up-to-date list of consortium members.

[2] The PAGE system developed and maintained at the German National Research Center for Artificial Intelligence (DFKI GmbH) is an advanced NLP core engine that especially facilitates the development of grammatical resources building on typed feature logics (e.g. for HPSG-style frameworks).

PAGE comprises a fair number of various (and often independent) modules and linguistic resources that allow for a flexible configuration according to different user requirements (see Uszkoreit et al. (1994) for details), of which primarily the unifier, type system, and parser play a role in the profiling experiments discussed presently.

- a new grammar release is issued by the coordinator.

For its first two years working on the ERG grammar, the consortium has mostly carried through the methodology and release schedule as expected; yet, the bulk of development and integration still resides with the coordinator.

2.2 ERG Evaluation Requirements

Obviously, in the evaluation of contributions and the preparation of new releases a well-defined quality metric is required, to allow both the contributors and the coordinator to (i) assess individual contributions and (ii) judge how changes to modules interact with the grammar as a whole; here, the three most relevant dimensions in evaluation are (i) grammatical coverage, (ii) overgeneration (or accuracy) of analyses[3], and (iii) system (i.e. primarily analysis) performance.

A coarse-grained three-level evaluation along those dimensions, however, is clearly insufficient for the ERG contributors and especially the coordinator in working towards a new grammar release. Instead, in both overall progress evaluation and finer-grained diagnostics developers need a large degree of flexibility to view and analyze evaluation results from various perspectives. In general, evaluators want to classify (aggregate) and summarize data into meaningful subsets (which may or may not correspond to what is taken to mean a module in the underlying grammar) and thus flexibly adjust the granularity in evaluation to their actual needs.

In contrast, traditional test suite approaches (like the still widely used HP test suite that will be taken as a reference point for comparison presently) typically only foresaw either the inspection of processing results on an individual (per item) level or the production of overall (average) results for a full test suite run. Without further information (i.e. linguistic or extra-linguistic distinctions) explicitly represented in the test suite, this restriction is impossible to overcome.

A final observation that, again, suggests a more eloborate evaluation setup is that in any typical NLP development context the engineering environment and processing components will change simultaneously to the grammar writing. Thus, not only will changes to the grammar itself potentially have a strong impact on system performance, but so too will changes made to system parameters or the actual software (e.g. a new version of the grammar development platform released by the external developer). Progress evaluation in such a setup is of an inherently iterative nature and has to control for numerous, often orthogonal, parameters.

2.3 TSNLP — Test Suites for Natural Language Processing

Recently, the *Test Suites for Natural Language Processing* (TSNLP)[4] project has delivered a multi-purpose and multi-language test suite package addressing the increasing demand in the NLP community (among developers and users alike) for reference data to evaluate various types of language technology applications.

The project produced the methodology and technology for the development and application of systematic test suites (see Lehmann et al. (1996b) and Oepen et al. (1997)). Substantial test suites of some 4500 items each were built for English, French, and German covering central syntactic phenomena. Test items are annotated using a rich annotation schema that is designed to be mostly neutral to linguistic theories and specific types of applications.

In comparison to first generation (or traditional) test suites, the TSNLP approach is novel in that the concept of a test suite as a monolithic set of test items has been abandoned in favour of a notion of a complex and highly structured test suite; the TSNLP test suites are implemented as a database in which test items are stored together with a rich inventory of associated linguistic and non-linguistic annotations. Thus, the test suite database serves as a virtual (or meta) test suite that provides the means to extract the relevant subset of the test data suitable for a specific task. Since the database concept allows the storage of application-specific coverage and performance measures as a precise and contiguous record of how various system and grammar versions (at various times) perform, it easily facilitates progress evaluation, regression testing, and comparative report generation.

3 A Case Study: TSNLP and ERG

From this brief summary of the TSNLP test suite package it already follows that in its design it should have the potential to address most, if not all, of the evaluation requirements in the ERG — and any similar — (distributed) grammar engineering setup. However, as TSNLP was designed as a multi-purpose tool (i.e. useful in different evaluation scenarios with different application types), it seemed interesting to evaluate its adequacy according to the specific needs found in this one particular environment. Only if the test data and tools prove flexible and (easily) adaptable, can the approach

[3] In information extraction terminology, the coverage and overgeneration dimensions correspond to the familiar recall vs. precision distinction. For the present discussion, both are often subsumed under the term grammatical competence and contrasted with the third dimension: system performance.

[4] Most of the project results and information on the consortium can be obtained from '`http://tsnlp.dfki.uni-sb.de/tsnlp/`'.

aiming for general, reusable test suite packages be fully validated.

3.1 Setup: Connecting TSNLP to the PAGE Grammar Development System

To take full advantage of the TSNLP virtual test suite concept and to allow for regular and flexible profiling and experimentation, the test suite database (tsdb) has been seamlessly integrated into the PAGE system; through a bidirectional interface the development platform is given full access to tsdb, thus enabling it to retrieve *and* store arbitrary data. For storing application- or user-specific data, the TSNLP data model foresees what is called a *user & application profile*, i.e. the ability to add new relations to the database in order to accommodate, for instance, additional annotations, specifications of expected output, and any number of result and performance metrics that the application wants recorded (see below for details).

The close coupling of the grammar engineering system and test suite apparatus adds (among others) the following functionality to PAGE:

(i) interactive retrieval and browsing of TSNLP test items using a simplified SQL-type query language;

(ii) listing of test suite vocabulary (and frequency of occurrence), expansion of necessary lexicon entries, and checking for missing items;

(iii) selection of (a subset of the available) test data, fail-tolerant batch processing, and storage of performance measures (figure 1 gives an excerpt from the resulting log); and

(iv) report generation summarizing grammar coverage and system performance in (simple) descriptive statistics (all figures used in this paper were automatically generated from this machinery).

While batch processing a selection from the test data, a large number of system-specific performance parameters are recorded in the database (associated with the respective test item and current test suite run); besides those shown in figure 1, these include: the total and per reading numbers of succeeding, failed, and filtered parser tasks (see section 3.5), and the time used, resulting feature structure size, derivation (tree), and semantic formula for each reading obtained.

Thus, the database not only serves as a data pool for various flexible views (report generation and statistical analysis) on the results obtained from a single test suite run, but just as well as a contiguous history of system development that, at any time, allows the assessment of previous competence and performance including comparative progress evaluation.

Regarding test suite vocabulary, the ERG lexicon was extended to incorporate the necessary lexical material used in TSNLP (some 580 word forms). Given the strict limitation in lexical variation in the test suite and the overall glass box evaluation scenario, it turned out simpler to adapt the application (i.e. the ERG lexicon) rather than the test suite because (i) TSNLP provides no support for automated lexical replacement to customize the test data (and customization in general remains an open problem) and (ii) the ERG consortium aims for a small but representative core lexicon anyway, to distribute it as part of the grammar (while larger lexicons developed at individual sites can in principle preserve a proprietary status). In identifying and adding missing lexical entries, the tsdb to PAGE integration already proved useful, in that it allowed the grammar writer to extract the necessary vocabulary from the test suite, feed it word by word through morphological and lexical processing, and obtain a list of lexical gaps.

Some of the phenomena in the TSNLP test suite (see below), however, inherently make reference to lexical properties: testing complementation patterns, for example, in an HPSG-type setup mostly amounts to testing the lexicon. And although the distinction between lexical and non-lexical competence in the grammar is somewhat blurred, the primary use of the TSNLP test suites is not seen in evaluating lexical coverage; thus, especially when adding lexicon entries for verbal complementation patterns (the TSNLP C_Complementation phenomenon), the grammar writers were allowed to inspect full test items (i.e. the context in which a particular word form occurs) to determine which usage of some verb was actually intended. Aiming for in-depth diagnosis and progress evaluation of how grammatical competence and system performance evolve rather than a competitive comparison among several applications, this approach seems to have no negative impact on the results obtained.

3.2 Grammatical Coverage

The linguistic coverage of a grammar is described within the TSNLP framework in terms of classes of phenenomena, with a top-level division into ten or so broad categories which can be further refined to arbitrary levels of precision. While the classes chosen do not of course exhaustively cover the range of important phenomena for a language, they will serve well enough for this study, to illustrate how a grammar developer can extract useful generalizations about the current state of the grammar from test data that facilitates the analysis in various degrees of granularity. In Figure 2 the ten categories chosen for the English TSNLP test items are shown with data from a run of the ERG grammar in October 1997, and include results

```
(22290101) ‘She arranges with him to work .’ [0] --- 1 (2.0|0.4:1.7 s) <24:223> (11.8M) [0].
(22290102) ‘He is arranged with to work .’ [0] (1 gc);
(22290102) ‘He is arranged with to work .’ =[0] --- (4.0:1.0|3.5 s) <47:306> (17.2M) [=1].
(22290103) ‘He is arranged with by her to work .’ [0] --- (5.5:1.0|4.7 s) <52:485> (27.0M) [0].
(22290104) ‘*To work is arranged with him .’ [0] --- (4.2:0.7|3.7 s) <47:336> (19.9M) [0].
(22290105) ‘*To work is arranged with him by him .’ [0] --- (13.0:6.5|12.2 s) <51:596> (36.7M) [0].
(22290106) ‘*It is arranged with him to work .’ [0] --- 7 (6.7:0.9|0.9:5.9 s) <49:627> (36.4M) [0].
(22290107) ‘It is arranged with him by him to work .’ [0] --- 15 (23.8:12.6|1.0:20.6 s) <53:1199> (70.5M) [0].
(22300101) ‘He considers him a competitor .’ [0] --- 1 (0.6|0.1:0.4 s) <9:82> (4.3M) [0].
(22300102) ‘He is considered a competitor .’ [0] --- 1 (2.5|0.8:2.0 s) <32:257> (14.8M) [0].
(22300103) ‘He is considered a competitor by him .’ [0] --- 3 (5.7|1.4:5.2 s) <36:559> (32.7M) [0].
(22300104) ‘*A competitor is him considered .’ [0] --- (1.5|1.1 s) <32:178> (10.3M) [0].
(22300105) ‘*A competitor is him considered by him .’ [0] (1 gc);
(22300105) ‘*A competitor is him considered by him .’ =[0] --- (4.0:0.9|3.5 s) <36:334> (20.4M) [=1].
(22310101) ‘He considers him as a competitor .’ [0] --- 3 (1.3|0.2:1.0 s) <13:174> (9.0M) [0].
(22310102) ‘He is considered as a competitor .’ [0] --- 4 (5.5:1.2|0.5:5.0 s) <36:461> (26.7M) [0].
(22310103) ‘He is considered as a competitor by him .’ [0] (1 gc);
(22310103) ‘He is considered as a competitor by him .’ =[0] --- 14 (16.9:2.8|0.8:14.7 s) <40:1436> (84.9M) [=1].
(22320104) ‘*Good is found the building .’ [0] --- (1.7|1.3 s) <32:152> (9.1M) [0].
(22320105) ‘*Good is found the building by him .’ [0] --- (6.9:3.7|6.0 s) <36:280> (17.2M) [0].
```

Figure 1: Snapshot from batch processing TSNLP test items. The log format aims to give a compact summary of some of the information gathered; besides the input string and an upper limit for chart edges (given in square brackets if available) the information following the triple dash is: the number of readings obtained, the time used to find the first reading and overall (exhaustive search) processing time (in parentheses) the number of lexical items involved and total number of edges in the chart (angle brackets), the amount of memory used, and the number of (global) garbage collections while parsing (square brackets); a leading ‘=’ sign indicates that the garbage collector was explicitly called before parsing this item which can either be triggered when reparsing an item because of a prior garbage collection during the parse process or by an explicit specification in the database.

Phenomenon	**total items** ♯	**positive items** ♯	**word string** ø	**lexical items** ø	**parser analyses** ø	**total results** ♯	**overall coverage** %
S_Types	175	75	4.55	20.65	1.24	72	96.0
C_Agreement	123	68	4.79	14.94	3.28	54	79.4
C_Complementation	1011	148	5.07	15.41	2.44	124	83.8
C_Diathesis-Active	57	54	5.17	16.33	2.09	44	81.5
C_Diathesis-Passive	220	99	7.00	33.67	4.47	53	53.5
C_Tense-Aspect-Modality	196	157	4.51	21.14	1.53	133	84.7
C_Negation	418	289	5.23	27.63	4.64	224	77.5
NP_Agreement	1196	201	2.51	5.12	1.11	118	58.7
NP_Modification	782	300	6.28	14.06	1.40	244	81.3
NP_Coordination	333	147	4.32	9.63	2.92	109	74.1
Total	**4511**	**1538**	**4.96**	**17.57**	**2.49**	**1175**	**76.4**

Figure 2: ERG Coverage Profile as of OCT-97. Columns are (from left to right): TSNLP phenomenon name, total number of test items, number of grammatical test items, average test item length, average number of lexical entries per test item, average number of readings per test item, total number of test items successfully parsed, percentage of grammatical items parsed. Comparing columns 4 and 5 provides a measure of lexical ambiguity, while column 6 indicates syntactic ambiguity. For example, passive test items exhibit significant lexical ambiguity because of multiple lexical entries for the copula and the passive participle; the latter also contributes to the higher measure of syntactic ambiguity for passives.

on examples testing sentence types, several clausal properties including agreement, verb complementation, some diathesis, tense and aspect, and negation, as well as some NP-internal properties including agreement, modification, and coordination (see Lehmann et al. (1996a) for a precise account of the phenomena classification used in TSNLP).

The test results summarized in Figure 2 show, for each phenomenon type, the total number of test items available, the number of grammatical items, the average number of words in each example, the average number of lexical items retrieved, the average number of distinct analyses obtained, the total number of grammatical examples that were successfully parsed, and the percentage of grammatical examples parsed. The criteria for deciding that a parse was successful can of course be complex, but for this table we merely record whether or not at least one parse was found for the given item. More informative metrics might record the correctness of labeled bracketings or the semantic representations computed, but these require more detailed annotations that are not yet available for the full suite. The lack of these annotations can quickly give rise to misleading results in a summary like that in Figure 2; for example, a grammar might completely lack an analysis of passives with a *by*-phrase, yet still provide a successful parse for such examples by treating the *by*-phrase as a simple locative. Such a lack of coverage might be visible in the record of the average number of analyses found, but will not be reflected in the percentage score for overall coverage at present. A more fine-grained classification of phenomena would also help to reveal that the relatively lower score on coverage of passives is due in large part to two inadequacies in the current grammar: a lack of an analysis implemented for pseudopassives, and an overly fixed ordering constraint on optional *by*-phrases relative to modifier phrases within the VP; besides, some of the valency frames used in constructing passive examples are (still) missing from the ERG lexicon (see below).

More generally, even though the test items were designed with the intention of excluding the presence of irrelevant analyses, this is difficult to guarantee for a wide-coverage grammar, and indeed many of the test items do admit of what are clearly unanticipated analyses. Obviously, an iterative process of refinement of the test suite is desired, where the grammar developer discovers test items that show unintended ambiguity, and replaces those with more carefully chosen examples. An alternative approach would be to annotate the overly ambiguous examples as they are identified, marking certain analyses as irrelevant. Some systematic unwanted analyses might also be eliminated by imposing phenomena-wide restrictions, such as an injunction that for the complementation and diathesis examples, all PPs are to be taken only as complements, thus eliminating the unwanted analysis of passive *by*-phrases as modifiers for those test items.

In general, the ability to break down a complete test run into smaller units, e.g. subsets of the test items corresponding to the TSNLP phenomena classification (or other aggregation schemes), often allowed the developers to spot unexpected behaviour within one class and then use the test suite database for further inspection of smaller aggregates or individual test items.

3.3 Overgeneration

The converse of measuring successful analyses for grammatical examples is to gather metrics on overgeneration, where the parser is not restrictive enough to exclude true ungrammatical examples. Here again, a rough measure of the lack of precision can be computed simply by seeing what percentage of the ungrammatical test items are assigned at least one successful analysis.

Like the metrics for positive examples, however, this kind of measure can be misleading, since it obscures several sources of confusion. First, as we saw with the grammatical examples above, it is difficult for the test suite developer to ensure that the ungrammatical item will not receive some kind of parse from the system, suggesting again that an iterative tuning process for the test suite itself will be a necessary part of the regular use of such technology in grammar development. A second source of confusion in measuring overgeneration with the coarse classifications we have adopted here comes from the lack of a notion of *prerequisite properties* in the test item specifications. For example, the numbers in Figure 3 show that in the October 1997 version of the ERG grammar, negation scores very well in not allowing any overgeneration at all, but an examination of the relevant items shows that the ungrammatical ones all test just one single property: the *not* must not appear before the finite verb that it negates. A similar kind of confusion reflected in the numbers in Figure 3 supports the idea that prerequisite properties should be identified for each test item, so the classification of a failure can be correctly assigned to the right phenomena. For example, NP_Modification in Figure 3 shows massive overgeneration by the grammar, but a look at the individual test items shows that almost all of the 400 or so ungrammatical examples are admitted by the grammar due to a single flaw: the relative pronouns *who* and *that* are wrongly allowed to appear in pied piping relative clauses as in '*the road on that she traveled.*' (in contrast to '*the road that she traveled on.*') If these NP_Modification examples had been annotated to require that lexical complementation properties must first be satisified, then the failure of the

Phenomenon	total items ♯	negative items ♯	word string ø	lexical items ø	parser analyses ø	total results ♯	overall coverage %
S_Types	175	95	4.75	18.17	1.22	49	51.6
C_Agreement	123	50	4.42	14.06	2.43	7	14.0
C_Complementation	1011	705	4.50	14.60	1.91	70	9.9
C_Diathesis-Active	57	2	8.50	23.50	5.00	2	100.0
C_Diathesis-Passive	220	111	6.59	33.86	2.83	18	16.2
C_Tense-Aspect-Modality	196	38	4.08	21.21	5.25	28	73.7
C_Negation	418	129	5.47	23.90	0.00	0	0.0
NP_Agreement	1196	928	2.95	5.85	1.09	131	14.1
NP_Modification	782	482	6.51	14.68	1.66	412	85.5
NP_Coordination	333	180	4.51	10.14	1.70	63	35.0
Total	**4511**	**2720**	**4.46**	**12.77**	**1.73**	**780**	**28.7**

Figure 3: ERG Overgeneration Profile as of OCT-97. The columns have the same meaning as in Figure 2 except for the difference in the underlying selection from the TSNLP test data: column three gives the number of test items marked as ungrammatical; hence, column eight is to be interpreted as coverage of illformed examples, i.e. overgeneration. In general, the ungrammatical test items are somewhat shorter (because the derivation of ungrammatical examples often builds on deletion of an obligatory element) and exhibit substantially less lexical and structural ambiguity.

Phenomenon	oct-96				oct-97			
	lexical ø	parser ø	in %	out %	lexical ø	parser ø	in %	out %
S_Types	12.50	2.16	78.7	40.0	19.63	1.23	96.0	51.6
C_Agreement	12.15	1.33	58.8	10.0	14.75	3.25	79.4	14.0
C_Complementation	10.85	2.19	62.2	12.1	14.00	1.90	83.8	9.9
C_Diathesis-Active	12.26	2.56	66.7	0.0	16.81	2.26	81.5	100.0
C_Diathesis-Passive	24.21	2.87	25.3	8.1	33.57	4.14	53.5	16.2
C_Tense-Aspect-Modality	12.01	1.57	66.9	73.7	21.11	2.17	84.7	73.7
C_Negation	17.90	2.12	74.7	0.0	26.48	4.64	77.5	0.0
NP_Agreement	4.48	1.06	47.8	14.8	5.84	1.09	58.7	14.1
NP_Modification	12.63	1.58	75.7	82.0	14.44	1.56	81.3	85.5
NP_Coordination	6.63	1.72	74.1	15.0	9.90	2.49	74.1	35.0
Total	**10.63**	**1.75**	**65.3**	**26.6**	**14.31**	**2.15**	**76.4**	**28.7**

Figure 4: ERG Competence Progress Profile comparing OCT-96 to OCT-97. The contrastive view summarizes key characteristics from both the coverage and overgeneration profiles, thus facilitating a comparison among two evolution stages; the '*in*' and '*out*' columns indicate coverage and overgeneration, respectively.

grammar to block *who* and *that* as complements of prepositions could have been identified first, and the grammar developer would have been led more directly to the source of the trouble.

This example of NP_Modification shows up a third kind of confusion in the numbers in Figure 3, one that comes from the lack of a balance in the raw numbers of examples illustrating each phenomenon. Since the overall coverage percentage is computed in terms of raw numbers of wrongly analyzed ungrammatical examples, one might expect that each variant of a given phenomenon would be illustrated by a handful of examples, and then another variant would be introduced. However, in the TSNLP test suite, like in others, some particular phenomenon can get far more examples than the next, partly because of necessary systematic variation, partly simply because the test suite developer chose to richly illustrate the first but not the second.

A fourth source of confusion in the overall coverage numbers can be more difficult to correct, since it arises from errors in the definitions of particular lexical entries. Since one of the design criteria for the test suite was to keep the number of lexical entries small, an error in any one of the included entries can affect a large number of test items across phenomena classes, obscuring the distinctions these classes attempt to draw. For example, an error in the entry for an auxiliary verb like *did* could lead to overgeneration for any example using this word, but would be hard to detect in test results at any granularity coarser than for individual lexical entries. For the results reported in Figure 3, one such lexical error is present, but fortunately does manifest itself even in the coarse classification scheme used, since its distribution in the test data is quite restricted, namely to the examples for C_Tense-Aspect-Modality. For this phenomenon class, the overgeneration percentage is surprisingly high in Figure 3, but the inspection of individual test items showed that the lexical entries for negative contracted auxiliaries (such as *won't*) contain an error in their morphology, leading to multiple spurious analyses for each such test item. In this case diagnosis of the problem was simplified by the fortunate accident that these negative contracted auxiliaries are not widely used for test items in other phenomena classes.

3.4 Progress Evaluation

As errors are diagnosed and corrected, or as the grammar is modified to extend coverage to additional phenomena, the developer often needs to see how a new instance of the grammar compares to a previous version. This evaluation of progress in grammar development is helped significantly by the ability to construct summary reports that concisely contrast salient characteristics for the two versions. One such view of progress is given in Figure 4, showing how grammatical competence changed over the course of one year with the ERG grammar with respect to the standard ten phenomena and key metrics from both the coverage and overgeneration profiles.

Some useful inferences about the status of the grammar can be drawn from even superficial examination of the test data. For example, it is reassuring to see that the coverage of sentence types improved while the average number of analyses for these test items dropped. In contrast, while coverage of sentence-level agreement also rose, so did the average number of analyses, suggesting that there may be a noticeable problem with overgeneration for these items in the October 1997 version of the grammar (and indeed, overgeneration was increased by a factor of 1.4).

An inspection of the two coverage ('*in*') columns shows that, as one would hope, coverage has improved for almost every phenomenon type, with the exception of NP_Coordination, which stayed flat (though a more detailed examination of the test results for this phenomenon revealed that two somewhat different sets of items are admitted in the two versions of the grammar, and accidentally result in the same number of correct items). Fortunately, while overgeneration also increased as coverage grew, it did not increase very much, and indeed fell for some phenomena, suffering severely only for NP_Coordination. Through the elimination of disjunctions from the grammar (see below), the average number of lexical entries grew markedly across the board, and while the average number of syntactic analyses also grew, the table shows much more variation in analyses by phenomenon.

Three of the phenomena which showed the largest increase in lexical ambiguity (Diathesis-Passive, Tense-Aspect-Modality, and Negation) all feature auxiliary verbs, suggesting that other systematic errors may have been introduced in these lexical entries (beyond the negative contraction flaw mentioned above), leading to spurious lexical ambiguity. Here the classification by phenomena along only one dimension partially obscures what may be a single source of error in the auxiliary verb system; this illustrates the potential benefit to the developer of allowing multiple classifications of phenomena for particular test items. On the other hand, this comparison at the coarse-grained level provides the developer with a reassuring brief overview of system behavior at two time points, and would quickly reveal any systematic errors corresponding to the chosen phenomena.

3.5 System Performance

While the TSNLP tools clearly hold the potential for significant benefit to the grammar developer in analyzing linguistic coverage, or competence profiling, they also enable the recording and analysis of detailed performance characteristics of a parser for a given grammar. As with competence properties, performance measures can be stored and retrieved at several levels of granularity, including broad phenomena, particular test items, or individual readings. These measures can include the time required to compute a parse, the number of parsing tasks attempted and completed, and the space consumed during parsing or in the resulting data structures.

Within the PAGE development system, several user-settable parameters are available for tuning the performance of the parser and unifier to the particular properties of the grammar. Parameters include the order of instantiation for daughters in a rule, the relative priority of individual rules, and even arbitrary properties of phrases such as the number of words they span, as well as simplification steps to be taken after unification of feature structures. Since the interactions among these parameters can be complex and even experienced grammar engineers often find their intuitions about the system behaviour incomplete, the developer must experiment with a variety of settings to derive the best performance from the system. The TSNLP tools enable the developer to approach the tuning task systematically, maintaining a rich, accurate record of past experimental results, in a uniform representation that allows performance properties to be associated with linguistic phenomena of varied granularity.

To take a familiar example in processing head-driven grammars, one can derive significant efficiency benefits by choosing the order in which the daughters in each rule are picked up: although experimenting with distinct rule expansion strategies confirmed the (rather obvious) prediction that an HPSG grammar will benefit in parsing from a head-first strategy, it also pointed the developer to a finer-grained tuning process where some particular rule schemata actually do better with an idiosyncratic ordering. When the data are studied on a rule-by-rule level (again taking advantage of the granularity variation supplied by the profiling machinery), they suggest, for example, that of the three specific filler-head schemata in the grammar, the two for WH and relative clauses reduce the average number of parser tasks if the nonhead (the filler) is picked up first, in contrast to the declarative variant, for ordinary topicalization. What the integrated profiling mechanism provides is a means to determine for a non-trivial grammar which lexical heads in which constructions should be picked up first in order to minimize processing costs.

Another longer-term experiment took advantage of the modular design of the PAGE environment and the fact that the ERG grammar does not deploy the full expressivity of the PAGE formalism. As — for both theoretical and practical reasons — the grammar makes no recourse to disjunctive or negated constraints, it allowed the ERG coordinator to substitute a simplified and optimized unifier in place of the standard PAGE unification engine.

While implementing and tuning the lightweight unifier and its interface to the type system and parser, the precise measurement of the resulting decrease in both space and time usage across rule and phenemona types allowed the identification of initial bugs in the algorithm — even though these only showed up in rare circumstances — and it also provided a detailed profile of which parameters have a strong impact on unification costs. Thus both competence and performance properties of the experimental module could be derived, providing guidance to ensure correctness of the algorithm, and presenting further targets for improved efficiency.

Figure 5 summarizes an intermediate stage of system optimization (as of October 1997) compared to an earlier version of the ERG grammar and PAGE software. Aggregating test items by string length demonstrates that input size has an obvious effect on processing costs (as is to be expected). Ignoring the relatively sparse populated class of test items with more than nine words for a moment, it seems to be the case that the speedup in processing time and reduction of memory usage manifests itself relatively uniformly across the classes; especially when normalized with respect to the increase in parser tasks (i.e. looking at the average time per parser action by crediting column nine to column ten), all four classes in fact show a time reduction between 75 and 83 %.

Yet, the classification by item length is primarily chosen to allow the comparison to a random sample of 96 sentences drawn from the English VerbMobil[5] corpus, a collection of task-oriented appointment scheduling dialogues. In general, there is an open question about whether performance metrics obtained on test suite data — which was artificially constructed to reduce ambiguity and phenomena interaction — can at all be meaningful for application profiling and optimization; obviously, some of the challenges in real-world (e.g. corpus) data are eliminated in a well-designed test suite. The profiling machinery discussed in section 3.1, however, is not restricted to the TSNLP (or HP) test data. For VerbMobil grammar development, for instance, the same engine is used to store and process application-specific corpora such that both types of data are available from a uniform source and can

[5] VerbMobil is a large-scale research project on spoken dialogue machine translation funded by the German national government. Some of the ERG grammar engineering at CSLI is carried out within the VerbMobil context.

Aggregate	total items ♯	oct-96 tasks ø	oct-96 time ø (s)	oct-96 space ø (kb)	oct-97 tasks ø	oct-97 time ø (s)	oct-97 space ø (kb)	reduction tasks %	reduction time %	reduction space %
$9 \leq length \leq 12$	80	1553	6.5	34872	4832	5.2	25522	-211.1	19.9	26.8
$6 \leq length < 9$	1343	1098	5.8	27331	1599	1.5	8837	-45.6	74.2	67.7
$3 \leq length < 6$	2849	467	2.1	14196	671	0.6	4027	-43.7	73.6	71.6
$0 \leq length < 3$	239	131	0.4	5235	162	0.1	988	-23.7	72.8	81.1
Total	**4511**	**653**	**3.2**	**17932**	**999**	**0.9**	**5705**	**-53.0**	**71.5**	**68.2**
Corpus	**96**	**1684**	**32.8**	**68629**	**3885**	**8.4**	**24373**	**-130.7**	**74.5**	**64.5**

Figure 5: ERG Performance Progress Profile comparing OCT-96 to OCT-97. TSNLP test items are aggregated into four classes (by string length) that directly correlate with the number of parser tasks, average processing time, and memory usage. All classes demonstrate a significant reduction in time and space resource consumption (by about a factor of four) even though by (i) extending coverage between the two grammar releases and (ii) eliminating disjunctive constraints lexical and structural ambiguity (reflected in the number of parser tasks here) were substantially increased. The bottom line relates the performance comparison for the artificially constructed test items to results obtained for a random sample of 96 sentences from the VerbMobil corpus (average sentence length: 8.4 words); although for the corpus data processing costs are in general higher because of more ambiguity, it is reassuring that the proportions in columns nine to eleven are mostly comparable.

Phenomenon	total items ♯	positive items ♯	word string ø	lexical items ø	parser analyses ø	total results ♯	overall coverage %
S_Types	235	179	6.83	24.15	2.91	118	65.9
C_Agreement	68	49	6.00	16.92	1.95	41	83.7
C_Complementation	179	108	5.46	16.78	2.46	99	91.7
C_Diathesis-Passive	35	27	6.63	26.78	5.50	24	88.9
C_Tense-Aspect-Modality	83	79	5.39	18.49	3.47	66	83.5
C_Negation	58	44	5.23	16.32	2.63	41	93.2
C_Coordination	79	57	8.33	23.84	5.42	43	75.4
C_Modification	174	121	7.14	21.39	2.65	80	66.1
NP_Agreement	46	37	4.86	17.03	2.48	31	83.8
NP_Modification	83	71	7.08	20.35	3.05	56	78.9
NP_Coordination	55	26	5.38	12.58	4.74	23	88.5
Total	**1095**	**798**	**6.40**	**20.32**	**3.12**	**622**	**77.9**

Figure 6: ERG Coverage Profiles (on HP test suite) as of OCT-97; see Figure 2 for comparison.

Phenomenon	total items ♯	negative items ♯	word string ø	lexical items ø	parser analyses ø	total results ♯	overall coverage %
S_Types	235	56	7.11	25.25	2.50	8	14.3
C_Agreement	68	19	3.42	8.63	1.18	11	57.9
C_Complementation	179	71	5.52	16.87	3.00	18	25.4
C_Diathesis-Passive	35	8	7.37	29.62	2.00	4	50.0
C_Tense-Aspect-Modality	83	4	6.50	21.00	4.75	4	100.0
C_Negation	58	14	4.71	11.00	0.00	0	0.0
C_Coordination	79	22	8.05	23.18	4.57	7	31.8
C_Modification	174	53	6.51	20.96	3.89	19	35.8
NP_Agreement	46	9	4.22	14.89	1.75	4	44.4
NP_Modification	83	12	7.92	21.58	1.00	1	8.3
NP_Coordination	55	29	4.90	11.31	3.15	20	69.0
Total	**1095**	**297**	**6.07**	**18.83**	**3.03**	**96**	**32.3**

Figure 7: ERG Overgeneration Profiles (on HP test suite) as of OCT-97; see Figure 3 for comparison.

deploy similar report generation techniques.

Going back to Figure 5, the comparison between the averages for the TSNLP data and the VerbMobil results reveals that indeed processing costs are in general much higher for the corpus sentences; thus, the test suite results cannot be taken as a direct predictor on analysis complexity relative to the input string length. Still, the VerbMobil data indicate a very similar overall time and space reduction as the corresponding class (six to nine words) in the TSNLP test suite; besides providing reassurance to the developers, it is demonstrated that the testing and optimization of the new lightweight unifier (and other modules) can often rely on comparative profiling using the artifically constructed data. After all, parsing the test suite data is much faster and cheaper than for most corpus sentences. Generally speaking, it is not at all clear where performance profiles obtained from test suites can be representative for a specific application and domain and where not. And while many of the metrics presented earlier should allow the developer to make an informed prediction about in which respects a particular (new) data sample and the available reference data differ and what effect there should be on system behaviour, clearly this area requires further investigation.

4 Comparing TSNLP to the HP Test Suite

Though the TSNLP test suite developers derived some design ideas and some test items from the HP test suite released in 1987, they pursued a more rigorous methodology in the construction of the data set, which should be visible in a comparison of the same ERG grammar against both suites (serving as a control experiment for the practical results reported earlier). For this purpose the HP test data was imported into the TSNLP test suite database and augmented with the minimal annotations required for the comparison. Above all, these included the classification into top-level TSNLP phenomena which was sometimes intricate because (i) the HP developers when constructing the data had assumed an underlying system that makes a number of grammar-specific distinctions and (ii) some of the HP items present multiple phenomena in (mostly unsystematic) combination, a state of affairs that, though potentially very rewarding, was deliberately excluded in the TSNLP data. Still, about 90 % of the HP data set could be meaningfully classified and taken into account for the comparison.

The data in Figures 6 and 7 summarizes the results of running the ERG grammar against the HP suite, and the list of phenomena is similar to that for the TSNLP suite, but with passive as the only example of diathesis alternations, and with the addition of two phenomena classes, for clause-level coordination and modification.

One quickly noticeable difference between the TSNLP and the HP sets is that the length of the average test item in the TSNLP collection is less than that of the HP examples, yet the average lexical ambiguity per test item is nearly the same for the two. Also obvious is the fact that there are proportionately about five times as many ungrammatical examples in the TSNLP suite as there are in the HP one. Although this is a natural consequence of the TSNLP methodology aiming for a systematic and exhaustive derivation of negative (i.e. ill-formed) test items, it will lead to a potential skewing of the results on overgeneration.

Other less obvious differences between the two suites which help to account for the variability in numbers comparing Figures 2 and 6 and Figures 3 and 7, respectively, are the following:

- there are no examples of stand-alone NPs in the HP test data;
- there are examples of sentence-level coordination in the HP but not in TSNLP, which contributes to longer average string length;
- there are no negative contractions in the HP suite (which helps boost the overall coverage);
- the HP suite contains examples of adverbial modification;
- there is almost no diathesis in the HP suite; and
- much less care was taken with the HP suite in trying to avoid unintended analyses for test items.

These significant differences in coverage of phenomena types and in methodology of construction make the comparison of the two test suites more difficult, but it is still reassuring that the percentage of overall coverage is very nearly the same for the two suites.

One striking difference between the two sets of numbers is that there is slightly greater lexical ambiguity in the TSNLP examples, in spite of a firm intent to minimize such ambiguity in the construction of test items. In contrast, another of the design goals of the TSNLP effort is vindicated by comparison: there is less structural ambiguity in the TSNLP items than in the HP sentences which, in fact, was among the inadequacies criticized more frequently.

While the comparison of the numbers in the two tables presents a strong parallelism of results for the ERG grammar, a more careful study of the test items themselves reveals that the TSNLP design presents the developer with the means to tune the items so they provide more illuminating data about the grammar being tested. More importantly, the developer can exploit the rich annotation capabilities provided in the TSNLP framework to dramatically improve the precision of the *successful analysis* notion for a given test item. Since unintended ambiguity, both lexical and structural, continues to be exhibited in both test suites built ten years apart, it seems clear that this ability to provide annotations about what properties of an item should be of interest will be crucial in making the test suite results even more revealing to the grammar developer.

5 Technological Assessment

Though it was not initially planned, the TSNLP project has devoted some effort to software building for test suite construction, maintenance, and application; among the main motives for including a set of test suite tools into the TSNLP package were that the consortium in an initial survey on publicly available test suites had found that virtually no specialized technology was available and, hence, the reuse, adaptation, extension, and application of existing test data is often severely hindered.

Judging from close to a full year of regular usage of the evaluation machinery sketched earlier, the most central tool from the TSNLP package, viz. the test suite database `tsdb`, has been found highly adequate in some ways and substantially limited in others. Oepen et al. (1997) give the reasons for implementing `tsdb` as a home-grown standalone relational database with a simplified query language as (i) suitability, (ii) extensibility, (iii) portability, and (iv) simplicity.

From the four desiderata, items (i), (iii) and (iv) are mostly met: the database compiles on a variety of platforms and is available in binary form for the most common (Un∗x) installations. Despite the lack of complete, detailed, and up-to-date user-level documentation, `tsdb` is easy to install and use; the simplified query language, though greatly restricted in functionality compared to full SQL, enables users to take advantage of the test suite structure and annotations without a full or even technical understanding of the underlying data model. The storage of `tsdb` data files in an ASCII representation that is directly accessible to standard Un∗x text processing utilities (`grep(1)` et al.) is often noted to increase the transparency and flexibility of the data set.

From an interface design perspective, integration into the PAGE grammar development environment was straightforward, using `tsdb` as a background process that communicates with PAGE through a standard (fifo) input and output channel.[6] Through a layer of wrapper functions in the (Common-Lisp) PAGE universe most of the `tsdb` functionality (retrieval and storage of data, setting parameters et al.) is made available to other modules (e.g. the report generator) as well as through the PAGE command-level user interface. Because the database does not require a centrally administered server process or other special privileges (as is the case for larger database systems), it can be distributed in binary form as part of the standard PAGE releases, such that someone installing and using the grammar engineering environment need not even be aware of the existence of the database.

Regarding extensibility, the user & application profile concept made it very easy to add relations to the database as needed to store PAGE- and ERG-specific profiling results and output specifications. Yet, database size and efficiency impose some limitations: given the wealth of information gathered during a single test run, it is not feasible to build up a progress profile reflecting several test runs in a single database. Instead individual test runs are stored as separate databases (i.e. directories), a design that delegates most of the bookkeeping over test runs into the filesystem; since comparative report generation is implemented within PAGE (in Common-Lisp) rather than within `tsdb`, the separation of databases poses no practical problem (and often simplifies queries). However, in a production environment where greater scalability was required — for example if developers wanted to store large amounts of data (e.g. complete features structures obtained as parsing results) as part of a profile — it might be desirable to substitute a full-blown (commercial) database system for `tsdb`; besides the extra installation and maintenance cost there should be no principled obstacle but a considerable gain in efficiency and general database functionality (full SQL).

Another important aspect in handling the test suites is visualization and editing support. TSNLP includes a graphical (form-based) editor for the core test data (test items, phenomena classification, and test sets); however, the tool does not allow the production or inspection of user & application profile data, because these can be of variable format. And although the bulk of the user & appliation profile is typically not produced manually (but gathered from a test suite run), at least the output specifications (e.g. the number of analyses expected for a test item, particular semantic formulae, upper limits for parser tasks or garbage collections) often require manual editing; as it stands, this currently amounts to text editing the ASCII representation of `tsdb` tables. Similar to the existing test data editor, a customizable browser and editor for the user & application profile layer of the annotation schema would be required.

In general, user interfaces remain an open issue. As both `tsdb` and the PAGE development shell are purely command- (or ASCII)-based, they nicely

[6] A similar approach proved feasible in integrating `tsdb` into the EU-funded ALEP grammar development system; see Oepen and Groenendijk (1997) for details.

match in their default mode of operation and give developers direct and full control over the capabilities of the underlying machinery. Yet, it can be difficult to train less experienced users (e.g. students experimenting with the grammar as part of a practical class; let alone linguistics professors) and enable them to benefit from the profiling engine and report generation facilities; the set of parameters in both creating a new database, doing a test suite run, and especially identifying the appropriate view on the data for report generation is substantial; here, again, a specialized graphical tool that visually presents the set of choices and hardwires a number of common interactions with the machinery would be expected to greatly enhance usability and acceptability. Strictly speaking, however, such a tool cannot be implemented as part of the test suite per se (and thus provided in the TSNLP package) because it closely interacts with the development system (i.e. the application to be evaluated). Here, again it seems, an inherent mutual dependency of the evaluation tool and system under evaluation manifests itself.

6 Conclusion: A Few Recommendations to Test Suite Developers and Users

The general conclusion on experimenting with the TSNLP data and tools within the ERG consortium is very positive. The test suite machinery proved to be an essential tool in practical grammar engineering and has enabled the developers to (i) precisely identify several (often systematic) deficiencies in various evolution steps of the ERG grammar and lexicon and (ii) greatly improved their understanding of the overall resource and thus guided and focussed the work on improving the grammatical competence and system performance. Above all, the TSNLP test data and technology were found sufficiently mature and flexible for regular deployment in a production environment; the methodology is sound and highly scalable.

Besides the criticism and suggestions for improvement presented in the earlier discussion, the following paragraphs summarize a few more general observations (and recommendations) that may be of interest to future test suite developers and users alike.

Iteration It is often assumed that application development and test suite construction should be completely independent processes (as was the case for most of the TSNLP effort) and that once a complete test suite is produced it will serve as a gold-standard reference. Obviously, this approach is impractical and, we argue, often just as much theoretically undesirable (a similar case is made in Gambäck (1997) for what is called compositionality evaluation there). At least for the grammar engineering evaluation scenario, a prototypical test run will provide feedback to the current state of the grammar, the processing components, the test data, and the evaluation machinery itself.

In initially setting up the testing environment, for example, a number of iterations (spread out over several months) were needed to

- implement and debug the tsdb to PAGE integration;
- add missing vocabulary to the ERG lexicon;
- identify useful competence and performance metrics and extract them from the processing components;
- make the test run processing fault-tolerant to various types of system errors;
- customize and correct the test data;
- define appropriate report generation strategies that present test results at different levels of granularity.

Obviously, both the grammar (plus development environment) and the test suite itself benefit significantly throughout each iteration. Thus, it seems plausible to integrate a similar process — invert the standard conception of the world and use a wide-coverage grammar as an evaluation metric for the test data — into future test suite building.

Use of Annotations One important use of annotations on test items is to make the intended use (or interpretation) of an item explicit, even if that item proves to have other analyses not considered by the developer. In the process of systematic derivation of ill-formed examples from grammatical test items, for example, within the C_Complementation (verb valency) phenomenon the developers have regularly applied a procedure of elimination of obligatory arguments. Accordingly, there is a large number of test items of the type '**Accounts for her.*' or '**Battles against it.*' (resulting from subject deletion) marked as ungrammatical, where apparently the test data authors either expected the initial capitalization to block a noun phrase analysis, or were simply unaware of the alternative reading. The ERG grammar (partly designed for processing speech recogniser output that has no reliable capitalization information), however, makes no use of the spelling or punctuation clues. Therefore, the profiling machinery records the noun phrase readings as genuine overgeneration.

Yet, the TSNLP annotations include an indication of the root category for all test items, such that the coverage and overgeneration scoring could easily be adjusted if the evaluators supplied a mapping from the categories derived by the ERG grammar (HPSG feature structures) to the annotations used in the test suite (atomic labels for morphosyntactic categories). Although such a transformation is trivially implemented (in fact, for the ERG grammar part of it already exists to allow the printing of phrase structure trees with atomic node labels), it is obviously dependent on the grammar and, again, requires maintenance as the grammar evolves throughout iterations.

Another obvious candidate to vastly improve the accuracy of automatically computed profiles would be the inclusion of further annotations — especially the underspecified dependency structure representation, chosen by the TSNLP developers as a (mostly) theory-neutral intermediate format that abstracts from idiosyncratic phrase structure assumptions — from the test suite into the scoring of results. Again, a mapping from actual parsing results to the format used in the test suite database would be required. As this mapping of HPSG tree representations into functor – argument structures is a non-trivial task however, it seems it will only be worth the effort if the dependency structure annotations in the test suite were complete and consistent; unfortunately, for all three (English, French, and German) test suites delivered by TSNLP this is not the case. Many test items have no or only incomplete dependency structure annotations; besides, manual inspection suggests there are remaining inconsistencies.

Summing up, the requirements on annotations imposed in a fully automated profiling approach are much higher than those for simple browsing of the test suite database. While the existing annotations already serve well to (i) explicate the underlying structure of the data (and some of the methodology applied to their construction) and (ii) facilitate the formation of various subsets that can be meaningfully interpreted, a better quality would be required for inclusion into the automated interpretation. Once more, it seems, an iterative approach building on mutual feedback and comparison between a large-scale grammar and the test suite development should be highly beneficial in improving the overall value of the diagnostic resource; after all, much of the knowledge and competence needed for test data writing is very similar to that required in grammar engineering.

Phenomena Dependencies As observed above, evaluation results can be significantly skewed where test items reflect more than one phenomenon being measured, unless the dependencies among phenomena are made explicit. While the TSNLP annotation schema already foresees marking one phenomenon as dependent on another (by means of a *presupposition* attribute in the phenomena description), this is often insufficient to capture individual relations between test items (from various phenomena), as would be required to allow the automated adjustment in coverage and overgeneration scoring.

To return to the example of passives discussed earlier, individual inspection of grammatical items that are rejected by the grammar reveals that the disappointing coverage for this phenomenon (see Figure 2) is partly due to missing complementation patterns in the lexicon; so, a number of failing test items are falsely charged to the passive analysis, even though they are essentially ruled out lexically already. And for phenomenon-internal motives it may even be the case that a particular verb frame has more statistical weight (i.e. a higher frequency in tokens) among the passive examples than in the C_Complementation phenomenon itself. Therefore, a simple-minded comparison of overall coverage on the phenomenon level cannot illuminate this issue.

Besides, the fact that the passive phenomenon lists complementation as a general presupposition is equally uninstructive for these cases. Instead it would be necessary to make the dependency among items explicit at the level of individual tokens. Thus, if and only if a passive item had an overt link to the corresponding usage of its matrix verb within the complementation data, the profiling machinery could detect where the failure of analysis originates and then automatically adjust the scoring. Clearly, this approach cannot be feasible for all types of dependencies that one may postulate between test items and may ultimately, at least in part, be specific to a particular application or grammar; it remains to be seen, to what extend the approach can be implemented.

The TSNLP annotation schema already foresees the grouping of (positive or negative) items into what is called a *test set*. For the existing data test sets are primarily used to relate (sets of) ungrammatical examples to the underlying well-formed item(s) from which the negative item(s) were derived. In principle, the same approach could be used to record dependencies across phenomena with only one extension that would be required, viz. to flag test sets for the nature (or purpose) of grouping that they represent. If such an attribute was added to the test set description, there could then be multiple layers of test item groupings, each encoding a different dependency type (purpose); the obvious example of passives, for instance, could use a label like 'presupposition' to specialize the overall dependency relation between phenomena classes on the test item level. Additional mechanisms may be needed to allow the developer to establish efficiently the relevant dependencies for a new test item, for example to support transitivity (or inheritance) of dependency relations, which would permit the automatic computation of the relevant background dependencies given some phenomenon.

Future Work While the present study is based on the hand-built TSNLP data, the tools are in fact well-suited for detailed analysis of data drawn from corpora, with the associated greater demands of larger vocabulary and greatly increased ambiguity. Since the test items can be readily annotated for alternate syntactic bracketings and semantic interpretations, the TSNLP machinery enables the developer to precisely monitor development of the grammar as it grows to accommodate such real-world data.

Based on a generalized notion of performance profiling for HPSG-type grammars (along the lines of section 3.5), it is expected that the close coupling of grammar development platform(s) and the TSNLP database approach will allow for an improved understanding of the inherent computational complexity in processing and key factors in processing costs. Aiming for an improved performance model, the profiling methods discussed will be used as an experimentation environment to evaluate how specialized control, learning, and compilation techniques can improve system behaviour. Thus, frequent profiling, analysis, and adaptation cycles become an integral part of regular system and grammar development.

Acknowledgements The research and implementation work has been carried out in close collaboration between CSLI Stanford and Saarbrücken University over the past few years. The authors are greatly indebted to numerous colleagues at the two institutions and their scientific vicinities for invaluable discussions and productive criticism. To name only a few, the feedback provided by John Carroll, Anne Copestake, Marius Groenendijk, Tibor Kiss, Sabine Lehmann, John Nerbonne, and Hans Uszkoreit has greatly contributed to the present results. In addition, audiences at Potsdam, Tusnad, Stuttgart, Tbilisi, and Heidelberg and several anonymous reviewers have given important comments on various versions of this paper.

Throughout the submission and a number of revision stages for the manuscript, Rob Gaizauskas has always been an exemplary editor — one equipped with a rare combination of patience, flexibility, and accuracy.

References

Black, Ezra, Steven Abney, Daniel P. Flickenger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings of the 4th DARPA Speech and Natural Language Workshop*. Pacific Grove, CA. Margan Kaufmann.

Carroll, John. 1994. Relating Complexity to Practical Performance in Parsing with Wide-Coverage Unification Grammars. In *Proceedings of the 31th Meeting of the Association for Computational Linguistics (ACL)*, 287 – 294. Las Cruces, New Mexico.

Erbach, Gregor, and Hans Uszkoreit. 1990. Grammar Engineering: Problems and Prospects. Technical Report 1. Saarbrücken: Computerlinguistik an der Universität des Saarlandes.

Flickinger, Daniel, John Nerbonne, Ivan A. Sag, and Thomas Wassow. 1987. Toward Evaluation of NLP Systems. Technical report. Hewlett-Packard Laboratories. Distributed at the 24th Annual Meeting of the Association for Computational Linguistics (ACL).

Gambäck, Björn. 1997. Processing Swedish Sentences. A Unification-Based Grammar and Some Applications. Technical Report (SICS Dissertation Series # 21). Swedish Institute of Computer Science.

Lehmann, Sabine, Dominique Estival, Kirsten Falkedal, Hervé Compagnion, Lorna Balkan, Frederik Fouvry, Judith Baur, and Judith Klein. 1996a. TSNLP User Manual. Volume 3: Test Data Documentation. Technical report. Istituto Dalle Molle per gli Studii Semantici e Cognitivi (ISSCO) Geneva, Switzerland.

Lehmann, Sabine, Stephan Oepen, Sylvie Regnier-Prost, Klaus Netter, Veronika Lux, Judith Klein, Kirsten Falkedal, Frederik Fouvry, Dominique Estival, Eva Dauphin, Hervé Compagnion, Judith Baur, Lorna Balkan, and Doug Arnold. 1996b. TSNLP — Test Suites for Natural Language Processing. In *Proceedings of COLING 1996*. Kopenhagen.

Nerbonne, John, Klaus Netter, Kader Diagne, Ludwig Dickmann, and Judith Klein. 1993. A Diagnostic Tool for German Syntax. *Machine Translation* 8:85–107.

Oepen, Stephan, and Marius Groenendijk. 1997. Towards Systematic Testing and Diagnosis Integrating TSNLP and ALEP. In *Proceedings of 3rd Alep User Group Workshop*. Saarbrücken.

Oepen, Stephan, Klaus Netter, and Judith Klein. 1997. TSNLP — Test Suites for Natural Language Processing. In *Linguistic Databases*, ed. John Nerbonne. CSLI Lecture Notes. Center for the Study of Language and Information.

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

Sparck Jones, Karen, and Julia Rose Galliers. 1995. *Evaluating Natural Language Processing Systems. An Analysis and Review.* Lecture Notes in Artificial Intelligence, Vol. 1083. Berlin - Heidelberg - New York: Springer.

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 15^{th} Conference on Computational Linguistics (COLING).* Kyoto.