Testing and Customisation of Test Items

Eva Dauphin, Veronika Lux, Sylvie Regnier-Prost,
Doug Arnold, Lorna Balkan, Frederik Fouvry,
Judith Klein, Klaus Netter, Stephan Oepen,
Dominique Estival, Kirsten Falkedal, Sabine Lehmann

Report to LRE 62-089 (D-WP4.2&3)
Test Suites for Natural Language Processing (TSNLP)
University of Essex
November 1995

Summary

Title	Testing and Customisation of Test Items
Project	LRE-62-089
Document ID	D-WP4.2&3
WP	Testing and evaluation
Date	November 1995
Principal Authors	Sylvie Regnier-Prost, Veronika Lux, Eva Dauphin Lorna Balkan, Frederik Fouvry Kirsten Falkedal Stephan Oepen
Authors	Doug Arnold, Lorna Balkan, Frederik Fouvry, Sylvie Regnier-Prost, Eva Dauphin, Veronika Lux, Judith Klein, Klaus Netter, Stephan Oepen, Dominique Estival, Kirsten Falkedal, Sabine Lehmann
Reviewers	The consortium
Status	public

Testing and Customisation of Test Items

INTRODUCTION

WP3.1 of TSNLP concentrates on the construction of general purpose test data, that is central to general language and applicable to a wide range of applications. In WP4, we have investigated the testing and customisation of this data to three applications - a parser, a grammar checker and a controlled language checker.[1]

The present report aims at presenting the methods used to customise, test and, to a certain extent, evaluate the general purpose test data produced in WP3 with three NLP systems. It also contains an account of the types of feedback provided to WP3.1 (in terms of necessary or desirable changes of the initial data) and to WP2 (first appreciation of the annotations' usefulness for testing purposes).

In addition to these testing experiences, some general principles and methodologies are addressed on the customisation and tailoring of general purpose test data for particular applications.

The result of this workpackage will be incorporated into a revised version of WP2.1 (see [*Balkan, Meijer, Falkedal et al., 1994*]) in the section on customisation.

This document reports on both testing and customisation (i.e. tasks WP4.1, WP4.2 and WP4.3) which were originally intended to be two separate reports. A more detailed report on the customisation and testing for SECC can be found in [*Balkan, Fouvry, and Kordoni, 1995]*.

Originally, the scope of WP4, as described in TSNLP+, was wider. It was intended to test the TSNLP test data on each application for at least two languages (except the controlled language checker, which was only to be used for English) and to investigate the tailoring of test suites to different corpora. However, following the mid term review in February 1995, it was agreed that we should concentrate on the production of more core test items (i.e. WP3.1), and the following reduced testing and customisation was agreed on:

Task 4.1 - Testing of the diagnostic test data on several different applications:

Subtask 4.1.1 - Testing of a grammar checker (GC)
- LC101 --- French by Aerospatiale

Subtask 4.1.2 - Testing of a controlled language checker (CLC)
- SECC --- English by the University of Essex

Subtask 4.1.3 Testing of monolingual analysis
- parser of DFKI HPSG system--- German by DFKI

Each testing site (Essex, DFKI and Aerospatiale) thus tested the TSNLP test data on only one application in one particular language and used as a testing basis a subset of the data produced in WP3 (since the data construction task was still in progress when these tests took place).

[1]Throughout the report, we assume familiarity with previous TSNLP reports and terminology, especially what concerns the annotation scheme, the TSCT data construction tool and the TSDB database.

Each of the systems concerned was tested under different conditions. DFKI was in a glass box situation with respect to the DFKI parser. Aerospatiale was in a black box situation with respect to LC101, which, however, allows the user to inspect details of the morphological and syntactic analysis provided for each test data. Essex was equally in a black box situation with respect to SECC, except that they had access to the controlled grammar and controlled language descriptions, but not to the system rules.

Further constraints on the testing which could actually be performed were imposed by the differences in access to the applications involved.
Essex tested their own data on a controlled grammar checker under development and had to agree with the SECC project on testing conditions (coverage of data and annotations tested, system access, partial confidentiality of the results, etc.)
Aerospatiale tested data mainly produced by ISSCO using a PC based commercial grammar checker.
Both Essex and Aerospatiale tested the test data prior to its import into the database.
DFKI, on the other hand, postponed testing until the database was available in order to be able to establish direct links between the data in the database and their parser.

Given these differences, and the inherent differences between the 3 languages covered by the project, we have chosen to keep the accounts of the testing from each of the sites in separate sections of the report and just provide a few general observations on the tasks of testing and customisation.

Finally, due to time and tool constraints acknowledged during the mid term review, we could not investigate the tailoring of test suites to different corpora, originally planned in WP4. By tailoring of test suites to different corpora, we mean establishing the frequency (and therefore relevance) of phenomena or combinations of phenomena in different corpora, with a view to filling in some of the "user specific" slots in the TSNLP annotation scheme (see [*Estival, Falkedal, Lehmann et al., 1994*]).

Some work on the checking of test suites against corpora was however carried out in D-WP3.2 (see [*Dauphin, Lux, et al., 1995*]) : in D-WP3.2 a chapter from a helicopter maintenance manual of Aerospatiale was analysed to establish what phenomena were frequent in this text type. It provided a way of evaluating the adequacy of the core data coverage with some real (technical) text. Most of the corpus analysis was done manually, and highlighted one of the main problems in tailoring test suites to text types or domains, namely the lack of adequate tools. We have thus observed that most of the existing corpus analysis tools concentrate on a morphologic parsing of large amounts of texts, identification of existing or potential domain specific terminologies, and statistic records on the lexical and typographic content of corpora. Such corpus annotations and inventories are clearly insufficient as a basis for automatic identification of higher level syntactic structures of the type needed to identify and count examples of e.g. subcategorisation frames or types of modifications in corpora.
This problem has also been discussed in D-WP5.2 (see [*Balkan, Fouvry et al.,1995*]).
So, until more and better tools become available for corpus analysis and/or for the mapping between test suite annotations and corpus tags, the tailoring of test suites to specific corpora remains a difficult problem.

The rest of this document is structured as follows:

- Section 1: General comments on testing
- Section 2 : General comments on customisation
- Section 3: Testing and Customising the French Test Items
- Section 4: Testing and Customising the English Test Items
- Section 5: Customising and Testing the German Test Items
- Section 6: Conclusion

Section 1: General comments on testing

1.1 Objectives of the testing

As already stated above, the purpose of the testing undertaken in WP4 was to use the different NLP systems to test various aspects of the adequacy of the TSNLP test data as diagnostic or evaluation tools, rather than to use the test data to test NLP systems (which would otherwise be the intended application of the test suites).

For this purpose, it has proven useful to distinguish between the actual test input strings (referred to as test inputs or test items in the present report) and their associated annotations as two separate objects to consider.

Ideally, a test of the adequacy of a test suite as a diagnostic or evaluation tool to reveal the coverage and/or deficiencies of an NLP system would have to establish at least the following 4 types of adequacy[2] :

1. the adequacy of the **test suite** (as a whole or for specific phenomena) to reveal reliably which syntactic phenomena are mastered by the system and which not ;
2. the adequacy of the **individual test inputs** to trigger the relevant types of response from the system ;
3. the adequacy of the **annotations** for retrieval, modification and extension of the test data ;
4. the adequacy of the **annotations** to help the tester account for and interpret the results of the testing.

In practice, it was not possible to test all 4 aspects of adequacy within the limits of WP4 :

1.1.1. Adequacy of the test data

- Testing the adequacy of the test suites to reveal reliably and exhaustively system deficiencies would have required that deficiencies of the systems used for the testing were known in advance.
  This knowledge was not available. However, the testing undertaken in WP4 enabled us to measure the major performance of the systems tested (i.e. both deficiencies and qualities) relative to the phenomena covered by the test data.

- Testing the adequacy of the individual test inputs (and systematic series of inputs for a specific phenomenon) to trigger the relevant types of system response means to test whether the inputs are adequate as tests representative of the phenomenon they are intended to illustrate, according to the annotations. This again amounts to testing that :
  * the vocabulary used in the test inputs is known to the system dictionaries ;
  * the vocabulary is non-ambiguous as to morpho-syntactic categories (semantic ambiguity falls beyond the scope of the TSNLP test suites but should be avoided as far as possible anyway) ;
  * the test inputs are structurally and semantically non-ambiguous with respect to the phenomena they are intended to illustrate ;
  * the format of the test inputs (sentential or sub-sentential units, capitalisation, punctuation, etc.) does not induce the system into erroneous performance.

[2]For a more detailed discussion of the scope and limitations of test suite testing, we refer to the WP2.1 reports.

The main bulk of WP4 testing has focused on this last aspect of adequacy, i.e. the adequacy of the test inputs to trigger the relevant types of response from the system. In this context, the availability of annotations significantly facilitated the testing procedures and largely enhanced the testing results.

Apart from testing the adequacy of the individual test items, we also tried to address the following issues :

- systematicity of the test items for each phenomenon,
- relevance of the different operations of derivation of ill-formed input (i.e. replacement, deletion and permutation) for each phenomenon, and
- relevance of relations between one or more negative test items and one unique positive test item and identification of other possible positive test items. Here, the testing experience provided us with some suggestions for the output specification of ill-formed test items, by pointing out differences between the well-formed test item associated with an ill-formed test item in TSNLP and the correction proposed by the NLP systems (GC and CLC) for this ill-formed test item.

1.1.2. Adequacy of the annotations

- Testing the adequacy of the annotations for various types of manipulation of the data. The richness of annotations made it easy to retrieve subsets of data based on different annotations (e.g. wellformedness, phenomenon, etc.)[3]

- Testing the adequacy of the annotations as a support in accounting for the results of testing was also undertaken, especially for subsets of annotations which proved the most informative for testing purposes, in the different testing environments available at partners' sites.

The testing experiments conducted in WP4 highlighted the usefulness of the information provided in the annotation scheme for testing purposes. It contains basically two types of information :

- the linguistic information, i.e. it provides the user of TSNLP with a detailed description of the linguistic characteristics of the test items and test sets and
- the information on the testing environment, i.e. it enables a user to precisely describe a specific testing environment and limit the relevance of test items to particular corpora, domains or applications.

For the testing experiments undertaken in WP4, the linguistic descriptions provided in the annotations attached to each test item were extensively used and proved useful both to facilitate the testing procedure and enhance the testing results.
Three types of linguistic information have been especially used :

- the linguistic description of the test items,
- the linguistic prerequisites for the use of test data , and
- the well / ill-formedness characteristics of the test data.

[3]Retrieval would have been even easier in the case of Aerospatiale and Essex, had the data been mounted on the database.

The information on the linguistic descriptions of the test items can be found :

- in the „analysis“ section of the item window
  The descriptions provided on the position, instance, category, function and domain of individual test items enable the user to be aware of the choice made by the test item writer and interpret the response of the system tested accordingly, especially in case of categorial or attachment ambiguities. For the ill-formed test items, the „comment“ field, in „phenomenon“ window or in „item“ for French, contains an explanation of the nature of the ill-formedness expressed in natural language (for example : *Preposition is wrong. It should be „de“.)*
- in the „name“ and ‚supertypes“ sections of the phenomenon window.
  The information provided under „name“, i.e. a structured description of the domain, the phenomenon and other divisions, help the tester to organise the testing, according to his own priorities. The use of a coherent and well-known terminology is also a significant advantage.
  The lists of phenomena given under „supertypes“ have been extensively used in the testing task.

The information on the linguistic prerequisites for the use of test data are available in the „Presupposition“ section in the phenomenon window. It is of particular use to interpret cases of unexpected response of the system tested (which may concentrate on phenomena non central for the test item writer).

The descriptions of the well / ill-formedness characteristics of the test data can be found :

- in the WF (well-formedness) section of the item window,
  The four values of well-formedness (1=correct, 0=not correct, 2=dubious, 3=relatively ill-formed w.r.t. the phenomenon) have been extensively used during the testing phase. Additional values or slots may be needed to express the ill-formedness relative to a particular application (for example, passives are well-formed constructions in the general language but are ill-formed for controlled language checkers based on the English Simplified Language).
- in the IDs of Positive and Negative Items of the set window,
  The identification references to „positive and negative items“ are essential to launch a testing procedure on a whole set of items, with the links between (most often) one positive TI and one or more negative TIs.

The testing experiments undertaken in WP4 showed that another level of information provided in the annotation scheme is potentially useful as a means to build, extract or retrieve specific or customised test items, in particular testing environments. Since the very goal of TSNLP is to build large amounts of general purpose test data, this kind of information was not really used, i.e. the default values have always been chosen. But future (academic or industrial) users may benefit from the availability of such information in the annotation scheme.
Three types of descriptions are possible in this part of the annotation scheme :

- description of a specific testing environment
- description of corpus or domain-specific test data
- description of application-specific test data.

The possibility to describe a specific testing environment is available :

- in the „purpose“ and „restriction“ sections of the phenomenon window :
  Instead of the default value „test“ currently used for „purpose“, a user may want to indicate the different releases of an NLP system tested, for non regression tests, for example.
  Instead of the default value „neutral“ currently used for „restriction“, a user may describe a specific application or a specific corpus (for example, to distinguish between phenomena specific to a helicopter maintenance manual and phenomena particular to a computer maintenance manual).

- in the „user“ section of the profile window
  This section is currently empty, to comply with the principles of TSNLP. It could be filled with information on the name and basic linguistic priorities of a user or a group of users.

The possibility to describe corpus or domain-specific test data is available :

- in the „origin“ and „register“ sections of the item window,
  Instead of the default value „invented“ currently used for „origin“, a user may refer to a specific corpus from which a user may want to extract test items.
  Instead of the default value „formal“ currently used for „register“, a user may describe other types of registers, such as „commercial“, „scientific publications“, „technical maintenance documentation“, „telex“ etc.. to distinguish between several types of texts or industrial domains.
- in the „interaction“ section of the phenomenon window :
  In this section, the default value „none“ is currently used, since it is the very goal of TSNLP to isolate phenomena represented in the test items. It can be filled with lists of phenomena which are combined in real texts.

The possibility to describe application-specific test data is available :

- in the „centrality“ section of the profile window
  Instead of the default value „core“ currently used for „centrality“, a user may want to prioritise test items according to his own needs or the degree of frequency in real corpora, and interpret the testing results accordingly.
- in the „relevance“ section of the profile window
  In the context of TSNLP, this section has not been extensively used, and most often, the default value „central“ is indicated for all three fields of relevance, i.e. parser, grammar checker, and CLC. A tester may however want to use this field to build a whole test set for one specific application.

As noted above, the testing experiences reported by each testing site in the present document do not cover the whole of the test data produced by the end of the TSNLP project, since the WP3 test construction was not finished when WP4 took place. A subset of such test data has been selected by each testing site on the basis of the quantity and maturity of test data available in each language when they actually began the testing tasks[4].
There may therefore be some differences from one language to another between the types of phenomena treated, the exhaustiveness of the annotations, and the ratio of positive/negative test items.

[4] It was agreed with SECC that between 500 and 1000 test items be produced to test SECC. Details of these items can be found in [*Balkan, Fouvry and Kordoni, 1995*].

Section 2: General comments on customisation

Customisation involves either taking existing general purpose data and tuning it, or writing additional new test data. In this work package, we investigated both options, though Aerospatiale mainly concentrated on the „tuning“ task while the University of Essex conducted both tasks in parallel.

In both customisation contexts, we tried to apply the general guidelines which were discussed in WP2.1 (see [*Balkan, Meijer, Falkedal et al., 1994*]) , including:

- Format of test items:
  It is very important that the format of the test items be compatible with that allowed by the system.
  Also, if output annotations are given, then they too should conform to the format produced by the system, to enable automatic comparison. Format includes not just the markup and character set of the test item, but also punctuation conventions and whether the item is a sentence or sub-sentential constituent.
- Avoid introducing unnecessary complications:
  Where a new test item needs to be produced, that is not covered by the core phenomena, then introduce it in as simple a context as possible where other phenomena are known (from previous testing) or must be assumed to be unproblematic.
  The TSNLP guidelines (WP3.1 section 2.4) on general data construction are meant to facilitate this.
- Lexical replacement:
  A lexical item which is chosen to replace a word from the original test suite should be compatible in the sense that it has no other features that could influence the results of the performance.
  It should have the same syntactic (and in some cases also semantic) requirements as the word it is replacing.

2.1 Tuning general purpose data

The processes involved in tuning general purpose data to a particular application or domain include:
- lexical replacement,
- adapting test items to the input required by the system,
- creating output specifications.

2.1.1. Lexical replacement

Lexical replacement is an important issue and may be needed for three major reasons :
- a word used in a test item is unknown to the NLP system used for the testing and is detrimental to the parsing of the test sentence. Lexical replacement is necessary as the test suite should not test for system behaviour in the presence of unknown words.
- a word used in a test item is syntactically or semantically ambiguous and may generate a different interpretation. Some such ambiguities can be detected already during an initial reading of the test suite where it is possible to adopt a more external and „neutral“ point of view on the test items than during the construction phase. Identification of further ambiguities are also the inevitable result of the testing phase where the system helps us to pinpoint them.
- the word is not allowed in the controlled language.

2.1.2. Adapting test items to the input required by the system

This includes formatting conventions and expanding sub-sentential component items to full sentences for systems that require full sentences as input.

Test suites may, e.g. contain test items which consist of just a noun phrase, an adjectival phrase or an adverbial phrase. NLP systems may, on the other hand, not be geared to treat such phrases outside of a larger context (e.g. APs as part of NPs or NPs as part of full sentences).

Although it makes sense to include some such phrasal test items in any test suite in order to discover how a system reacts to them, both testing of systems and validation of test suites require that they also appear in sentence contexts.
If sentential equivalents of phrasal test items are not already part of the test data (which they were not in all cases in the WP3.1 data), this kind of customisation is necessary and may be carried out at different stages.

Customisation may prove necessary, before testing, as an initial step in rendering the test data formally adequate for a given application. This was the case for Essex who had to customise most of the English core data for the SECC controlled language checker (see section 4 on customisation)

In other contexts, customisation proves necessary as a result of a first round of testing. This was the case for Aerospatiale who was in a position to submit the initial French core data to the French grammar checker directly and *a posteriori* perform the customisation that eventually appeared necessary.
The initial hypothesis (based on the descriptions of the system‘s capacities provided in the user manual) was that the system used for testing (LC101) could correctly parse and correct sub-sentential components without context (especially isolated APs, PPs and NPs).
The erroneous diagnoses produced by the French grammar checker on such sentence components highlighted the necessity to adapt and customise the initial test items for the two following reasons :

- the syntactic analysis of these sub-sentential test items was wrong, because of these TIs‘ categorial ambiguity (for example, NPs analysed as sentences),
- the use of punctuation had a strong influence on the parsing results (depending on the presence or the absence of a full stop at the end of the test item, LC101 yielded different results and corrections).

The augmentation of phrasal test items into full sentences must of course follow the general TSNLP guidelines, in this case, the one stating that each phenomenon should be represented in the core data in as much isolation as possible in order to minimise potential „noise“ from cooccurring phenomena present in the context. Where representing a phenomenon in complete isolation is not possible, it is a language specific problem - and may also be a system dependent problem - to decide what the simplest possible context is. We especially examined two concrete examples, i.e. embedding isolated NPs and APs into larger phrases or full sentences and came to the following conclusions.

- Embedding an isolated NP into a full sentence

  If a given system does not accept, or does not respond reliably to NPs presented out of sentence context, two solutions of embedding the isolated NP may be considered, at least for languages like French and English which do not have to consider case markings. One of these solutions would be to make the NPs the subjects of intransitive verbs, thus allowing for simple copying of already existing annotations. However, if the phenomenon represented by the test items is agreement inside NPs, interference with subject-verb agreement cannot be excluded, especially not in case of ill-formed test items.

The alternative solution (especially for French and English) would be to embed the NPs as objects of transitive verbs where the risk of interference between NP agreement and subject-verb agreement does not exist. In either case, initial tests should ensure that the intransitive or transitive verbs used are indeed treated as such by the system.

- Embedding an isolated AP into a larger phrase or a full sentence

  Similarly, if APs cannot be properly analysed by a system without any context, two possible ways of embedding such APs can be considered, i.e. embedding them in an NP or in a full sentence, as predicate.
  However, several examples showed that embedding an ill-formed AP exemplifying adjectival complementation in an NP creates PP attachment ambiguities, so that the result is likely to be a well-formed NP.
  Examples for French test data :
  * Well-formed AP : *cordial avec l'ingénieur*
  * Ill-formed AP: **cordial de l'ingénieur*
  * AP embedded in a (well-formed) NP: *Le respect cordial de l'ingénieur.*

  We thus concluded that an AP should preferably be embedded in a sentence as predicate, even though some potential ambiguities may still remain in the resulting sentence (i.e. the PP may be attached to the verb and several agreement phenomena may also be introduced i.e. subject-verb and subject-predicate agreements, especially for French).
  * customised example of well-formed AP : *L'homme est cordial avec l'ingénieur*
  * customised example of ill-formed AP (embedded in a sentence as predicate) : **L'homme est cordial de l'ingénieur.*

For in-depth testing/evaluation purposes, it is arguable that a test suite should contain test items for e.g. NP internal phenomena both in isolation (ending with a full stop or not), in subject position and in other functions in sentences. In practice, it may however be worthwhile to try to determine, based on just a few examples, whether these various options really trigger different system behaviours or if not, opt for just one of them.

2.1.3. Creating output specifications

Output specifications may include suggestions for corrections in the case of grammar checkers and controlled language checkers, indication of rule violations, parse trees, etc.

The University of Essex especially concentrated on output specifications, as SECC was most interested in this type of customisation. The creation of an output-data file for SECC is described in section 4.

Aerospatiale, on the other hand, concentrated on the type of customisation described in 2.1.1 and 2.1.2. Some discussions on the problems of defining output specifications for a grammar checker can however be found in section 3.

2.2 Creating new test items

It is unlikely that any general purpose test suite will contain just the set of phenomena or combinations of phenomena that one is interested in testing, and it is to be expected that some expansion of the data will be required for any real application.

This was particularly the case for SECC, where many new items had to be created to test phenomena not covered in the core data. Due to time constraints, the writing of test items was limited to the testing of „simple“ phenomena but an interesting extension would be to write test items specifically testing for various combinations of cooccurring phenomena, and to investigate some application specific items such as „traps“ for grammar checkers (of the types discussed in WP2.1).

Section 3: Testing and customising the French test items

3.1 The testing process

3.1.1. General presentation of the French grammar checker LC101

Le Correcteur 101 (LC101) is a commercial spelling and grammar checker/corrector, developed by Machina Sapiens. It is available for Macintosh (we did the testing on Macintosh) and Windows and works either in an interactive or in a batch mode.

The batch mode allows the user to submit lists of test items (TIs) to the system and to get lists of diagnoses.
In this output, the diagnoses are presented in particular layouts, so that we can distinguish :

- TIs considered as correct ,
- TIs considered as incorrect and which LC101 can correct (the corrected parts are in bold, suggestions for correction are in italics) .
- TIs that could only be partially analysed and that are considered as incorrect though LC101 cannot correct them. Such TIs appear with slashes (the slashes should indicate where the analysis has failed).
- TIs with a missing word,
- TIs with an unknown word (the word appears between brackets) or an unknown grammatical structure,
- TIs which could not be analysed for lack of time or memory.

In the interactive mode, the diagnosis on the selected part of text is presented in a similar layout, and the user has access to further information: on-line explanations are attached to the corrections and to the correction suggestions as well as to the ambiguities detected. On the user request, LC101 also provides an analysis of the sentence (which, according to the user manual, is in conformity with the grammar of French as defined by Grevisse[5]).

The user can also look up the LC101 dictionary, which contains „*almost all the words of Le Petit Robert, Le Petit Larousse and Multidictionnaire"*. He can express personal needs concerning the vocabulary by creating personal dictionaries, taken into account by LC101.

LC101 is described in the user manual as „*le seul correcteur capable de faire une analyse grammaticale complète de phrases complexes en français, comme le ferait Monsieur Grevisse lui-même* „ (sic) *(the only checker which is able to achieve complete grammatical analysis of complex sentences, as Monsieur Grevisse himself would do.*). The user manual highlights the fact that LC101 really performs an analysis of the sentences, and is based on large grammars as well as on „*innovative technologies from AI, linguistics and computer engineering"*.

3.1.2. Description of the testing procedure

The subset of TSNLP French test data used by Aerospatiale and tested on the French grammar checker „Le Correcteur 101", illustrated four phenomena (complementation, tense_aspect_modality, agreement and word_order) and three domains (Clause, NP and AP) distributed as follows :

[5] Grevisse is also a reference for TSNLP.

French Phenomenon	gram. test items	ungram. test items	dubious test items	Total
Adjectival_Phrase_Complementation	9	23	0	32
Clause_Agreement	60	103	0	163
Clause_Complementation	80	253	0	333
Clause_Tense-Aspect-Modality	73	273	2	348
Noun_Phrase_Agreement	138	539	0	677
Noun_Phrase_Complementation	6	11	3	20
Noun_Phrase_Wordorder	29	31	0	60
TOTAL	395	1233	5	1633

The general results obtained are presented in the table below (a more detailed analysis of such results can be found in the following subsections) :

TSNLP data	LC101 results		
Well-formed test items	items corrected (√)	items considered correct (•)	items not analysed (? or *)
395 TIs	3,3 % (13 TIs)	96 % (379 TIs)	0,7 % (3 TIs)
Ill-formed test items			
1238 TIs	58 % (719 TIs)	16,4 % (203 TIs)	25,6 % (316 TIs)

We had basically two objects under consideration : the test items themselves and their related annotations. The testing method thus followed this distinction and was organised in the three following phases :

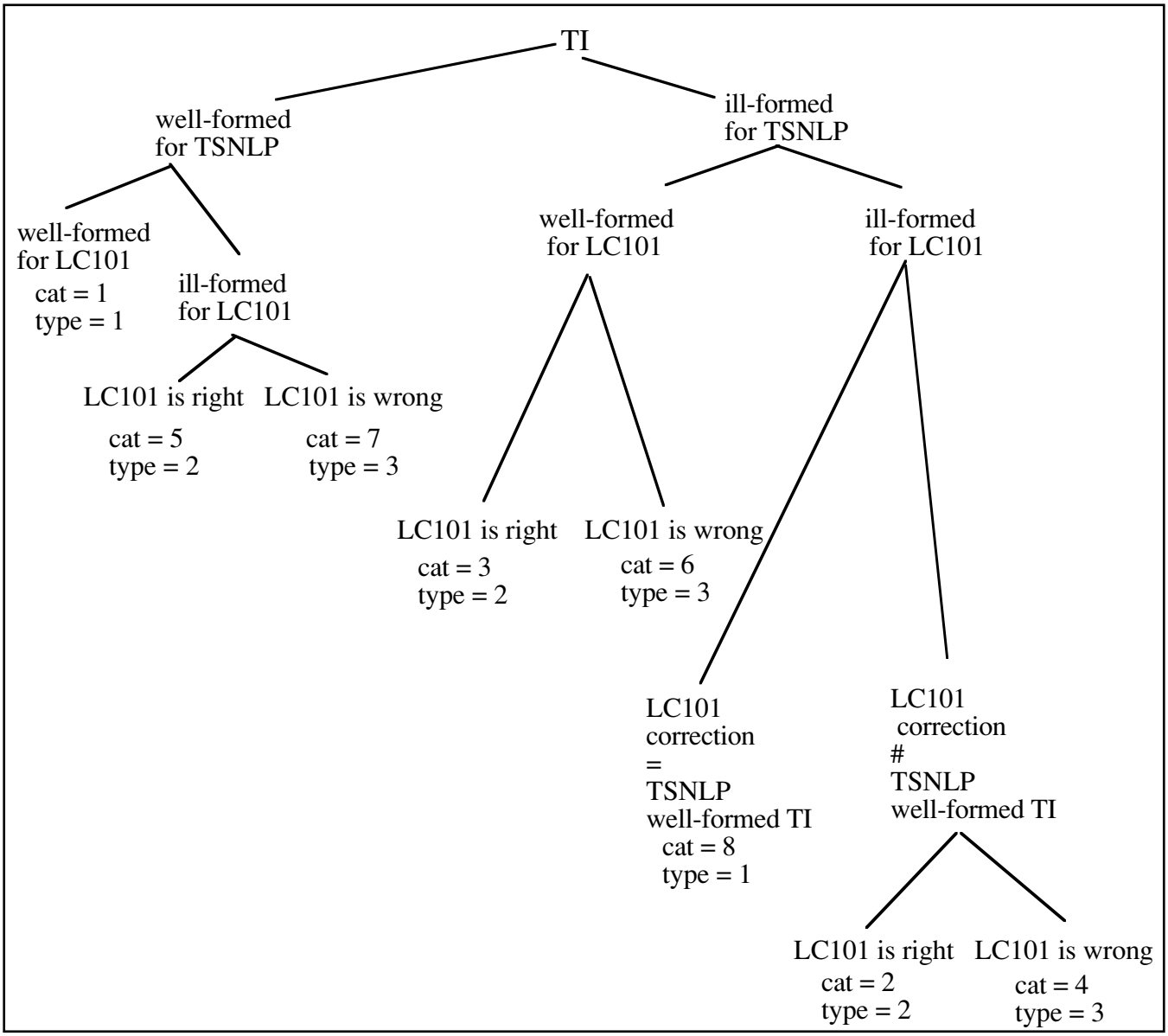
- testing of the test items ;
- testing of the annotations. A subset of the annotations has been selected and extensively tested, when using the LC101 grammar checker ; it consisted mainly in the information on test item phenomena and domains, the notion of „TI sets“, the information on the test items‘ well-formedness and, in some cases (especially cases of ill-formed TIs) the comment field.
- correction and customisation of test data.

In practice, the testing consisted in:

- feeding the test items to the system tested,
- checking that the well-formed test items are treated as such (i.e. checking that „no error is detected“),
- checking that the ill-formed test items are treated as such (i.e. checking that at least one error is detected by the system).

The explanations given by the system for the correction of ill-formed test items were then compared with the corresponding TSNLP annotations (given in terms of phenomenon names, parameters or as comments). If the correction proposed by the system for a given ill-formed test item did not correspond to the well-formed test items defined in TSNLP, further inspection of the analysis given by the system was undertaken in order to determine whether the system analysis was deficient or, if not, whether and how the test item was inadequate.

The results of this comparison fell into 8 categories, distributed in three major types (excluding the cases where the LC101 system could not produce any diagnostic), as shown on the following figure:



Type 1 : The diagnoses produced by LC101 correspond to TSNLP TIs and annotations, in the following two cases :

- Category 1 : A TI is considered well-formed both by TSNLP and by the system.
  Ex.: *Le travail* -> no error detected

- Category 8: A TI is considered ill-formed both by TSNLP and by the system and the correction proposed by the system corresponds to the associated well-formed TSNLP TI (or to one of the associated well-formed TIs).
  Ex.: **La travail* -> corrected as: *Le travail*

Type 2 : The diagnoses produced by LC101 do not correspond to TSNLP TIs and annotations but are plausible, in the following three cases :

- Category 2: A TI is considered ill-formed both by TSNLP and by the system. The system proposes a correction that is plausible, though it is not the associated well-formed TSNLP TI (nor one of the associated well-formed TIs).
  Ex.: **L'avenir l'entreprise.* -> corrected as : *L'avenir l'entreprit.*

- Category 3: A TI is considered ill-formed by TSNLP but well-formed by the system. Indeed the system provides an analysis that shows that the TI can be well-formed.
  Ex.: **Accepte l'invitation.* -> no error detected

- Category 5: A TI is considered well-formed by TSNLP and ill-formed by the system. The system is right and proposes a well-formed correction.
  Ex.: *divers entreprises* -> corrected as *diverses entreprises*

Type 3 : The diagnoses produced by LC101 do not correspond to TSNLP TIs and annotations and are erroneous, in the following three cases :

- Category 4: A TI is considered ill-formed by both TSNLP and by the system. The system proposes a correction that is ill-formed.
  Ex.: **Il neiger.* -> corrected as: **Il neigé.*

- Category 6: A TI is ill-formed for TSNLP and well-formed for the system. The system is wrong (= silence).
  Ex.: **tout hommes* -> no error detected

- Category 7: A TI is considered as well-formed by TSNLP and ill-formed by the system. The system is wrong (= noise).
  Ex.: *Ce sont les hommes.* -> corrected as: *C'est l'homme.*

Test items which yielded results falling into one of the types 1 and 3, i.e. into categories 1, 8, 4, 6 and 7, can, at least within the limits of the present round of testing, be said to be adequate as tests for the phenomena and parameters they represent (either they help us to define the system's capacities - categories 1 and 8 - or they help us to track system's deficiencies - categories 4, 6 and 7 -). Adequacy of the TIs is examined in more details in section 3.2.

Test items which yielded results falling into type 3, i.e. into categories 2, 3 and 5 are obviously not adequate as tests of what they are intended to test. The analysis and corrections given by LC101 can here be used to pin down the inadequacy which will most frequently be a categorial, lexical or syntactic ambiguity which the test data constructor had not been aware of. In these cases, the test items needed to be changed and re-tested before they could be classified as adequate, as explained in section 3.3.

For the testing of the adequacy of the French data in WP4, we decided to concentrate on ensuring that the test items triggered either the „correct" correction relative to a given parameter or an erroneous correction.

In the sections below, we will discuss our „positive" and „negative" findings in more detail and we will especially describe some of the cases where mismatches between expected output and actual corrections led to revision of the test data.

3.2 Adequacy of the test data to reveal system deficiencies

The evaluation of the systematicity of the system's deficiencies when dealing with a particular clearly-defined phenomenon, led to a classification of the results obtained for each phenomenon according to the following matrix:

LC101 / WP3	positive items	negative items
systematic	+ -	+ -
incoherent	- / +	- / +

For certain phenomena, we observed that all test items labelled as ill-formed by TSNLP for a given phenomenon in a subdivision were systematically wrongly parsed by LC101 and always produced erroneous diagnoses - either such test items were considered well-formed by the system and no error was detected, or the corrections proposed by the system were ungrammatical.

We have thus identified a series of data which have proven adequate to reveal system's deficiencies. Results are presented below in a rough classification, based on the type of system deficiency hypothesised:

- lexical deficiency
- (morpho-) syntactic deficiency

3.2.1. Lexical deficiency

The TSNLP test data is mainly a syntactic test suite and should not be used as a tool to track a system's lexical deficiency. Though, in one case, using a subset of the French data to test NP agreement, we end up postulating a lexical deficiency of the system.

For NP agreement, some TIs showed that the LC101 properly deals with determiner/noun agreement.

Le travail. -> no error detected
**La travail.* -> corrected as: *Le travail*.
LC101 explanation: *The article agrees with the noun it determines.*
Les travaux. -> no error detected
**Les travail.* -> corrected as: *Les travaux*.
LC101 explanation: *The noun agrees with the article it determines* .

But we also found the following (potential) counterexample :

La plupart. -> no error detected
Le plupart. -> *?* LC101 detects a mistake but cannot describe nor correct it.

In order to understand why the LC101 properly corrected **La travail*. but not **Le plupart*, we had to go further in the testing and write additional test items (lexical replacement). However, it is doubtful that these items have to be added to the Test Suite, as they already have „syntactic representatives" in the Test Suite.

- First lexical replacement : *décision* instead of *plupart*

Through this lexical replacement, we wanted to check the following hypothesis : LC101 detects a feminine article instead of a masculine one but not the opposite.

**Le décision.* -> corrected as: *La décision*.
LC101 explanation: *The article agrees with the noun it determines.*

This result showed that our hypothesis was wrong.

- Second lexical replacement : *majorité* and *totalité* instead of *plupart*

Through this lexical replacement , we wanted to check the following hypothesis : LC101 produces similar diagnoses with other words similar to „*plupart*“, requiring a complement (*La plupart de*...) and expressing a quantity.

**Le majorité*. -> corrected as: *La majorité*.
LC101 explanation: *The article agrees with the noun it determines*.

**Le totalité*. . -> corrected as: *La totalité*.
LC101 explanation: *The article agrees with the noun it determines*.

This result shows that our second hypothesis is also wrong.

Here, we have failed to generalise our observation and cannot postulate a syntactic deficiency. We only can say: on det_N agreement, there is something wrong with LC101 when dealing with the word „plupart“.
Careful inspection of the error diagnosis given by the system would, however, allow us to detect that LC101 has encoded „*la plupart*“ as a single lexical entry, classified as a 3rd person singular pronoun. The set of French test items for determiner-noun agreement therefore allowed us to conclude that the phenomenon seems well mastered by the system provided the words involved are actually coded as determiners and nouns but that certain, pragmatically determined lexical encodings limit the reliability of the corrections which can be expected. A certain (desirable) redundancy of the test items for the same phenomenon also allowed us to assume that the amount of such deviant codings is limited.

3.2.2. (Morpho-)syntactic deficiency

The French data used for the testing of LC101 allowed us to make some hypotheses on the (morpho-)syntactic phenomena mastered by LC101 and on the (morpho-) syntactic phenomena that are not properly treated.

a. Noun/adjective agreement in NPs

This first series of examples shows the results obtained when testing the French TIs on noun/adjective agreement in NPs :

La bonne question -> no error detected
Les bonnes questions -> no error detected
**Le bonne question* -> corrected as: *La bonne question*.
**La bon question* -> corrected as: *La bonne question*.
**Le bon question* -> corrected as: *La bonne question*.
**Les bonne question* -> corrected as: *Les bonnes questions*.
**La bonne questions* -> corrected as: *Les bonnes questions*.
**La bonnes question* -> corrected as: *La bonne question*.
**La bonnes questions* -> corrected as: *Les bonnes questions*.
**Les bonnes question* -> corrected as: *Les bonnes questions*.
**Les bonne questions* -> corrected as: *Les bonnes questions*.
**Le bonnes question* -> corrected as: *La bonne question*
**La bons questions* -> corrected as: *Les bonnes questions*.
**Le bons questions* -> corrected as: *Les bonnes questions*.
**Le bonnes questions* -> corrected as: *Les bonnes questions*.
**Le bonnes question* -> corrected as: *La bonne question*.
**La bons question* -> corrected as: *La bonne question*.
**Les bons question* -> corrected as: *Les bonnes questions*.
La question la plus habituelle -> no error detected
Les questions les plus habituelles -> no error detected

**Le question la plus habituelle*	-> corrected as*: La question la plus habituelle*.
**La question le plus habituel*	-> corrected as*: La question la plus habituelle*.
**Le question le plus habituel*	-> corrected as*: La question la plus habituelle*.
**Les question la plus habituelle*	-> corrected as*: Les questions les plus habituelles*.
**La questions la plus habituelle*	-> corrected as*: Les questions les plus habituelles*.
**La question les plus habituelles*	-> corrected as*: La question la plus habituelle*.
**La questions les plus habituelles*	-> corrected as*: Les questions les plus habituelles*.
**Les questions la plus habituelle*	-> corrected as*: Les questions les plus habituelles*.
**Les questions les plus habituels*	-> corrected as*: Les questions les plus habituelles*.
**Les question les plus habituels*	-> corrected as*: Les questions les plus habituelles*.
**Les question le plus habituel*	-> corrected as*: Les questions les plus habituelles*.
**La questions les plus habituels*	-> corrected as*: Les questions les plus habituelles*.
**La questions le plus habituel*	-> corrected as*: Les questions les plus habituelles*.
**Le questions la plus habituelle*	-> corrected as*: Les questions les plus habituelles*.
**Le question les plus habituelles*	-> corrected as*: La question la plus habituelle*.

A first conclusion was that LC101 seems to master agreement phenomena in NPs, and in particular, the noun/adjective agreement. There is no problem with adjectives in the superlative. However, the following TIs seem to show that LC101 does neither detect nor correct errors on agreement between a noun and its modifier adjective in an NP, if the adjective is an irregular superlative.

**Les pire question*	-> no error detected
**Le pire question*	-> no error detected.
**Les meilleurs question*	-> no error detected
**La meilleur travail*	-> corrected as*: Là meilleur travail* or not corrected
**Le meilleure travail*	-> corrected as*: La meilleure travail*
**La meilleure travail*	-> corrected as*: Là meilleur travail* or not corrected
**Les meilleur travail*	-> no error detected
**Les meilleures travaux*	-> no error detected

b. Quantifiers, complex determiners (in NP_agreement)

A second example led us to a hypothesis on a deficiency in the treatment of quantifiers and complex determiners.

**toute homme*	-> no error detected
**tout hommes*	-> no error detected
**toutes question*	-> no error detected
**toute questions*	-> no error detected

Further testing allowed us to discover that the system‘s performance changed to correct error analysis and correction of both superlatives and quantifiers under either of the following conditions :

- when the phrasal test items were embedded into full sentences with or without a full stop at the end. This observation corresponded to what we expected ;
- when the phrasal test items were ended by a full stop. This behaviour was not foreseen by the WP3.1 test data construction practice according to which full stops

were used in sentential test items but not in phrasal items. We concluded from this that a test suite should contain both phrasal items without full stops (as e.g. NPs in headings, lists, etc. in real texts most frequently occur without punctuation) and items with full stops.

On the other hand, we also found examples of erroneous corrections like :

**mon quelques entreprise* -> corrected as: *ma quelque entreprise*
**le quelque entreprise* -> corrected as: *la quelque entreprise*

which remained the same, both when the items were given to the system followed by a full stop and embedded in a sentence. This thus allowed us to detect a systematic deficiency in the system's treatment of quantifiers as part of complex determiners (where they are analysed as ordinary adjectives).

Another set of data seemed to indicate problems with complementation of nouns designating quantities :

**une foule de personne.* -> corrected as*: no error detected*
**tant de personne* -> corrected as*: no error detected*
**un grand nombre de personne* -> corrected as*: no error detected*

However, the two examples below are corrected as expected :

**un bouquet de fleur* -> corrected as*: un bouquet de fleurs.*
**une quantité de livre* -> corrected as: *une quantité de livres*

If we consider the results of the first three test items, we might be tempted to conclude that the LC101 has no knowledge of the countable/uncountable distinction, but this supposition is invalidated by the results of the last two test items.

Inspection of the analysis reveals that the problem resides in the use of „*personne*" which is categorially ambiguous between noun and pronoun.

c. NP and Clause Complementation

This third series of examples aims at illustrating the system's deficiency on clause complementation. When using the French TIs on this phenomenon, we had (relatively often) discovered ambiguities that turned the ill-formed TIs in potential well-formed TIs (in particular because of the PP which can be both complement or modifier of the verb) and had therefore ended up with very few ill-formed TIs for clause complementation.
Some of these (see appendix 1) showed that, in certain cases, LC101 can deal with this type of ill-formedness. The following examples however showed that it is not always the case.

**L'entreprise suppose de l'ingénieur.* -> no error detected
**L'entreprise regrette partir.* -> no error detected
**L'entreprise suppose si l'ingénieur vient.* -> no error detected

The test items for this phenomenon thus allowed us to state that the system does use the subcategorisation information given in the dictionary entries in its analysis but that the syntactic variations allowed for the „object" of transitive verbs are too general.

d. Subject-verb agreement (ill-formed TIs with a verb in the infinitive instead of a conjugated verb)
This last example shows a case where we concluded with both a hypothesis on the system's deficiency and a question on the test data used.

Many TIs on subject verb agreement showed that LC101 can deal with this phenomenon. But we were a little confused when testing the following TIs.

Example 1:
Il neiger. -> corrected as*: *Il neigé.*
LC101 explanation: *„Le verbe aurait du être au participe passé."*
In such a case, we did not understand the diagnosis (*the verb should have been a past participle*) as the correction proposed was not a well-formed phrase.

Example 2:
Il travailler. -> corrected as*: Il travailla*
LC101 explanation: *„Le verbe aurait du être au participe passé."*
At this stage, we wondered why the diagnosis was not at all the same as in example 1 and we wondered why „*travailler*" was corrected as „*travailla*" (i.e. simple past) rather than as the present „*travaille*" (possible correction that was graphically near to „*travailler*") or the imperfect „*travaillait*" (phonetically near to „*travailler*").

Example 3:
Vous travailler. -> no error detected
Now, we wondered why LC101 did not behave as it did in example 2.

It is arguable that the number of possible corrections for these test items amounts to all finite verb forms which could agree with the subject and that, consequently, the test items are ill-defined.

It should probably not be counted as a deficiency of a grammar checker if it does not suggest all possible corrections in this case but, as the examples reveal, the system exhibits an inconsistent and therefore unreliable behaviour, ranging over :
- wrong diagnosis and erroneous suggestion for correction ;
- wrong diagnosis but possibly correct suggestion ;
- acceptance of ill-formed structure.

The set of test items for subject-verb agreement did, thus, allow the tester to state that the phenomenon is mastered when the verb is inflected for person and number but that system behaviour is inconsistent if the ill-formedness of a test item consists in using an infinitive instead of a finite verb form.

In conclusion, the TSNLP French data tested seems to allow us to detect some of the system's deficiencies, although a syntactic test suite by its very nature does not allow us to measure or estimate precisely the extent of the detected deficiencies where these are due to peculiarities of lexical codings.

The analysis of the results obtained are described below :

TSNLP data		LC101 results		
Well-formed test items		items corrected (√)	items considered correct (•)[6]	items not corrected (? or *)
C_Comp :	80 TIs		100 % (80 TIs)	
C_TAM :	73 TIs		100 % (73 TIs)	
NP_Comp :	6 TIs		100 % (6 TIs)	
NP_WO :	29 TIs		100 % (29 TIs)	
NP_Agr :	138 TIs	7 % (10 TIs)	93 % (128 TIs)	
AP_CO :	9 TIs		77 % (7 TIs)	23 % (2 TIs)
C_Agr :	60 TIs	5 % (3 TIs)	93 % (56 TIs)	2 % (1 TI)

[6]Types of system responses expected (i.e. „items considered correct „ for well-formed TSNLP TIs and „items corrected" for ill-formed TSNLP TIs, even if the correction proposed by the system does not always correspond to the corresponding well-formed TI defined by TSNLP).

TSNLP data		LC101 results		
Ill-formed test items		items corrected (√)[6]	items considered correct (•)	items not corrected (? or *)
C_Agr :	103 TIs	99 % (102 TIs)		1 % (1 TI)
NP_Agr :	539 TIs	92 % (498 TIs)	7 % (37 TIs)	1 % (4 TIs)
C_TAM :	275 TIs	35 % (97 TIs)	9 % (24 TIs)	56 % (154 TIs)
NP_Comp :	14 TIs	14 % (2 TIs)	57 % (8 TIs)	29 % (4 TIs)
C_Comp :	253 TIs	7 % (18 TIs)	47 % (118 TIs)	46 % (117 TIs)
NP_WO :	31 TIs	6 % (2 TIs)	17 % (5 TIs)	77 % (24 TIs)
AP_CO :	23 TIs		48 % (11 TIs)	52 % (12 TIs)

3.3 Inadequacies of the test data discovered during testing

When applying French test items to the LC101, cases occurred in which the diagnoses produced by the system were relevant though different from the description provided by TSNLP. Such cases :

- allowed us to improve the WP3 data and suggest amendments to the Test Suite construction methodology given in WP2 (see sub-section. 3.3.1 below),
- led to some grammar checker dedicated customisation of the French test suite (see section 2 on customisation),
- showed the problem of component testing inside systems (see sub-section 3.3.2 below)
- raised questions on the TSNLP strategy to derive ill-formed test items (see section 3.4 below)

3.3.1. The testing allowed us to improve the WP3 data and suggest amendments to the Test Suite construction methodology given in WP2.

Not only were some typing and encoding mistakes found out, but the ambiguity of many test items was revealed by the system: many test items that were encoded as ill-formed test items during WP3 turned out to be grammatically correct constructions (see appendix 3 : *Feedback for the WP#3: data construction*). This situation can come from the fact that a test item writer, concentrating on a specific phenomenon and on systematic derivation of ill-formed variants, „forgets“ possible ambiguities of this item.

Example : **L'entreprise regrette*.
is an ill-formed TI for C_complementation (missing direct object). But the TI turns inadequate, because *regretter* has also an intransitive use.

First replacement tried out : **L'entreprise célèbre*.
The TI should illustrate the same ill-formedness. Note that we have carefully chosen a verb that has no intransitive use. But the TI turns inadequate again, because *célèbre* can be both a verb and an adjective, so that the intended ill-formed sentence can be a well-formed NP.

Second replacement tried out : **L'entreprise fonde*.
This TI can be considered as non-ambiguous.

Ambiguities in the TIs can also be a consequence of the use of the well-formedness value 2 to mean both „questionable grammaticality“ and „ill-formed test item with respect to that phenomenon only“.
This policy has now changed :

- the value 2 now refers to „dubious grammaticality“ ;
- a new value 3 has been introduced to refer to „grammatical in another context“.

We try to present the following examples, in a first (and non exhaustive) classification, according to the reason for their inadequacy:

a. inadequacy due to morphological ambiguity

In French, for example, such an ambiguity often occurs between:

- the 1rst and 3rd persons of a verb of the first group (i.e. ending in -er) conjugated in the present tense and in the indicative mode,
- the 1rst person of a verb of the first group conjugated in the present tense and in the indicative mode and the 2nd person of the same verb in the present tense and imperative mode.
- the accusative and the dative of a pronoun.

Examples:

**Accepte l'invitation.*

is an ill-formed test item for phenomenon clause complementation (missing subject in an active declarative sentence) but a correct imperative sentence.

** Elle lui vient.*

is an ill-formed test item in phenomenon clause complementation (if "lui" is meant as an accusative pronoun) but it is a correct test item as "lui" is also a dative pronoun.

** Je suis travaillé.*

is an ill-formed item in phenomenon Clause Tense-Aspect-Modality (with the corresponding well-formed item *J'ai travaillé.*) but it is a correct passive sentence, as *travailler* also has a transitive acceptance.

b. inadequacy due to ambiguity in the morpho-syntactic category

Many French words belong to several morpho-syntactic categories (as taggers and parsers know). The combination of such ambiguities in a TI can lead to ambiguity of TI category (for example, the same TI can be an NP as well as a sentence).

Examples:

* *cordial de l'ingénieur*

is an ill-formed item in phenomenon AP complementation (with the corresponding well-formed item *cordial avec l'ingénieur)* but it is a correct NP, as *cordial* can also be a noun.

* *La belle livre.*

is an ill-formed NP with regard to det_N_adj agreement (with *belle* as adjective and *livre* as noun) but it could also be an ill-formed sentence with regard to C complementation (with *belle* as noun and *livre* as verb, with a missing direct object).

Because of this type of ambiguity, the use of certain types of TSNLP ill-formed items turned out to be critical.

Examples:

**La cause la rupture .*

-> corresponding correct NP in TSNLP: *La cause de la rupture.*

-> corresponding correct Clause for LC101: *Là cause la rupture.*

In TSNLP an ill-formed item with phrase type X is associated with a well-formed item with phrase type X. But the LC101 does not / cannot take this into account (and we have no way to provide this information to the system). Hence when the system

tries to correct an ill-formed item, phrase type X by an ill-formed item, phrase type Y, the test item used is inadequate (for the system tested), because it does not allow us to test what we wanted to test. To avoid this „phrase type switching“ as much as possible, the only solution that we have found is to make sentences out of all the test items (see section on customisation).

Note that, in a sense, all this is but a re-expression of the need for output specifications in the case of ill-formed test items.

c. inadequacy due to syntactic or functional ambiguity

In this category, we would in particular mention the ambiguity between PP as modifiers and PP as complements or between conditional clauses and other subordinate clauses introduced by *si*. We have found this ambiguity extremely embarrassing when trying to build ill-formed TIs for complementation: not only should we choose a word with a restrictive complementation frame but the intended „wrong“ complements will mainly be potential (correct) modifiers.

Examples:

* *L'ingénieur rêve par l'entreprise.*

is an ill-formed test item for phenomenon clause complementation (as *rêver* does not accept a p-obj(par), but a well-formed test item (though semantically strange) if „*par l'entreprise*.“ is considered as a modifier of the verb.

* *L'ingénieur rêve si la femme vient.*

is an ill-formed test item in phenomenon clause complementation but it is a well-formed test item as „*si la femme vient* „ can also be a conditional clause.

d. inadequacy due to lexical semantic ambiguity

This is the most evident type of ambiguity (at least a posteriori).

Examples:

* *La belle livre.*

is an ill-formed test item if *livre* is a masculine noun meaning *book* but a correct NP if livre is a feminine noun meaning *pound*.

**L'entreprise suppose que l'ingénieur vienne*

is an ill-formed sentence with regard to complementation but a well-formed sentence, as *entreprise* also means *undertaking* and *supposer* also mean *to presuppose*.

e. other types of inadequacies

Example:

**Beaucoup de vins.*

is an ill-formed partitive construction but a correct test item otherwise. We wonder if, in French, any TI can unambiguously illustrate the ill-formedness that this TI is to illustrate.

Two types of corrective action are needed for these types of inadequacy :

- replacement in both well-formed and ill-formed test items of ambiguous words with unambiguous ones which will render the derived ill-formed items genuinely ill-formed ;
- omission of ill-formed items derived by certain operations if these operations will

always result in grammatically correct constructions representing a different phenomenon (e.g. deletion of the subject of transitive verbs will in most cases result in a well-formed imperative). Where possible, construction of new test sets where the derivation operation in question will result in a genuinely ill-formed item.

Moreover, it should be added to the general guidelines for test suite construction (WP2.1) and to the discussion of the annotation scheme and its possible values (WP2.2) that, at least for the purpose of grammar checker testing, the use of test items with the new well-formedness value = 3 seems of doubtful interest.

3.3.2. The testing shows the problem of component testing inside systems

Testing the TS as a purely syntactic TS is difficult, because we have no means to dissociate the grammar checking functions and the spelling checking functions in LC101.

Examples:

**La vient*
-> corresponding well-formed test item in TSNLP: *Elle vient*.
-> correction proposed by LC101: *La vie*.

**Je travailler*.
-> corresponding well-formed test item in TSNLP: *Je travaille*.
-> corrections proposed by LC101: J*e travaillai*. AND *Je travaille*.

As already discussed above, since potentially all 1st person verb forms could be correct here, as shown in example 2, the system seemingly switches to the rules for spelling correction in order to propose the most likely corrections.

In its corrections, LC101 not only uses syntactic rules and morphological verification but also uses:
- phonetic rules (trying to replace a word A by a word B the pronunciation of which is similar to A's. Ex.: port(harbour)/pore(pore)/porc(pig, pork),
- graphical rules (trying to replace a word A by a word B the writing form of which is similar to A's),
- both phonetic and graphical rules (trying to replace a word A by a word B which is a paronym A. Ex.: conjoncture(conjuncture, situation)/ conjecture(conjecture)),
- knowledge on word spelling (LC101 is both a spelling checker and a grammar checker),
- knowledge on word ambiguity (ambiguity in the grammatical category).

Because the use of these not syntactic rules interferes with syntax, LC101 corrections are often not the same as TSNLP well-formed items[7] . To correct **La vient*. by *La vie*. For example, rules on homography (or typing mistakes) are probably used, while rules on homophony would be used to correct **Je travailler*. by *Je travaillai*.

In these cases, it is difficult to make a judgement on the pertinence or inadequacy of the TSNLP test items for testing grammar checkers. We could only add test items dedicated to grammar checkers allowing the testing of the syntactic correction only.
When testing a specific grammar checker (it is of course always possible, in cases like the „*la vient*" example, to try to construct customised test items for the phenomenon in question which avoid interference from the spelling checker). We do, however, doubt that the writing of such test items, i.e. items made out of words that

[7] All the more as TSNLP did not manage to take homography or homophony into account when choosing the TS vocabulary.

are graphically and phonetically similar to none, is generally feasible.

Testing, so it seems, allows us find more adequate material for the testing (and this makes the building of a Test Suite a long iterative process).

3.4 Relevance of the error types illustrated in the test data

When we laid down the principles of building ill-formed test items in WP2 and defined the relevant operations (deletion, permutation, etc.) to derive ill-formed items from well-formed items, our aim was to be as systematic and exhaustive as possible. We thus adopted the strategy of artificial data construction for well-formed data as well as for ill-formed data, in order to ensure systematicity.
However, it appears that, when applied and tested, some ill-formed test items, built systematically along the principles defined previously, are so unlikely to occur in a real life situation that grammar checker developers, who clearly foresee other types of errors than TSNLP, have not foreseen these and not provided appropriate responses to them.

For example, it seems that the use of deletion to form ill-formed item leads to sentences and phrases for which LC101 cannot propose any correction (deletion does not seem an „operation“ considered by LC101 in priority: the message „missing word ?“ is extremely rare in LC101 diagnostics).

In D-WP2.1 (see [*Balkan, Meijer, Falkedal et al., 1994*]), we actually discussed different approaches to building ill-formed test items. Systematic constraint violation was chosen for several reasons :

- it was relevant to most types of NLP systems ;
- it was feasible since it did not require additional research and empirical investigations into real life errors of the types discussed in D-WP2.1 ;
- it was hypothesised that systematic constraint violation might turn out to be an economic way of revealing the same system aspects as those which real life data could capture.

Another option consists in collecting ill-formed test data extracted from corpora and trying to build typologies of errors (these typologies are likely to be specific to classes of writers/authors, such as people writing in a non native language, children, etc.) or to derive rules to create ill-formed data, etc.

It is a topic for future research to investigate the potentially higher relevance and adequacy for the testing of grammar checkers of such error types by, for example, devising a grammar of mistakes (such grammars have already been developed by second language teachers and could be inferred from grammar books and corpora studies). Results on this topic are likely to be interesting for TSNLP.

It is equally a topic for future research to determine whether the so-called medium related errors, i.e. errors introduced by various kinds of computational text manipulation, can be classified in a way which will render them useful for test suite testing, and whether this type of data would prove more adequate as test data for grammar checkers than artificial data obtained by systematic violations of grammatical constraints.

As expected, the TSNLP French test items allow us to identify some of the system‘s weaknesses and, in some cases, to give a detailed description of such deficiencies.

We also experienced the usefulness of several annotation types:

- the annotations on phenomena and domains (provided in the TI window) allow us to structure the testing and suggest some sort of protocol for the use of the data

(i.e. we successively used TIs on NP_complementation, C_agreement, ...). Together with the TI sets, they helped us to test the systematicity of TIs inside subsets of the TS (though the systematicity of the TS as a whole cannot be tested, since the TS is still being built),
- the usefulness of the annotation WF (well-formedness) is evident,
- the comment field has proved especially useful for the ill-formed TIs, as it helped us discovering possible differences between the TSNLP analysis and the system‘s diagnostic.

Furthermore, the WP4 testing (together with the WP3.2 experiments) suggests possible values for annotations that presently systematically have a default value (since the existing TIs are core data).

WP4 thus confirmed the basic adequacy of the French TSNLP data for system testing and also occasioned a number of corrections and improvements to the initial test items produced in WP3.1.

Section 4: Testing and Customising the English test items

This section describes the customisation and testing of TSNLP data for SECC. Some details are confidential and are reported at greater length in *[Balkan, Fouvry, and Kordoni, 1995]* and [*Macken, 1995*]

4.1. The Customisation of test suites for SECC

The customisation of test suites for SECC was carried out by Essex and included the following tasks :

4.1.1. Tuning general purpose data

A subset of the core data was customised for SECC. The following operations were carried out :

1. Lexical replacement

Originally, we had intended to exploit the Lexical Replacement Tool which we devised in WP5 (see [*Arnold, Rondell and Fouvry*, *1994*]) in the customisation of test items for SECC in WP4, but the unavailability of an on-line dictionary from SECC made this impossible. We did, however, use the Lexical Replacement tool in the construction of core data for WP3.

For the core data, we have a lexicon containing categorial information (see appendix 2) which was devised as an aid to data construction, independent of the lexical replacement task, and obviously could be enhanced to make it more useful for lexical replacement (by, for example, having more semantic information). As pointed out previously in WP5.1, the lexical replacement tool, which in its current version is only meant to work for simple replacement with hardly any knowledge of the structure of a record, would also need to be enhanced to make it truly useful.

2. Adapting test items to the input required by the system

This included writing items of the category level required by the system.

3. Creating output specifications

This was a very important aspect of the customisation phase, and a number of meetings were arranged to establish the exact output specifications required by SECC.

For already existing positive data, some extra ill-formed data had to be created, based on the violation of constraints outside those tested for in the core ill-formed data.

4.1.2. Creating new test items

New test items needed to be created for the controlled language checker SECC since many rules are based on particular lexical items, and in the general purpose test suite a small, restricted vocabulary was used. Controlled language rules prohibit all uses of a word in some cases, or restrict its use to a particular context in others, or in still other cases, allow a word to be used only with a certain meaning. An example of the last includes the following rule from SECC (example from Adriaens):

Use only "*because*", never "*since*" in a subclause of reason:

-- Since a DBCS manages the System 12 database, physical storage is transparent to the users

+ Because a DBCS manages the System 12 database, physical storage is transparent to the users[8]

The new test items were created using the TSNLP guidelines stated in WP3.1 [*Estival , Falkedal, and Lehmann*, *1995*] and repeated here for convenience :

- general
  - no ambiguities
  - no modifiers and adjuncts
  - unmarked word order
- sentence level
  - declarative sentences
- verb form
  - 3rd person singular
  - present tense
  - indicative
  - active
  - unambiguous verbal morphology
  - no auxiliaries / modals
- NPs
  - explicit case marking where appropriate or possible
  - fixed definite determiners (no demonstratives, possessives etc. unless necessary) ; alternatively (simple) pronouns where possible
- PP
  - no complex prepositions

Thus this data construction was also provided the opportunity to test the TSNLP methodology for test suite construction.

Some of the problems we encountered in test suite customisation were the following:

1. Incomplete ti-analysis in expanded test items

When sub-sentential test items were expanded to full sentences for the purposes of testing, their ti-analysis is underspecified. They thus need to be distinguished from test items which have a full ti-analysis.

2. Output specifications

It was not always easy to predict output specifications, particularly corrections. This was due to underspecification and or incompleteness in the grammar.

3. Relating system-specific linguistic analysis and classifications to TSNLP analysis and classification.

Although we chose as neutral linguistic descriptions of our data as possible, work may still be required to map them onto system specific descriptions. Likewise, as described in D-WP2.1, the naming conventions (of, for example, subcategorisation frames) used by system developers may not coincide with those used by TSNLP, but it is hoped that the extensive documentation in TSNLP will help the test suite user relate TSNLP data to his system.

[8]SECC also has rules referring to formatting, not all of which can be represented in the TSCT (the TSNLP input tool) format. One solution would be to adopt LATeX or SGML conventions for representing lists, for example. Text-level rules would also pose a problem for TSCT (at the level of ti-analysis, for example), but we were not required to write any for SECC, and they are anyway outside the scope of TSNLP.

In general the customisation of core data was quite time consuming and less trivial than might be imagined due mainly to the vagueness and incompleteness of the CL grammar.

4.1.3. Integrating customised data into the database

Since customised data for SECC was created prior to the implementation of the database, the format for output annotations had to be made compatible with TSCT format. (No output window had been created for the core data, since output annotations are system specific).

Having agreed on the output annotations for SECC, we created a template for these annotations. We used the forms-mode from *Emacs*, that allows one to write forms as templates to fill in values next to attribute names. This information is written to files with record and field separators, as in TSCT. This tool has one restriction: it can only be used for records with a fixed number of fields, but this was not a problem here: there are no slots that can be repeated.

Where output annotations were being written for existing core data, an *awk* script was used to fill in some of the slots in the template automatically from the items file (for example, the input string and id-number). This provided the link with the other files in the test suite.

While the input string slot was first inserted for pragmatic reasons, it became clear that it could be used to increase the amount of useful information. The TSCT tool has certain requirements with respect to punctuation. Not all applications will adopt these requirements, only because they do not comply with the TSCT punctuation rules. So an edited version of the input string in the items file can be used as the value of the input slot in the output window.

An output slot (possibly in some markup language) can be provided to make automatic comparison of system output possible.

4.2. Results of testing

Testing of the data was carried out by SECC and is reported at length in [*Macken, 1995*]. Only one testing cycle was performed, although more would be required to fine-tune the data.

The TSNLP data and methodology were validated by the testing phase in the following ways :
First, the construction of the data following the TSNLP guidelines helped to avoid the introduction of unnecessary complexity into the data.
Secondly, the use of the construction tools helped ease of construction and consistency of data.
Thirdly, the data is annotated in such a way as to allow easy classification and retrieval of phenomena.
Fourthly, the annotation scheme proved sufficiently flexible to enable user specific information to be accommodated (such as output specifications).
Finally, and most importantly, the TSNLP test data proved very valuable in highlighting deficiencies in the system performance, as well as in the rule descriptions and in giving pointers to the possible source of those errors. Details of these performance errors and rule description errors can be found in [*Macken, 1995*].

Section 5: Customising and Testing the German test items

5.1 A Glass Box: Connecting TSNLP to an HPSG System

To validate the TSNLP annotation schema and German test data using a constraint-based parser, DFKI deployed the HPSG system developed at DFKI[9] for German.

Whereas for the grammar and controlled language checkers there was no or only limited access to internal data structures or the control flow within the application (i.e. the application as a whole seen from the outside is a *black box*), the evaluation situation for the DFKI HPSG parser for German was different:: since the test suite and the parser have been developed by the same group, there is full access to the internal structures of individual modules as well as to the flow of control. Testing TSNLP results in this setup amounted to an ideal *glass box* evaluation.

In order to take full advantage of the favourable evaluation situation and the full depth of TSNLP annotations, the work plan was to:

- identify and tailor relevant test data (including lexical replacement if necessary);
- investigate the bidirectional interface to the TSNLP database allowing the application to retrieve *and* store data (through Common-Lisp foreign function interface and the $tsdb_1$ interface library);
- at the application side: establish a mapping between structure derived from the grammar and the TSNLP morpho-syntactic and relational annotations;
- alternatively or supplementarily: add output specifications (the number of readings, phrase structural or semantic representations et al.) to a DFKI *user & application profile* ;
- run automated *retrieve, process, and compare* cycles in grammar diagnosis and progress evaluation; and to
- employ the TSNLP database to store additional application-specific data (e.g. performance measures).

Through the bidirectional connection to the TSNLP database the DFKI HPSG system presumably is one of the few grammar development platforms that is now seamlessly integrated with a general-purpose, multi-lingual test suite.[10]

---

[9]The DFKI HPSG system has continuously been developed over more than five years by the DFKI Computational Linguistics group. The current development version combines

- a powerful state-of-the-art grammar development system including a rich typed feature formalism and unifier, a parameterizable parser and generator, a command shell, and several development tools;
- linguistic competence for German (morphology, syntax, and semantics);
- new methods for linguistic performance modelling; and
- an interesting sample application for appointment and calendar management.

The system has been delivered to several international research institutions (including CSLI Stanford, Brandeis and Simon Fraser Universities, and German sites taking part in the international VerbMobil machine translation project); see [*Uszkoreit et al., 1994*] for more details.

[10]Although in the Hewlett-Packard NL system already there was a close connection to the Hewlett-Packard test suite (see [*Flickinger et al., 1987*]), the link was unidirectional rather than bidirectional: since the HP test suite (similar to most first generation test suites) is organized as a set of ASCII files, the NL system would retrieve a set of test items, process them in a batch job and compare the results to the (shallow) annotations available.

The close coupling to the TSNLP database has enriched the DFKI HPSG system with a validated and comparable evaluation facility and provides for highly valuable coverage and performance measures during the ongoing grammar development for (currently) German and English.[11] Additionally, the mapping of TSNLP categories and annotations at lexical and phrasal levels (see section 3 below) into HPSG feature structures as they are derived by the DFKI grammar allows for the validation of the type and depth of annotations chosen.

5.2 Diagnostic Results Obtained

For the diagnosis of the DFKI HPSG system and especially the validation of the TSNLP test suites as a useful tool in continuous progress evaluation (i.e. the automated comparison of actual system performance - e.g. in terms of coverage and efficiency - after changing the software or grammar to the expected or former behaviour) the German test data had to be customised with respect to the application. However, other than in the customisation experience with SECC (see section 4) no data had to be replaced or removed in order to meet the requirements of the application. Instead, customising the German test suite for the DFKI HPSG system resulted in a monotonic extension of the data following the *user & application profile* approach: everything relevant to the DFKI application is stored into its designated place in the database (viz. two specific and separate relations) such that the generality of the German test suite is not affected (see figure 1 for details).

i-id	system	version	readings	first	total	user	date	error
11001	paradice	4711	1	0.9	2.0	oe	23-sep-1995	
11006	paradice	4711	2	3.6	8.8	oe	23-sep-1995	
11178	paradice	4711	-1			oe	23-sep-1995	bus error

i-id	nll (normalised meaning representation)
11001	direct(theme:^(∃ ?x ?y addressee(instance:?x) work(instance:?y agent:?x)))
11006	direct(theme:^(∃ ?x ?y addressee(instance:?x) work(instance:?y agent:?x)))
11006	ask(theme:^(∃ ?x ?y addressee(instance:?x) work(instance:?y agent:?x)))

Figure 1 : Information added in customising the German test suite for testing the DFKI HPSG system: the DFKI *user & application profile* records processing measures (e.g. the grammar version and date of testing, the number of readings obtained, the processing times for finding the first reading and overall total, and the type of processing error where applicable) for each of the test items; rather than deploying bracketed phrase structure trees as an output specification, a normalised semantic representation serves as a reference point to judge the quality of analysis and to identify spurious ambiguities.

[11] Since the TSNLP test suites are now integrated into a well-established grammar development environment, it is expected that the further distribution of the DFKI HPSG system will serve as a catalyst for the dissemination of TSNLP results.
Additionally, Cray Systems has already indicated a strong interest in integrating TSNLP into the ALEP (Advanced Linguistic Engineering Platform) framework such that the test suites could ultimately become a widely accepted diagnostic and reference tool for multi-lingual grammar development.

Using the customised German test suite and the test data available at the beginning of the testing phase (3674 test items for 9 phenomena) three complete test cycles (each of more than two full days in processing time[12]) were carried out in order to:

(1) debug the interface and database technology allowing the elimination of TSNLP software bugs;

(2) determine the system coverage and precision and to eliminate data flaws resulting in the identification of systematic failure sources (lexical gaps, rule application, overgeneration, software bugs et al.) in the application;

(3) compute reference data for future diagnosis (including manual inspection and correction of NLL output specifications).

5.2.1. Feedback for the Test Suite

The overall feedback from testing the German test suite with the DFKI HPSG system is very positive: both, the test data and the database technology have proven highly adequate and sufficiently flexible to provide for a very valuable diagnostic tool in the (progress) evaluation of a constraint-based parser. The test suite, annotations, and software could be taken unchanged as they were developed by TSNLP yet allowing the smooth and seamless integration into a complex grammar development system.

Besides, comparing the German test data to the analyses derived by the DFKI HPSG grammar revealed some encoding mistakes during the data construction, concerning above all unexpected and unintended ambiguities detected by the system as well as other inconsistencies. Given that the substantial test data construction (more than 5000 test items as of October 1995) was carried out in only about half a year and by several test data authors in parallel, it was to be expected that human failures had introduced misclassifications, inconsistencies, and even typing and spelling errors that were not detected during the database import and consistency checking. Accordingly, processing the data using a constraint-based grammar and inspecting the cases were the expected and actual system output do not match (see section 2.2 below for details) allowed for the identification and correction of several erroneous grammaticality and category values in the German test suite. However, even without this additional validation and feedback the test data already turned out to be of very high quality.

5.2.2. Feedback for the Application

Although the testing of TSNLP in conjunction with the DFKI HPSG system had its focus on the validation of the TSNLP methodology, test data, and technology, naturally, diagnostic results for the application were obtained at the same time. Since some of the feedback for the DFKI HPSG system could straightforwardly be generalised and applied to similar grammar development systems, it is reported briefly.

Fundamental in both judging the current status of the application as well as in using the test suite in continuous progress evaluation while the system evolves is the classification of *output mismatches* , i.e. test items for which the expected performance of the application is different from the actual result obtained.

[12]In order to obtain comparable efficiency measures, the system is run in development mode and reset to its initial state for each of the test items; hence, even though the actual parsing time totals less than two hours, the batch processing mode adds a substantial time overhead.

Using the information stored in the DFKI *user & application profile* and the figures obtained in test cycles two and three (see section 2 above), the database retrieval engine (see [*Oepen et al., 1995*]) can be used to extract and count the following classes of output mismatches:

mismatch of expected vs. actual output	**2nd cycle**	**3rd cycle**
wellformed items without analysis	868	751
illformed items with analysis	235	111
spurious ambiguities (≥ 3 readings)	27	19
fatal system errors	7	0
total reduction of output mismatches	**23 %**	

While the first class of output mismatches (grammatical test items that could not be analysed by the DFKI HPSG system) is mostly determined by lexical gaps in the grammar[13] and restrictions on the treatment of coordination, the second class (supposedly ill-formed test items that nevertheless have one or more analyses) predominantly indicates overgeneration in the application and allowed the grammar writer to identify two systematic sources of failure (viz. spurious rule application and lexical misclassification). However, most of the remaining 111 mismatches in this class reveal that ill-formed-looking test items like

(1) **das [alter Männer]*$_{gen}$
?the [one of] old men

(2) **keines [alter Projekte]*$_{gen}$
?none [of] old projects

actually *can* be analysed as being grammatical under the assumption that German allows elliptical noun phrases (formed by a determiner plus a phonologically empty head noun) to combine with genitive post-modifiers. However, since examples like (1) and (2) would usually require some larger context to be acceptable, the grammaticality value in the TSNLP test suite was not changed, while it is recorded in the DFKI *user & application profile* that the system nevertheless is not to be blamed for accepting these items.

5. 3 Mapping TSNLP Categories into HPSG

In order to automatically match the TSNLP annotations with the HPSG analyses produced by the DFKI parser, it becomes necessary to map the HPSG typed feature structures onto the atomic category names used in the TSNLP database and to compute appropriate dependency relations from a derivation tree (a parsing result).

Without going into the details of HPSG (see [*Pollard and Sag, 1994*]), the following simplified examples are meant to illustrate the method and technology used to implement a suitable mapping procedure that can be applied to the result structures of the DFKI parser (i.e. to HPSG typed feature structures in general). The approach taken borrows from current compiler technology (see [*Nerbonne et al., 1993*] for an application of compiler technology to NLP) and general term rewrite systems: it fundamentally builds on the typed feature structure rewrite engine zebra[14] ([*Konrad, 1995*]) developed at DFKI that allows to

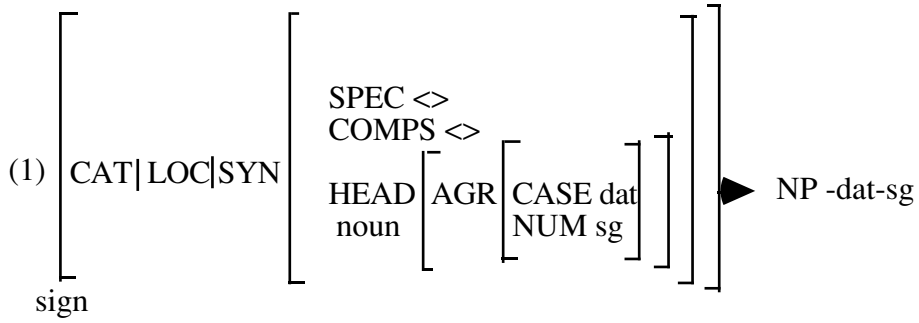
[13]Since the testing for different complementation patterns (e.g. verbal complementation in the *C_Complementation* phenomenon) is partly a matter of lexical rather than of syntactic overage, obviously further test suite or grammar customization would be required.

[14]The zebra system in turn was inspired by --- and in a sense can be seen as an extension of --- the work by Joachim Laubsch and John Nerbonne (of Hewlett-Packard Laboratories and DFKI) on zebu and NLL, a parser generator and meaning representation language respectively.

compile rewrite rules over typed features structures into Common-Lisp procedures. Individual or sets of compiled procedures are then applied recursively using a parameterisable tree traversal driver.

Through the partitioning of rewrite rules into disjoint sets that are applied sequentially, it is possible to process heavily nested feature structures inside-out; for example for the mapping into TSNLP annotations it seems plausible (i) to rewrite embedded categories (i.e. lexical and phrasal nodes) first, then (ii) to deploy another set of rewrite rules to collect the results from (i), and (iii) finally to compute the dependency relations in a third stage.

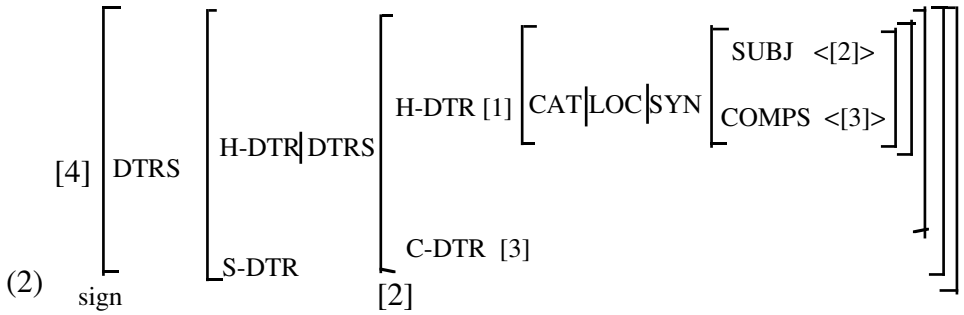
Example (1) is a simplified version of the rule that rewrites the HPSG representation of a (dative singular) saturated noun phrase into the TSNLP equivalent.[15] The context for an application of rule (1) (i.e. the pattern on the left hand side of (1)) is strictly local: it is restricted to the token phrase being rewritten.



Because the HPSG account of subcategorisation builds on list cancellation (i.e. in a fully saturated verbal projection SUBJ and COMPS are empty lists), the rules to compute dependency relations typically require a larger context. In order to identify the subject and object(s) of a verbal head (a functor in the TSNLP sense), the entire phrase structure tree headed by the verb (the domain of the functor) has to be taken into account: even though the grammatical functions can be determined from the valency features of the verbal head (structure [1] in example(2)), the remainder of the derivation tree (represented in (2) as a HPSG DTRS structure) contains the information about the actual realisation of the subcategorised complements and the corresponding string positions.[16]

[15] Although in the rewrite rule examples for expository reasons type names are omitted where they are not strongly required to constrain the context, in the zebra system each of the embedded feature structures is assumed to bear an appropriate type. Because the types used in the specification of rewrite rules can be taken from exactly the same hierarchy that constitutes the parsing grammar itself, the set of TSNLP rewrite rules can be interpreted as a proper and declarative extension to the DFKI HPSG grammar.

[16] In example (2) the notation ‚[1]' is used to indicate the position of the substring corresponding to the feature structure [1]. String positions in HPSG can in general be read of the PHON feature (a glossed list-valued representation); however, since the DFKI system incorporates a chart-based parsing algorithm, a more straightforward option is to extract the position values immediately from the chart.



a-position	a-category	a-function	a-domain
'[1]'	[1]	func	'[4]'
'[2]'	[2]	subj	'[1]'
'[3]'	[3]	obj	'[1]'

For similar types of morpho-syntactic categories and other dependency relations involving HPSG selector features (e.g.MOD and SLASH) analogous rules are deployed to compute the TSNLP correspondences.

As of October 1995 the categorial mapping for root categories of TSNLP test items (i.e. the *category* field in the *item* relation) has been established and used to automatically compare the results obtained from the DFKI HPSG system with the annotations found in the TSNLP test suite. Analogously to the grammaticality results (see section 2), the categorial information in several cases has proven to be highly valuable in testing the DFKI application. Regarding feedback for the test suite, the mapping of HPSG feature structures into TSNLP categories made it possible to detect two systematic error sources in the data: firstly, it greatly helped in spotting incorrect or inconsistent categorial annotations (e.g. two different test data authors had used slightly distinct categories for the same lexical item and type of construction); secondly, it provided the basis for the identification of categorial ambiguities that test data authors had not been aware of (e.g. the use of the form *arbeiten* (work) as both a verb and a noun).

However, unless the similar exercise is completed for TSNLP functional annotations, it remains difficult to judge to what extent an exhaustive mapping onto the TSNLP annotations can be established and inhowfar the granularity of test data and level of annotations suffice for the thorough and automated matching against a highly formalised theory of grammar such as HPSG.

Section 6: Conclusion

The experiences of testing the adequacy of TSNLP test data described in the present report show that the methods used and the types of results obtained largely depend not only on the language tested but also on the material and information access conditions under which they have been performed.
However, the general and common conclusion we have all acknowledged is that, in most cases, the subset of the TSNLP test items tested in WP4 proved to be adequate and were actually representative of the phenomena they were intended to illustrate.[17]

For the testers, and especially for actual and potential future industrial users, the TSNLP Test Suite turns out to be a well adapted linguistic tool to detect and characterise NLP system linguistic capacities and deficiencies.

The TSNLP Test Suite annotations on phenomena and domains are relatively easy to understand for non linguist users, and make the Test Suite relatively easy to manipulate.

Furthermore, WP4 has also been the place for some feedback to WP3 : the testing of (a subset of) the French Test Suite has enabled us to provide a list (see appendix 3) of inadequate or miscoded test items, some of which would have been difficult to detect manually.
It has also highlighted some general problems with the data. One concerned categorial ambiguity of some of the lexical items in the test suite vocabulary. These have been replaced by categorially unambiguous lexical items. The second problem concerned the confusing use of the value 2 as the grammaticality indicator, both for test items of dubious grammaticality and for test items which are ill-formed with respect to one phenomenon but well-formed examples of other phenomena (so called relative ill-formed test items).
Hence, an output of the WP4 is also an improved TSNLP Test Suite.

Finally, WP4 also results in some feedback to WP2 : our customisation and testing experiences in this work package allowed us to add the following issues to our guidelines:

1. the need for close collaboration between the test suite writer and user
2. the need for several test runs when tuning the data
3. the need for adequate dictionary for automatic lexical replacement
4. the need for corpus tools to obtain user profile information (weightings of phenomena with respect to text type and corpora)
5. the need for a careful choice of vocabulary in order to avoid :
    - categorial ambiguities
    - lexical items, which when used in ill-formed test items trigger interference with another system component in case of component testing, e.g. interference from the spelling checker when testing a combined spelling and grammar checker.

In a more general context, WP4 (together with WP3.2 and WP5.2) gave us the first opportunity to position the TSNLP Test Suite in the whole evaluation procedure, in an industrial testing environment where both artificially constructed test data and corpora would be used.

---

[17] These conclusions are mainly based on the experience and results of experiments carried out at Aerospatiale and Essex.

Hence, a first graduated methodology for the evaluation of NLP systems can be suggested, that would successively use:

- a linguistically well-founded general purpose test suite: the TSNLP Test Suite (as emerging from WP3),
- intermediate test items resulting from a domain or a system oriented customisation,
- corpora.

The intermediate level items are crucial for an industrial user, who is likely to proceed with corpora firstly, with intermediate test items secondly (if the system has failed on corpora), and with a general purpose Test Suite thirdly (in order to provide feedback to the system developer), i.e. in reverse with reference to the TSNLP methodology. The intermediate level items lead the user to the general purpose Test Suite and show him the link between his real text and a purely linguistic artificially constructed object.

In WP4, questions arose on the customisation necessary for the constitution of these intermediate level test items:

- some customisation has been achieved in TSNLP WP4, showing that even the customisation of core data to a specific system (SECC for example) or a particular domain is not always straightforward. For system oriented customisation, the ease of construction depends to a large extent on the clarity and completeness of the grammar. For text type oriented customisation, a descriptive grammar is naturally unlikely to exist, and, as mentioned in WP3.2, it is difficult to extract test items combining linguistic phenomena from corpora, because corpora above all provide very complex phenomena combinations which cannot be identified based on present day corpus tagging methods.

- ill-formed test items typical for a particular text type (or expected by a particular system) are also part of the intermediate test items. In Section 3, we have raised the question: „how valid are our ill-formed items, as data for grammar checkers, for example, which are designed to detect errors people actually make, rather than (possibly systematic) violations of grammatical constraints, as found in TSNLP data ?“
  Again, corpora and analysis tools are lacking to build such customised ill-formed test-items.

These questions boiled down to the problem of how to bridge the gap between the relatively simple but systematic core test data and the complex but unsystematic combinations of phenomena occurring in real life texts.
The task would be to combine the advantages of test suites (especially their systematicity and interpretability) with the advantages of using real life data or corpora which could help identify, among the almost infinite many possible systematic combinations of phenomena and of errors (which a test suite could produce), the ones that actually most frequently occur in real texts.

REFERENCES:

[*Adriens and Schreurs, 1992*] G. Adriaens and D. Schreurs (1992) : From Cogram to Alcogram: Toward a Controlled English Grammar Checker, in Proceedings of Coling 1992, Nantes.

[*Arnold, Rondell and Fouvry, 1994*] D. Arnold, M. Rondell and F. Fouvry (1994) : Design and Implementation of Test Suite Tools, Report to LRE 62-089 (D-WP5.1), TSNLP, University of Essex.

[*Balkan, Fouvry et al.,1995*] L. Balkan, F. Fouvry et al. (1995) : Corpus-based test suite generation, Report to LRE 62-089 (D-WP5.2), TSNLP, University of Essex.

[Balkan, Fouvry, and Kordoni, 1995] L. Balkan, F. Fouvry, and E. Kordoni (1995) : Report on Customisation and Testing of Test Suites for SECC, Report to LRE 62-089 (D-WP4.2\3), TSNLP, University of Essex.

[*Balkan, Meijer, Falkedal et al., 1994*] L. Balkan, S. Meijer, and K. Falkedal et al. (1994) : Test Suite Design and Methodology, Report to LRE 62-089 (D-WP2.1), TSNLP, University of Essex.

[*Dauphin, Lux, et al., 1995*] E. Dauphin, and V. Lux et al. (1995) : Checking coverage against corpora, Report to LRE 62-089 (D-WP3.2), TSNLP, University of Essex.

[*Estival, Falkedal, Lehmann et al., 1994*] D. Estival, K. Falkedal, and S. Lehmann (1994) et al. : Test Suite Annotation Scheme, Report to LRE 62-089 (D-WP2.2), TSNLP, University of Essex.

[*Estival , Falkedal, and Lehmann, 1995*] D. Estival, K. Falkedal, and S. Lehmann (1995) et al. : The Construction of Test Material, Report to LRE 62-089 (D-WP3.1), TSNLP, University of Essex.

[*Flickinger et al., 1987*] D. Flinckinger, J. Nerbonne, I. A. Sag, and T. Wassow : Toward evaluation of NLP systems. Technical report, Hewlett-Packard Laboratories, 1987. Distributed at the 24th Annual Meeting of the Association for Computational Linguistics (ACL)

[*Konrad,1995*] K. Konrad : Abstrakte Syntaxtransformation mit getypten Merkmalstermen. Technical report, Deutsches Forschungszentrum für Künstliche Intelligenz, 1995. Forthcoming.

[*Nerbonne et al., 1993*] J. Nerbonne, J. Laubsch, A. Kader Diagne and S. Oepen : Software for applied semantics. In Chu-Ren Huang, Claire Hsun hui Chang, Keh jiann Chen and Cheng-hui Liu, editors, *Proceedings of Pacific Asia Conference on Formal and Computational Linguistics*, pages 35-56, Talpei, 1993. Academica Sinica. Also available as Research report RR-92-55, Deutsches Forschungszentrum für Künstliche Intelligenz.

[*Oepen et al., 1995*] S. Oepen (principal author), K. Netter, J. Baur, T. Fettig, J. Klein, and F. Oberhauser : The TSNLP Database. From tsct to tsdb, Report to LRE 62-089 D-WP6.1, DFKI Saarbrücken, Germany, 1995.

[*Macken, 1995]* : L. Macken (1995) : Testing of TSNLP Test Suites, Report to LRE62-064 (WP6A1).

[*Pollard and Sag, 1994*] C. Pollard and I. A. Sag : *Head-driven Phrase Structure Grammar*. Studies in Contemporary Linguistics. The University of Chicago Press, Stanford, California, 1994.

[*Uszkoreit et al., 1994*] H. Uszkoreit, R. Backofen, S. Busemann, A. Kader Diagne, E. A. Hinkelmann, W. Kasper, B. Kiefer, H.U. Krieger, K. Netter, G. Neumann, S. Oepen, and S. P. Spackman : DISCO - An HPSG-based NLP system and its Application for Appointment Scheduling. In *Proceedings of the 15th Conference on Computational Linguistics (COLING)*, Kyoto, 1994

APPENDICES

APPENDIX 1 Examples of testing results on Clause Complementation

APPENDIX 2 Customisation procedure
- *A sample of the lexicon*
- *Example of extraction script*

APPENDIX 3 Feedback for the WP 3 on data construction
1 Ill-formed TIs which have been coded as well-formed TIs in the TSNLP TS
2 Well-formed TIs which have been coded as ill-formed TIs in the TSNLP TS
3 Miscellaneous
4 Semantic TIs
5 Vocabulary used though it does not belong to the original TSNLP vocabulary

APPENDIX 1

Examples of testing results on Clause Complementation

C_complementation (clause complementation) is an interesting phenomenon on which the judgement of the TS pertinence/system capacity are not evident. LC101 seems to have some knowledge of clause complementation. When dealing with ill-formed test items on clause complementation:

- LC101 can detect the mistake,
- LC101 sometimes utters a diagnostic.

In case of a missing complement, LC101 can evidently not suggest a correction (because it cannot guess what the complement missing was intended to be).

Example

**Il apporte.*

LC101 diagnosis *„si il s'agit du verbe apporter, je ne parviens pas à trouver son complément direct: apporter QUOI ? apporter QUI ?"*

When there are too many verbal complements, LC101 either detects the mistakes and cannot utter a diagnosis (example 1) or does not even detect the mistake (example 2). Especially if the complement too many is a PP, LC101 analysis it as a modifier.

Example 1:

**L'ingénieur vient l'entreprise.* *-> ? L'ingénieur vient l'entreprise.*

Example 2 :

**L'ingénieur accepte de l'invitation.* *-> no error detected*

In the TSNLP Test suite, this test item illustrates a PP complement introduced by the preposition DE instead of a direct object complement.

In fact, LC101 does not use semantics and the test item is acceptable modulo some lexical replacement:

Examples of well-formed test items :

L'ingénieur accepte de bon coeur. => *„de bon coeur" i*s verb modifier
L'ingénieur accepte de l'argent. => *„de l'argent"*, NP with a partitive, is a direct object complement

When the expected verbal complement is replaced by an unexpected one, LC101 mainly detects there is a mistake but cannot produce the diagnosis in compliance with the TSNLP annotations.

Examples :

**A l'ingénieur vient.* *-> ? A l'ingénieur vient.*
**Moi vient.* *-> ? Moi vient.*

APPENDIX 2

Customisation procedure

A sample of the lexicon

A competitor	NP_sg
a competitor	NP_sg
a manager	NP_sg
about	ADVP
about her	PP
about the news	PP
abroad	ADVP
Abstains	V
abstains	V
Accounts	V
accounts	V
across	ADVP
across it	PP
Adds	V
adds	V

Example of awk extraction script

```
awk -F@ WORD=<...> '{if ($1 == WORD) print $2}' <lexicon file>
```

will give you all the _categories_ of the word to be replaced (that are present in the lexicon).

There remains the task of searching all entries with the same category, and to list them.

APPENDIX 3

Feedback for the WP3 on data construction

1 Ill-formed TIs which have been coded as well-formed TIs in the TSNLP TS

LC101 detected mistakes in the following TIs, which were considered correct in the TSNLP test suite.

Le question habituelle et intelligent	--3-6-6-2-1
Tous la hommes	--3-6-10-2-6
Mon quelques entreprises	--3-6-10-6-5

2 Well-formed TIs which have been coded as ill-formed TIs in the TSNLP test suite

LC101 accepted the following TIs which were ill-formed items of the TSNLP TS. Most of the time, in such cases, the TI had indeed a possible correct analysis.

TSNLP test input	Results of testing	TSNLP annotations	Modifications on the initial test input
heureux si l' ingénieur vient	correct as a conditional construction (ex.: „L'organisateur est heureux si l'ingénieur vient.")	-- 4-4-2-1-4	WF = 03
Neige.	can be a correct NP (in a tittle)	-- 2-4-1-2-3	Pleut. (declarative sentence with a missing subject) (phrase déclarative avec sujet impersonnel manquant).
Accepte l' invitation .	correct imperative	-- 2-4-4-1-3	Acceptent l'invitation. (declarative sentence with a missing subject) (phrase déclarative avec sujet manquant)
Accepte la question .	correct imperative	-- 2-4-4-2-3	Acceptent la question. (declarative sentence with a missing subject) (phrase déclarative avec sujet manquant)
A l' ingénieur pose la question .	correct as an imperative („Pose la question à l'ingénieur."), with inversion of the indirect object.	-- 2-4-4-2-2	A l'ingénieur mérite la question (declarative sentence with an indirect object instead of a subject) (phrase declarative avec objet indirect à la place du sujet)
Eux viennent .	correct in the paragraph: „La plupart des gens refusent de se déplacer. Eux viennent."	-- 2-4-3-6-5	Toi viens. (if this TI is not already in the TS) Note that modulo punctuation, this TI is well-formed: Toi, viens!
Eux l' acceptent.	idem	-- 2-4-5-7-7	Toi l'accepte. (if this TI is not already in the TS)
Lui l' accepte	correct in the paragraph: „La plupart des gens refusent cette réalité. Lui l'accepte."	-- 2-4-5-3-8	Moi l'accepte. (if this TI is not already in the TS)
Lui vient .	idem	--2-4-3-3-5.	Moi viens. (declarative sentence with a dative complement instead of an accusative complement) (if this TI is not already in the TS)

Elle lui vient .	correct (could be a pronominalisation for „Une idée lui vient.“)	-- 2-4-3-3-9	Elle moi vient. (declarative sentence with a dative complement too many) (if this TI is not already in the TS)
Elles lui viennent.		----2-4-3-6-9	Elles moi viennent. (phrase declarative, complement datif en trop) (if this TI is not already in the TS)
Ils lui viennent.		--2-4-3-6-8	Ils moi viennent. (declarative sentence with a dative complement too many) (if this TI is not already in the TS)
L' entreprise faut pour l' ingénieur.	correct if the verb is not FALLOIR but FAILLIR	-- 2-4-6-1-7	verbe impersonnel transitif direct avec sujet personnel = ?
L' entreprise faut un ingénieur .	-	- 2-4-6-1-2	L‘entreprise se peut un ingénieur. (impersonal verb supporting a direct object, with a personal subject instead of the expected impersonnal one). Note: we could not find a better verb than „se pouvoir“.
L' entreprise faut si l' ingénieur vient.	idem	-- 2-4-6-1-8	WF = 03 Problem to derive an unambiguous ill-formed TI: si_compl (intended to be misplaced) will always be a potential (correct) conditional clause.
Il faut .	idem	--2-4-6-1-11	Il se peut.
L' entreprise faut .	idem	--2-4-6-1-12	L‘entreprise se peut.
Il faut pour un ingénieur .		-- 2-4-6-1-3	Il se peut pour un ingénieur.
Il faut si l' ingénieur vient.		-- 2-4-6-1-4	Il se peut si l‘ingénieur vient.
L' ingénieur rêve si la femme vient .	correct as a conditional construction (A if B)	-- 2-4-8-1-5	? Problem to derive an unambiguous ill-formed TI: si_compl (intended to be misplaced) will always be a potential (correct) conditional clause.
L' ingénieur rêve par l' entreprise .	syntactically possible (cf. „L‘ingénieur rêve par	- 2-4-8-1-2	? Problem to derive an unambiguous ill-formed TI: the PP that should be a misplaced complement will mainly (always ?) be a potential correct modifier.
L' ingénieur rêve l' entreprise .	correct as „rêver“ has a transitive acceptation	-- 2-4-8-1-3	L‘ingénieur luit l‘entreprise (declarative sentence with an indirect object too many).
L' entreprise suppose que l'ingénieur vienne .	correct as „entreprise“ can mean „undertaking“, and „supposer“ can mean „to presuppose“	--2-4-10-1-3	L‘entreprise apprend que l‘ingénieur vienne. (declarative sentence, transitive verb, c_completive in the wrong mode).
L' ingénieur pense l' entreprise .	correct as „penser“ has a transitive acceptation	-- 2-4-8-2-3	L‘ingénieur appartient l‘entreprise. (declarative sentence, direct object instead of an indirect object)
L' entreprise regrette .	correct, as „regretter“ has an intransitive acceptation	--2-4-10-2-4	L‘entreprise atteint. (declarative sentence with a missing object)
Je me/te/le/la/les/nous/vous rêve.	syntactically correct as „rêver“ has a transitive acceptation. Improbable for semantic reason (because it implies that an object is „speaking“).	-- 2_4_9_2_2 to 2_4_9_2_10	Je me/te/le/la/les/nous/vous médis. (declarative sentence, pronominalised direct object instead of a de_obj)
Tu me/te/le/la/les/nous/vous rêves.	syntactically correct as „rêver“ has a transitive acceptation	-- 2_4_9_3_2 to 2_4_9_3_10	Tu me/te/le/la/les/nous/vous médis. (declarative sentence, pronominalised direct object instead of a de_obj)

Il me/te/le/la/les/nous/vous rêve.	syntactically correct as „rêver“ has a transitive acceptation	-- 2_4_9_4_2 to 2_4_9_4_10	Il me/te/le/la/les/nous/vous médit. (declarative sentence, pronominalised direct object instead of a de_obj)
Nous me/te/le/la/les/nous/vous rêvons.	syntactically correct as „rêver“ has a transitive acceptation	-- 2_4_9_5_2 to 2_4_9_5_10	Nous me/te/le/la/les/nous/vous médisons. (declarative sentence, pronominalised direct object instead of a de_obj)
Vous me/te/le/la/les/nous/vous rêvez.	syntactically correct as „rêver“ has a transitive acceptation	-- 2_4_9_6_2 to 2_4_9_6_10	Vous me/te/le/la/les/nous/vous médites. (declarative sentence, pronominalised direct object instead of a de_obj)
Ils me/te/le/la/les/nous/vous rêvent.	syntactically correct as „rêver“ has a transitive acceptation	-- 2_4_9_7_2 to 2_4_9_7_10	Ils me/te/le/la/les/nous/vous médisent. (declarative sentence, pronominalised direct object instead of a de_obj)
Je te/nous/vous/les suppose .	syntactically correct as „suposer“ has a transitive acceptation and the pronouns te/nous/vous/les are either accusative or dative. Sometimes improbable for semantic reason (the use of „me“, „nous“, „vous“ implies that the supposed object are „speaking“ or are spoken to).	-- 2-4-11-1-2 to -- 2-4-11-1-5	Je te/nous/vous/les luis. (declarative sentence with a pronominalised direct object too many)
Tu me/nous/vous/les/le supposes .	syntactically correct as „suposer“ has a transitive acceptation.	-- 2-4-11-2-2 to -- 2-4-11-2-6	Tu me/nous/vous/les/le luis. (declarative sentence with a pronominalised direct object too many)
Il me/te/nous/vous/les suppose .	syntactically correct as „suposer“ has a transitive acceptation.	-- 2-4-11-3-2 to -- 2-4-11-3-6	Il me/te/nous/vous/les luit. (declarative sentence with a pronominalised direct object too many)
Elle me/te/nous/vous/les suppose .	syntactically correct as „suposer“ has a transitive acceptation.	-- 2-4-11-4-2 to -- 2-4-11-4-6	Elle me/te/nous/vous/les luit. (declarative sentence with a pronominalised direct object too many)
Nous me/te/vous/les supposons .	syntactically correct as „suposer“ has a transitive acceptation.	-- 2-4-11-5-2 to -- 2-4-11-5-5	Nous me/te/vous/les luisons. (declarative sentence with a pronominalised direct object too many)
Vous me/te/nous/les supposons .	syntactically correct as „suposer“ has a transitive acceptation.	-- 2-4-11-6-2 to -- 2-4-11-6-5	Vous me/te/nous/les luisez. (declarative sentence with a pronominalised direct object too many)
Ils me/te/nous/vous/les supposent .	syntactically correct as „suposer“ has a transitive acceptation.	-- 2-4-11-7-2 to -- 2-4-11-7-6	Ils me/te/nous/vous/les luisent. (declarative sentence with a pronominalised direct object too many)
Elles me/te/nous/vous/les supposent .	syntactically correct as „suposer“ has a transitive acceptation.	-- 2-4-11-8-2 to -- 2-4-11-8-6	Elles me/te/nous/vous/les luisent. (declarative sentence with a pronominalised direct object too many)
L' entreprise suppose l' ingénieur .	syntactically correct as „supposer“ is a transitive verb. Improbable for semantic reasons.	-- 2-4-10-1-4	L‘entreprise luit l‘ingénieur. (phrase declarative avec un objet direct en trop)
Nous travaillant	correct clause in „Nous travaillant, il était seul à la maison“.	-- 2-3-1-5-7	Nous travaillant. With a final full stop ie. as a sentence, this TI is unambiguously an ill-formed one.
Vous travaillant	idem	-- 2-3-1-6-7	Vous travaillant. With a final full stop ie. as a sentence, this TI is unambiguously an ill-formed one.

Je suis travaillé	correct (though not probable without agent complement) because travailler has a transitive acceptation)	-- 2-3-2-1-2	Je suis marché.
Je serai travaillé	idem	-- 2-3-2-18-2	Je serai marché.
Tu es travaillé	idem	-- 2-3-2-2-2	Tu es marché.
Tu sera travaillé	idem	-- 2-3-2-19-2	Tu sera marché.
Il est travaillé	idem	-- 2-3-2-3-2	Il est marché.
Il sera travaillé	idem	-- 2-3-2-20-2	Il sera marché.
Vous êtes travaillé.	idem (polite singular VOUS)	-- 2-3-2-6-2	Vous êtes marché.
Vous serez travaillé.	idem (polite singular VOUS)	-- 2-3-2-23-2	Vous serez marché.
Il a travaillé aujourd'hui .	correct if the sentence is uttered in the evening	-- 2-3-3-1-2	? Il a travaillé demain (semantic test item)
Il travaille demain .	the use of the present for a future is correct	-- 2-3-3-3-2	? Il travaille hier (semantic test item).
Il travaillera aujourd'hui .	correct if the sentence is uttered in the morning	-- 2-3-3-1-5	? Il travaillera hier (semantic test item).
un litre de vins	correct NP as wine can be countable (ex. „Il apporte un litre de vins: un demi-litre de Beaujolais et un demi-litre de Bourgogne“)	-- 3-6-2-2-3	WF = 01
un litre des vins	correct NP as wine can be countable ?	-- 3-6-2-2-4	WF = 02
deux litres de vins	correct NP as wine can be countable (ex. „Il apporte deux litres de vins: un litre de Beaujolais et un litre de Bourgogne“)	-- 3-6-2-3-3	WF = 01
deux litres des vins	correct NP as wines can be countable ?	-- 3-6-2-3-4	WF = 02
deux kilos de sucres	correct NP as sugar can be countable (ex. „Il apporte deux kilos de sucres: un kilo de sucre roux et un kilo de sucre blanc.)	-- 3-6-2-4-3	WF = 01
deux kilos des sucres	correct NP as sugar can be countable ?	-- 3-6-2-4-4	WF = 02
Beaucoup des vins	correct NP as wines can be countable ?	-- 3-6-9-2-4	WF = 02
Beaucoup de vins	correct NP as wines can be countable	-- 3-6-9-2-3	WF = 01
trop de vins	correct NP as wines can be countable	-- 3-6-9-4-3	WF = 01
trop des vins	correct NP as wines can be countable ?	-- 3-6-9-4-4	WF = 02
assez de vins	correct NP as wines can be countable	-- 3-6-9-6-3	WF = 01
assez des vins	correct NP as wines can be countable ?	-- 3-6-9-6-4	WF = 02
assez des personnes	?	-- 3-6-9-7-2	WF = 02
La plus belle livre	correct NP as „livre“ can also be a feminine noun meaning „pound“	-- 3-6-11-1-5	La plus belle ingénieur.

Les plus belles livres	• Les plus belles livres √ Les plus •**beaux** livres correct NP as „livre“ can also be a feminine noun meaning „pound“	-- 3-6-11-1-12	Les plus belles ingénieurs.
cordial de l' ingénieur	correct NP as „cordial“ can also be a noun	-- 4-4-1-3-2	affectueux de l‘ingénieur
cordial à l' ingénieur	correct NP as „cordial“ can also be a noun (improbable for semantic reason because an engineer is unlikely to be a cordial ingredient)	-- 4-4-1-3-3	affectueux à l‘ingénieur
difficile pour l' entreprise pour comprendre	syntactically correct if the second PP (pour l‘entreprise) is attached to „l‘entreprise“	-- 4-4-4-1-3	WF = 03
La décision de l' ingénieur à venir	correct NP though semantically improbable („the decision of the engineer to come“)	-- 3-4-5-1-3	WF = 03
L 'invitation de l' ingénieur de la négociation	syntactically correct if the PP „de la négociation“ is attached to „l‘ingénieur“	-- 3-4-4-1-2	WF = 03

3 Miscellaneous

- C_ TenseAspectModality: 0Vous instead of Vous

0Vous avez travaillez	-- 2-3-2-6-5
0Vous êtes venez	-- 2-3-2-15-5
0Vous aurez travaillez	-- 2-3-2-23-5
0Vous serez venez	-- 2-3-2-32-5

- C_COMplementation: remaining *

*A l' entreprise pleut	-- 2-4-1-1-4
*Il pleut l' entreprise.	-- 2-4-1-1-5
*Il pleut venir.	-- 2-4-1-1-6

- In the NP_agreement phenomena, some items begin with a capital, others do not.
- Some IDs look like --a--b--c--d--e instead of --a-b-c-d-e.
- In the TIs, the determiner „l‘“ is followed by a space. This is not in compliance with the typing rules (so that LC101 repeatedly signals misplaced spaces).
- In the TIs ending with a full stop, the full stop is followed by a space. This is not in compliance with the typing rules (so that LC101 repeatedly signals misplaced spaces).

4 Semantic TIs

Il a travaillé aujourd'hui .	-- 2-3-3-1-2
Il avait travaillé aujourd'hui .	-- 2-3-3-1-3
Il travaille hier .	-- 2-3-3-2-2
Il travaillera hier .	-- 2-3-3-2-3
Il va travailler hier .	-- 2-3-3-2-4
Il a travaillé demain .	-- 2-3-3-3-3
Il avait travaillé demain .	-- 2-3-3-3-4

5 Vocabulary used though it does not belong to the original TSNLP vocabulary

VERB

- pleuvoir
- neiger

NOM

- cérémonie
- fleur
- femme
- littérature
- livre (an ambiguous noun)
- personne
- maison
- minorité
- vin
- huile
- farine
- sucre

ADJ

- civile
- religieuse
- espagnole
- italienne