Computational Grammar Development: What is it good for?

Miriam Butt
(University of Konstanz)
and
Tracy Holloway King (eBay Inc.)

Kathmandu 2012

Outline

- What is a deep grammar and why would you want one?
- XLE and ParGram
- Robustness techniques
- Generation and Disambiguation
- Some Applications:
  - Question-Answering System
  - Murrinh-Patha Translation System
  - Computer Assisted Language Learning (CALL)
  - [Text Summarization]

Deep grammars

- Provide detailed syntactic/semantic analyses
  - HPSG (LinGO, Matrix), LFG (ParGram)
  - Grammatical functions, tense, number, etc.

    Mary wants to leave.

    subj(want~1,Mary~3)
    comp(want~1,leave~2)
    subj(leave~2,Mary~3)
    tense(leave~2,present)
- Usually manually constructed

Why don't people use them?

- Time consuming and expensive to write
  – shallow parsers can be induced automatically from a training set
- Brittle
  – shallow parsers produce something for everything
- Ambiguous
  – shallow parsers rank the outputs
- Slow
  – shallow parsers are very fast (real time)
- Other gating items for applications that need deep grammars

Why should one pay attention now?

New Generation of Large-Scale Grammars:

- Robustness:
  - Integrated Chunk Parsers/Fragment Grammars
  - Bad input always results in some (possibly good) output
- Ambiguity:
  - Integration of stochastic methods
  - Optimality Theory used to rank/pick alternatives
- Speed: comparable to shallow parsers
- Accuracy and information content:
  - far beyond the capabilities of shallow parsers.

XLE at PARC

- Platform for Developing Large-Scale LFG Grammars

- LFG (Lexical-Functional Grammar)
  - Invented in the 1980s
    (Joan Bresnan and Ronald Kaplan)
  - Theoretically stable ⇔ Solid Implementation

- XLE is implemented in C, used with emacs, tcl/tk
- XLE includes a **parser**, **generator** and **transfer** (XFR) component.

Demos:

1) **IBM Watson**
2) **Q&A System**

Project Structure

- Languages: Arabic, Chinese, Danish, English, French, Georgian, German, Hungarian, Irish Gaelic, Indonesian, Japanese, Malagasy, Murrihn-Patha, Norwegian, Polish, Tigrinya, Turkish, Urdu, Welsh, Wolof…
- Theory: Lexical-Functional Grammar
- Platform: XLE
  - parser
  - generator
  - machine translation
- Loose organization: no common deliverables, but common interests.

Grammar Components

Each Grammar contains:

- Annotated Phrase Structure Rules (S --> NP VP)
- Lexicon (verb stems and functional elements)
- Finite-State Morphological Analyzer
- A version of Optimality Theory (OT):

  used as a filter to restrict ambiguities
  and/or parametrize the grammar.

The Parallel in ParGram

- Analyze languages to a degree of abstraction that reflects the common underlying structure (i.e., identiy the subject, the object, the tense, mood, etc.)

- Even at this level, there is usually more than one way to analyze a construction

- The same theoretical analysis may have different possible implementations

- The ParGram Project decides on common analyses and implementations (via meetings and the feature committee)

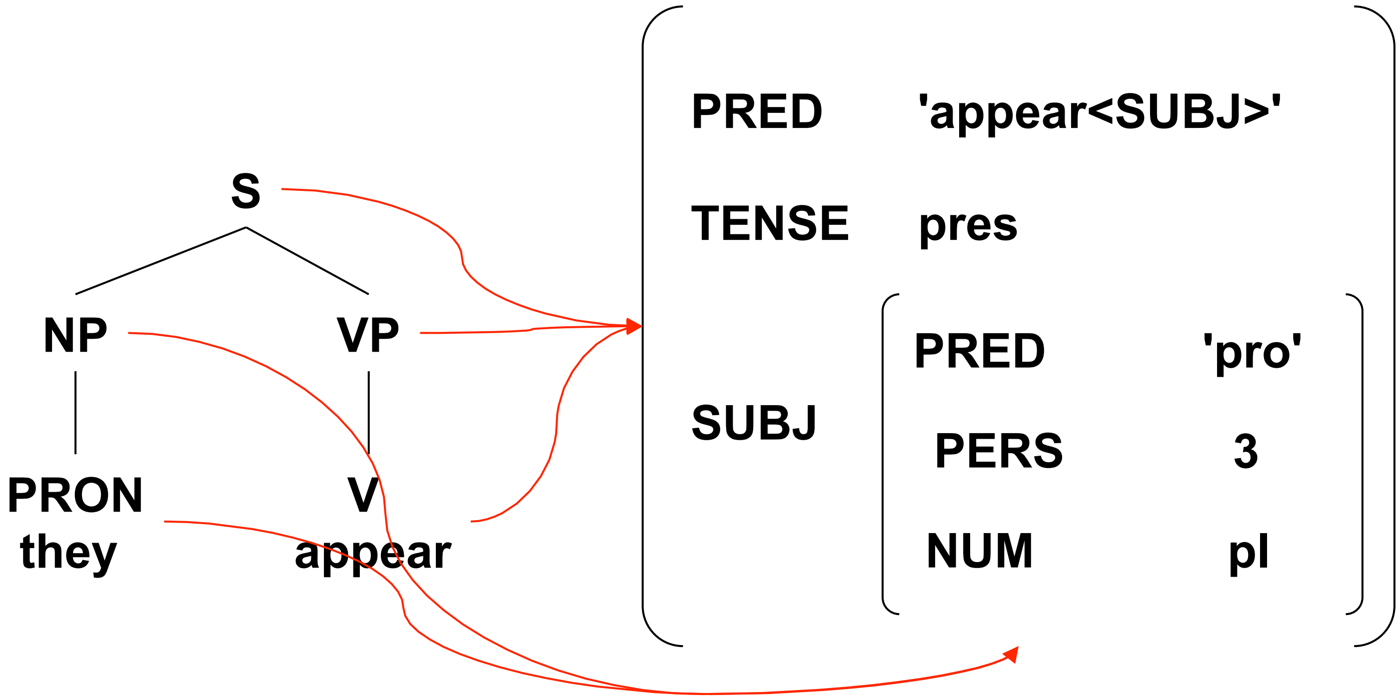
The Parallel in ParGram

- Analyses at the level of c-structure are allowed to differ (variance across languages)
- Analyses at f-structure are held as parallel as possible across languages (crosslinguistic invariance).
- **Theoretical Advantage**: This models the idea of UG.
- **Applicational Advantage**: machine translation is made easier; applications are more easily adapted to new languages (e.g., Kim et al. 2003).

Basic LFG

- Constituent-Structure: tree
- Functional-Structure: Attribute Value Matrix

universal



The Parallel in ParGram

- Sample Structures from the last ParGram Meeting at Bali, Indonesia
- ParGram Structure Comparison, Summer 2012
- Next Meeting will be in Debrecen, Hungary just after the LFG13 conference

Syntactic rules

- Annotated phrase structure rules

```
Category --> Cat1: Schemata1;
             Cat2: Schemata2;
             Cat3: Schemata3.

S --> NP: (^ SUBJ)=!
          (! CASE)=NOM;
      VP: ^=!.
```

Another sample rule

```
                                     "indicate comments"
VP --> V: ^=!;                       "head"
      (NP: (^ OBJ)=!                 "() = optionality"
           (! CASE)=ACC)
      PP*: ! $ (^ ADJUNCT).          "$ = set"
```

VP consists of:

- a head verb
- an optional object
- zero or more PP adjuncts

Lexicon

- Basic form for lexical entries:

```
word Category1 Morphcode1 Schemata1;
       Category2 Morphcode2 Schemata2.

walk V * (^ PRED)='WALK<(^ SUBJ)>';
       N * (^ PRED) = 'WALK' .

girl   N * (^ PRED) = 'GIRL'.

kick  V * { (^ PRED)='KICK<(^ SUBJ)(^ OBJ)>'
           |(^ PRED)='KICK<(^ SUBJ)>'}.

the   D * (^ DEF)=+.
```

Templates

- Express generalizations
  - in the lexicon
  - in the grammar
  - within the template space

No Template

```
girl N * (^ PRED)='GIRL'
         { (^ NUM)=SG
           (^ DEF)
          |(^ NUM)=PL}.
```

With Template

```
TEMPLATE: CN = { (^ NUM)=SG
                 (^ DEF)
                 |(^ NUM)=PL}.
girl N * (^ PRED)='GIRL' @CN.
boy N * (^ PRED)='BOY' @CN.
```

Template example cont.

- Parameterize template to pass in values

```
CN(P) = (^ PRED)='P'
          { (^ NUM)=SG
          (^ DEF)
           |(^ NUM)=PL}.
```

```
girl N * @(CN GIRL).
boy N * @(CN BOY).
```

- Template can call other templates

```
INTRANS(P) = (^ PRED)='P<(^ SUBJ)>'.
TRANS(P) = (^ PRED)='P<(^ SUBJ)(^ OBJ)>'.
OPT-TRANS(P) = { @(INTRANS P) | @(TRANS P) }.
```

Parsing a string

- create-parser grammar1.lfg
- parse ”Hans sleeps”
- Ungrammatical via Unification, etc.

Simple Demo

Outline: Robustness

Dealing with brittleness

- Missing vocabulary
  - you can't list all the proper names in the world
- Missing constructions
  - there are many constructions theoretical linguistics rarely considers (e.g. dates, company names)
- Ungrammatical input
  - real world text is not always perfect
  - sometimes it is really horrendous

Dealing with Missing Vocabulary

- Build vocabulary based on the input of shallow methods
  - fast
  - extensive
  - accurate
- Finite-state morphologies

  falls -> fall +Noun +Pl

  fall +Verb +Pres +3sg
- Build lexical entry on-the-fly from the morphological information

Guessing words

- Use FST guesser if the morphology doesn't know the word

  – Capitalized words can be proper nouns
    Saakashvili -> Saakashvili +Noun +Proper +Guessed

  – *ed* words can be past tense verbs or adjectives
    fumped -> fump +Verb +Past +Guessed
    fumped +Adj +Deverbal +Guessed

Ungrammatical input

- Real world text contains ungrammatical input
  - typos
  - run ons
  - cut and paste errors
- Deep grammars tend to only cover grammatical input
- Two strategies
  - robustness techniques: guesser/fragments
  - disprefered rules for ungrammatical structures (useful for CALL applications)

Harnessing Optimality Theory

- Optimality Theory (OT) allows the statement of preferences and dispreferences.
- In XLE, OT-Marks (annotations) can be added to rules or lexical entries to either prefer or disprefer a certain structure/item.

  `+Mark` = preference

  `Mark` = dispreference

- The strength of (dis)preference can be set variably.

OT Ranking

- **Order of Marks**: Mark3 is preferred to Mark4

  `OPTIMALITYORDER Mark4 Mark3 +Mark2 +Mark1.`

- **NOGOOD Mark:** Marks to the left are always bad. Useful for parametrizing grammar with respect to certain domains

  `OPTIMALITYORDER Mark4 NOGOOD Mark3 +Mark2 +Mark1.`

- **STOPPOINT Mark:** slowly increases the search space of the grammar if no good solution can be found (multipass grammar)

  `OPTIMALITYORDER Mark4 NOGOOD Mark3 STOPPOINT Mark2 STOPPOINT Mark1.`

Rule Annotation (O-Projection)

- Common errors can be coded in the rules

  mismatched subject-verb agreement

  ```
  Verb3Sg = { (^ SUBJ PERS) = 3
              (^ SUBJ  NUM) = sg
            | @(OTMARK BadVAgr) }
  ```

- Disprefer parses of ungrammatical structure
  - tools for grammar writer to rank rules
  - two+ pass system

Demo
Robustness

grammar2.lfg (OT Marks)

english.lfg (FST Morphology, Fragments)

Generation Outline

- Why generate?
- Generation as the reverse of parsing
- Constraining generation (OT)
- The generator as a debugging tool
- Generation from underspecified structures

Why generate?

- Machine translation

  Lang1 string -> Lang1 fstr -> Lang2 fstr -> Lang2 string
- Sentence condensation

  Long string -> fstr -> smaller fstr -> new string
- Question answering
- Grammar debugging

Generation: just reverse the parser

- XLE uses the same basic grammar to parse and generate
  - Parsing: string to analysis
  - Generation: analysis to string

- Input to Generator is the f-structure analysis

- Formal Properties of LFG Generation:
  - Generation produces Context Free Languages
  - LFG generation is a well-understood formal system (decidability, closure).

Generation: just reverse the parser

- Advantages
  - maintainability
  - write rules and lexicons once
- But
  - special generation tokenizer
  - different OT ranking

Restricting Generation

- Do not always want to generate all the possibilities that can be parsed

- Put in special OT marks for generation to block or prefer certain strings
  - fix up bad subject-verb agreement
  - only allow certain adverb placements
  - control punctuation options

- GENOPTIMALITYORDER
  - special ordering for OT generation marks that is kept separate from the parsing marks
  - serves to parametrize the grammar (parsing vs. generation)

Generation tokenizer

- White space
  - Parsing: multiple white space becomes a single TB

    John   appears. -> John TB appears TB . TB

  - Generation: single TB becomes a single space (or nothing)

    John TB appears TB . TB -> John appears.

    *John   appears   .

Generation morphology

- Suppress variant forms
  - Parse both *favor* and *favour*
  - Generate only one

Ungrammatical input

- Linguistically ungrammatical
  - They walks.
  - They ate banana.
- Stylistically ungrammatical
  - No ending punctuation: They appear
  - Superfluous commas: John, and Mary appear.
  - Shallow markup: [NP John and Mary] appear.

Too many options

- All the generated options can be linguistically valid, but too many for applications

- Occurs when more than one string has the same, legitimate f-structure

- PP placement:
  - In the morning I left.   I left in the morning.

Example: Prefer initial PP

```
S --> (PP: @ADJUNCT)
        NP: @SUBJ;
       VP.
VP --> V
        (NP: @OBJ)
        (PP: @ADJUNCT @(OT-MARK GenGood)).
GENOPTIMALITYORDER NOGOOD +GenGood.
```

parse: In the morning they appear.

generate: without OT: In the morning they appear.
They appear in the morning.

with OT: They appear in the morning.

Generation commands

- XLE command line:
  - regenerate "They appear."
  - generate-from-file my-file.pl
  - (regenerate-from-directory, regenerate-testfile)
- F-structure window:
  - commands: generate from this fs
- Debugging commands
  - regenerate-morphemes

Underspecified Input

- F-structures provided by applications are not perfect
  - may be missing features
  - may have extra features
  - may simply not match the grammar coverage
- Missing and extra features are often systematic
  - specify in XLE which features can be added and deleted
- Not matching the grammar is a more serious problem

Creating Paradigms

- Deleting and adding features within one grammar can produce paradigms
- Specifiers:
  - set-gen-adds remove "SPEC"

    set-gen-adds add "SPEC DET DEMON"
  - regenerate "NP: boys"

{ the | those | these | } boys

etc.

Summary: Generation and Reversibility

- XLE parses and generates on the same grammar
  - faster development time
  - easier maintenance
- Minor differences controlled by:
  - OT marks
  - FST tokenizers

Demo Generator

Applications — Beyond Parsing

- Machine translation
- Sentence condensation
- Computer Assisted Language Learning
- Knowledge representation

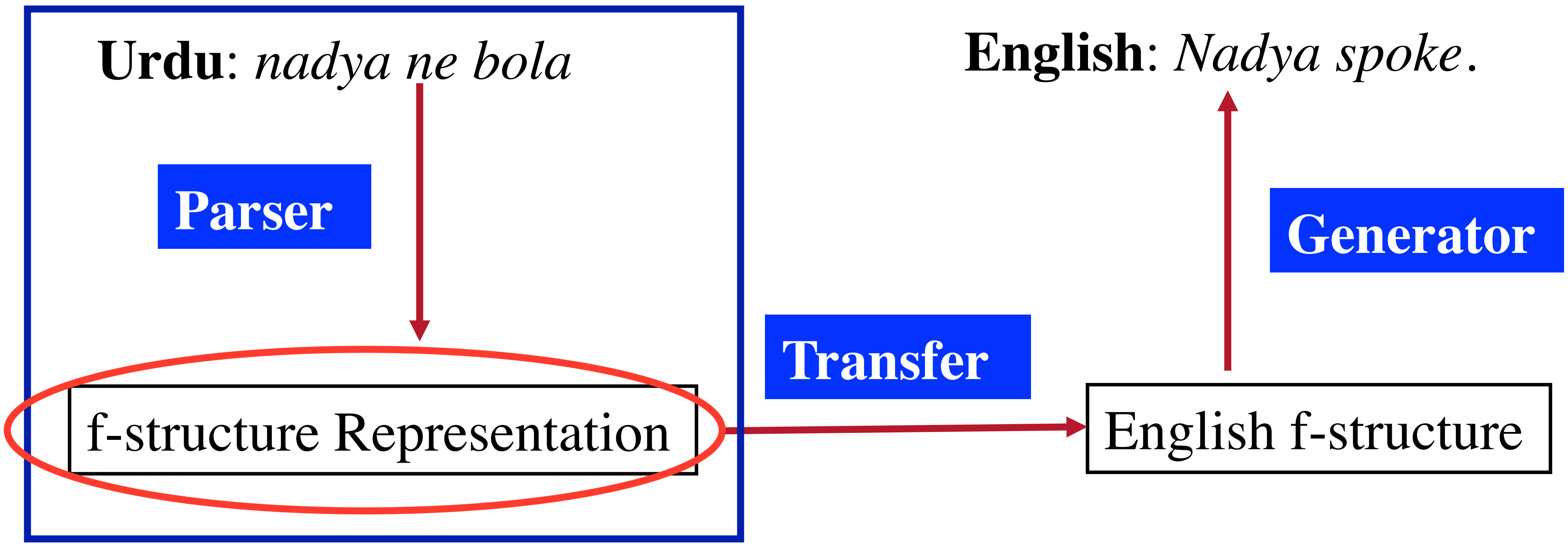
Machine Translation

- The Transfer Component
- Transferring features/F-structures
  - adding information
  - deleting information
- Examples

Basic Idea

- Parse a string in the source language
- Rewrite/transfer the f-structure to that of the target language
- Generate the target string from the transferred f-structure

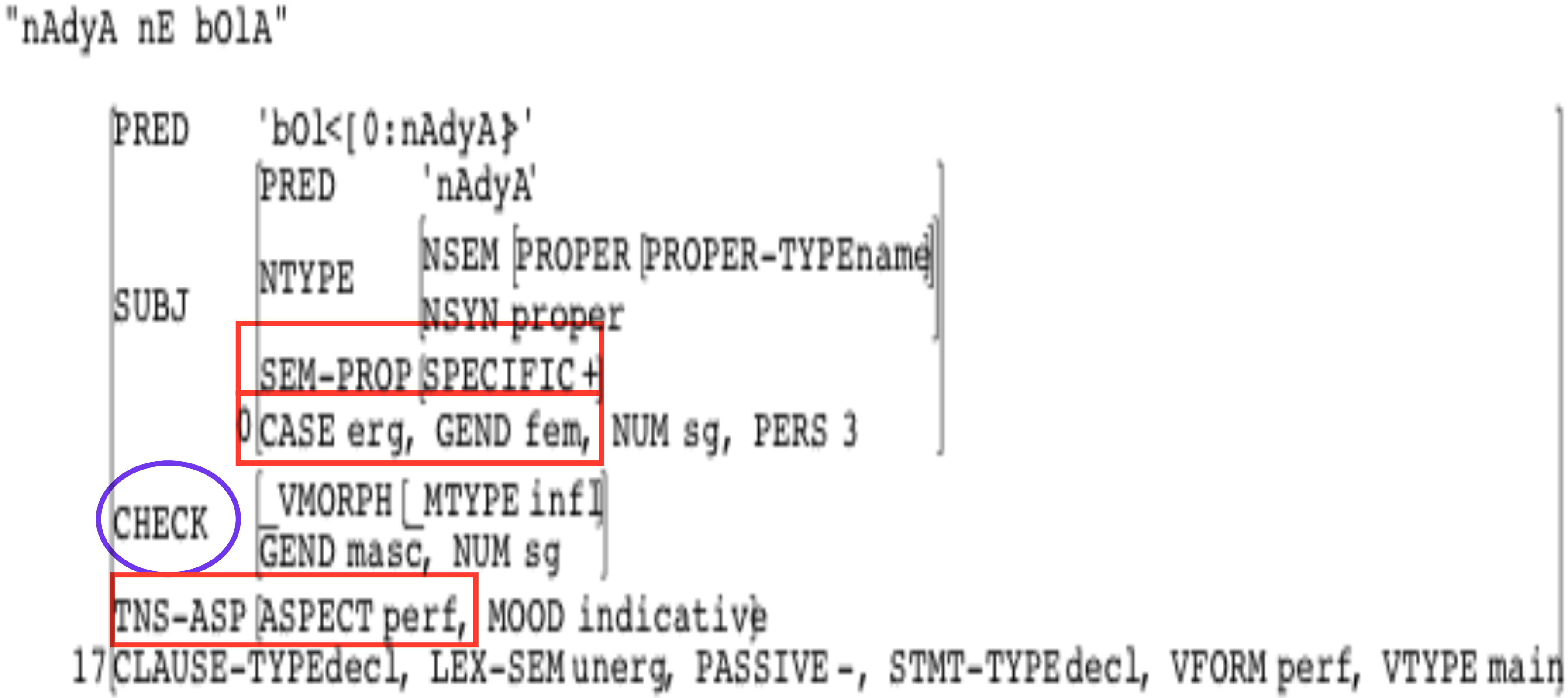
Urdu to English MT



from Urdu structure …

parse: *nadya ne bola*

Urdu f-structure



… to English structure

Urdu f-structure → **Transfer** → English f-structure → **Generator**

```
 [PRED      'speak<[3:Nadya]>'
 |          [PRED      'Nadya'
 |          |          [NSEM 6[PROPER 7[PROPER-TYPE name]]
 |          |NTYPE    5[NSYN  proper
 |SUBJ      |GEND-SEM  [=[<a:2  female>]]
 |          |HUMAN     [=[<a:2  +>]]
 |         3[CASE nom, NUM sg, PERS 3
 |TNS-ASP  9[MOOD indicative, PERF -, PROG -, TENSE past]
0[CLAUSE-TYPE decl, PASSIVE -, STMT-TYPE decl, VTYPE main]
```

English:

Nadya spoke.

The Transfer Component

- Prolog based
- Small hand-written set of transfer rules
  - Obligatory and optional rules (possibly multiple output for single input)
  - Rules may add, delete, or change parts of *f*-structures
- Transfer operates on packed input and output
- Developer interface: Component adds new menu features to the output windows:
  - transfer this f-structure
  - translate this f-structure
  - reload rules

Sample Transfer Rules

Template

```
verb_verb(%Urdu, %English) ::
   PRED(%X, %Urdu), +VTYPE(%X,%main) ==>
                              PRED(%X,% English).
```

Rules

```
verb_verb(pI,drink).
verb_verb(dEkH,see).
verb_verb(A,come).
```

```
%perf plus past, get perfect past
   ASPECT(%X,perf), + TENSE(%X,past) ==>
            PERF(%X,+), PROG(%X,-).
%only perf, get past
   ASPECT(%X,perf) ==> TENSE(%X,past), PERF(%X,-),
                        PROG(%X,-).
```

Generation

- Use of *generator as filter* since transfer rules are independent of grammar
  – not constrained to preserve grammaticality

- Robustness techniques in generation:
  – Insertion/deletion of features to match lexicon
  – For fragmentary input from robust parser grammatical output guaranteed for separate fragments

Adding features

- English to French translation:
  - English nouns have no gender
  - French nouns need gender
  - Solution: have XLE add gender

    the French morphology will control the value
- Specify additions in configuration file (xlerc):
  - set-gen-adds add "GEND"
  - can add multiple features:

    set-gen-adds add "GEND CASE PCASE"
  - XLE will *optionally* insert the feature

Note: Unconstrained additions make generation undecidable

Example

The cat sleeps. -> Le chat dort.

[PRED 'dormir<SUBJ>'
SUBJ [PRED 'chat'
NUM sg
SPEC def]
TENSE present]

[PRED 'dormir<SUBJ>'
SUBJ [PRED 'chat'
NUM sg
GEND masc
SPEC def]
TENSE present]

Deleting features

- French to English translation
  - delete the GEND feature
- Specify deletions in xlerc
  - set-gen-adds remove "GEND"
  - can remove multiple features

    set-gen-adds remove "GEND CASE PCASE"
  - XLE *obligatorily* removes the features

    no GEND feature will remain in the f-structure
  - if a feature takes an f-structure value, that f-structure is also removed

Changing values

- If values of a feature do not match between the input f-structure and the grammar:
  - delete the feature and then add it
- Example: case assignment in translation
  - set-gen-adds remove "CASE"
    set-gen-adds add "CASE"
  - allows dative case in input to become accusative
    e.g., exceptional case marking verb in input language but regular case in output language

Machine Translation

MT Demo – Murrinh Patha

Computer Assisted Language Learning (CALL) Outline

- Goals
- Method
- Augmenting the English ParGram Grammar via OT Marks
- Generating Correct Output

XLE and CALL

- Goal: Use large-scale intelligent grammars to assist in grammar checking
  - identify errors in text by language learners
  - provide feedback as to location and type of error
  - generate back correct example
- Method: Adapt English ParGram grammar to deal with errors in the learner corpus

XLE CALL system method

- Grammar: Introduce special `UNGRAMMATICAL` feature at f-structure for feedback as to the type of error
- Parse CALL sentence
- Generate back possible corrections
- Evaluated on developed and unseen corpus
  i. accuracy of error detection
  ii. value of suggestions or possible feedback
  iii. range of language problems/errors covered
  iv. speed of operation

Adapting the English Grammar

- The standard ParGram English grammar was augmented with:
  - OT marks for ungrammatical constructions
  - Information for feedback: Example: Mary happy.

    UNGRAMMATICAL {missing-be}

    top level f-structure

- Parametrization of the generator to allow for corrections based on ungrammatical input.

F-structure: Mary happy.

```
"Mary happy."

   PRED            'be<[22:happy]>[0:Mary]'
                   PRED   'Mary'
                          NSEM [PROPER [PROPER-TYPE name]]
   SUBJ            NTYPE
                          NSYN proper
                 0 CASE nom, GEND-SEM female, HUMAN +, NUM sg, PERS 3
                   PRED   'happy<[0:Mary]>'
   XCOMP           SUBJ   [0:Mary]
                22 ATYPE predicative, DEGREE positive
   TNS-ASP         MOOD indicative, PERF -_, PROG -_, TENSE pres
   UNGRAMMATICAL {missing-be}
73 CLAUSE-TYPE decl, PASSIVE -, STMT-TYPE decl, VTYPE copular
```

Example modifications

- Missing copula (Mary happy.)
- No subj-verb agreement (The boys leaves.)
- Missing specifier on count noun (Boy leaves.)
- Missing punctuation (Mary is happy)
- Bad adverb placement (Mary quickly leaves.)
- Non-fronted wh-words (You saw who?)
- Missing *to* infinitive (I want disappear.)

Using OT Marks

- OT marks allow one analysis to be prefered over another
- The marks are introduced in rules and lexical entries

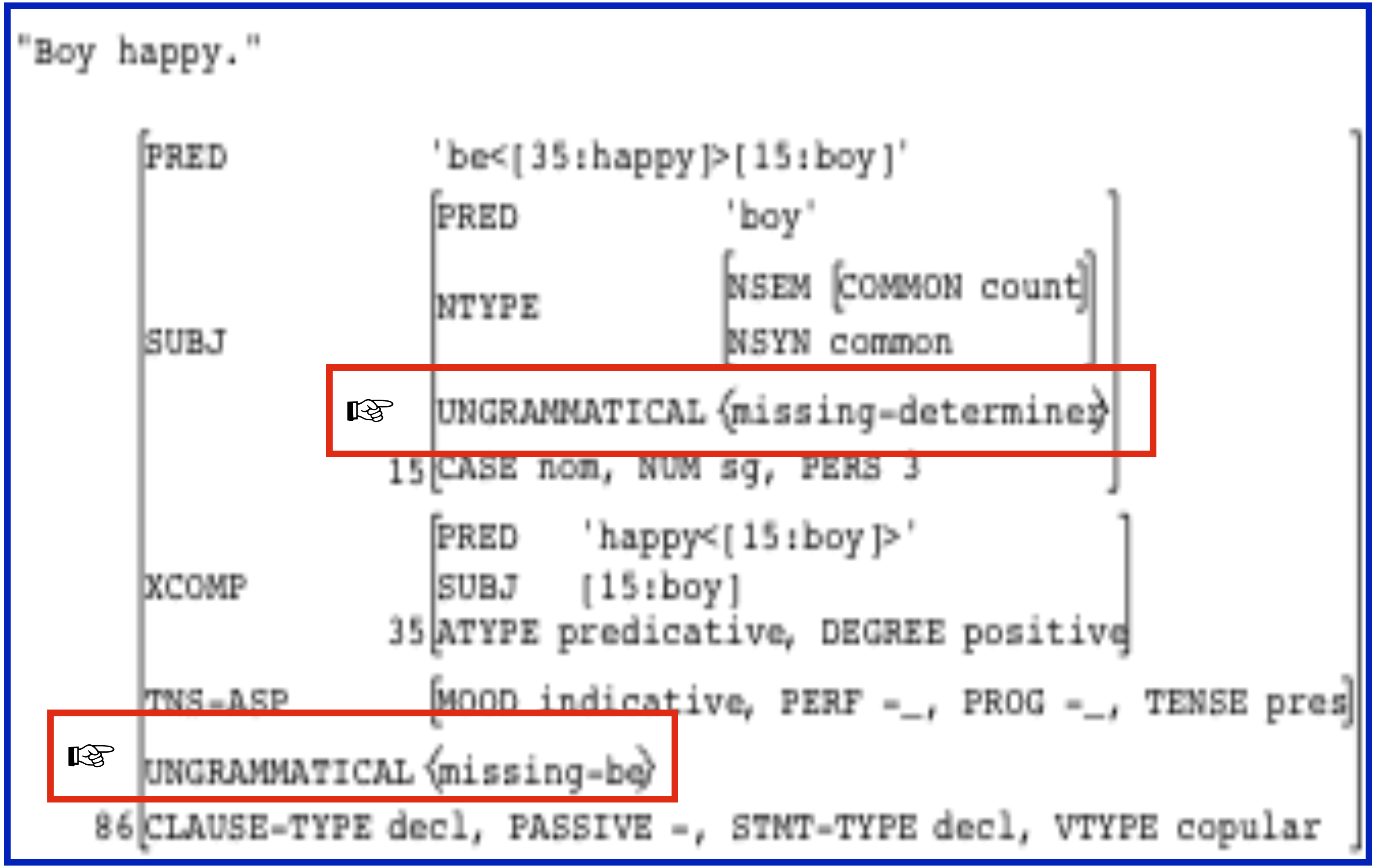
    @(OT-MARK ungrammatical)

- The parser is given a ranking of the marks
- Only the top ranked analyses appear

OT Marks in the CALL grammar

- A correct sentence triggers no marks
- A sentence with a known error triggers a mark ungrammatical
- A sentence with an unknown error triggers a mark fragment
- no mark < ungrammatical < fragment
  - the grammar first tries for no mark
  - then for a known error
  - then a fragment if all else fails

F-structure: Boy happy.



Generation of corrections

- Remember that XLE allows the generation of correct sentences from ungrammtical input.
- Method:
  - Parse ungrammatical sentence
  - Remove UNGRAMMATICAL feature for generation
  - Generate from stripped down ungrammatical f-structure

Underspecified Generation

- XLE generation from an underspecified f-structure (information has been removed).
- Example: generation from an f-structure without tense/aspect information.

John sleeps (w/o TNS-ASP)

→ All tense/aspect variations

```
John
  {  { will be
       |was
       |is
       |{has|had} been}
    sleeping
   |{{will have|has|had}|} slept
   |sleeps
   |will sleep}
```

CALL Generation example

- parse "Mary happy."
  generate back:
  Mary is happy.

- parse "boy arrives."
  generate back:
  { This | That | The | A } boy arrives.

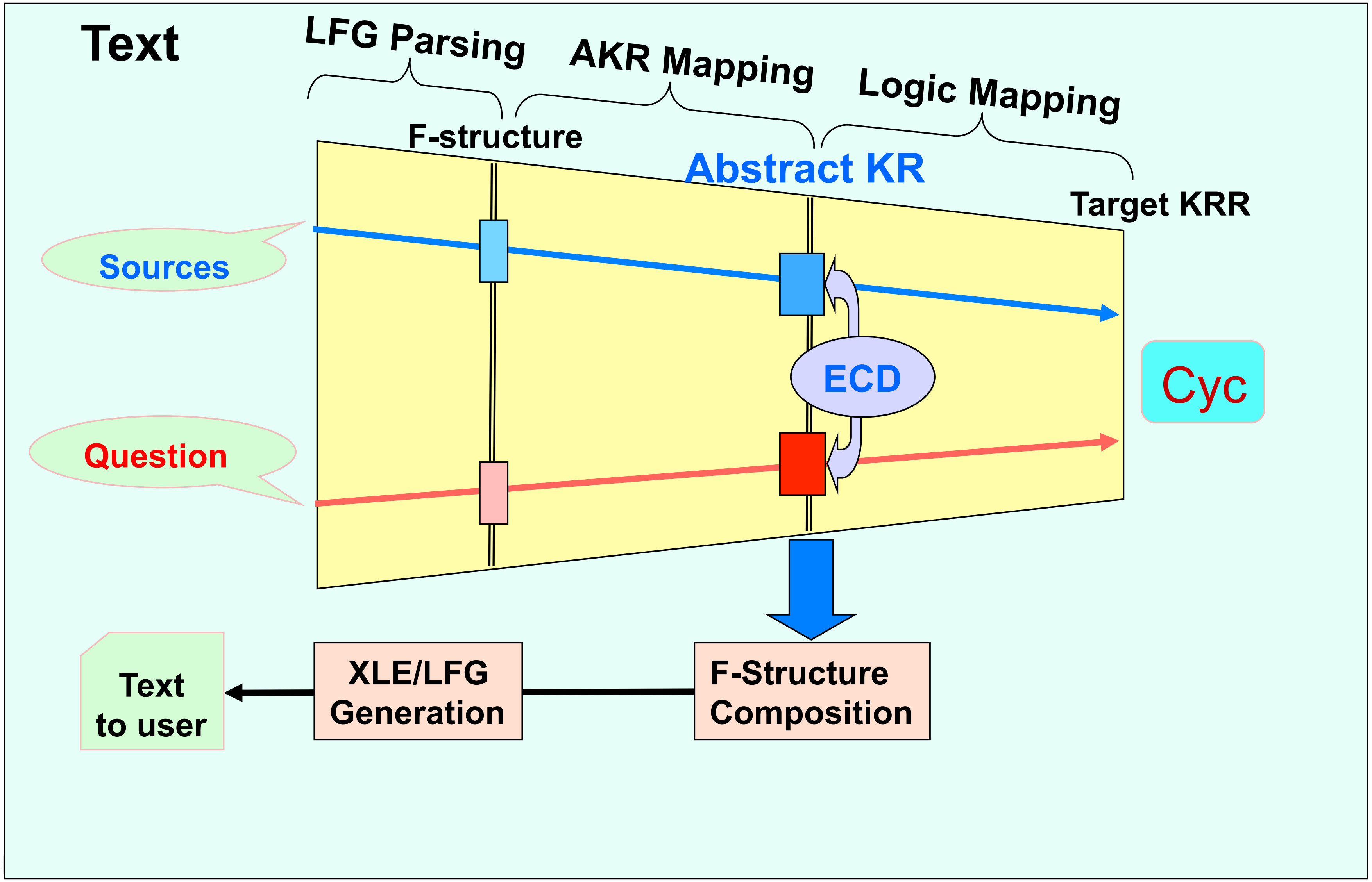
CALL evaluation and conclusions

- Preliminary Evaluation promising:
  - Word 10 out of 50=20% (bad user feedback)
  - XLE 29 out of 50=58% (better user feedback)
- Unseen real life student production
  - Word 5 out of 11 (bad user feedback)
  - XLE 6 out 11 (better user feedback)

Knowledge Representation

- From Syntax to Semantics
- From Semantics to Knowledge Representation
- Text Analysis
- Question/Answering

Text – KR – Text



Text
LFG Parsing
AKR Mapping
Logic Mapping
F-structure
Abstract KR
Target KRR
Sources
Question
ECD
Cyc
Text
to user
XLE/LFG
Generation
F-Structure
Composition

Rewrite Rules for KR mapping

All operate on packed representations:

- F-structure to semantics
  - Semantic normalization, verbnet roles, wordnet senses, lexical class information
- Semantics to Abstract Knowledge Representation (AKR)
  - Separating conceptual, contextual & temporal structure
- AKR to F-structure
  - For generation from KR
- Entailment & contradiction detection rules
  - Applied to AKR

Semantic Representation
Someone failed to pay

```
in_context(t, past(fail22))
in_context(t, role(Agent, fail22, person1))
in_context(t, role(Predicate, fail22, ctx(pay19)))
in_context(ctx(pay19), cardinality(person1, some))
in_context(ctx(pay19), role(Agent, pay19, person1))
in_context(ctx(pay19), role(Recipient, pay19, implicit_arg94))
in_context(ctx(pay19), role(Theme, pay19, implicit_arg95))

lex_class(fail22, [vnclass(unknown), wnclass(change),
                   temp-rel, temp_simul, impl_pn_np, prop-attitude])
lex_class(pay19, [vnclass(unknown), wnclass(possession)])),
word(fail22, fail, verb, 0, 22, t, [[2505082], [2504178], …, [2498138]])
word(implicit_arg:94, implicit, implicit, 0, 0, ctx(pay19), [[1740]])
word(implicit_arg:95, implicit, implicit, 0, 0, ctx(pay19), [[1740]])
word(pay19, pay, verb, 0, 19, ctx(pay19),
            [[2230669], [1049936], …, [2707966]])
word(person1, person, quantpro, 0, 1, ctx(pay19),
            [[7626, 4576, …, 1740]])
```

AKR

Someone failed to pay

Conceptual Structure:

subconcept(fail22, [[2:2505082], [2:2504178], …, [2:2498138]])
role(Agent, fail22, person1)
subconcept(person1, [[1:7626, 1:4576, …, 1:1740]])
role(cardinality_restriction, person1, some)
role(Predicate, fail22, ctx(pay19))
subconcept(pay19, [[2:2230669], [2:1049936], …, [2:2707966]])
role(Agent, pay19, person1)

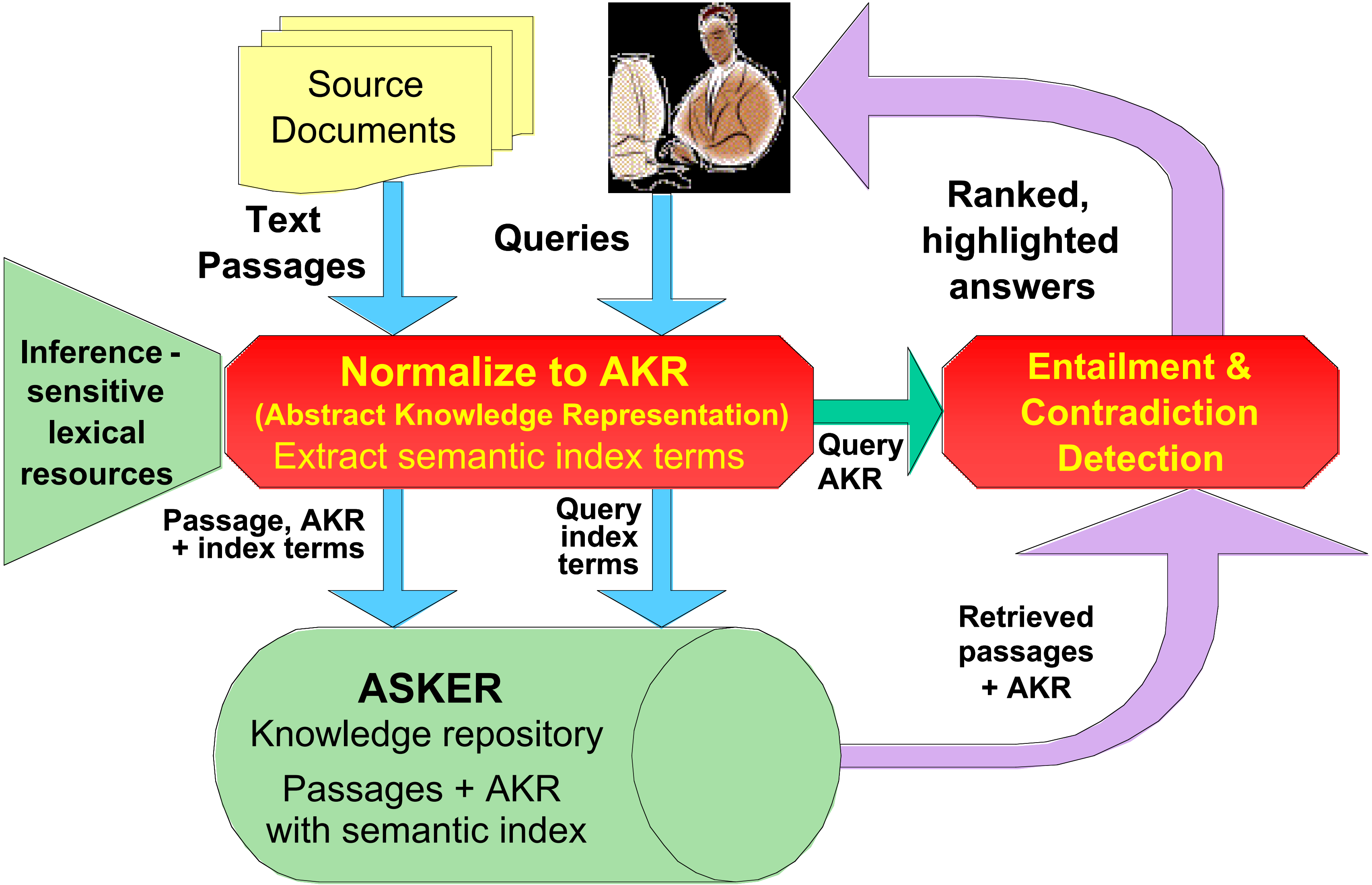
Contextual Structure:

context(t) context(ctx(pay19))
context_lifting_relation(antiveridical, t, ctx(pay19))
context_relation(t, ctx(pay19), Predicate(fail22))
instantiable(fail22, t)
uninstantiable(pay19, t)
instantiable(pay19, ctx(pay19))

Temporal Structure:

temporalRel(startsAfterEndingOf, Now, fail22)
temporalRel(startsAfterEndingOf, Now, pay19)

Semantic Search Architecture



Entailment & Contradiction Detection

1. Map texts to packed AKR
2. Align concept & context terms between AKRs
3. Apply entailment & contradiction rules to aligned AKRs
   1. eliminate entailed facts
   2. flag contradictory facts
4. Inspect results
   1. Entailment = all query facts eliminated
   2. Contradiction = any contradiction flagged
   3. Unknown = otherwise

- Properties
  – Combination of positive aspects of graph matching (alignment) and theorem proving (rewriting)
  – Ambiguity tolerant

ECD: Illustrative Example

- "*A little girl disappeared*" entails "*A child vanished*"

- A trivial example
  – Could be handled by a simpler approach (e.g. graph matching)
  – Used to explain basics of ECD approach

Representations

AKR: *A little girl disappeared.*	➔	AKR: *A child vanished*
context(t), instantiable(disappear4, t) instantiable(girl3, t) temporalRel(startsAfter, Now, disappear4) role(Theme, disappear4, girl3) role(cardinality_restriction, girl3, sg) subconcept(disappear4, [[422658], …, [220927]]) subconcept(girl3, [[9979060…1740], [9934281…9771976…1740], …, [9979646…1740]])		context(t), instantiable(vanish2, t) instantiable(child1, t) temporalRel(startsAfter, Now, vanish2) role(Theme, vanish2, child1) role(cardinality_restriction, child1, sg) subconcept(vanish2, [[422658], …, [2136731]]) subconcept(child1, [[9771320, …1740], [9771976, …1740], …, [9772490, …, 1740]])

Contextual, temporal and conceptual subsumption indicates entailment

Alignment

- Align terms based on conceptual overlap

```
***TABLE of possible Query-Passage alignments ***

vanish2    [1.0–disappear4,   0.0–little1,    0.0–girl3]
child1     [0.78–girl3,   0.0–little1,   0.0–disappear4]
t          [1.0–t]
```

```
subconcept(vanish2,
              [[422658],  …,  [2136731]])
subconcept(disappear4,
              [[422658],  …,  [220927]])
```

- Determined by subconcepts
  - Degree of hypernym overlap

vanish:2 = disappear:4 on sense 1

child:1 ⊂ girl:3 on sense 2

```
subconcept(child1,
              [[9771320,  …1740],
               [9771976,  …1740],
               …,  [9772490,  …,  1740]])
subconcept(girl3,
              [[9979060…1740],
               [9934281…9771976…1740],
               …,   [9979646…1740]])
```

Impose Alignment & Label Facts

girl3 // child1
disappear4 // vanish2

P-AKR: *A little girl disappeared.*

P:context(t)
P:instantiable(vanish2, t)
P:instantiable(child1, t)
P:temporalRel(startsAfter, Now, vanish2)
P:role(Theme, vanish2, child1)
P:role(cardinality_restriction, child1, sg)
P:role(subsective, child1, little1)
P:subconcept(little1, [[1443454…], …])
P:subconcept(vanish2,
[[422658], …, [220927]])
P:subconcept(child1,
[[9979060…1740],
[9934281…9771976…1740],
…, [9979646…1740]])

Q-AKR: *A child vanished*

Q:context(t),
Q:instantiable(vanish2, t)
Q:instantiable(child1, t)
Q:temporalRel(startsAfter, Now, vanish2)
Q:role(Theme, vanish2, child1)
Q:role(cardinality_restriction, child1, sg)
Q:subconcept(vanish2,
[[422658], …, [2136731]])
Q:subconcept(child1,
[[9771320, …1740],
[9771976, …1740],
…, [9772490, …, 1740]])

- Combined P-AKR and Q-AKR used as input to entailment and contradiction transfer rules

Entailment & Contradiction Rules

- Packed rewrite rules that
  - Eliminate Q-facts that are entailed by P-facts
  - Flag Q-facts that are contradicted by P-facts
- Rule phases
  - Preliminary concept subsumption
  - Refine concept subsumption via role restrictions
  - Entailments & contradictions from instantiable / uninstantiable facts
  - Entailments & contradictions from other relations

Preliminary Subsumption Rules

- Example rules:

e.g. "girl" and "child"

Q:subconcept(%Sk, %QConcept)
P:subconcept(%Sk, %PConcept)
{%QConcept ⊂ %PConcept}
==>
prelim_more_specific(%Sk, P).

e.g. "disappear" and "vanish"

Q:subconcept(%Sk, %QConcept)
P:subconcept(%Sk, %PConcept)
{%QConcept = %PConcept}
==>
prelim_more_specific(%Sk, mutual).

- Apply to subconcept facts to give:

prelim_more_specific(vanish2, mutual)
prelim_more_specific(child1, P)

Role Restriction Rules

- Example rules:

  "little girl" more specific than *"child"*

```
prelim_more_specific(%Sk,   %PM)
{ member(%PM,   [P,   mutual]) }
P:role(%%,   %Sk,   %%)
-Q:role(%%,   %Sk,   %%)
==>
more_specific(%Sk,   P).
```

- Rules apply to give:

```
more_specific(child1,   P)
more_specific(vanish2,   P)
```

Instantiation Rules

- Remove entailed instantiabilities and flag contradictions:

Q-instantiability entailed

```
more_specific(%Sk,   P),
P:instantiable(%Sk,  %Ctx)
Q:instantiable(%Sk,  %Ctx)
==>
0.
```

Q-uninstantiability contradicted

```
more_specific(%Sk,  P),
P:instantiable(%Sk, %Ctx)
Q:uninstantiable(%Sk, %Ctx)
==>
contradiction.
```

ECD Summary

- Combination of graph matching and inference on deep representations
- Use of transfer system allows ECD on packed / ambiguous representations
  - No need for early disambiguation
  - Passage and query effectively disambiguate each other
- ECD rules currently geared toward very high precision detection of entailments & contradictions

Semantic/AKR Indexing

- ECD looks for inferential relations between a question and a candidate answer
- Semantic/AKR search retrieves candidate answers from a large database of representations
- Text representations are indexed by
  - Concepts referred to
  - Selected role relations
- Basic retrieval from index
  - Find text terms more specific than query terms
  - Ensure query roles are present in retrieved text

Semantic/AKR Indexing

- Semantic/AKR search retrieves candidate answers from a large database of representations
  - Simple relevance retrieval (graph/concept subsumption)
    A girl paid. Did a child pay?
    - Text term more specific than query term
- Inferentially enhanced retrieval
  - Recognizing when text terms need to be less specific than query
    Someone forgot to pay. Did everyone pay?
    - Text term is less specific than query term
  - Looser matching on roles present in text
- Retrievals are then fed to ECD module

Semantic Lexical Resources

- Semantics/KR applications require additional lexical resources
  - use existing resources when possible
  - XLE transfer system incorporates basic database to handle large lexicons efficiently
- Unified (semantic) lexicon
  - Ties existing resources to XLE lexicons (WordNet, VerbNet, ontologies, …)
  - Additional annotation of lexical classes (*fail* vs *manage*, *believe* vs *know*)
  - Used in mapping f-structures to semantics/AKR

- **Demo**
- **AKR and ECD**

Advancing Open Text Semantic Analysis

- Deeper, more detailed linguistic analysis
  - Roles, concepts, normalization of f-structures
- Canonicalization into tractable KR
  - (un)instantiability
  - temporal relations
- Ambiguity enabled semantics and KR
  - Common packing mechanisms at all levels of representation
  - Avoid errors from premature disambiguation

Driving force: Entailment & Contradiction Detection (ECD)

ECD and Maintaining Text Databases

Tip 27057

Problem: Left cover damage

Cause: The left cover safety cable is breaking, allowing the left cover to pivot too far, breaking the cover.

Solution: Remove the plastic sleeve from around the cable. Cutting the plastic off of the cable makes the cable more flexible, which prevents cable breakage. Cable breakage is a major source of damage to the left cover.

Tip 27118

Problem: The current safety cable used in the 5100 Document Handler fails prematurely causing the Left Document Handler Cover to break.

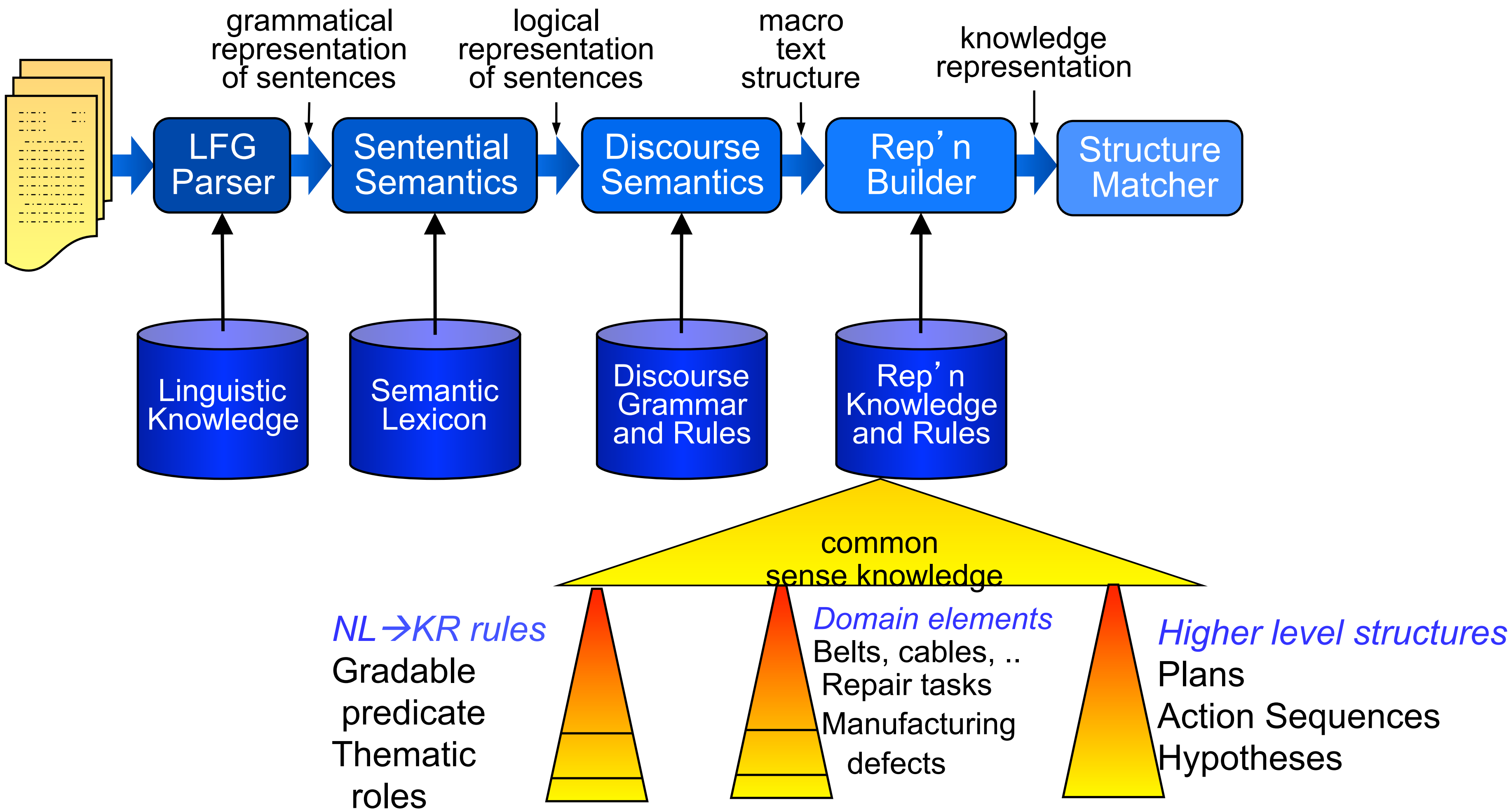
Cause: The plastic jacket made the cable too stiff. This causes stress to be concentrated on the cable ends, where it eventually fails.

Solution: When the old safety cable fails, replace it with the new one [12K1981], which has the plastic jacket shortened.

Maintain quality of text database by identifying areas of redundancy and conflict between documents

Deep, canonical, ambiguity-enabled semantic processing is needed to detect entailments & contradictions like these.

Architecture for Document ECD



XLE: Summary

- XLE
  - parser (tree and dependency output)
  - generator (reversible parsing grammar)
  - powerful, efficient and flexible rewrite system
- Grammar engineering makes deep grammars feasible
  - robustness techniques
  - integration of shallow methods
- Ordered rewrite system to manipulate grammar output

XLE: Applications

- Many current applications can use shallow grammars
- Fast, accurate, broad-coverage deep grammars enable extensions to existing applications and new applications
  - semantic search
  - summarization/condensation
  - CALL and grammar checking
  - entity and entity relation detection
  - machine translation

XLE: Applications

- Powerful methods that allow innovative solutions:
  - Integration of shallow methods (chunking, statistical information)
  - Integration of optimality marks
  - rewrite system
  - innovative semantic representation

Contact information

- Miriam Butt
  miriam.butt@uni-konstanz.de
  http://ling.uni-konstanz.de/pages/home/butt

- Tracy Holloway King
  thking@microsoft.com
  http://www.parc.com/thking

- Many of the publications in the bibliography are available from our websites.

- Information about XLE (including link to documentation):
  http://www.parc.com/istl/groups/nltt/xle/default.html

Bibliography

XLE Documentation: http://www2.parc.com/isl/groups/nltt/xle/doc/xle_toc.html

Butt, M., T.H. King, M.-E. Niño, and F. Segond. 1999. *A Grammar Writer's Cookbook*. Stanford University: CSLI Publications.

Butt, Miriam and Tracy Holloway King. 2003. Grammar Writing, Testing, and Evaluation. In A. Farghaly (ed.) *Handbook for Language Engineers*. CSLI Publications. pp. 129-179.

Butt, M., M. Forst, T.H. King, and J. Kuhn. 2003. The Feature Space in Parallel Grammar Writing. *ESSLLI 2003 Workshop on Ideas and Strategies for Multilingual Grammar Development*.

Butt, M., H. Dyvik, T.H. King, H. Masuichi, and C. Rohrer. 2002. The Parallel Grammar Project. *Proceedings of COLING2002, Workshop on Grammar Engineering and Evaluation* pp. 1-7.

Butt, M., T.H. King, and J. Maxwell. 2003. Productive encoding of Urdu complex predicates in the ParGram Project. In *Proceedings of the EACL03: Workshop on Computational Linguistics for South Asian Languages: Expanding Synergies with Europe*. pp. 9-13.

Butt, M. and T.H. King. 2003. Complex Predicates via Restriction. In *Proceedings of the LFG03 Conference*. CSLI On-line Publications. pp. 92-104.

Cetinoglu, O. and K.Oflazer. 2006. Morphology-Syntax Interface for Turkish LFG. Proceedings of COLING/ ACL2006.

Crouch, D. 2005. Packed rewriting for mapping semantics to KR. In *Proceedings of the International Workshop on Computational Semantics*.

Crouch, D. and T.H. King. 2005. Unifying lexical resources. In *Proceedings of the Verb Workshop*. Saarbruecken, Germany.

Crouch, D. and T.H. King. 2006. Semantics via F-structure rewriting. In *Proceedings of LFG06*. CSLI On-line Publications.

Frank, A., T.H. King, J. Kuhn, and J. Maxwell. 1998. Optimality Theory Style Constraint Ranking in Large-Scale LFG Grammars *Proceedings of the LFG98 Conference*. CSLI On-line Publications.

Frank, A. et al. 2006. Question Answering from Structured Knowledge Sources. *Journal of Applied Logic, Special Issue on Questions and Answers: Theoretical and Applied Perspectives*.

Kaplan, R., T.H. King, and J. Maxwell. 2002. Adapting Existing Grammars: The XLE Experience. *Proceedings of COLING2002, Workshop on Grammar Engineering and Evaluation,* pp. 29-35.

Kaplan, Ronald M. and Jürgen Wedekind. 2000. LFG generation produces context-free languages. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING2000), Saarbrücken*.

Kaplan, R.M., S. Riezler, T. H. King, J. T. Maxwell III, A. Vasserman, R. Crouch. 2004. Speed and Accuracy in Shallow and Deep Stochastic Parsing. In *Proceedings of the Human Language Technology Conference and the 4th Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL'04), Boston, MA.*

Kaplan, R. M. and P. Newman. 1997. Lexical resource reconciliation in the Xerox Linguistic Environment. In *Computational environments for grammar development and linguistic engineering*, pp. 54-61. Proceedings of a workshop sponsored by the Association for Computational Linguistics, Madrid, Spain, July 1997.

Kaplan, R. M., K. Netter, J. Wedekind, and A. Zaenen. 1989. Translation by structural correspondences. In *Proceedings of the 4th Meeting of the EACL*, pp. 272-281. University of Manchester: European Chapter of the Association for Computational Linguistics. Reprinted in Dalrymple et al. (editors), *Formal Issues in Lexical-Functional Grammar*. CSLI, 1995.

Karttunen, L. and K. R. Beesley. 2003. *Finite-State Morphology*. CSLI Publications.

Kay, M. 1996. Chart Generation. *Proceedings of the ACL* 1996, 200-204.

Khader, R. 2003. *Evaluation of an English LFG-based Grammar as Error Checker*. UMIST MSc Thesis, Manchester.

Kim, R., M. Dalrymple, R. Kaplan, T.H. King, H. Masuichi, and T. Ohkuma. 2003. Multilingual Grammar Development via Grammar Porting. *ESSLLI 2003 Workshop on Ideas and Strategies for Multilingual Grammar Development*.

King, T.H. and R. Kaplan. 2003. Low-Level Mark-Up and Large-scale LFG Grammar Processing. *On-line Proceedings of the LFG03 Conference*.

King, T.H., S. Dipper, A. Frank, J. Kuhn, and J. Maxwell. 2000. Ambiguity Management in Grammar Writing. *Linguistic Theory and Grammar Implementation*Workshop at European Summer School in Logic, Language, and Information (ESSLLI-2000).

Masuichi, H., T. Ohkuma, H. Yoshimura and Y. Harada. 2003. Japanese parser on the basis of the Lexical-Functional Grammar Formalism and its Evaluation, *Proceedings of The 17th Pacific Asia Conference on Language, Information and Computation* (PACLIC17), pp. 298-309.

Maxwell, J. T., III and R. M. Kaplan. 1989. An overview of disjunctive constraint satisfaction. In *Proceedings of the International Workshop on Parsing Technologies*, pp. 18-27. Also published as `A Method for Disjunctive Constraint Satisfaction', M. Tomita, editor, *Current Issues in Parsing Technology*, Kluwer Academic Publishers, 1991.

Riezler, S., T.H. King, R. Kaplan, D. Crouch, J. Maxwell, and M. Johnson. 2002. Parsing the Wall Street Journal using a Lexical-Functional Grammar and Discriminative Estimation Techniques. *Proceedings of the Annual Meeting of the Association for Computational Linguistics, University of Pennsylvania*.

Riezler, S., T.H. King, R. Crouch, and A. Zaenen. 2003. Statistical sentence condensation using ambiguity packing and stochastic disambiguation methods for Lexical-Functional Grammar. *Proceedings of the Human Language Technology Conference and the 3rd Meeting of the North A merican Chapter of the Association for Computational Linguistics (HLT-NAACL'03)*.

Seiss, Melanie. 2012. A morphological guesser for a morphologically rich language. Poster presented at the DGfS Jahrestagung, 06.-09.03.2012, Frankfurt. http://ling.uni-konstanz.de/pages/home/seiss/publications.html

Seiss, Melanie and Rachel Nordlinger. 2011. An Electronic Dictionary and Translation System for Murrinh-Patha. *Proceedings of the EUROCALL 2011 conference*, University of Nottingham.

Shemtov, H. 1996. Generation of Paraphrases from Ambiguous Logical Forms. *Proceedings of COLING* 1996, 919-924.

Shemtov, H. 1997. *Ambiguity Management in Natural Language Generation.* PhD thesis, Stanford University.

Umemoto, H. 2006. Implementing a Japanese Semantic Parser Based on Glue Approach. Proceedings of The 20th Pacific Asia Conference on Language, Information and Computation.