Hybrid Parsing

ELSNET Summer School 2006

Kilian Foth, Wolfgang Menzel

Department for Informatics
Hamburg University

12. Oktober 2006

Overview

1. Parsing
2. Architectures
3. Parsing as Constraint Satisfaction
4. Weighted Constraint Dependency Grammar
5. Information Fusion with Weighted Constraints

Parsing

Parsing

assigning structural descriptions to sentences

Hybrid parsing

using a range of heterogeneous methods for parsing

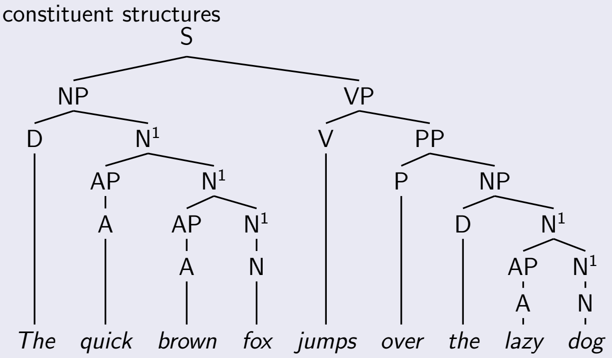
Parsing

- Syntactic Structures
- Deep Parsing
- Shallow parsing

Syntactic Structures

most popular:

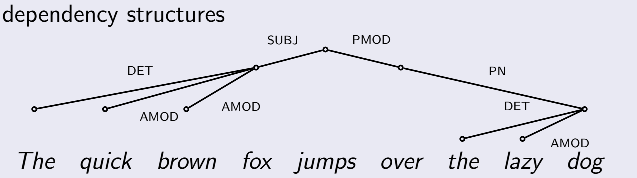
constituent structures



Syntactic Structures

alternative view:

dependency structures



Syntactic Structures

- different conventions for building trees and labelling them
  → annotation guidelines
- treebanks: large collections of sentences annotated with trees
  - English: Penn-Treebank
  - Czech: Prague Dependency Treebank
  - German: Negra-Treebank, Tiger-Treebank
  - ...

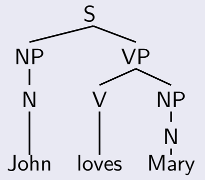
Syntactic Structures

- Why do we need syntactic structures?
  → guiding the semantic interpretation of an utterance
- applications for ...
  - information extraction
  - machine translation
  - corpus linguistics
  - ...

Parsing

deep parsing

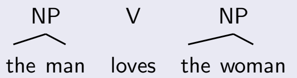
building recursively embedded tree structures



((John) (loves (Mary)))

shallow parsing

building flat syntactic descriptions

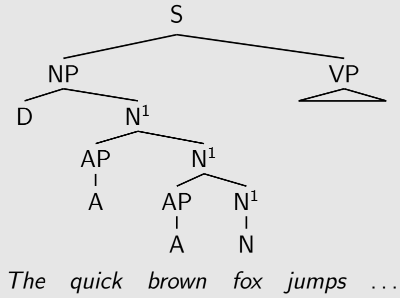


(The man) (loves) (the woman)

Deep Parsing

- using partial trees as building blocks

S → NP VP
NP → D N¹
N¹ → AP N¹
AP → A
N → N¹ . . .



Deep Parsing

realistic grammars are ambiguous:

$N^1 \rightarrow AP\ N^1$
$N^1 \rightarrow N^1\ NP$
$N^1 \rightarrow N^1\ PP$
$N^1 \rightarrow N$
$N^1 \rightarrow N\ PP$
. . .

lexical items are ambiguous:

V → jumps
N → jumps
A → brown
N → brown
. . .

- combinatorial search for a spanning tree licensed by the grammar

Deep Parsing

- structural descriptions and well-formedness conditions are closely connected

Deep Parsing

- worst case: ambiguities multiply out
  → extremely high degree of output ambiguity

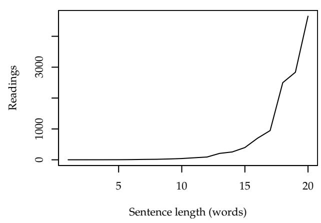
Hinter dem Betrug werden die gleichen Täter vermutet, die während der vergangenen Tage in Griechenland gefälschte Banknoten in Umlauf brachten.

The perpetrators of this fraud are supposed to be the same as those who brought into circulation fake bills in Greece over the last few days.

- Paragram (Kuhn und Rohrer 1997): 92 readings
- Gepard (Langer 2001): 220 readings

Deep Parsing

- Gepard: average ambiguity over a corpus of newspaper text (avg. 11.43 words): 78 readings
- $6.4875 \cdot 10^{22}$ readings for a single sentence (Block 1995)
- Alpino (HPSG of Dutch, van Noord & Malouf 2004):



Deep Parsing

- refining the rules
  - using agreement constraints
  - using subcategorization information

  does not really help

Stochastic parsing

- trained on a treebank
- learning preferences from observations
- state-of-the-art systems are based on
  - markovization: generate rules by means of a stochastic model
  - lexikalization: condition probabilities on lexical items (e.g. phrase heads)
- pro: full disambiguation by determining the most likely structure
- but: loss of perspicuity, accountability and diagnostic capability

Shallow Parsing

- problem with deep parsing:
  the parser has to make decisions, without the grammar providing enough distinguishing information
- alternative: shallow parsing
  use simpler target structures to avoid decisions which cannot be taken reliably
- simpler model structures
  mostly locally restricted relationships → machine learning techniques can be applied

Shallow Parsing

- different shallow components available. e.g.
- part-of-speech tagger

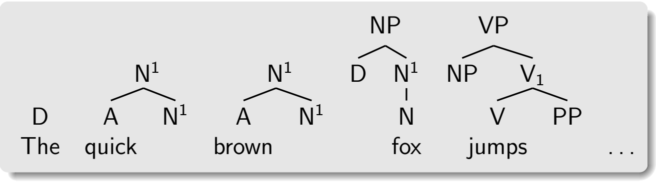
The/DT quick/JJ brown/JJ fox/NN jumps/VB over/IN the/DT lazy/JJ dog/NN

- phrase chunker

[The quick brown fox]$_{NP}$ [jumps]$_{VP}$ [over the lazy dog]$_{PP}$

Shallow Parsing

- supertagger



Shallow Parsing

- shallow components are beneficial for some applications: e.g. information extraction
- can their predictions be used to help a deep parser?
  → problem of information fusion

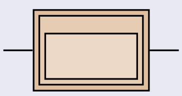
Shallow Parsing

- problem with shallow predictor components
  - relatively high error rate
  - particularly bad: inconsistent predictions
    e.g. two finite verbs in a sentence
    e.g. contradicting supertags
  - if two or more predictor components are combined: conflicting predictions between components

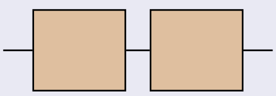
Overview

Architectures

tight integration

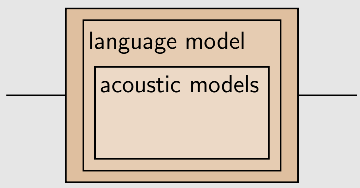


loose integration



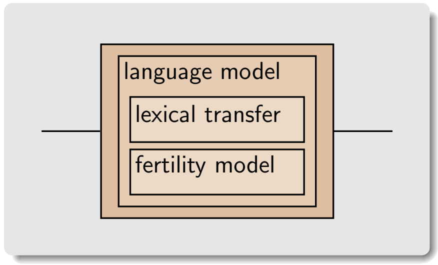
Tight integration

- combining models in a single (global) decision space
  → no local decisions in the preceding component
- e.g. word recognition (Jelinek 1976)



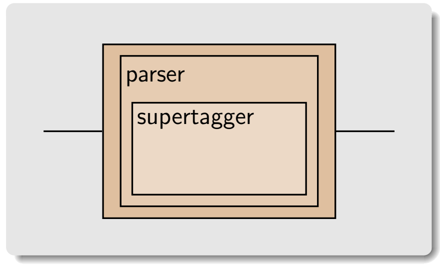
Tight integration

- e.g. stochastic machine translation (Brown et. al 1990)



Tight integration

- e.g. dependency parsing (Wang and Harper 2002)

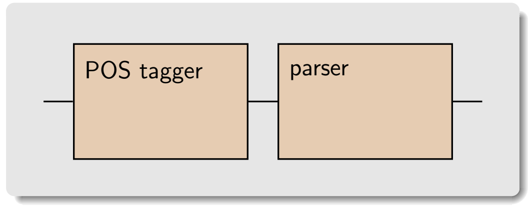


Tight integration

- suboptimal decisions are avoided
- but: tight integration cannot always be easily achieved
  - models must be of the same type (i.e. probabilistic)
  - simple mapping between the structures dealt with by the different models
  - search spaces must be (efficiently) combined

Loose integration

- one component makes independent (local) decisions and communicates them to a subsequent one (*filtering*)
- e.g. parsing

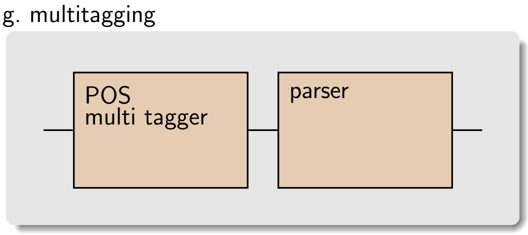


- problems:
  - suboptimal decisions
  - error propagation

Loose integration

possible remedies:

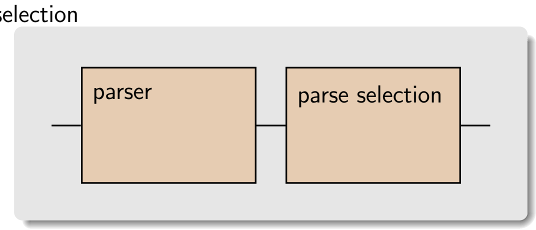
- learn to correct errors
  The subsequent component is trained on the erroneous output of the preceding one.
- select from a number of alternatives (*pipeline*)
  e.g. multitagging



Loose integration

possible remedies

- select from a number of alternatives (cont.) e.g. parse selection



- tolerate conflicting evidence

Loose integration

- How to tolerate erroneous predictions?
  - use an overgenerating backbone
    → distinction between right and wrong is lost
  - constraint retraction (*fallback*)
    → sensitivity to deviant input is kept
  - soft constraints
    - general mechanism in case of failure
    - sensitivity to many levels of error

Examples of Hybrid Parsers

- XTAG (Srinivas 1995):
  - filtering and guiding via POS tags and supertags
  - consistency checks find provably impossible eltrees
  - only the top 3 elementary trees are used for parsing...
  - ... unless parsing fails altogether
  - global preferences e.g. for low PP attachment
- Heart of Gold (Callmeier et al. 2004):
  - XML-based middleware integrates the PET HPSG parser, named entity detector, shallow clause detector, shallow parser SPPC, lexical semantics (GermaNet), stochastic topology parser, and POS tagger
  - no fixed order of processing (*blackboard*)
  - shallow components used for their robustness and throughput

Examples of Hybrid Parsers

- DIENES & DUBEY 2003: find and resolve empty elements in phrase structure trees
  - a maximum-entropy *trace tagger* finds trace locations
  - a PCFG attaches dislocated elements in those places
  - better than entrusting both tasks to the PCFG
- CHARNIAK & JOHNSON 2005: feature-rich PCFG parsing
  - a simplified PCFG generates a packed parse forest of likely candidates
  - the forest is pruned by marginal properties
  - the full PCFG is ranks the remaining possible trees
  - new record for parsing the WSJ corpus

Examples of Hybrid Parsers

- ZEMAN & ZABOKRTSKY 2005: dependency parsing of Czech
  - seven different parsers are run in parallel (*ensemble*)
  - individually, 64% to 85% accuracy
  - various combination policies are investigated
  - weighted voting: trust each prediction in proportion to the overall reliability of its source
  - altogether, 87% structural accuracy
  - "Diversity of opinion is more important to success than individual excellence."

Overview

1. Parsing
2. Architectures
3. Parsing as Constraint Satisfaction
4. Weighted Constraint Dependency Grammar
5. Information Fusion with Weighted Constraints

Constraint Satisfaction

- A *constraint* is a piece of declarative knowledge which restricts the solution space of a given problem.
- How is the structure of the solution space defined?
  - How many variables?
  - Is this number known in advance?
  - Are variables introduced dynamically?
  - Which kind of values can be attached to variables?

A popular Constraint Satisfaction Problem

1	2			5	7	3	6	
					4			7
	3	4		2				
	7	5			6	4		
	9	1			3		5	
2					5			8
5			4			8		
			3			9		
		3			1	6		

- Each row, column and subsquare must contain all 9 digits.
- 81 variables, 9 values, 30 unary / 27 9-ary constraints

A popular Constraint Satisfaction Problem

1	**2**	8	9	**5**	**7**	**3**	**6**	4
6	5	9	1	3	**4**	2	8	**7**
7	**3**	**4**	6	**2**	8	5	9	1
3	**7**	**5**	8	1	**6**	**4**	2	9
8	**9**	**1**	2	4	**3**	7	**5**	6
2	4	6	7	9	**5**	1	3	**8**
5	1	2	**4**	6	9	**8**	7	3
4	6	7	**3**	8	2	**9**	1	5
9	8	**3**	5	7	**1**	**6**	4	2

- Each row, column and subsquare must contain all 9 digits.
- 81 variables, 9 values, 30 unary / 27 9-ary constraints
- This particular problem has a unique solution.

The CS View

- TSANG 1993
- A fixed number of variables is bound to different data types
- Variables have a predefined meaning
- Different variables can have different domains
- Constraints are n-ary relations between specific variables, i.e. subsets of the corresponding Cartesian product of their domains

The HPSG View

- POLLARD AND SAG 1987
- Only a single variable with a recursive feature structure as its value
- Feature structures may again contain variables (e.g. to establish coreference).
- constraints are implications over feature structures
  - require to unify a feature structure with the consequence if it is subsumed by the premise
  - embedded variables are instantiated (information accumulation)

$$\begin{bmatrix}\text{DTRS} & \begin{bmatrix}\textit{head-struc}\end{bmatrix}\end{bmatrix} \rightarrow \begin{bmatrix}\text{SYNSEM|LOC|CAT|HEAD} & \boxed{1} \\ \text{DTRS|HEAD-DTR|SYNSEM|LOC|CAT|HEAD} & \boxed{1}\end{bmatrix}$$

The HPSG View

- structure construction as part of information accumulation
- alternative interpretations (variable bindings)
  - use underspecification
  - need to be enumerated

The LP View

- constraint handling rules (FRÜHWIRTH 1992)
- application to NLP: CHR-Grammar (CHRISTIANSEN 2002)
- constraints
  - are derived from the input and the grammar
  - describe the structure of the input
- high-level expectations can be effectively integrated
- difficulties with alternative interpretations of an input sentence

Other Views

- Property Grammar (BLACHE 1996)
  - elements from a fixed set of constraints describe NL input in a bottom-up manner
  - constructions are identified based on constraints holding for a candidate set of words
  - problem: where do the candidate sets come from?
- Mozart (Oz) (DUCHIER 2001)
  - grammar is described by means of set-valued constraints (dependency structures)
  - multi-level representations (DEBUSSMAN AND DUCHIER 2004)
    - syntax, topological fields, predicate-argument structure, scopus

Constraint Grammar

- KARLSSON 1995
  - attaching underspecified dependency relations to the word forms of an utterances

`@+FMAINV`	finite verb of a sentence
`@SUBJ`	grammatical subject
`@OBJ`	direct Object
`@DN>`	determiner modifying a noun to the right
`@NN>`	noun modifying a noun to the right

Constraint Grammar

- typical CS problem:
  - constraints: conditions on the (mutual) compatibility of dependency labels
  - indirect definition of well-formedness: everything which does not violate constraint explicitly is acceptable
- strong similarity to tagging procedures

Constraint Grammar

- two important prerequisites for robust behaviour
  - inherent fail-soft property: the last remaining category is never removed even if it violates a constraint
  - possible structures and well-formedness conditions are fully decoupled: missing grammar rules do not lead to parse failures
- complete disambiguation cannot always be achieved

Bill	*saw*	*the*	*little*	*dog*	*in*	*the*	*park*
`@SUBJ`	`@+FMAINV`	`@DN>`	`@AN>`	`@OBJ`	`@<NOM`	`@DN>`	`@<P`
					`@<ADVL`		

Constraint Grammar

- size of the grammar (English): 2000 Constraints
- quality:

	without heuristics	with heuristics
precision	95.5%	97.4%
recall	99.7 . . . 99.9%	99.6 . . . 99.9%

- TAPANAINEN & JÄRVINEN 1995:
  - reuse KARLSSON's Constraint Grammar unchanged
  - add handwritten rules that generate specific dependencies
  - report 95.3%/87.9% dependency accuracy for *Bank of English* text

Constraint Dependency Grammar

- MARUYAMA 1990
- each word form of a sentence corresponds to a variable.
  → number of variables is a priori unknown.
  → no predefined meaning for variables.
- every constraint must hold for each variable or a combination thereof.
- all value assignments for all variables are taken from the same domain: $W \times L$ (attachment values).
- fully specified dependency relations $D \in W \times W \times L$
- originally invented to express all possible preposition attachments concisely for Japanese

Constraint Dependency Grammar

- lexicon items and levels of analysis define the conceivable structures
- constraints make linguistically motivated restrictions
- an assignment which satisfies all constraints is by definition a solution
- $\longrightarrow$ parsing is structural disambiguation

Constraint Dependency Grammar

- constraints are logical conditions which *every* dependency edge must fulfill
- therefore constraints are typically implications of the form `condition -> restriction`
- constraints can be
  - unary: considering only one edge at a time or
  - binary: considering pairs of edges at a time
- constraints with higher arity are possible, but usually very expensive

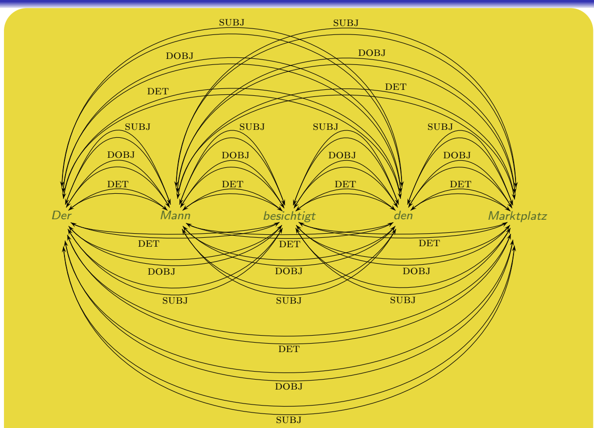
Constraint Dependency Grammar

- complete Constraint Satisfaction procedure
  - removal of incompatible dependency edges
  - constraint propagation via Waltz-Filtering
- interactive disambiguation: Increasingly domain specific constraints are applied if no full disambiguation can be achieved (MARUYAMA 1990)
- CDG is mildly context sensitive
- time complexity: $\mathcal{O}(|C| \cdot n^4)$
  - n length if the input
  - C constraint set
- Extraction of all parses can be NP complete!

Hypothesis Space

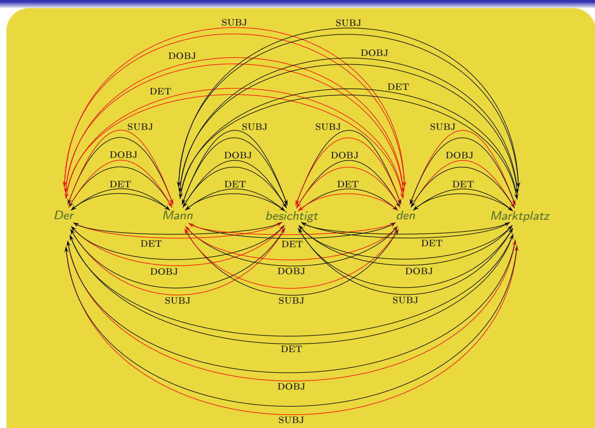
root/nil	root/nil	root/nil	root/nil	root/nil
det/2	det/1	det/1	det/1	det/1
det/3	det/3	det/2	det/2	det/2
det/4	det/4	det/4	det/3	det/3
det/5	det/5	det/5	det/5	det/4
subj/2	subj/1	subj/1	subj/1	subj/1
subj/3	subj/3	subj/2	subj/2	subj/2
subj/4	subj/4	subj/4	subj/3	subj/3
subj/5	subj/5	subj/5	subj/5	subj/4
dobj/2	dobj/1	dobj/1	dobj/1	dobj/1
dobj/3	dobj/3	dobj/2	dobj/2	dobj/2
dobj/4	dobj/4	dobj/4	dobj/3	dobj/3
dobj/5	dobj/5	dobj/5	dobj/5	dobj/4
Der	*Mann*	*besichtigt*	*den*	*Marktplatz*
1	2	3	4	5

Constraining structures



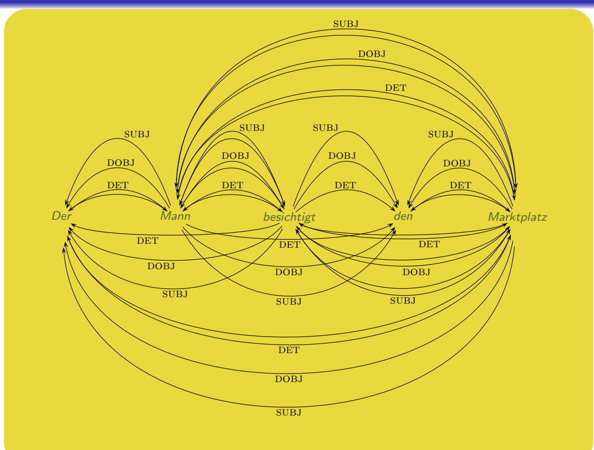
Initial state of a parsing problem with three labels (DET, SUBJ, DOBJ)

Constraining structures

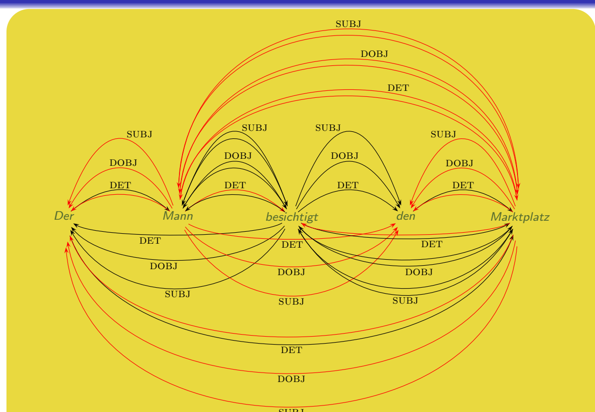


{X} : DetNom : Det : 0.0 : X↓cat=det → X↑cat=noun ∧ X.label=DET

Constraining structures

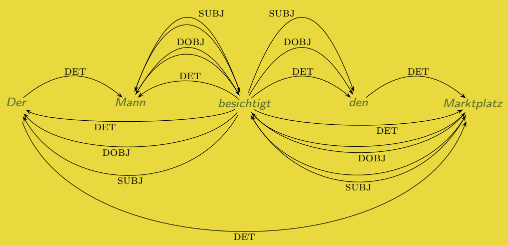


Constraining structures

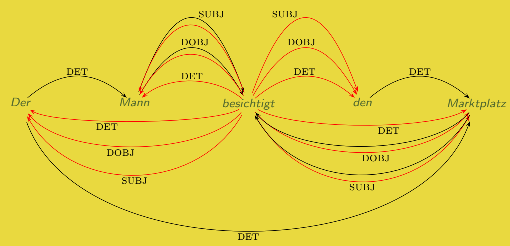


{X} : SubjObj : Verb : 0.0 :
X↓cat=noun → X↑cat=vfin ∧ X.label=SUBJ ∨ X.label=DOBJ

Constraining structures

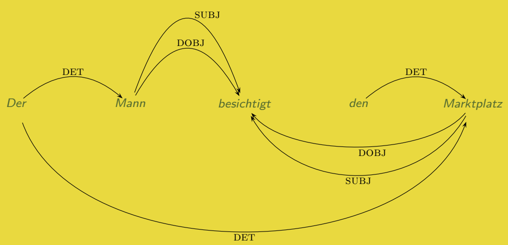


Constraining structures

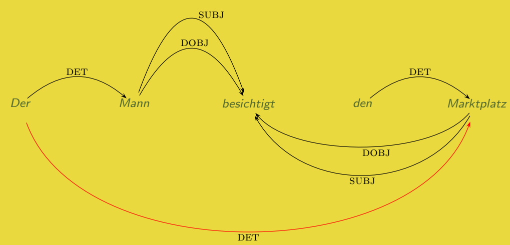


$\{X\}$: Root : Verb : 0.0 :
$X{\downarrow}\text{cat}=\text{vfin} \rightarrow X{\uparrow}\text{cat}=\text{nil}$

Constraining structures

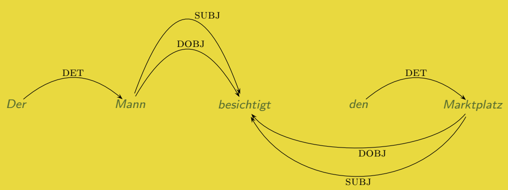


Constraining structures

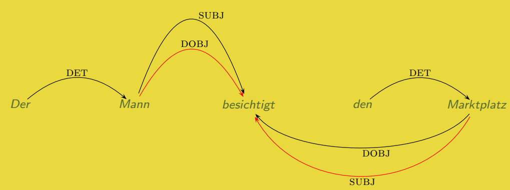


$\{X,Y\}$: Unique : General : 0.0 :
$X{\uparrow}\text{id}=Y{\uparrow}\text{id} \rightarrow X.\text{label}\neq Y.\text{label}$

Constraining structures

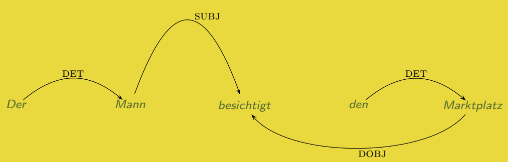


Constraining structures



$\{X,Y\}$: SubjAgr : Subj : 0.0 :
$X.\text{label}=\text{SUBJ} \wedge Y.\text{label}=\text{DET} \wedge X{\downarrow}\text{id}=Y{\uparrow}\text{id} \rightarrow Y{\uparrow}\text{case}=Y{\downarrow}\text{case}=\text{nom}$

Constraining structures



Benefits of Constraints for Parsing

- ideally suited for information fusion
  - multiple information sources are crucial to NLU
  - all sources can be integrated into the same computation
  - but each rule need only deal with one source (modularity)
- potential for fail-soft behaviour
  - rules can denote preferences as well as laws
  - preferences of different strengths can be modelled
- no equivalence of structures and rules necessary
  - for English, all allowed configurations can be listed
  - in other languages it is easier to describe the forbidden configurations instead
  - allows detailed diagnoses such as "syntactically correct, but misinflected"

Overview

1. Parsing
2. Architectures
3. Parsing as Constraint Satisfaction
4. Weighted Constraint Dependency Grammar
5. Information Fusion with Weighted Constraints

Extensions

- relational view on dependency structures instead of a functional one:
  - → SCHRÖDER (1996): access to lexical information at the modifying *and* the dominating node
- recognition uncertainty / lexical ambiguity
  - → HARPER AND HELZERMAN (1996): hypothesis lattice additional global constraint (path criterion) introduced
- existence quantors and long-distance dependencies
  - → FOTH (2002): arbitrary global constraints more than two edges can be restricted in some configurations

Local Constraints

A rule from DIAGRAM (ROBINSON 1982):

```
NP1RULE NP = (D= {A / DDET / DETQ} NOMHD (NCOMP)

CONSTRUCTOR (PROGN (COND ((@D)
                      [COND ((MASS? D)
                                (OR (MASS? NOMHD)
                                    (F.REJECT (QUOTE F.MASS]
                      [COND ((MASS? NOMHD)
                                (OR (NOT (@A))
                                    (F.REJECT (QUOTE F.MASS]
                      [COND ((@NCOMP)
                        (SET NBR(@INTERSECT NBR D NOMHD NCOMP)))
                        (T (SET NBR (@INTERSECT NBR D NOMHD]
                        ((AND (SG? NOMHD)
                          (NOT MASS? NOMHD)))
                        (@FACTOR (QUOTE F.NODET)
                                 UNLIKELY))
                        ((@NCOMP)
                       (SET NBR (@INTERSECT NBR NCOMP NOMHD)))
                    (T @FROM NOMHD NBR)))
                    [AND (@THANCOMP NCOMP)
                      (OR (@THANCOMP NOMHD)
                          F.REJECT (QUOTE F.THANC]
                    (@FROM NOMHD TYPE)))
```


Local Constraints

- a generative system must deal with category checks, order, concord of number, lexical issues ('than') and presence of a determiner in one rule
- a constraint grammar can use five rules:

```
{X!SYN} : NP :
  X.label = NP -> X↓cat = DT & (X↑cat = NN | X↑cat = NNS);

{X!SYN} : NP_order :
  X.label = NP -> X/;

{X!SYN} : NP_concord :
  X.label = NP -> X↓number = X↑number;

{X!SYN\Y!SYN} : NP_than :
  X↓word = than & Y.label = NP ->
  has(Y↑id, find_comparative, Labels, NP_scope);

{X!SYN} : NP_determiner:
  X↓cat = NN & ~exists(X↓mass_noun) -> has(X@id, Determiner);
```


Nonlocal Constraints

Constraints can also capture information from different levels of analysis:

- Quantor plausibility: if the VP is modified by 'usually', then the quantor 'every' is implausible
- Euphony: two successive words should not have the same phonetic form
- Style: the word forms in a sentence usually belong to the same register and epoch

Weighted Constraints

- penalty factors reduce the preference for hypotheses which violate a constraint
- $w(c) = 0$: hard constraint, must always be satisfied e.g. licensing structural descriptions
- $0 < w(c) < 1$: soft constraint may be violated if no better alternative is available
  - $w(c) << 1$: strong, but defeasible well-formedness conditions
  - $w(c) >> 0$: defaults, preferences, etc.

  $w(c) = 1$: no effect, neutralizes the constraint
- penalties can also depend on the specific subordination, e.g. the closer, the better

Weighted Constraints

Why weighted constraints?

- Weights help to fully disambiguate a structure.
  - Hard constraints are not sufficient (HARPER ET. AL 1995).
- Many language regularities are preferential and contradictory.
  - extraposition
  - linear ordering in the German mittelfeld
  - topicalization
- Weights are useful to guide the parser towards promising hypotheses.
- Weights can be used to trade speed against quality.

Weights In Collision

A phrase from Roman poetry: "... *nympham amabat sol* ..."

- Contradicting information sources:
  - morphology: *nympham* is accusative, therefore an object
  - syntax: *nympham* precedes the verb, therefore a subject
  - semantics: nymphs can love, but the sun cannot
  - pragmatics: the beauty of the nymph is the topic here
- a generative system would have to include all aspects into the same rule
- WCDG can write four different rules instead
- in this case, PRAG > MOR > SEM > SYN

Examples of Constraint Weights

Some reasonable assumptions about English:

- Subjects and objects appear only under verbs; $w(c) = 0$
  "We won." / *"We winners."
- Finite verbs almost always have subjects; $w(c) \approx 0$
  "And so it goes." / *"And so goes."
- Infinitives should not be split; $0 < w(c) << 1$
  "Try not to think of it." / *"Try to not think about it."
- Transitive verbs usually have objects; $0 << w(c) < 1$
  "We sell cars at fair prices." / *"We sell at fair prices."
- Plural nouns are slightly rarer than singulars; $w(c) \approx 1$
  "I feed the fish/sg." / "I feed the fish/pl."

Comparison to Optimality Theory

- There is no generative backbone.
  → Everything that is not forbidden is possible.
- Weights instead of ranking.
  → Several constraint weights can be equal.
- Scores are accumulated.
  → Several weak constraints
  can gang up on a stronger one.
- Rules, not principles, are scored.

Accumulating Scores

- accumulating (multiplying) the weights for all constraints violated by a partial structure
  → both single dependency relations and tuples have combined scores
- local scores are multiplied into a global one

$$w(t) = \prod_{e \in t} \prod_{c.\text{violates}(e,c)} w(c) \cdot \prod_{(e_i,e_j) \in t} \prod_{c.\text{violates}((e_i,e_j),c)} w(c)$$

- determining the optimal global structure

$$t(s) = \arg\max_t w(t)$$

→ parsing becomes a constraint optimization problem

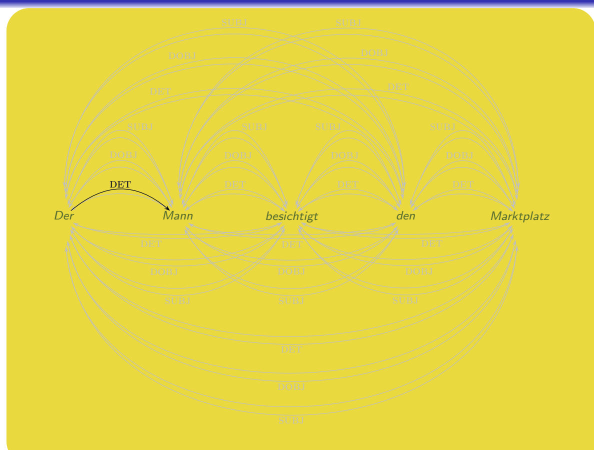
Constraint Optimization

- **consistency**: works only for hard constraints
- **pruning**: successively remove the least preferred dependency relations
- **search**: determine the optimum dependency structure
- **structural transformation**: apply local repairs to improve the overall score

Note: According to the $\mathcal{NP}$ hypothesis,
there can be no efficient solution method for WCDG!

Search

Search

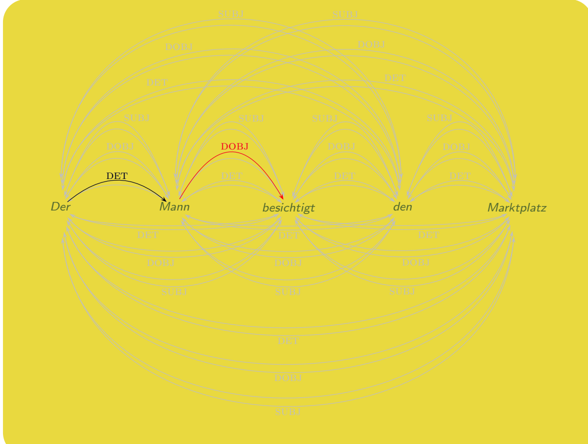


Search

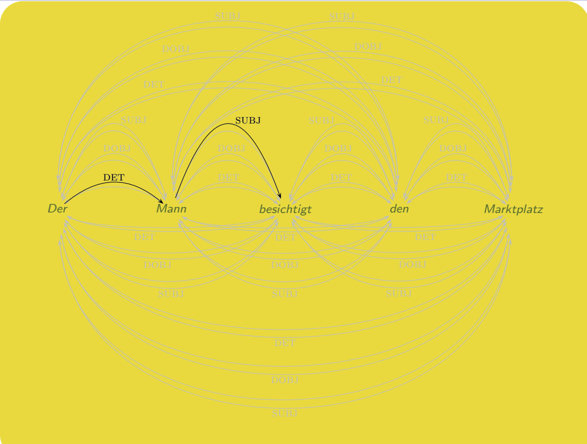
Search

Search

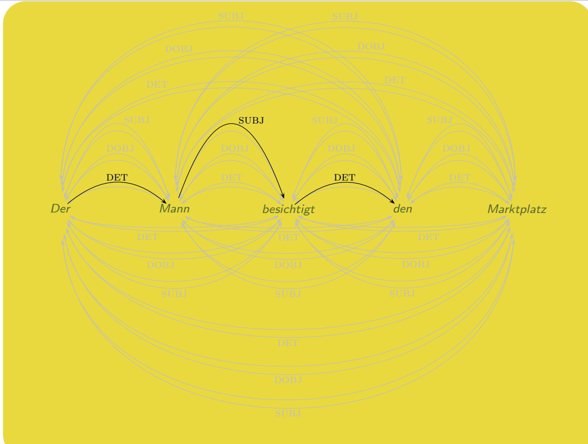
Search



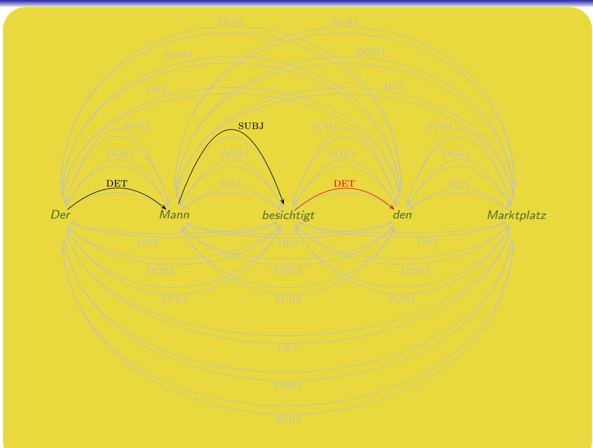
Search



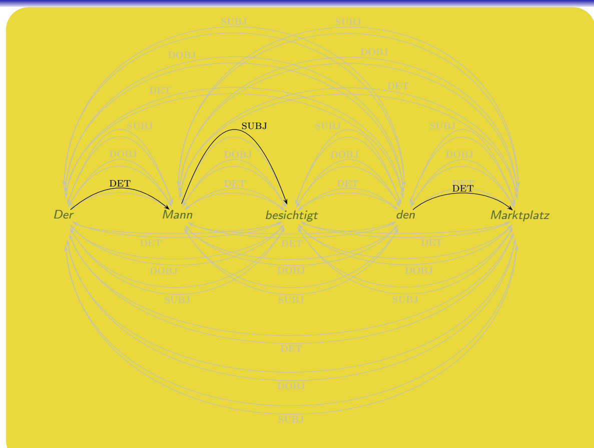
Search



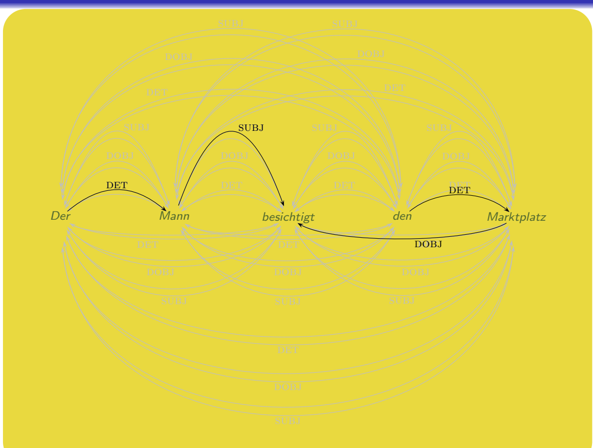
Search



Search



Search

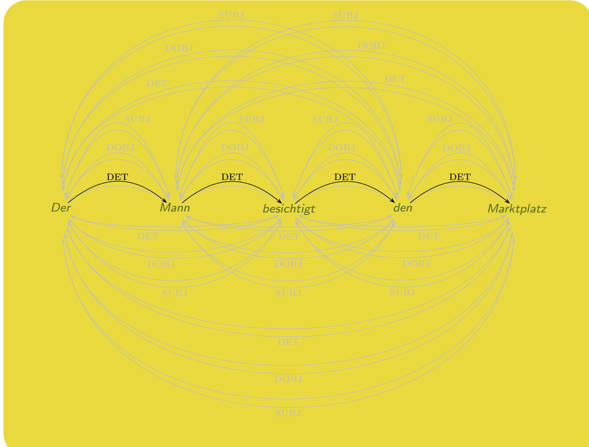


Structural Transformation

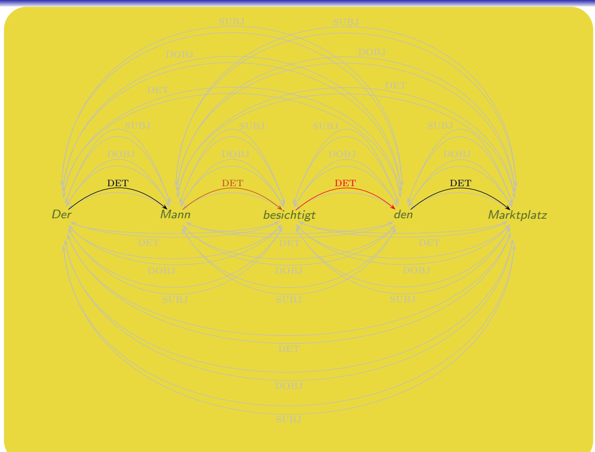
- elementary repair operations:
  - change a subordination
  - change the label of an edge
  - choose a lexical reading
- many degrees of freedom:
  - how many changes during one step?
  - how many alternative steps are tried at a time?
  - which transformation is tried first?
  - is the selection (partially) random?
  - when do we give up?

Structural Transformation

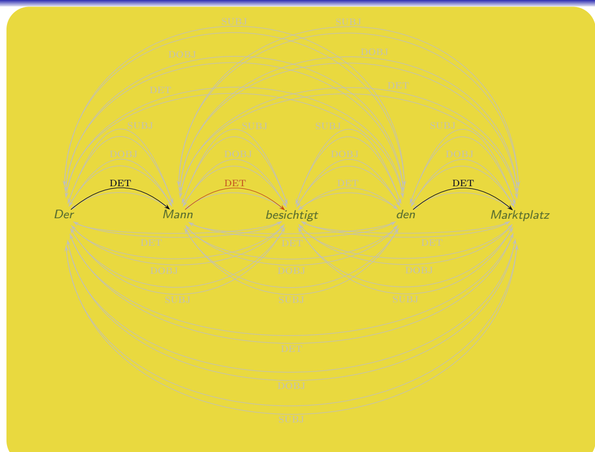
Structural Transformation



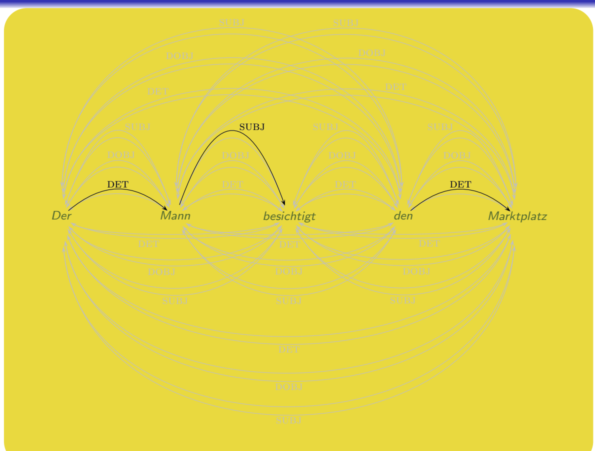
Structural Transformation



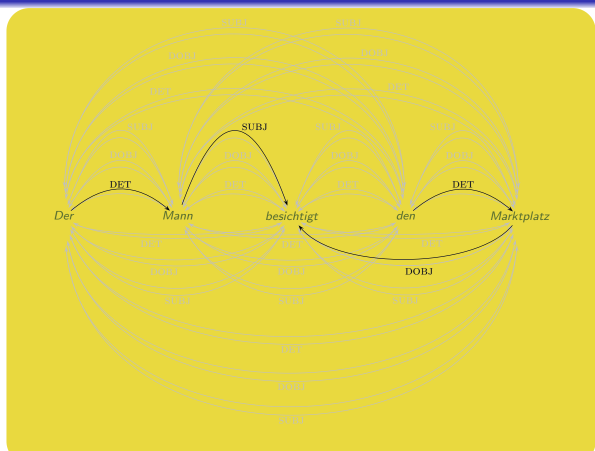
Structural Transformation



Structural Transformation



Structural Transformation



Structural Transformation

- Usually local transformations result in inacceptable structures
  - sequences of repair steps have to be considered.
  - e.g. swapping SUBJ and DOBJ

a)	syntax	...
der$_1$	det/2	...
mann$_2$	**dobj/3**	...
besichtigt$_3$	root/nil	...
den$_4$	den/5	...
marktplatz$_5$	**subj/3**	...

⟹

b)	syntax	...
der$_1$	det/2	...
mann$_2$	**subj/3**	...
besichtigt$_3$	root/nil	...
den$_4$	det/5	...
marktplatz$_5$	**dobj/5**	...

Frobbing*

- gradient descent search
- escaping local minima:
  increasingly complex transformations ⟶ local search
- heuristically guided tabu search
  - transformation with perfect memory
  - propagation of limits for the score of partial solutions
- faster than best-first search for large problems
- inherently anytime
- to date the best option for solving WCDG

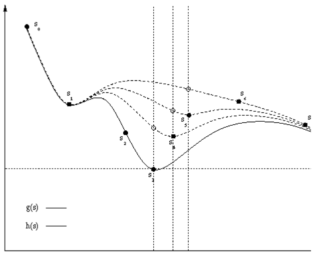
* *frobbing*: randomly adjusting the settings of an object, such as the dials on a piece of equipment or the options in a software program. (The Word Spy)

Guided Local Search

- memory as energy landscape defined over the hypothesis space (VODOURIS 1997)
- transformation with imperfect memory (SCHULZ 2000)
- augmented scoring function:

$$h(s) = g(s) + \lambda \cdot \sum_{i=1}^{|\mathcal{F}|} n_i \cdot I_i^f(s)$$

f_i solution features



Guided Local Search

- utility: where to change the weights of the scoring function

$$util(s_*, f_i) = I_i^f \cdot \frac{c_i}{1 + n_i}$$

- policy:
  - high costs $\longrightarrow$ high utility
  - repeated repair $\longrightarrow$ lower utility

Overview of Solution Methods

- Consistency: makes no mistakes, but leaves far too many choices (bad precision)
- Pruning: leaves fewer choices, but can destroy the solution (bad recall)
- Search: correct *and* complete, but unaffordable (bad time and space complexity)
- Heuristic Transformation: incomplete, but good in practice, and interruptible (bad theoretical foundation)
- Chart-like bottom-up parsing: avoids some inefficiency of backtracking, but not enough (good idea, bad in practice)
- Genetic algorithms: semi-randomized transformation, takes far too long to converge
- Shift-reduce parsing: more of that later

Modelling Syntax with Constraints

- WCDG ideally supports grammar development by providing *diagnostic information*
  - overgeneration appears as high scores for wrong analyses
  - undergeneration appears as low scores for the right analysis
  - the responsible constraints are immediately obvious
- typical development cycle:
  1. parse a sentence with a draft grammar
  2. correct the structure manually (and store it)
  3. change the constraints violating the gold standard; introduce constraints prohibiting misanalyses
  4. parse the sentence with the modified grammar
  5. repeat

Modelling Syntax with Constraints

- Writing constraints is counter-intuitive.
  - CFG: to extend coverage, *add* or *extend* a rule
  - CDG: to extend coverage, *remove* or *weaken* a constraint
- but covering new phenomena often requires introducing new labels or even levels
  $\longrightarrow$ extending coverage usually does lead to more constraints
- (and also to more complicated ones)
- describing an entire language is very hard work in any formalism

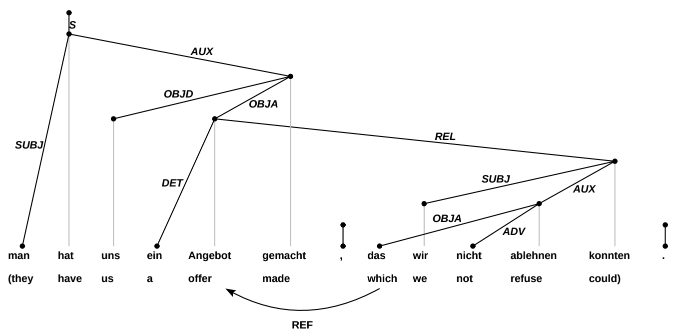
Examples of WCDG Constraints

```
// Subjects and objects appear only under verbs
{X:Syntax} : 'subject definition' : 0.0 :
  X.label = subject -> X↑cat = VB;

// Finite verbs almost always have subjects
{X:Syntax} : 'missing subject' : 0.1 :
  X↓cat = VBP -> has(X↓id, subject);

// Infinitives should not be split
{X/Syntax/\Y/Syntax} : 'split infinitive' : 0.2 :
  X↓word = to & X↑cat = VB ->  Y↓from < X↓from;

// Transitive verbs usually have objects
{X:Syntax} : 'missing object : 0.8 :
  X↓cat = VBP & exists(X↓transitive) -> has(X↓id, dobject);

// Plural nouns are slightly rarer than singulars
{X:Syntax} : plural : 0.99 :
  X↓number != plural;
```


The Grammar of German

- only two levels: syntax, reference
- about 1000 handwritten constraints
- allows non-projective dependency structures if necessary
- strongly lexicalized: e.g. valence information for verbs and prepositions
- An overview of relation types: ADV, APP, ATTR, AUX, AVZ, CJ, DET, ETH, EXPL, GMOD, GRAD, KOM, KON, KONJ, NEB, NP2, OBJA, OBJA2, OBJD, OBJG, OBJC, OBJI, OBJP, PAR, PART, PN, PP, PRED, REL, S, SUBJ, SUBJC, VOK, ZEIT, ".

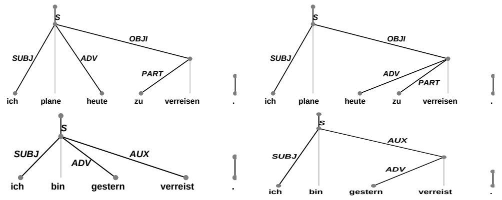
An Example Annotation



"They made us an offer we could not refuse."

Modelling Language With WCDG

- some ambiguity is *meaningful*, and some is *spurious*

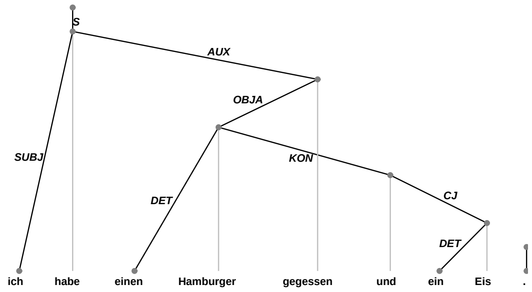


- spurious ambiguity should be normalized
  (if you're concerned about parsing accuracy):

```
{X!SYN/\Y!SYN} : 'VP lowering' : 0.1:
  X.label = AUX -> Y.label != ADV;
```


Modelling Language With WCDG

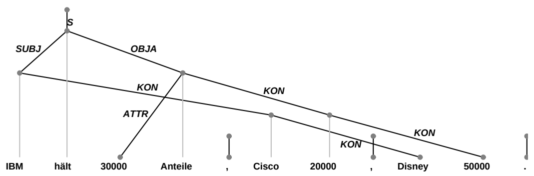
- WCDG allows *logical* dependency to be expressed as *physical* dependency:



- just don't enforce projectivity

Modelling Language With WCDG

- projectivity constraints are still needed for other phenomena
  → make exceptions for coordinations
- but know where to stop:



is this a useful syntax annotation?

Modelling Language With WCDG

- many conditions concern more than two dependencies
- but high-arity constraints are extremely expensive
  - approximate the condition with several binary constraints
  - introduce additional levels (MARUYAMA)
  - introduce operators that extend binaryness in a controlled way
- Example: German vorfeld
  "If two constituents precede the finite verb, then the verb itself is not labelled as *S* (main clause)."
- this involves three dependency edges!
- but the third is always the parent of the two others
  → introduce a "look above" operator: `is()`

Practical Constraint Writing

```
// Vor einem Hauptsatzverb dürfen nicht
// zwei Konstituenten stehen:
//  'Heute gehen wir in den Zoo.'
// *'Heute wir gehen in den Zoo.'
{X/SYN/\Y/SYN} : Vorfeld : 0.1 :
  X↑cat = VVFIN -> ~is(X↑id, S, Label, Konjunkt);
```

- constraints need detailed comments just like subroutines
- example and a counter-example to demonstrate that the constraint works
- contraint names are important for developer and user
- should a name describe the correct case or the error case?
- in this case, we waffled on the issue

Practical Constraint Writing

Unfortunately, there are many exceptions:

- **Hätte** ich ihr vertraut, **ich** hätte sie nicht verloren.
- Was es auch **ist**, das **Phänomen** muß untersucht werden.
- Wenn das Salz nicht mehr **salzt**, **womit** soll man salzen?
- Daß es so ein Erfolg **würde**, **damit** rechnete niemand.
- **Scheint** es gleich wirr, **so** hat es doch Methode.
- **Freilich**, der **Bundesrat** muß noch zustimmen.
- **Vom** Abend **bis** zum Morgen geht die Feier.
- **Das jedoch** scheint fraglich.
- **Wir im**- und exportieren Hardware.

Practical Constraint Writing

- when two principles clash, you can simply write two constraints with different weights
- where applicable, the stronger constraint will 'win'
- but it is better to allow special cases explicitly
- one rule cannot make exceptions from another rule
  → all exceptions must be part of the rule itself
- the *real* 'Vorfeld' constraint is 86 lines long!
- many rules consist predominantly of exceptions
  → generativity via back door

Sample Constraints

Subjects are nouns:

```
//  'Das ist gut.'
// *'Demnächst ist gut.'
//  '"Demnächst" ist gut!'
{X!SYN} : 'SUBJ-Kategorie' : category : 0.0 :
  X.label = SUBJ ->
  isa(X↓,Nominal) & X↓cat != PRF | X↓cat = ADJA |
  quoted(X);
```

Only a single subject is allowed:

```
//  'Ich hatte viel Bekümmernis.'
// *'Ich, ich, ich, ich hatte viel Bekümmernis.'
{X!SYN/\Y!SYN} : 'doppeltes Subjekt' : uniq : 0.0 :
  subsumes(Label, Subjekt, X.label)
    -> ~subsumes(Label, Subjekt, Y.label);
```

Sample Constraints

The subject is least oblique argument:

```
//  'Heute tanzt der König das Menuett.'
// *'Heute tanzt das Menuett der König.'
{X!SYN/\Y!SYN} : 'Subjekt-Position' : order : 0.9 :
 X.label = SUBJ & subsumes(Label, Nominalobjekt, Y.label)
 ->
 X↓from < Y↓from |

 // 'falls sich/OBJA nicht ein Investor/SUBJ findet'
 Y↓cat = PRF |

 // 'ein Mann, dem/OBJD man/SUBJ vertrauen kann'
 (Y↓cat = PWS | Y↓cat = PRELS | Y↓cat = PWAT |
  Y↓cat = PWAV | Y↓case = PRELAT |
  has(Y↓id, find_initial));
```


Sample Constraints

Subject and verb agree in number:

```
//  'Wir gehen in den Zoo.'
// *'Wir gehe in den Zoo.'
{X!SYN} : 'Subjekt-Numerus' : agree : 0.1 :
  X.label = SUBJ &
  exists(X↓number) & exists(X↑number)
  ->
  quoted(X) |

  compatible(Features, X↓number, X↑number) |

  // 'Peters Noten waren furchtbar, und Annas waren nicht viel besser.'
  X↓subcat = Vorname & X↓case = gen |

  // 'Eine Menge Leute sind hier.'
  X↑number = pl & (X↓set = yes | X↓word = Art | X↓sort = number) |

  // '80 Prozent mehr wurden verkauft als im letzten Jahr.'
  X↓cat=PIS & exists(X↓degree) |

  // '1992 war ein gutes Jahr.'
  X↓cat = CARD & X↓number = pl & X↓value < 2100 |

  // was für
  X↓word = was & X↓cat = PWS &
  (has(X↓id, find_für) | has(X↑id, find_für, Label, AUX_OBJI)) |

  // 'Das sind ganz üble Gesellen!'
  (X↓word = das | X↓word = dies | X↓word = es | X↓word = "'s" | X↓word = was) &
  (X↑base = sein | X↑base = werden | has(X↑id, find_sein_werden, Label, AUX_OBJI)) |

  // 'Yahoo und Amazon haben bestätigt, daß Wasser naß ist.'
  X↑number = pl &
  (X/ & has(X↓id,KON,Label,APP_KON, 0, X↑from) |
   X\ & has(X↓id,KON,Label,APP_KON));
```


The Lexicon

- full-forms for all closed-class items
- 8,500 verb stems, 27,000 noun stems
- compound analysis
- lexical templates for unknown words

```
der := [ cat:ART, case:nom, number:sg, gender:masc, definite:yes ];

ganz := [cat:ADV,subcat:grade, likes_positive:yes,
   modifies: <Adjektiv,1,Adverb,1, Pronomen,1,KOKOM,1,Verb,0.99> ];

ausgelöschten := [cat:ADJA,base:auslöschen,partizipial2:yes,
                 degree:positive,avz:allowed,perfect:haben,
                 valence:'a?',case:gen_dat,
                 number:sg,gender:bot,flexion:weak,suffix:en];

'^[A-ZÄÖÜ]\\\.$' =~
 [ pattern:Initial, cat:NE, subcat: Vorname,
   case:bot, person:third, gender:bot, number:sg, sort:bot ];
```


Evaluation

- 1000 sentences from the NEGRA corpus (German newspaper text)
- same as the one used by SCHIEHLEN (2004)
- superset of the one used by DUBEY (2005) who limited sentence length to 40 words
- dependency structures automatically extracted from the phrase structure of the treebank (DAUM ET AL. 2004)

Results

corpus	# of sentences	unlabelled edges	labelled edges
all sentences	1000	92.3%	90.9%
<60 words	998	92.3%	90.9%
<40 words	963	92.7%	91.3%
<20 words	628	94.1%	92.5%
<10 words	300	95.0%	92.9%

Comparison

- DUBEY (2005)
  - purely stochastic parser (modified Collins parser)
  - sister-head dependencies
  - treebank transformation
- SCHIEHLEN (2004)
  - probabilistic CFG parser
  - strong support from external lexical resources
  - enhanced treebank information used during training

	test set	constituent structures labelled precision/recall/f-score	dependency structures labelled precision/recall/f-score
DUBEY	≤ 40	70.9%/71.3%/71.09%	—/—/76.08%
SCHIELEN	all	—/—/69.36%	—/—/81.69%
FOTH ET AL.	all	—/—/—	90.9%/90.9%/90.9%

Comparison

MCDONALD et al. 2006:

- two-stage dependency parser:
- unlabelled attachment through local feature-based stochastic tree learning
- label classification via global Markov model
- many millions of features
- CoNLL task (different corpus and annotation standard): 90.4% / 87.3% for German (13 languages in all)

Processing Time

time limit per sentence	time used	unlabelled edges	labelled edges
600 seconds	68.0s	89.0%	87.0%
400 seconds	59.3s	88.7%	86.8%
200 seconds	44.9s	88.2%	86.2%
100 seconds	31.8s	87.1%	85.0%
50 seconds	21.6s	84.6%	82.3%

Text Type Influence

text type	sentences	average length	unlabelled edges	labelled edges
trivial literature	9547	14	94.2%	92.3%
law text	1145	19	90.7%	89.6%
Verbmobil dialogues	1316	8	90.3%	86.3%
Bible	2709	16	93.0%	91.2%
online news	10000	17	92.0%	90.9%
serious literature	68	34	78.0%	75.4%

Relative Importance of Information Sources

- Since constraints do not generate structures, no constraint is strictly necessary.
- Because of information fusion, switching off one constraint often does not change the analysis.
- We can leave out entire constraint classes at a time.
- This indicates which sorts of rules are the most important for parsing.
- Possible use for grammar development, psycholinguistic interpretation. . .

Relative Importance of Information Sources

Class	Purpose	Example	Importance
init	hard constraints	appositions are nominals	3.70
pos	POS tagger integration	prefer the predicted category	1.77
root	root subordinations	only verbs should be tree roots	1.72
cat	category cooccurrence	adverbs do not modify each other	1.13
order	word-order	determiners precede their regents	1.11
proj	projectivity	disprefer nonprojective coordination	1.09
exist	valency	finite verbs must have subjects	1.04
punc	punctuation	subclauses are marked with commas	1.03
agree	rection and agreement	subjects have nominative case	1.02
lexical	word-specific rules	“entweder” requires “oder”	1.02
dist	locality principles	prefer short attachments	1.01
pref	default assumptions	assume nominative case by default	1.00
sort	sortal restrictions	“sein” takes only local predicatives	1.00
uniq	label cooccurrence	there can be only one determiner	1.00
zone	crossing of marker words	conjunctions must be leftmost dependents	1.00

Availability

The reference implementation of WCDG is Free Software.

- online demo
  http://nats-www.informatik.uni-hamburg.de/Papa/ParserDemo
  - Does only time-limited analysis (for interactivity)
  - Contact us for bulk parsing
- download
  http://nats-www.informatik.uni-hamburg.de/download
  - Runs on x86-Linux; ports are planned
  - Contains program, grammar and annotation manual of German

Overview

1. Parsing
2. Architectures
3. Parsing as Constraint Satisfaction
4. Weighted Constraint Dependency Grammar
5. Information Fusion with Weighted Constraints

Stochastic Helper Components

- Why are handwritten rules not enough?
  - Language understanding is largely guided by preferences
  - In particular, preferences between alternatives that are both “correct”
  - Intuitive knowledge is not easily made explicit
  - Empirical models can capture it more reliably
- Trying to gain the best of both worlds
- Helpers: POS Tagging, Supertagging, Chunk parsing, PP attachment, Shallow dependency parsing

Hybrid Parsing: POS Tagger

- The grammar rules disambiguate most sentences correctly. . .
- . . . assuming we know the word categories
- Wide-coverage parsing requires an extremely broad view of categories:

  “Die Xeon-Prozessoren mit 256 KByte L2-Cache auf dem **Die** brauchen 133 MHz Front-Side-Bustakt.”
- Even closed-class items like “die” might be something different!

Hybrid Parsing: POS Tagger

- POS tagging is well-understood
- It’s not perfect, but we don’t have believe it completely
- Method:
  - Call TnT (Brants 1996) on a sentence before parsing
  - Map its probabilities to scores
  - Prefer the predicted categories:

```
{X:SYN} : tagger : [ predict(X↓id, POS, X↓cat) ] :
  predict(X↓id, POS, X↓cat) = 1.0;
```

- Effect: smaller hypothesis space, better guidance towards probable solutions
- Typically more than halves the error rate
  → POS tagging is an enabler for large-scale WCDG

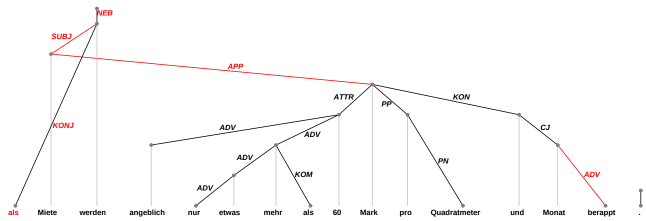
Hybrid Parsing: POS Tagger

Intricacies of tagger-parser integration:

- TnT and WCDG use different lexica
  → suppress out-of-lexicon predictions
- TnT calculates probabilities, WCDG uses penalties
  → normalize the highest p to 1
- TnT can use different beam widths
  → find a suitable one through experimentation
- Tagger errors can propagate to the parser
  → this one needs more effort

Hybrid Parsing: Tagger Errors

- TnT makes around 5% errors on unseen input
- 2% of these are *hard errors*
- Some of them can be overridden. . .
- . . . but many can't:



- One tagger error causes three attachment errors

Hybrid Parsing: Hybrid Tagging

- Many tagger errors are obvious to a human expert:
  "Die/ART Organisation/NN hatte/**VAFIN** am/APPRART Dienstag/NN einen/ART Waffenstillstand/NN erklärt/**VVFIN**."
  There are virtually never two finite verbs in a clause!
- By combining lexicon and sentence-global knowledge, we can figure out the truth: 'erklärt' is a past participle!
- But neither trigrams nor constraints can easily express this
- The answer is (of course) hybrid pre-processing.

Hybrid Parsing: Hybrid Tagging

- Apply automatic correction rules to TnT's output:
  - If two finite verbs co-occur, one of them is really infinite
  - "als" is not a conjunction if the verb is in the middle of the clause
  - An oblique personal pronoun near the corresponding base form is almost certainly a reflexive pronoun instead
  - Words in CamelCase are almost certainly proper nouns
- Over 50 rules are used altogether
- Tagging accuracy rises from 97.2% to 97.7% (on NEGRA)
- Parsing accuracy rises from 89.0% to 89.7%
  (of a possible 90.4%)

Hybrid Parsing: Hybrid Tagging

- POS tagging by transformation is not new
- In fact, transformation alone can be used (Brill 1992)
- But automatically learnt rules must follow strict templates
- By hybridising the task, we can focus on those errors particularly harmful to parsing
- . . . and use complicated rules where necessary
- Again, the advantages of both worlds can be combined

Hybrid Parsing: Supertagging

- *Supertagging* (Joshi 1999) extends tagging
- also predicts relation type, attachment direction, child nodes
- invented for LTAG, which is quite similar to WCDG
- used as a filter, it proved an enabling technology there
- But WCDG does not use elementary trees, only edges
- How to adapt subtree prediction to WCDG?

Hybrid Parsing: Supertagging

- A supertag might predict
  - the edge label
  - the subordination direction (left/right/not)
  - labels of complements (pre- and postmodifiers)
  - labels of all modifiers
  - the exact sequence of modifiers
- We define four subpredictions that can be made:
  label, direction, premodifiers, postmodifiers
- different combinations of these can be tested
- accuracy can be measured by exact tag or by subprediction

Hybrid Parsing: Supertagging

- transform the NEGRA corpus into dependency format
- extract a generalized supertag for each word:
  `PP+S/N+AUX,KON,SUBJ`
- project these onto the various features sets
- re-train TnT on these data
- call this model before parsing
- integrate the predictions with four new constraints, e.g.:

```
{X:SYN} : 'ST:direction' : stat : 0.9 :
  X/ & predict(X↓id, ST, dir) = R |
  X\ & predict(X↓id, ST, dir) = L |
  X| & predict(X↓id, ST, dir) = N;
```

Hybrid Parsing: Supertagging

Model	edge label	edge direction	dependent labels	dependent directions	#tags	Supertag accuracy	Component accuracy
A	yes	no	none	no	35	84.1%	84.1%
B	yes	yes	none	no	73	78.9%	85.7%
C	yes	no	obligatory	no	914	81.1%	88.5%
D	yes	yes	obligatory	no	1336	76.9%	90.8%
E	yes	no	obligatory	yes	1465	80.6%	91.8%
F	yes	yes	obligatory	yes	2026	76.2%	90.9%
G	yes	no	all	no	6858	71.8%	81.3%
H	yes	yes	all	no	8684	67.9%	85.8%
I	yes	no	all	yes	10762	71.6%	84.3%
J	yes	yes	all	yes	12947	67.6%	84.5%

- larger tag sets are generally harder to predict
- but not always, e.g. direction is particularly difficult
- richer context compensates that up to a point

Hybrid Parsing: Supertagging

- The best supertag model J increases structural parsing accuracy from 89.3% to 91.9%
- surprising: big improvement even though 1/3 of all supertags are wrong
- of course, *perfect* supertags would bring us to 97.2%...
- → supertagging is rightly called "almost parsing"
- Nevertheless, we can now directly benefit from future supertag research

Hybrid Parsing: Chunk Parsing

- ABNEY 1991: two-stage parsing model
  "[When I read] [a sentence],
  [I read it] [a chunk] [at a time]."
- syntax within chunks is regular, chunk attachment is more complex
- advantage: small-scale ambiguities are not multiply combinatorially
- has been successfully used to speed up some parsers
- WCDG could profit e.g. from noun phrase detection

Hybrid Parsing: Chunk Parsing

Problems with the chunk assumption:

- Abney assumes chunks for psycholinguistic reasons
- assumption: chunks cannot nest, but clauses can
- this seems to be untrue for German:
  "Die Verfassung und **[das [von [den Organen] [der Union] [in Ausübung] [der [der Union] übertragenen Zuständigkeiten] gesetzte] Recht]** haben Vorrang vor dem Recht der Mitgliedstaaten."
  "The Constitution and **[the laws created] [by organs] [of the Union] [in the exercise] [of the authority vested] [in the Union]** take precedence over the laws of the member states."

Hybrid Parsing: Chunk Parsing

Method:

- compute chunk boundaries with TreeTagger (SCHMID 1994)
- choose a *head* for each chunk with simple rules
- constraints require the following:
  - all words modify heads
  - only head words attach outside their chunk

```
{X!SYN} : 'chunk-1' : chunker : 0.9 :
 X↑to > X↓chunk_end |  X↑from < X↓chunk_start ->
 chunk_head(X↓id);
{X!SYN} : 'chunk-2' : chunker : 0.9 :
 chunk_head(X↓id) ->
 X↑to > X↓chunk_end |  X↑from < X↓chunk_start;
```


Hybrid Parsing: Chunk Parsing

- parsing accuracy increases to 89.8%
- this is only an error rate reduction of 5%
- why does chunk parsing not help more?
  - Bad input: perhaps the chunk parser is too inaccurate
    → no: an idealized chunker does hardly better
  - Bad integration: perhaps too much weight for the new constraint?
    → no: other constraint weights do not improve things
  - Bad idea: it seems that WCDG does not make many errors at the chunk level as it is.

Hybrid Parsing: PP Attachment

Label	occurred	retrieved	percentage	no. of errors
	2350	2350	100.0	0
DET	2030	2001	98.6	29
PN	1725	1684	97.6	41
PP	1695	1133	**66.8**	562
ADV	1235	936	75.8	299
SUBJ	1210	1130	93.4	80
ATTR	1143	1106	96.8	36
S	1142	997	87.3	145
AUX	635	595	93.7	40
OBJA	604	522	86.4	82

Spot the obvious weak link in our rule set!

Hybrid Parsing: PP Attachment

- almost one word in 30 is a mis-attached preposition
- the problem is well-known to be difficult
- many different factors contribute to PP attachment
- some of them are very hard to formalize
- (all the following examples also work in German)
- "The girl glanced idly through the telescope."
  → "through" cannot modify "girl" because of projectivity
- "The statue in the harbour was made in France."
  → "in" cannot modify "was", because "statue" already does

Hybrid Parsing: PP Attachment

- "The bill was vetoed by the House of Lords."
  → "House of Lords" is a fixed expression
- "This chair was instituted on the orders of the king."
  → "on the orders" is incomplete without a preposition
- "The holding bought 1,000,000 shares for \$15 a share."
  → "shares for \$15 a share" would be bad style
- "Please wash the infection with soap."
  → "infection with soap" would make no sense
- "I bought a stamp for sixpence."
  → "for" might modify either "bought or "stamp" with no meaning change

Hybrid Parsing: PP Attachment

- so far we use only syntactic and some idiom rules
- many of the other criteria are almost inexpressible
- some of them might be approximated by simple lexicalization
- but word-specific constraints would number many millions
- again, empirical knowledge might be helpful

Hybrid Parsing: PP Attachment

Luckily, data on prepositions are plentiful.

- we own around 100,000 trees of German sentences
- raw text is available almost without limitation
- idea: assume that co-occurrence correlates with subordination
- in trees, count preposition/head pairs
- in raw text, count co-occurrence of prepositions and nouns/verbs
- assume that subordination preference can be approximated as

$$LA(w,p) := \frac{f_{w+p}}{t} \Big/ \left(\frac{f_w}{t} \cdot \frac{f_p}{t}\right)$$

Hybrid Parsing: PP Attachment

- the assumption that co-occurrence ≈ subordination is far truer of verbs than of nouns
- previous research (VOLK 2001) suggests the following method:
  - count co-occurring verbs and prepositions
  - count only nouns and adjacent prepositions
  - unskew the counts with a noun correction factor
- because of data sparseness, undo compounding, inflection etc. before counting
- treat all numbers, proper names etc. as one class
- treat verbs occurring with different separable prefixes as different verbs

Hybrid Parsing: PP Attachment

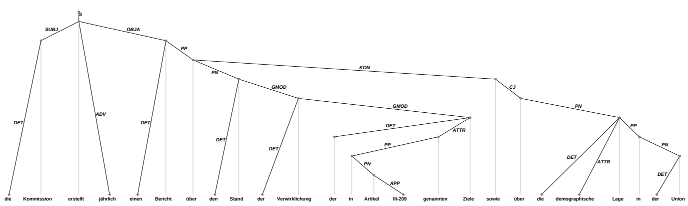
- LA scores must be mapped to constraint weights somehow
- we choose: $p(w,p) = \frac{\max(1, \min(0.8, 1-(2-\log_3(LA(w,p)))/50))}{\mu}$
- this ensures a constraint penalty between 0.8 and 1
- the exact formula is relatively unimportant
- we use supervised counts and back off to unsupervised counts where $f_w < 1000$
- parsing accuracy rises from 89.3% to 90.6%
- this means that about half the preposition attachment errors are corrected

Hybrid Parsing: PP Attachment

- we achieve half the theoretically possible gain
- that is rather good, considering our attacher is far below the state of the art
- for instance, we ignore potentially useful kernel nouns
  "Buy the car with the warranty" vs.
  "Buy the car with your credit card"
- co-occurrence of more than two words can be misleading:
  "The ministers met last Friday in Rome."
  → "Friday" and "in are correlated,
  but virtually never subordinated
- "We talk to the director of over 50 films."
  → both "to" and "of" are good modifiers for "talk",
  but not at the same time

Hybrid Parsing: Shift-Reduce Parsing

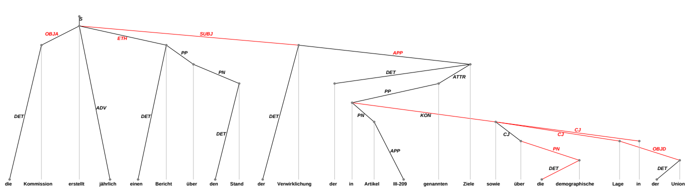
Here is a sentence that appears to be not particular difficult:



"The Commission compiles yearly reports about the state of the realization of the goals named in §III-209, and about the demographic situation in the Union."

Hybrid Parsing: Shift-Reduce Parsing

Yet it is analysed quite wrongly by our parser.



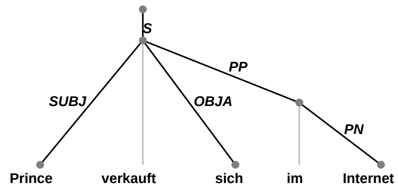
(Well, not really — only if you give it too little time.)

Hybrid Parsing: Shift-Reduce Parsing

- even a much simpler model could have gotten this sentence right
- but simple models don't have all the nice coverage
- again, we should combine advantages of both worlds
- a fast parser could deliver an initial guess ...
- ... and transformation can choose a better solution if it finds one
- this would improve both anytime behaviour and accuracy in the limit

Hybrid Parsing: Shift-Reduce Parsing

- we choose a deliberately simple model
- shift-reduce parsing (NIVRE 2003) creates projective dependency trees in linear time
- idea: at each word, on of four moves can be made: shift onto stack, reduce from stack, attach to the right, or attach to the left



Hybrid Parsing: Shift-Reduce Parsing

- Here's the correct sequence of moves:

Configuration	Move	Edge
[ ] . Prince verkauft sich im Internet	SHIFT	
[Prince] . verkauft sich im Internet	LEFT	Prince $\bullet\xrightarrow{\text{SUBJ}}$ verkauft
[ ] . verkauft sich im Internet	SHIFT	
[verkauft] . sich im Internet	RIGHT	sich $\bullet\xrightarrow{\text{OBJA}}$ verkauft
[verkauft sich] . im Internet	REDUCE	
[verkauft] . im Internet	RIGHT	im $\bullet\xrightarrow{\text{PP}}$ verkauft
[verkauft im] . Internet	RIGHT	Internet $\bullet\xrightarrow{\text{PN}}$ im
[verkauft im Internet] . $	REDUCE	
[verkauft im] . $	REDUCE	
[verkauft] . $	REDUCE	verkauft $\bullet\xrightarrow{\text{S}}$ null

- note that precisely $2n$ moves are necessary
- but how do we know which move to make?
- Nivre originally made the first move that was possible
- we need an arbitration policy

Hybrid Parsing: Shift-Reduce Parsing

- we must train a parse move predictor for a given state
- by iterating it, we get a predictor for complete trees
- optionally, we could annotate LEFT and RIGHT with edge labels
- what features in a parse state could we use for training?
  - the top stack word (word form or POS tag)
  - the context of this word (its regent and its dependents so far)
  - the next input word
  - the distance between both words
  - the words in a fixed lookahead window

Hybrid Parsing: Shift-Reduce Parsing

- policy: simple tuple counting
- back-off to simpler tuples in fixed order
- features: 2 words lookahead, edge labels, no lexicalisation
- no exhaustive tuning was attempted
- result: 85% structural accuracy on NEGRA
- the oracle solves the earlier example sentence correctly (that's why it was chosen)

Hybrid Parsing: Shift-Reduce Parsing

- again, constraints prefer edges that match the predictions:

```
{X!SYN} : 'SR:regent' : stat : 0.9 :
  predict(X↓id, SR, gov) = X↑to;

{X|SYN} : 'SR:NIL' : stat : 0.9 :
  predict(X↓id, SR, gov) = 0;

{X:SYN} : 'SR:Label' : stat : 0.9 :
  predict(X↓id, SR, lab) = X.label;
```

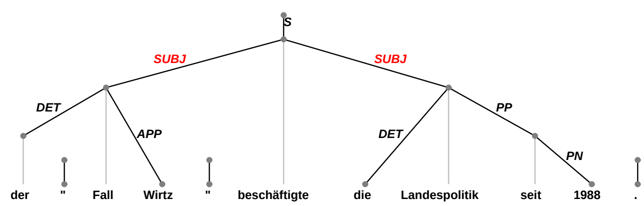
- the 0.9 *was* tuned exhaustively
- it guarantees that transformation usually starts from the exact oracle parse

Hybrid Parsing: Shift-Reduce Parsing

- parsing accuracy improves from 89.3% to 91.5%
- both parsers in combination are a lot better than either
- WCDG runs an order of magnitude faster at the same quality level
- makes you wonder what a really good oracle parser would do for us
- unfortunately, the published German stochastic parsers all generate phrase-structure
- what could be improved about our oracle?

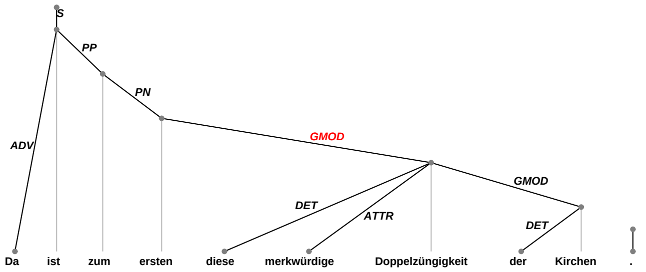
Hybrid Parsing: Shift-Reduce Parsing

Here are some obvious defects:



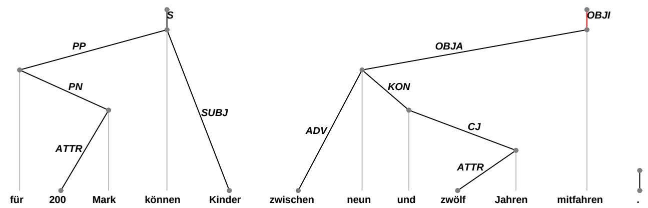
- Why are two subjects predicted?
- → Because our context feature is not that detailed!
- (Alas, extending it does not improve accuracy.)

Hybrid Parsing: Shift-Reduce Parsing



- How can it be a GMOD without a genitive determiner?
- Again, our model does not look that far into the context
- This could be captured with a corner feature (YAMADA ET AL. 2003)

Hybrid Parsing: Shift-Reduce Parsing



- Why is the subclause assumed to be a fragment?
- Well, that's what we have to do to our treebank before training. . .
- Newer approaches allow nonprojective training data (NIVRE 2005)

Hybrid Parsing: Shift-Reduce Parsing

- our parse state model assumed a fixed set of features
- a model that can adapt its feature selection would be more adequate
- e.g. decision trees, support vector machines. . .
- however, this simple model is enough to prove the value of the hybrid approach
- again, as shallow parsing improves, we can plug it right in

Hybrid Parsing: Combining Predictors

- What if we use more than one predictor?

		Accuracy	
Experiment	Predictors	structural	labelled
1	none	72.6%	68.3%
2	TnT	89.0%	87.1%
3	hybrid POS	89.3%	87.5%
4	POS+CP	89.8%	88.0%
5	POS+ST	91.9%	90.5%
6	POS+PP	90.6%	88.9%
7	POS+SR	91.5%	89.8%
8	POS+PP+SR	91.4%	89.6%
9	POS+PP+ST	92.0%	90.6%
10	POS+ST+SR	92.2%	90.7%
11	all five	92.3%	90.9%

- three predictors are even better than two
- all five predictors are best
- spot the exception!

Hybrid Parsing: Combining Predictors

- the only unsuitable hybrid model is PP attachment + shift-reduce parsing

		Accuracy	
Experiment	Predictors	structural	labelled
3	hybrid POS	89.3%	87.5%
6	POS+PP	90.6%	88.9%
7	POS+SR	91.5%	89.8%
8	POS+PP+SR	91.4%	89.6%

- the SR parser predicts regents of prepositions. . .
- . . . but without lexical information
- this duplication of work is apparently harmful
- exempting prepositions from SR predictions fixes the anomaly

Hybrid Parsing: Lessons Learnt

- POS tagging is an enabling technology for WCDG
- stochastic models can replace grammar writing with data collecting
- cheap, simple empirical models can usefully complement a heavy-weight deep model
- two parsers are better than either one
- good heuristics can massively affect the time/space trade-off

Future Work: Morphology Tagging

"Man hat uns[case:dat] ein[case:acc,gender:neut] Angebot[case:acc] gemacht, das[case:acc] wir nicht ablehnen konnten[person:first]."

- Syntactic category prediction is very useful, but morphology predictions would be even more useful
- German adjectives have up to 26 underlying feature combinations
- In simulation, morphology information adds another 1% of parsing accuracy
- However, that is with an error-free morphology tagger
- How good must a morphology component be to yield a net benefit?

Future Work: Named Entity Tagging

"Die [IBM Visual Age Micro Edition] läuft jetzt auch unter [Mac OS X]."

- Multiword expressions often function like single words: "Es spielt das NDR Sinfonieorchester."
- Long names, titles, institutions etc. create many opportunities for totally wrong subordinations
- At the same time they are highly recognizable
- → we should pre-detect these and reduce ambiguity considerably

Future Work: Nuclei

- Nuclei are actually an ancient concept in dependency grammar (TESNIÈRE 1959)
- could be useful for other things than proper names:
  - verb phrases: "Ich weiß nicht, was ich [hätte tun sollen]."
  - category-changing idioms: "Es sieht [alles andere als] gut aus."
- a preprocessor could replace known nuclei with new word hypotheses
- problem: idioms can have compositional homonyms
- solution: WCDG can already deal with alternatives in lattices

Future Work: Solution Methods

- Feature-based stochastic dependency tree learning (MACDONALD ET AL. 2005)
  - Edge probabilities are learnt solely through *unary* features. . .
  - . . . but over 13,000,000 of them
  - Extra work to guarantee the result is a tree
- Example-based parsing (KONG ET AL. 1998)
  - We have many thousands of ready-parsed sentences
  - Yet we often fail to produce good results for very similar sentences if they are very long
  - The known structure should be utilized at least as an initial guess

Future Work: Applications

- extend the reference resolution capabilities
  - So far we do only relative pronouns
  - But personal pronouns, possessives, nouns and even verbs can also refer
  - Problem: antecedents are often found in previous sentences
- test psycholinguistic adequacy claims on the parser
  - needed: left-to-right incrementality
  - needed again: multi-level representations
- use the diagnostic ability for language learning purposes
  - optimizing the grammar for non-native language
  - disambiguating multi-level representations