

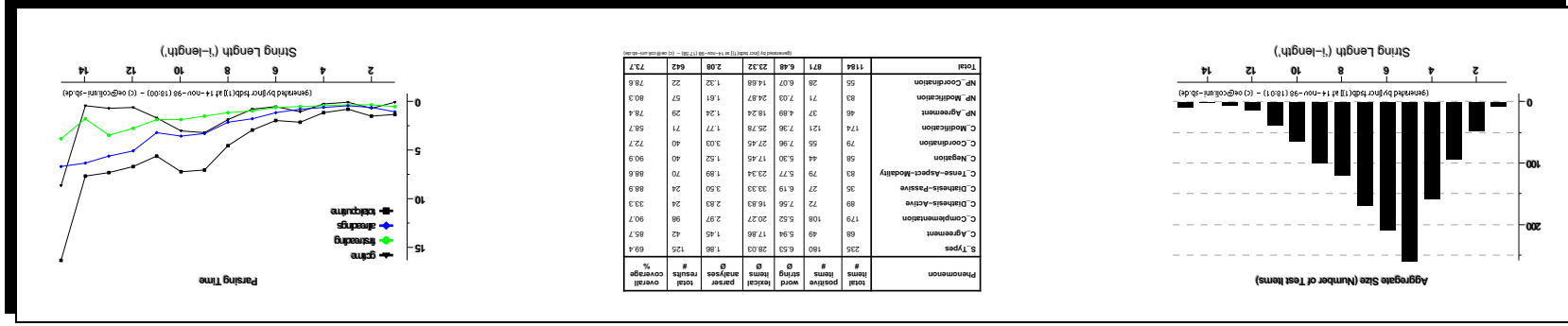
# Implementing Grammars Large and Small

— HPSSG Engineering for Education and Industry —

Stephan Oepen

CSLI Stanford and Y Y Technologies

oecsli.stanford.edu



# Unification-Based Processing Underway to Dot Com

Tutorial at the 38<sup>th</sup> Annual Meeting of  
the ACL (Hong Kong, October 2000)

**Dan Flickinger**

CSLI, Stanford University  
and Y Y Software Corporation  
dan@csli.stanford.edu

**Stephan Open**

Computational Linguistics  
Saarland University  
oe@coli.uni-sb.de

## Based on Research and Contributions of

Ulrich Callmeier, John Carroll, Liviu Ciortuz,  
Ann Copestake, Dan Flickinger, Bernd Kiefer,  
Hans-Ulrich Krieger, Takaki Makino,  
Rob Malouf, Yusuke Miyao,  
Stefan Müller, Mark-Jan Nederhof,  
Günter Neumann, Takashi Ninomiya,  
Kenji Nishida, Ivan Sag, Melanie Siegel,  
Kentaro Torisawa, Jun-ichi Tsujii,  
Hans Uszkoreit, and many others.

# Outline: What We Are About to Do

## Present

- large, multi-national collaboration on efficient HPSG engineering;
- open-source repository of multi-purpose grammars and systems;
- broad progress achieved over a development period of six years.

## Motivate

- collaborative approach to grammar and system development;
- frequent, systematic, precise progress evaluation and comparison;
- emerging utility of 'deep' processing for practical applications.

# Why Build Natural Language Grammars by Hand?

## Why Computational Grammars

- **research** formalize linguistic theories with complex interactions of language phenomena; identify cross-language generalizations; e.g. Dependency Grammar, Categorical Grammar, LFG, HPSG et al.
- **theory validation** enable direct feedback on predictions made by an implemented model; immediate empirical hypotheses testing.
- **education** teach frameworks or analyses in formal morphology, syntax, and semantics; autonomous student experimentation.
- **applications** embed grammar-based natural language analysis in research prototypes and (increasingly) commercial applications; e.g. machine translation, auto response, deep information extraction.

# Student Experimentation — Immediate Gratification



## Implementing Grammars Large and Small (6)

# 'Deep' Grammars Can Provide Semantic Precision

## Composition of Meaning from Words and Phrases

- Computational grammars encode linguistically motivated analyses;
- syntax – semantics interface is explicit and clearly defined;
- generic schemata, lexical classes, and (idiosyncratic) lexical entries.

## Spurious Results Can Be Avoided

- Grammars distinguish well-formed from ungrammatical utterances;
- fine-grained distinctions interact well with general principles;
- declarative grammar can serve both for parsing and for generation.

# Third Generation of Broad-Coverage Grammars

## Characteristics

- multi-dimensional, hierarchical representation of knowledge;
- declarativity and reversibility: support parsing and generation;
- focus on engineering methodologies and efficient processing;
- multi-lingual development, often distributed across several sites.

## Examples

- ParGram (PARC, Xerox Grenoble, IMS Stuttgart, Bergen);
- LINGO (Stanford, Saarbrücken, Tokyo, Cambridge, Groningen);
- F|XTAG (University of Pennsylvania, Université Paris 7, Korea).

# Collaboration is Essential in this Millennium

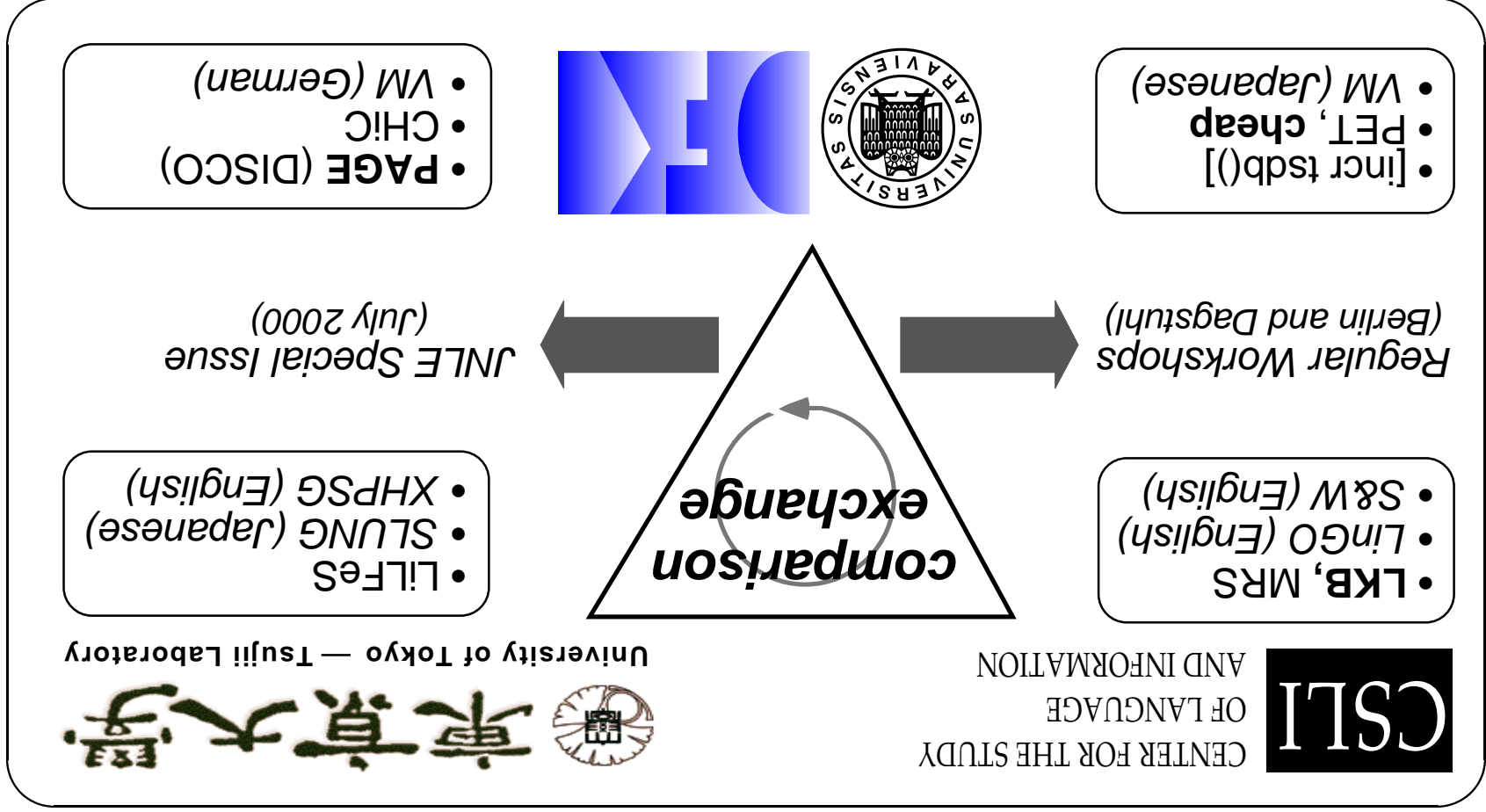
## Unification-Based Grammars Benefit from Exchange

- Multitude of phenomena to be analyzed in natural languages;
- designing, implementing, and validating good analyses requires both linguistic expertise and competence with chosen formalism;
- collaborators have already demonstrated benefits from exchanging analyses and techniques (e.g. ParGram, HPSG community).

## Advances in Processing Technology Can Be Combined

- Groups worldwide continue work on better development tools and processing algorithms for unification-based grammars;
- sustained and detailed interaction among such developers has been demonstrated to be highly beneficial to all (e.g. JNLE, March 2000).

# Setup: International, Multi-Site Collaboration



# Common Background Assumptions and Goals

## Linguistic Framework and Descriptive Formalism

- Established frameworks: HPSG and Minimal Recursion Semantics;
- mutual feedback between theory development and implementation;
- convergence on typed feature structure formalism (details below);

## Joint Goals

- establish joint repository of grammars, processors, reference data;
- advance theory building and implementation through synergy;
- reusability: make as much as possible available open-source;
- devise efficient and flexible technology for practical applications.

# Convergence and Cross-Fertilization: Some Examples

## Theoretical

- **Methodology** experimental, empirically-driven engineering;
- **Benchmarking** regular precise and systematic comparison;
- **Unification** graph unification vs. abstract machines;
- **Parsing** specialized bottom-up control; novel ambiguity packing.

## Practical

- **Profiling** uniform data format and analysis environment;
- **Comparability** common reference grammars and test data;
- **Communication** regular work-shops, mutual exchange of staff;
- **Exchange** transfer of specific algorithms and components.

# Collaborative Research on HPSG Processing

## Convergence in Descriptive Formalism

- Conservative selection from existing multitude of formal devices;
- blend of [Carpenter, 1992], [Copestake, 1992], and [Krieger, 1995].

## Common Reference Grammars on Multiple Platforms

- LingO grammar (CSLI): 8082 types (5552 leaves); 6897 (17917) lexical entries; 29 lexical and 45 phrase structure rules (22-may-00);
- Japanese grammar (DFKI): 3710 types (2485 leaves); 3654 (3654) lexical entries; 0 lexical and 28 phrase structure rules (10-apr-00);
- PAGE (DFKI), LKB (CSLI), LILFES (Tokyo), ABC Light (DFKI), PET (COLI), Cali (TU Delft), ALE (Gerald Penn), TFS (Martin Emelé).

# Collaborative Research on HPSG Processing (2)

## Common Sets of Test and Reference Data

- *'csll'* 1348 manually constructed test item (HP test suite); simple; 96 randomly selected VerbMobil sentences; medium;
- *'aged'* 2161 VerbMobil items; balanced up to 20 words; hard.

Set	Aggregate	total items	word string	lexical entries	total results	parser analyses	passive edges	fs size
<i>'csll'</i>	wellformed	918	6.45	15.3	732	2.16	115	302
	illformed	375	6.11	14.9	85	2.31	84	298
<i>'aged'</i>	wellformed	96	8.41	23.1	72	7.00	292	315
<i>'fuse'</i>	wellformed	1975	11.62	42.9	1265	69.55	1895	336
	illformed	186	12.54	48.0	36	31.64	1381	317

# Choice of Descriptive Formalism

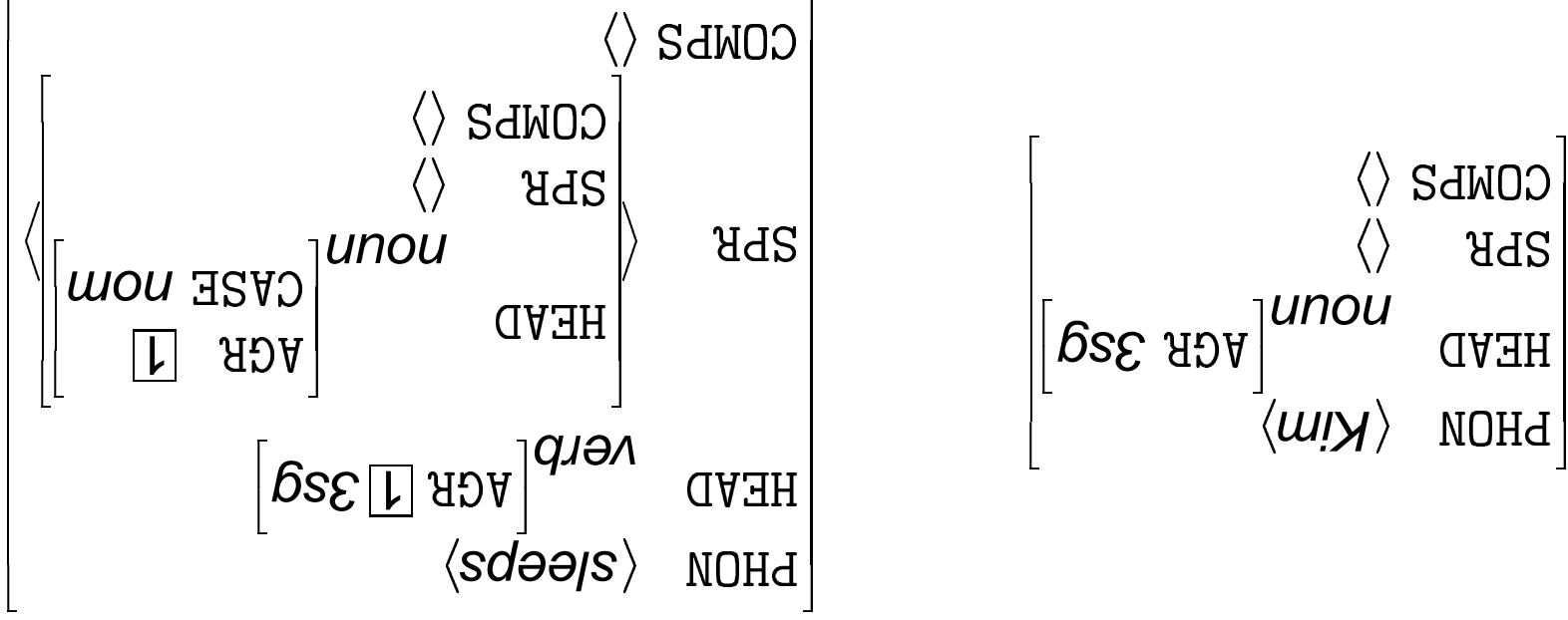
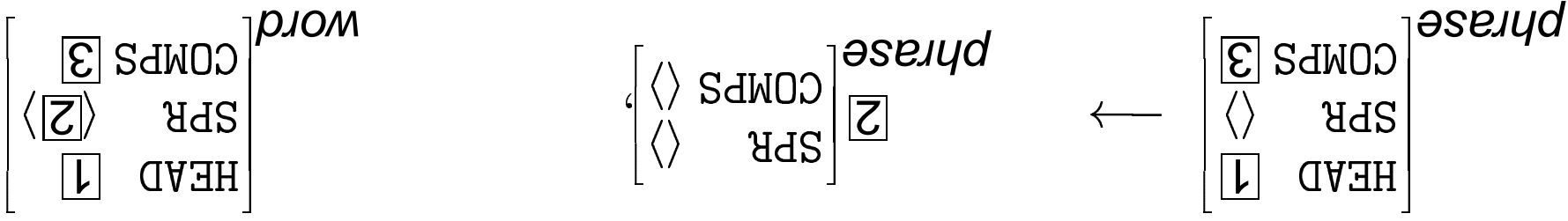
## Desiderata

- **Declarativity** clear separation of declarative and procedural knowledge: very same grammar is used for parsing and generation;
- **Adequacy** non-redundant account of linguistic generalizations;
- **Efficiency** build on known techniques: aim for near real-time.

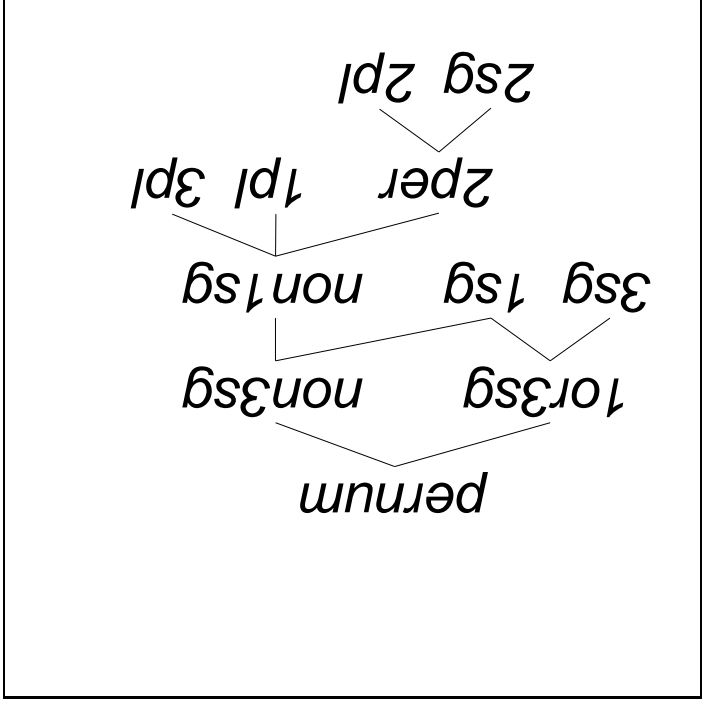
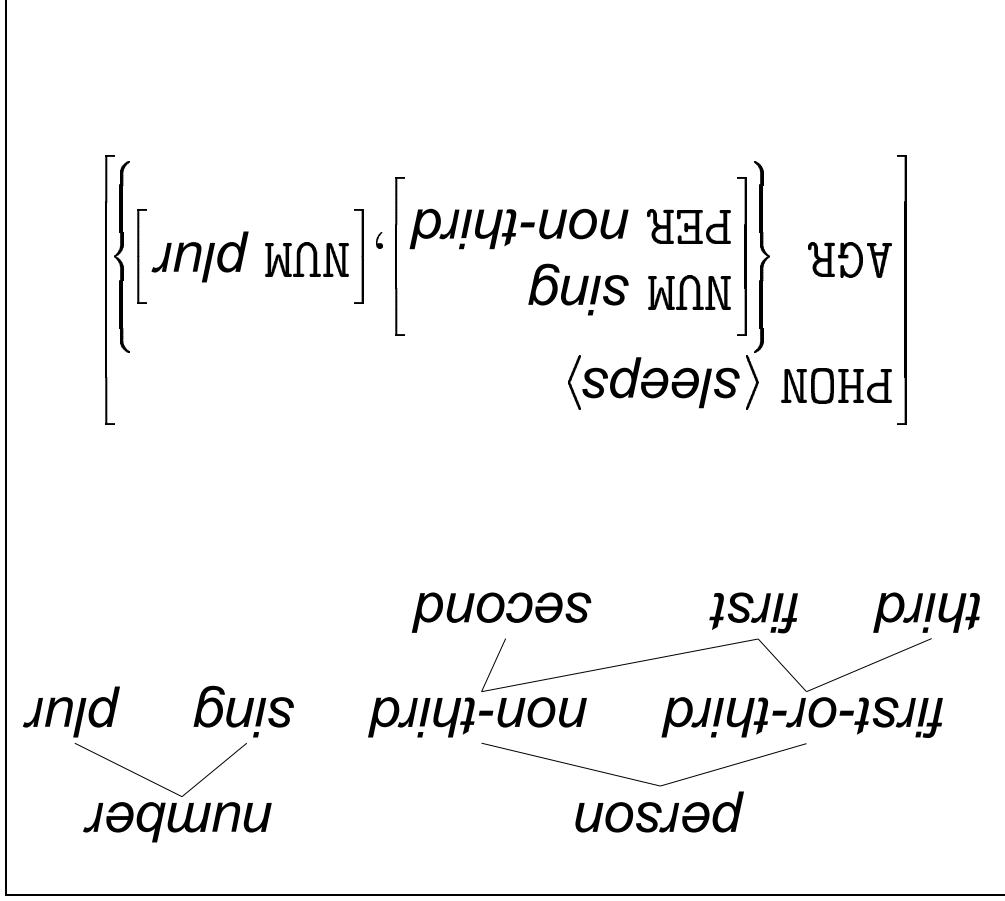
## Design Decisions

- Unification of typed feature structures is the central operation;
- multiple inheritance, strict appropriateness, complex constraints;
- closed world; no disjunctive, implicational, or relational constraints

# Reminder: Feature Structures and Unification



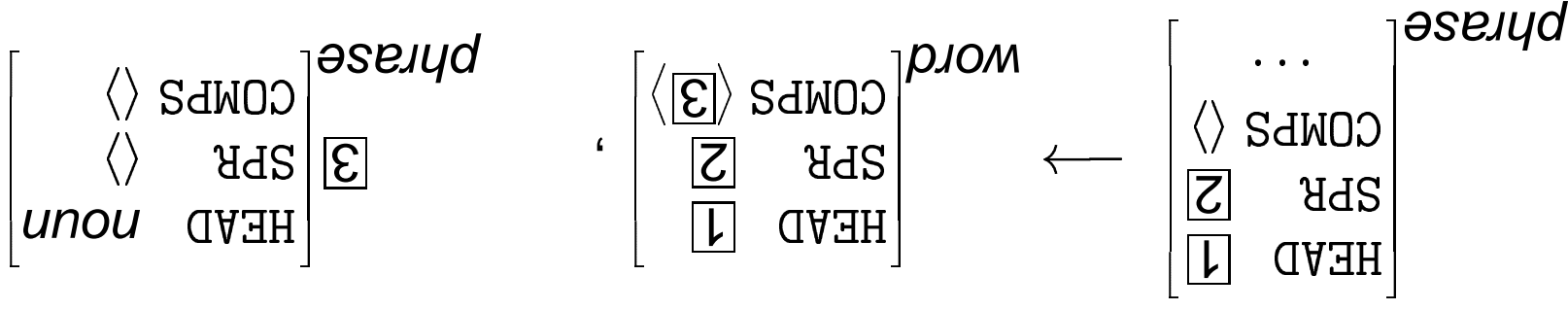
# Type Hierarchies: Avoiding Disjunctive Specification



# All Linguistic Objects are Typed Feature Structures

## Basic Inventory

- **Lexicon** rich collection, indexed by stem and semantic relation;
- **Principles** constraints on types, (limited) application at run-time;
- **Schemata** fixed-arity phrase structure rules with TFS categories;
- **Start Symbol(s)** TFS specification(s) that must unify with results.



# An Example: Lingo English Resource Grammar

## Development history (1993 – present)

- First prototype in ALE, then in PAGE with disjunctive constraints;
- revision in PAGE, using types instead of discussions;
- more recently, adoption of strict typing using LKB system.

## Design

- HPSG [Pollard & Sag 1994]: specifiers, complements, modifiers;
- hierarchies of types defining feature structures for lexicon and rules;
- feature structures provide both syntactic and semantic constraints;
- no macros, no negation, no disjunctive feature specifications;
- avoidance of ungrammaticality: generator using same grammar.

# Lingo English Grammar: Coverage and Size

## Linguistic Coverage

- 85 % of 12,000 transcribed dialogue turns from Verbobil domains;
- average 9-word utterances, ranging from 1 – 40 words in length;
- 80 % of phenomena-based examples in HP-derived test suite.

## Size of Grammar (as of May 2000)

- 8082 types (5552 leaf types) for lexicon, rules, and semantics;
- 6897 lexical entry stems (corresponds to 17917 inflected forms);
- 29 lexical (6 inflectional) and 45 phrase structure rules;
- 24,000 source lines for type definitions (excluding lexical entries);
- 30,000 lines for hand-built lexicon (more recently as DB).

## Sample Verbobil Dialogue (from CD 13) Analyzed by LINGO English Grammar

- *Are you free on the afternoon of Thursday the first, or the morning of Friday the second? Those are pretty open days for me.*
- *Well, on the first I can only meet you after four pm, so that might be tight. On Friday I have a seminar from ten to two thirty.*
- *Are you going to be around anytime on the fifth? I am free all day on the fifth.*
- *Well, on the fifth I have a seminar from ten to two and a lecture from three to five, so I could, from nine o'clock to ten in the morning, if that is not too early for you?*
- *That is perfectly fine with me, if it is okay with you.*

# Grammar Development Tools

## Requirements

- Well-formedness and consistency checking of source definitions;
- flexible views into data structures, both static and dynamic;
- adequate processing engines (see below);
- profiling tools for competence and performance (see below);

## An Illustration: The LKB System

- Developers: Copestake, with Carroll, Malouf, and Open;
- implementation: Allegro CL, Macintosh CL, (LispWorks, CMU CL);
- available in open-source and binary form for common platforms.

# The Linguistic Knowledge Building (LKB) System

## Error Checking of Type Definitions

- Grammar definition errors identified at load time by position in file;
- inherited attributes and values checked by type expansion;
- inheritance and appropriateness tracked by type and attributes;
- batch check and expansion of full lexicon on demand.

## Processing Engines

- Bottom-up passive or (hyper-)active chart parser, all-paths or n-best;
- chart-based Shake-N-Bake style generator [Carroll et al. 1999];
- both map between strings and flat semantic representations.

# Recent Breakthrough in Processing Efficiency

## Current State-of-the-Art

- Exhaustive parsing of 25-word sentences in less than one second;
- moderate space consumption: run-time module around 25 mbytes.

## Most Important Engineering Achievements (JNLE March 2000)

- Built on rich literature: three decades of research in CL and CS;
- exchange of implementation experience, eclectic engineering;
- improved feature structure encodings, unification, and copying;
- specialized parsing and generation algorithms (see below);
- better indexing and filtering; optimal factoring of local ambiguity;
- detailed documentation of resulting encodings and algorithms.

## A Prototypical Example: The PET System

- Synthesize best current knowledge from collaborative experience;
- provide extensible building blocks: various feature structure encodings, unify and copy algorithms, task filters, and parsing regimes;
- experiment with different parameter settings and configurations;
- integrate, scale, adapt, and improve encodings and algorithms;
- build small, well-engineered, and optimized implementation in C<sub>++</sub>;
- document [Callmeier 2000] (Diploma Thesis at Coli Saarbrücken).

# PET at Work: A Parser Instantiation

The (Current) Best Parser: *cheap* — Main Ingredients

- [Ait-Kaci, 1991] (WAM-style) stack-based memory allocation;
- [Tomabechi, 1991] quasi-destructive graph unification;
- [Gerdemann, 1995] partial expansion and constraint unfilling;
- [Malouf et al., 2000] subgraph sharing, (pre-unification) quick check;
- [Kiefer et al., 1999] static rule filter, cache glb computation;
- [Open & Carroll, 2000] (bidirectional, key-driven) hyper-active parser;
- [Callmeier, 2000] dag size, type encoding, quick check improvements.

# Quantifying Progress: October 1996 vs. August 1999

Test Set	test	October 1996	August 1999
Test Set items	#	4463	4463
lexical parser	$\phi$	2.32	2.21
in	$\phi$	1.75	2.67
out	$\phi$	65.3	2.67
lexical parser	$\phi$	2.55	7.00
in	$\phi$	65.8	75.0
out	$\phi$	2.11	—
'aged'	95	2.11	—
'tsnlp'	4463	2.32	2.21
'aged'	95	2.11	2.74

(generated by [incr tsdb()] at 5-nov-1999 (17:11 h))

Version	Platform	Test Set	filter	etasks	pedges	tcpu	space
August 1999	cheap	'tsnlp'	93.9	170	55	0.03	333
October 1996	PAGE	'aged'	51.3	1763	97	36.69	79093
August 1999	cheap	'aged'	95.1	753	292	0.14	1435
October 1996	PAGE	'fuse'	95.5	3084	1140	0.65	10589
Version	Platform	Test Set	%	$\phi$	$\phi$	$\phi$ (s)	$\phi$ (kb)

(generated by [incr tsdb()] at 5-nov-1999 (21:23 h))

# Semantics Engineering and Practical Applications

## Design of Minimal Recursion Semantics (MRS)

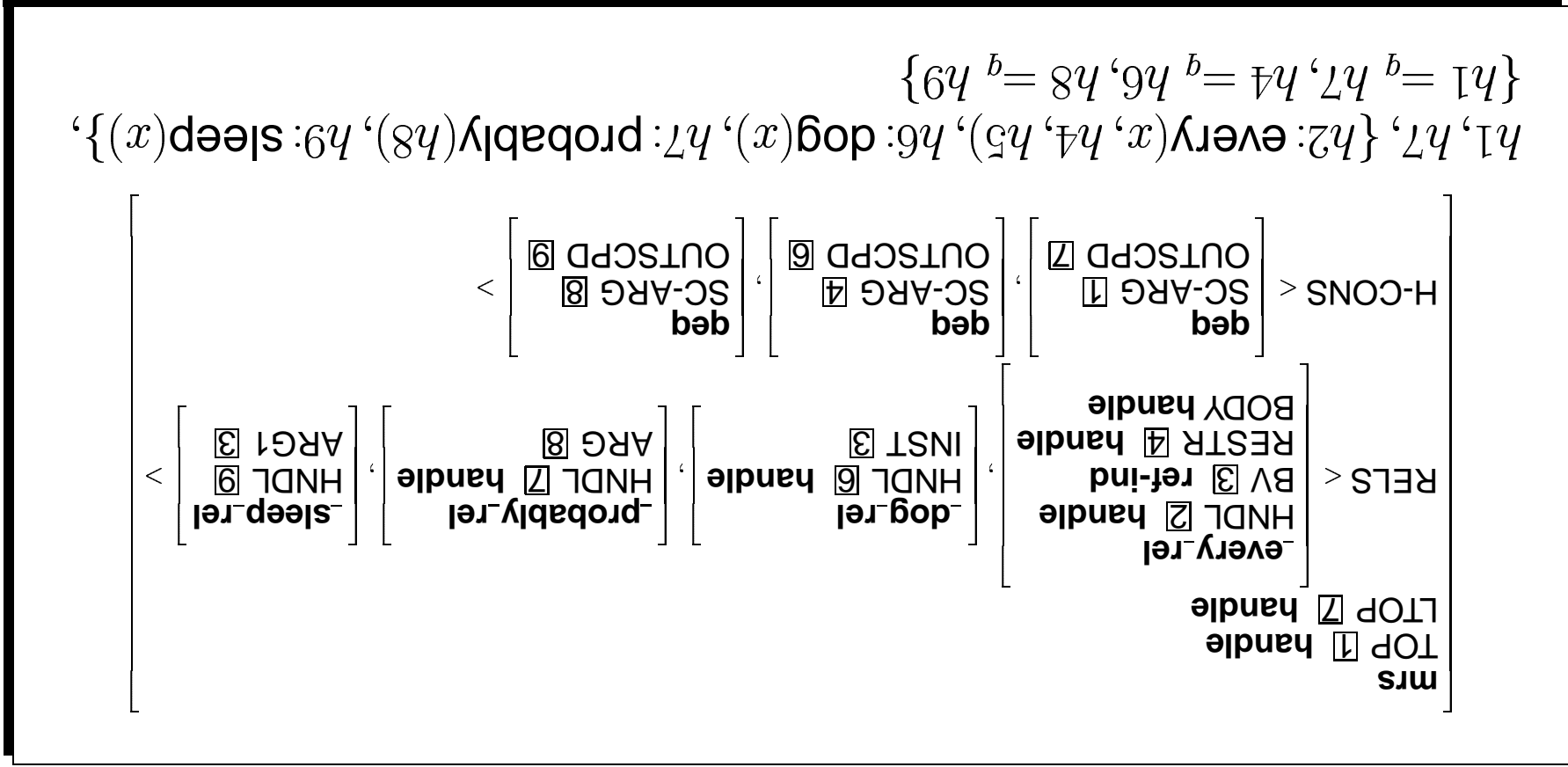
- flat, non-recursive unordered bags of relations with handles;
- explicit 'handle constraints' to represent scopal relations;
- provision for underspecification of relation types and scope;
- semantic composition in grammar delivers initial semantics;
- well-formedness checking and resolution in post-processing.

## Use of MRS in LINGO English Resource Grammar

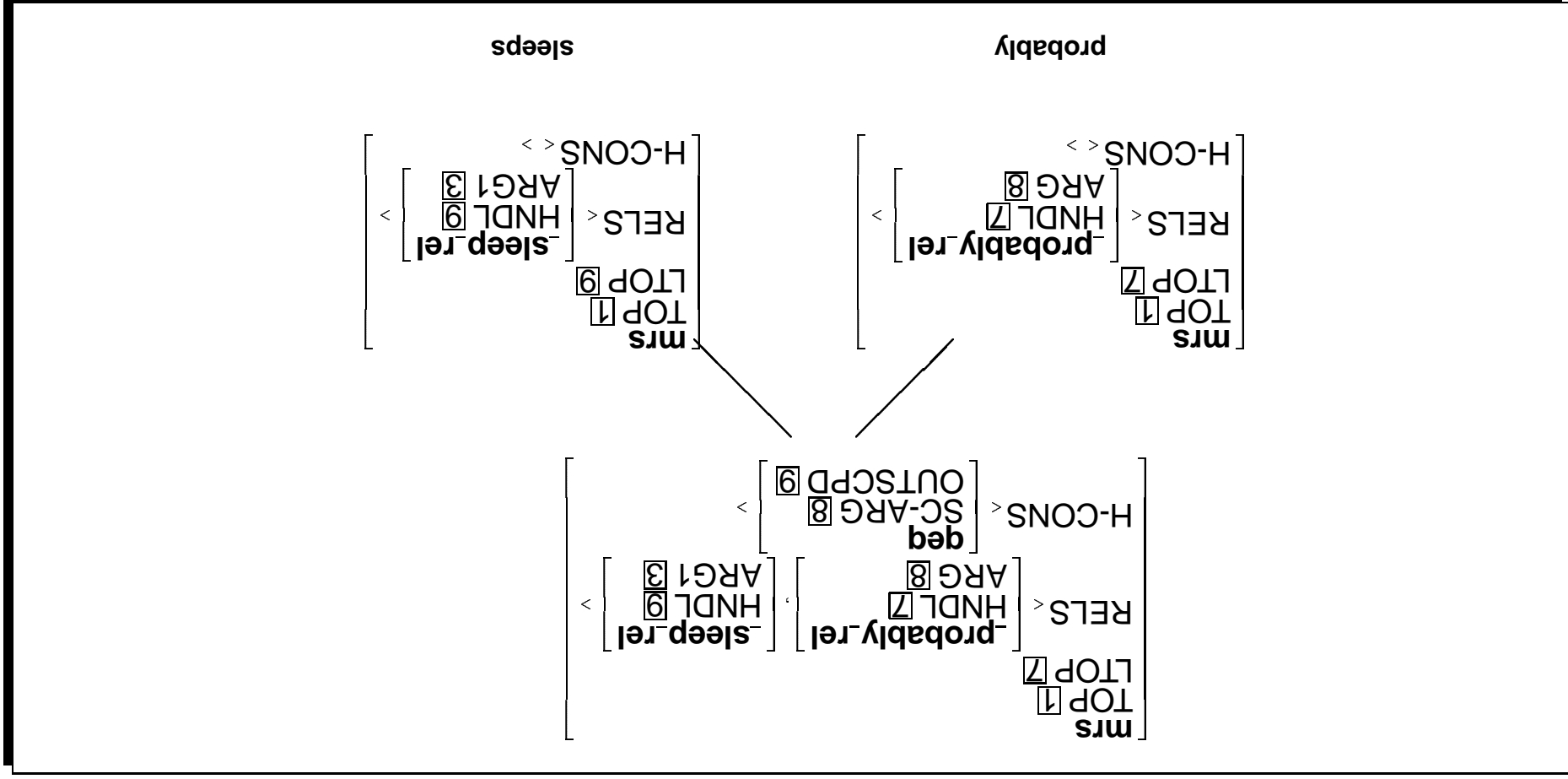
- unique predicate (relation) for almost every lexical entry;
- limited inventory of attributes (thematic roles) to simplify linking.

# MRS Feature Structure and Logical Form

*Every dog probably sleeps*



# MRS Composition: *probably sleeps*



## Conclusions

- 'Deep' grammar-based processing is still a very relevant paradigm;
- precise linguistic grammars are required for certain applications;
- building large-scale grammars and lexicons requires sustained efforts;
- collaborative engineering has boosted processing efficiency for HPSG;
- specialized tools are indispensable for grammar engineers and developers;
- HPSG parsing has matured to applicability in commercial contexts.

*Invitation: Additional collaborators will be very welcome*

# Outlook — Literature and Further Pointers

## Selection of Ongoing Activities

- [Torisawa et al., 2000]: CFG approximation for LINGO (JNLE);
- [Kiefer & Krieger, 2000]: CFG approximation for LINGO (IWPT);
- [Open & Carrol, 2000]: Optimal HPSG ambiguity packing (NAACL);
- [van Lohuizen, 2001]: Distributed Parsing with LINGO (TU Delft);
- active deployment of HPSG grammars in (at least) three Dot Coms.

## Background Reading; On-line Resources

- Dan Flickinger, Stephan Open, Jun-ichi Tsujii, Hans Uszkoreit (editors). *Journal of Natural Language Engineering* 6 (1). *Special Issue on Efficient Processing with HPSG: Methods, Systems, Evaluation*.
- Further pointers, on-line at: <http://lingo.stanford.edu/ac100/>.