Holistic Regression Testing for High-Quality MT

— Some Methodological and Technological Reflections —

**Stephan Oepen[♣♠], Helge Dyvik[◇], Dan Flickinger[♠],
Jan Tore Lønning[♣], Paul Meurer[◇], and Victoria Rosén[◇]**

[♣]Universitetet i Oslo, Boks 1102 Blindern; 0317 Oslo (Norway)
[◇]Universitetet i Bergen, Sydnesplassen 7, 5007 Bergen (Norway)
[♠]Center for the Study of Language and Information, Stanford, CA 94305 (USA)

{oe | helge | danf | jtl | paul | victoria}@emmtee.net

Abstract

We review the techniques and tools used for regression testing, the primary quality assurance measure, in a multi-site research project working towards a high-quality Norwegian – English MT demonstrator. A combination of hand-constructed test suites, domain-specific corpora, specialized software tools, and somewhat rigid release procedures is used for semi-automated diagnostic and regression evaluation. Based on project-internal experience so far, we comment on a range of methodological aspects and desiderata for systematic evaluation in MT development and show analogies to evaluation work in other NLP tasks.

1 Background

The Norwegian national initiative LOGON (Oepen et al., 2004) is a four-year research project aiming to deliver a high-quality, domain-adapted MT system for written texts. The project involves research groups at the Universities of Oslo, Bergen, and Trondheim and targets the domain of tourism-related information—specifically the translation of Norwegian instructional documents on back-country activities into English.[1] To investigate the portability of the general approach and re-usability of individual components, a Japanese – English instantiation of the system serves as a secondary test-bed, but is far less developed than the Norwegian – English main branch.

Emphasizing translation quality more than breadth of system coverage, the consortium has adapted a relatively conventional approach, viz. semantic transfer of logical-form meaning representations obtained from a broad-coverage parser of Norwegian and subsequent grammar-based generation from target language semantics. On top of this symbolic backbone, LOGON incorporates stochastic components at all processing levels, primarily to rank and select among competing hypotheses and, to a lesser degree, increase end-to-end robustness. Given significant progress in computational linguistics in the past two decades, a central goal of the LOGON initiative is to evaluate state-of-the-art grammatical frameworks and processing schemes as to the contribution they can make to a high-quality end-to-end MT system.

The specific meaning representation language used in LOGON is Minimal Recursion Semantics (MRS; Copestake, Flickinger, Sag, & Pollard, 2003), a family member of the class of flat, underspecified semantics that have been popular in computational semantics since the early 1990s. Syntactic analysis of Norwegian is based on an existing Lexical-Functional Grammar (LFG) implementation, NorGram (Dyvik, 1999), under development on the Xerox Linguistic Environment (XLE) since around 1999. The grammar has a lexicon comprising some

[1] See the public project web pages at ‘http://www.emmtee.net’ for additional information, including research groups and people involved, a project bibliography, and pointers to on-line publications.

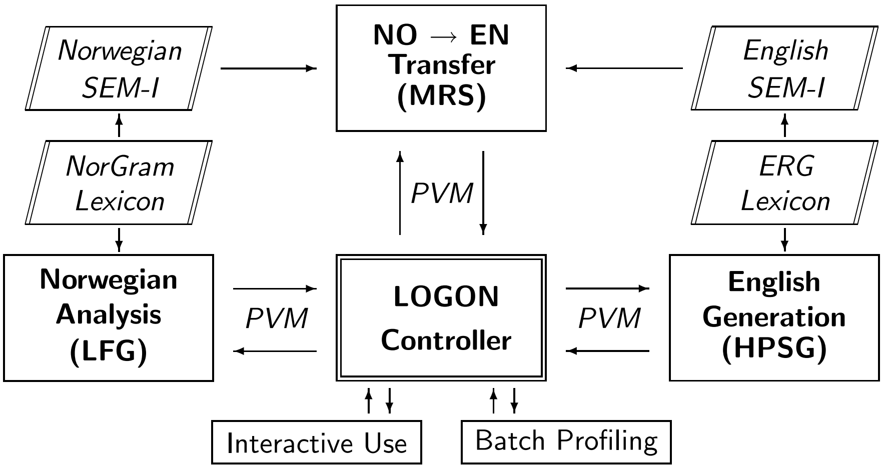

Figure 1: Schematic LOGON system architecture: the three core processing components are managed by a central controller that passes intermediate results (MRSs) through the translation pipeline. The Parallel Virtual Machine (PVM) layer facilitates distribution, parallelization, failure detection, and roll-over, and also provides the API to the profiling tools.

80,000 lemmas and, for use in LOGON, is substantially extended and augmented with an MRS module, deriving semantic representations suitable as input to transfer. English realization, in turn, employs the LinGO English Resource Grammar (ERG; Flickinger, 2000), a computational resource grammar in the Head-Driven Phrase Structure Grammar (HPSG) paradigm, and parts of the DELPH-IN open-source processing suite (Copestake, 2002; Carroll, Copestake, Flickinger, & Poznanski, 1999). Figure 1 shows a block diagram of the main components and flow of control; we will discuss the make-up of the LOGON controller and profiling modules and use of the Parallel Virtual Machine (PVM) further in Section 2 below. For the purpose of the present discussion, nothing much will depend on the specific linguistic or engineering details of these components. Let it suffice that the system-internal diversity of grammatical frameworks and externally-supplied pieces of software (each with several dozens of configuration options) greatly contribute to the inherent complexity of the LOGON demonstrator and, thus, the project-internal need for precise and largely automated quality assurance.

2 Competence and Performance Profiling for MT

While software configuration and release management, quality assurance, regression testing, et al. are standard fare even in small-ish commercial (software) development efforts, this is true to a lesser degree in academic engineering. In the following we will argue—by means of reflection on LOGON experience to date, assertion of conclusions to be drawn, and a series of example diagnostics—that a technology-oriented research initiative of any size not only must implement basic quality assurance measures, but potentially can gain a lot from organizing all development around a scalable, uniform, precise, and automated evaluation regime. LOGON has some eight active developers at three sites contributing to the source code repository of its core demonstrator and (as of January 2005) around 650 megabytes (or 11,000 files) of software (430,000 lines of code in six programming languages) and linguistic resources (some 488,000 lines of source) in version control, including seven third-party components. The project aims to complete at least three integration cycles per calendar year, i.e. test and release to other consortium members and outside partners new system revisions in each spring, fall, and winter. Within these integration cycles, major components (software and grammars alike) on average undergo revision every other week, closer to an integration deadline often once per day. Each component has an appointed maintainer, i.e. a developer who has primary responsibility for the well-functioning of the module both in isolation and as part of the demonstrator at the time of module submission. Component maintainers use a suite of automated regression tools to assess the effects of new revisions prior to submission. Holistic regression testing, in our universe, is a methodology of tracking system evolution in painstaking detail—specifically aiming at the automated and immediate identification of (potentially) detrimental changes.

Oepen & Flickinger (1998) and Oepen & Callmeier (2004) have proposed a technique dubbed *competence and performance profiling* (in analogy to profiling in software engineering) for the development of computational grammars and parsers. Profiling in their sense characterizes a strongly empiricist approach that makes frequent and detailed measurements of grammar and system behavior—in combination with in-depth analysis and automated comparison to earlier versions—a focal point in the engineering cycle. Thus, profiling could be viewed as glass box evaluation at the highest possible level. In a nutshell, the LOGON quality assurance approach to guiding developers in revising and integrating parts of the MT demonstrator is a generalization of the competence and performance profiling methodology and tools. A profile is a record of a large number of indicators of linguistic 'competence' and system 'performance' obtained from a batch run on some data set, i.e. a test suite or development corpus. For each input in the batch, the profile records measures like the number of candidate analyses, associated linguistic representations and probability scores for each of them, central parameters of resource consumption (use of cpu time vs. real time, memory use, et al.) and processing search space explored, and many more. Linguistic representations can comprise derivation trees, feature structures, semantic formulae, and others, while processing metrics may include counts of chart edges in parsing or generation, successful vs. failed unifications, copy operations, effectiveness of ambiguity factoring, and others. All data is recorded in a database, such that over time a detailed history of evolution is accumulated. At the same time, the database enables subsequent inspection (using a specialized tool) of individual profiles, further automated testing, and comparison to previous revisions, i.e. older profiles. In LOGON, profiling is regularly applied to each of the three major components—analysis, transfer, generation—in isolation, as well as to the end-to-end MT system as a whole.

The open-source [incr tsdb()] tool[2] provides a powerful, yet relatively easy-to-use interface to the profiling and analysis facilities, such that component maintainers in LOGON are able to perform their own diagnostic and regression evaluation at the same level of precision as is used by the coordinator in final integration testing. See Section 3 below for a few examples of standard regression tests and diagnostic routines applied in LOGON. For use in the MT scenario, the [incr tsdb()] profiler (which was originally developed for parsing with DELPH-IN HPSG implementations and specifically the LinGO ERG) has been interfaced to the XLE parsing system for LFG, and adapted for use with the LOGON transfer component and generator. While test inputs for the parser are plain strings, typically each an individual root-level utterance, the latter two components take a complete profile as their input—essentially using the MRS meaning representations produced by an earlier processing stage in the pipeline (analysis or transfer, respectively) as their inputs. The extended [incr tsdb()] database schema comprises some 175 fields in 18 relations, in contrast to the 48 fields in the original TSNLP data model (Oepen, Netter, & Klein, 1997).

Another benefit of the use of [incr tsdb()] in LOGON is its built-in support for distributed and parallel computation using the Parallel Virtual Machine (PVM; Geist, Bequelin, Dongarra, Manchek, & Sunderam,

[2] See '`http://www.delph-in.net/itsdb/`' for on-line access to the software, sample data sets for six languages, rudimentary documentation, and pronunciation guidelines ('*tee ess dee bee plus plus*').

1994) protocol for inter-process communication across (standard) networked workstations. Using a mini-HPC cluster of four dual-Xeon Linux nodes, batch processing of the main development corpus for LOGON can be accomplished in a matter of minutes, while a strictly sequential batch on a single cpu would take time on the order of an extended lunch break. Given internal complexity of each of the components and multiple dimensions of possible interactions, the ability to obtain an up-to-date system snapshot, empirically assessing the impact of most recent changes, is an important element in our highly data-driven approach to system engineering. Furthermore, the LOGON controller itself—brokering intermediate results within the translation pipeline—utilizes the [incr tsdb()] API in communication with individual modules, such that component-level profiling and end-to-end evaluation are merely nested instances of the same protocol. Thus, workload distribution, parallelization, failure detection, and roll-over are all capabilities that can be employed at either level.

3 Test Suites, Development Corpora, and Testing Routines

For the first year and a half or so of the project life-cycle, engineering in LOGON has revolved largely around foundational aspects of building an end-to-end translation system, establishing interfaces and agreement on the use of the MRS meaning representation language. These included the design and implementation of its MRS-based transfer engine (Oepen et al., 2004) and design or consolidation of semantic analyses in the source and target language grammars. Accordingly, controlled, hand-constructed test suites have played a central role in organizing development so far. Among the key advantages of hand-constructed test items are (a) the ability to provide systematic, often exhaustive coverage of select phenomena (including infrequent ones), where (b) each phenomenon can be presented in isolation or controlled interaction with other phenomena, (c) irrelevant variation and ambiguity ('noise' from a diagnostic evaluation point of view) can be avoided, and (d) vocabulary size remains limited to a minimum breadth (Flickinger, Nerbonne, Sag, & Wasow, 1987). Test suites typically include some degree of linguistic and extra-linguistic annotation. In the case of LOGON, these consist primarily of an internal break-down by linguistic phenomena and the determination of the intended or preferred reading(s) per test item, as with grammars of relatively broad coverage some ambiguities are inevitable. The latter annotation, in turn, is part of our regular treebanking (or MRS banking) of LOGON test suites and domain corpora (in the spirit of the LinGO Redwoods approach; Oepen et al., 2002). It is not only needed to maintain training material for stochastic processes (like parse and realization ranking), but at the same time makes regression testing a lot more concise. A general ability to track all changes in system outputs or intermediate analyses is very useful already, but the added information about which hypotheses are actually in-focus clearly facilitates more targeted diagnostics.

The two main test suites used so far exemplify aspects of semantic composition (e.g. variation in complementation and linking, modification, quantification, scopal relations, et al.) and various types of closed-class items (determiners, pronouns, conjunctions, et al.) as basic syntactic and semantic building blocks. The first of these, dubbed '*mrs*', is an adaptation of the existing LinGO MRS test suite to Norwegian, where the overall goal was to preserve semantic phenomena (where they are applicable to Norwegian) rather than produce direct translations; thus, the resulting 120 or so test items were then back-translated into English. Our closed-class test suite is called '*base*' and, at this point, has close to 300 items reflecting some 15 top-level phenomena (of which most are further sub-divided, for a total of 66 phenomena). Hand-constructed test data was complemented by a 110-item development sample of actual tourism text, where utterances were selected in order to present a balanced representation of phenomena typical

LOGON Coverage Profiles (Release 0.2 of December 1, 2004)

Phenomenon	**total items** ♯	**word string** ϕ	**analysis results** ϕ	**transfer results** ϕ	**generation results** ϕ	**distinct outputs** ♯	**overall coverage** %
Conjunctions	29	5.2	1.3	1.1	2.0	1.8	75.9
Definiteness	23	2.8	1.4	2.1	1.7	1.6	100.0
Demonstratives	12	4.6	2.8	5.6	2.8	2.6	100.0
Interrogatives	14	2.9	1.2	1.2	4.1	3.8	92.9
MassNouns	23	3.4	1.5	2.5	2.3	2.0	91.3
Modals	26	3.8	1.6	2.1	2.1	2.0	92.3
Partitives	14	4.5	1.0	2.7	1.8	1.6	85.7
Possessives	44	3.5	1.7	3.7	5.2	5.0	100.0
Pronouns	28	2.5	1.7	2.0	2.3	2.1	96.4
Quantifiers	42	3.5	1.3	2.9	2.9	2.7	88.1
Reflexives	13	3.5	1.9	1.3	1.4	1.4	100.0
Other	22	3.5	1.1	1.4	1.6	1.6	95.5
Total (‘*base*’)	**290**	**3.6**	**1.5**	**2.5**	**2.8**	**2.6**	**92.8**
Corpus (‘*tur*’)	**104**	**12.8**	**7.6**	**37.6**	**761.8**	**211.9**	**70.2**

(generated by [incr tsdb()] at 31-jan-2005 (10:35 h))

Figure 2: End-to-end coverage and average ambiguity rates (for analysis, transfer, and generation, respectively) on ‘*base*’ test set vs. ‘*tur*’ development corpus. Test suite results are broken down by linguistic phenomena, highlighting variation in input complexity and average ambiguity and, of course, remaining problem areas. Differences between the average number of generator results and the total number of final outputs indicate that there can be multiple paths through the LOGON pipeline resulting in identical (looking) translations, for example where different modifier attachments are not reflected at the surface.

for the genre. Test items in this development corpus, dubbed ‘*tur*’, are quite a bit longer (ranging to up to 30 words, at an average of 13) than the test suite examples, exemplifying phenomena like instructional imperatives, abundant directionals and locatives, complex coordination structures of all major syntactic categories, and of course rich combinations of these. As of late, the project is shifting emphasis to more corpus data, preparing to now scale up demonstrator development to target a 50,000-word corpus of running tourism text (see Section 4 below).

To provide at least some indication of how profiling for MT works for LOGON in practice, Figure 2 presents a summary view of translation coverage (on the ‘*base*’ and ‘*tur*’ sets) from the [incr tsdb()] profile database. Obviously, the core parameters of the two types of data sets (and LOGON system performance) show significant differences and often directly point to remaining development opportunities. In addition to such aggregated summary views on a profile, the [incr tsdb()] environment supports item-level analysis and off-line well-formedness checking of individual results, for example the identification of syntactically fragmented analyses (a deliberate robustness measure) or semantically anomalous MRSs (with respect to variable binding, internal cycles, or scope relations, say). We are in the process of combining this facility with an explicit semantic interface specification for each component (the resources labeled ‘SEM-I’ in Figure 1), in order to further sharpen unit testing: for each grammar, its SEM-I essentially enumerates the inventory of valid semantic predicates and appropriate roles and value constraints. Such explicit specification of inter-component interfaces will enable automated flagging of non-compliant intermediate results and further disentangle developer dependencies for ‘downstream’ components. Another example of the use of profiling in LOGON is given in Figure 3, where the incremental evolution of coverage in the transfer component is plotted over an integration cycle of five months. Despite occasional regression between snapshots, tracking the performance of a single module over time guarantees a positive long-term development trajectory. At the same time, of course, it is

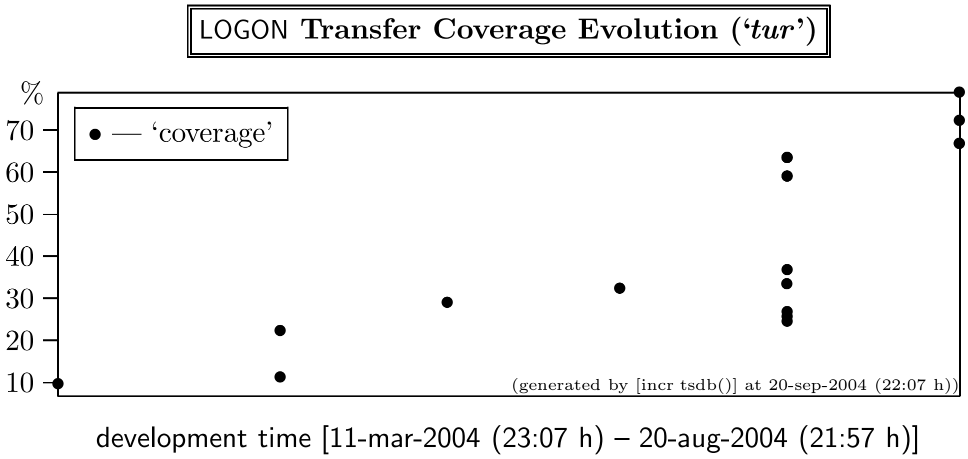

Figure 3: Evolution of grammatical coverage in the transfer component (when viewed in isolation) over a five-month development cycle. The two near vertical clusters of data points reflect intense development periods targeting this specific test corpus and daily system snapshots against it.

re-assuring to confirm that two focused engineering weeks against the '*tur*' corpus resulted in dramatic coverage gains (although more work remains).

4 Reflections, Preliminary Conclusions, and Outlook

The quality assurance measures we propose will sound largely obvious to some (industrial developers, say) and unnecessarily complex to others (some of our colleagues in academia, possibly). This may be a necessary dilemma in methodological arguments. The LOGON glass box set-up, for example, is in several respects similar to the MT system developed by Sågvall Hein, Forsbom, Weijnitz, Gustavii, & Tiedemann (2003), even though the range of diagnostic parameters and analysis techniques in LOGON is probably broader. More importantly, however, we believe that scalable, systematic regression testing requires both (a) structured, annotated input data (rather than flat text files) and (b) a database-like organization of test results (instead of a collection of system logs). Both seem necessary to accomplish a high degree of automation of diagnostics, in-depth precision, and user-level flexibility—all prerequisites in our view to establishing frequent regression testing as a pivotal point in development.

The by far larger Verb*mobil* project on spoken dialogue MT (Wahlster, 2000) had designated teams working on integration and testing. However, unlike in LOGON, few partners were able to both perform unit testing on their own modules and assess its performance when embedded within the full demonstrator. At least some Verb*mobil* participants in retrospect seem to judge a 'one-way' delivery paradigm (hand-over of modules to integrators) as a limiting factor to component-level development. Accordingly, LOGON strives to always make the full system easily accessible to all partners and, in fact, requires component maintainers to complete unit testing and end-to-end regression testing of new revisions prior to submission of each component.

At this stage, regression testing always runs the LOGON pipeline in exhaustive fan-out mode, although we expect to (have to) move to cascaded n-best mode—based on probability distributions over intermediate results—as ambiguity further increases (a production version could then be assembled by simply setting the final n to 1). For evaluation purposes, fan-out mode is not only attractive because in some cases early pruning would eliminate a globally best output, but also to make sure that all branches of processing (and some seemingly dis-preferred hypotheses) receive first-class consideration. As the project moves into the next level of empiricist, data-driven engineering, viz. against a bi-lingual 50,000-word text corpus, we expect that the profiling methodology and tools will enable the consortium to maintain the same high standards of controlling system outputs and diagnostic accuracy that we have found essential to our parallel and distributed development efforts already.

References

Carroll, J., Copestake, A., Flickinger, D., & Poznanski, V. (1999). An efficient chart generator for (semi-)lexicalist grammars. In *Proceedings of the 7th European Workshop on Natural Language Generation* (pp. 86 – 95). Toulouse, France.

Copestake, A. (2002). *Implementing typed feature structure grammars.* Stanford, CA: CSLI Publications.

Copestake, A., Flickinger, D., Sag, I. A., & Pollard, C. (2003). *Minimal Recursion Semantics. An introduction.* In preparation, CSLI Stanford, Stanford, CA.

Dyvik, H. (1999). The universality of f-structure. Discovery or stipulation? The case of modals. In *Proceedings of the 4th International Lexical Functional Grammar Conference.* Manchester, UK.

Flickinger, D. (2000). On building a more efficient grammar by exploiting types. *Natural Language Engineering, 6 (1) (Special Issue on Efficient Processing with HPSG)*, 15 – 28.

Flickinger, D., Nerbonne, J., Sag, I. A., & Wasow, T. (1987). *Toward evaluation of NLP systems* (Technical Report). Hewlett-Packard Laboratories. (Distributed at the 24th Annual Meeting of the Association for Computational Linguistics)

Geist, A., Bequelin, A., Dongarra, J., Manchek, W. J. R., & Sunderam, V. (Eds.). (1994). *PVM. Parallel Virtual Machine. A users' guide and tutorial for networked parallel computing.* Cambridge, MA: The MIT Press.

Oepen, S., & Callmeier, U. (2004). Measure for measure. Towards increased component comparability and exchange. In H. Bunt, J. Carroll, & G. Satta (Eds.), *New developments in parsing technology.* Dordrecht, The Netherlands: Kluwer Academic Publishers.

Oepen, S., Dyvik, H., Lønning, J. T., Velldal, E., Beermann, D., Carroll, J., Flickinger, D., Hellan, L., Johannessen, J. B., Meurer, P., Nordgård, T., & Rosén, V. (2004). Som å kapp-ete med trollet? Towards MRS-based Norwegian – English Machine Translation. In *Proceedings of the 10th International Conference on Theoretical and Methodological Issues in Machine Translation* (pp. 11 – 20). Baltimore, MD.

Oepen, S., & Flickinger, D. P. (1998). Towards systematic grammar profiling. Test suite technology ten years after. *Journal of Computer Speech and Language, 12 (4) (Special Issue on Evaluation)*, 411 – 436.

Oepen, S., Netter, K., & Klein, J. (1997). TSNLP — Test Suites for Natural Language Processing. In J. Nerbonne (Ed.), *Linguistic Databases* (pp. 13 – 36). Stanford, CA: CSLI Publications.

Oepen, S., Toutanova, K., Shieber, S., Manning, C., Flickinger, D., & Brants, T. (2002). The LinGO Redwoods treebank. Motivation and preliminary applications. In *Proceedings of the 19th International Conference on Computational Linguistics.* Taipei, Taiwan.

Sågvall Hein, A., Forsbom, E., Weijnitz, P., Gustavii, E., & Tiedemann, J. (2003). MATS. A glass box machine translation system. In *Proceedings of the 9th Machine Translation Summit* (pp. 24 –). New Orleans, LO.

Wahlster, W. (Ed.). (2000). *Verbmobil. Foundations of speech-to-speech translation.* Berlin, Germany: Springer.

Wilson, P. R., Johnstone, M. S., Neely, M., & Boles, D. (1995). Dynamic storage allocation: A survey and critical review. In *Proceedings of the International Workshop on Memory Management.* Kinross, Scotland: Springer.