Shallow Natural Language Parsing

Günter Neumann

LT lab, DFKI

(includes modified slides from Steven Bird & Junichi Tsujii)

Parsing of unrestricted text

- Complexity of parsing of unrestricted text
  - Robustness
  - Large sentences
  - Speed
  - Input texts are not simply sequences of word forms
    - Textual structure (e.g., enumeration, spacing, etc.)
    - Combined with structual annotation (e.g., SGML tags)

Light Parsing: Overview

- Difficulties with full parsing
- Motivations for parsing
- Light (or “partial”) parsing
- Chunk parsing (a type of light parsing)
  - Introduction
  - Advantages
  - Implementations
- SMES: a German Shallow Parser

Full Parsing

Goal: build a *complete parse tree* for a sentence.

- Problems with full parsing:
  - Low accuracy
  - Slow
  - Domain Specific
- These problems are relevant for both symbolic and statistical parsers

Full Parsing: Accuracy

Full Parsing gives relatively low accuracy

- Exponential solution space
- Dependence on semantic context
- Dependence on pragmatic context
- Long-range dependencies
- Ambiguity
- Errors propagate

Full Parsing: Domain Specificity

Full parsing tends to be domain specific

- Importance of semantic/lexical context
- Stylistic differences

Full Parsing: Efficiency

Full parsing is very processor-intensive and memory-intensive

- Exponential solution space
- Large relevant context
  - Long-range dependencies
  - Need to process lexical content of each word
- Too slow to use with very large sources of

text (e.g., the web).

Motivations for Parsing

- Why parse sentences in the first place?
- Parsing is usually an intermediate stage
  - Builds structures that are used by later stages of processing
- Full Parsing is a *sufficient* but not *necessary* intermediate stage for many NLP tasks.
- Parsing often provides more information than we need.

Light Parsing

Goal: assign a *partial structure* to a sentence.

- Simpler solution space
- Local context
- Non-recursive
- Restricted (local) domain

Output from Light Parsing

- What kind of *partial structures* should light parsing construct?
- Different structures useful for different tasks:
  - Partial constituent structure

    [NP I] [VP saw [NP a tall man in the park]].
  - Prosodic segments (phi phrases)

    [I saw] [a tall man] [in the park]
  - Content word groups

    [I] [saw] [a tall man] [in the park].

Chunk Parsing

Goal: divide a sentence into a sequence of chunks.

- Chunks are non-overlapping regions of a text

  [I] saw [a tall man] in [the park]

- Chunks are non-recursive
  - A chunk can not contain other chunks
- Chunks are non-exhaustive
  - Not all words are included in the chunks

Chunk Parsing Examples

- Noun-phrase chunking:
  - [I] saw [a tall man] in [the park].
- Verb-phrase chunking:

  The man who [was in the park] [saw me].
- Prosodic chunking:

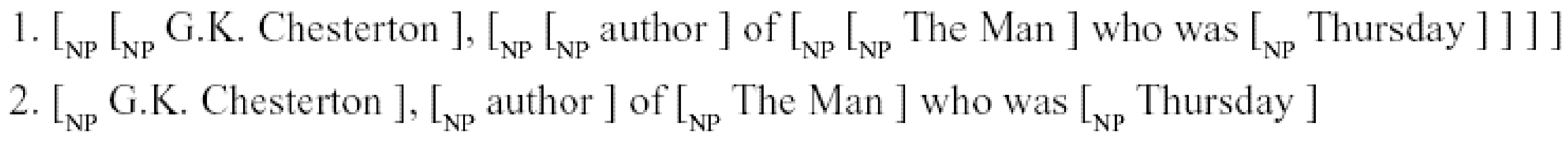
  [I saw] [a tall man] [in the park].

Chunks and Constituency

Constituents: ***[[a tall man] [in [the park]]].***

Chunks: ***[a tall man] in [the park].***

- A constituent is part of some higher unit in the hierarchical syntactic parse
- Chunks are *not constituents*
  - – Constituents are recursive
- But, chunks are typically subsequences of constituents
  - – Chunks do not cross major constituent boundaries



1. [$_{\text{NP}}$ [$_{\text{NP}}$ G.K. Chesterton], [$_{\text{NP}}$ [$_{\text{NP}}$ author] of [$_{\text{NP}}$ [$_{\text{NP}}$ The Man] who was [$_{\text{NP}}$ Thursday]]]]
2. [$_{\text{NP}}$ G.K. Chesterton], [$_{\text{NP}}$ author] of [$_{\text{NP}}$ The Man] who was [$_{\text{NP}}$ Thursday]

Chunk Parsing: Accuracy

Chunk parsing achieves higher accuracy

- Smaller solution space
- Less word-order flexibility *within* chunks than *between* chunks
  - Fewer long-range dependencies
  - Less context dependence
- Better locality
- No need to resolve ambiguity
- Less error propagation

Chunk Parsing: Domain Specificity

Chunk parsing is less domain specific

- Dependencies on lexical/semantic information tend to occur at levels "higher" than chunks:
  - Attachment
  - Argument selection
  - Movement
- Fewer stylistic differences with chunks

Psycholinguistic Motivations

Chunk parsing is psycholinguistically motivated

- Chunks are processing units
  - Humans tend to read texts one chunk at a time
  - Eye movement tracking studies
- Chunks are phonologically marked
  - Pauses
  - Stress patterns
- Chunking might be a first step in full parsing
  - Integration of shallow and deep parsing

Chunk Parsing: Efficiency

Chunk parsing is more efficient

- Smaller solution space
- Relevant context is small and local
- Chunks are non-recursive
- Chunk parsing can be implemented with a finite state machine
  - Fast (linear)
  - Low memory requirement (no stacks)
- Chunk parsing can be applied to a very large text sources (e.g., the web)

Chunk Parsing Techniques

- Chunk parsers usually ignore lexical content
- Only need to look at part-of-speech tags
- Techniques for implementing chunk parsing
  - Regular expression matching
  - Chinking
  - Cascaded Finite state transducers

Regular Expression Matching

- Define a regular expression that matches the sequences of tags in a chunk
  - A simple noun phrase chunk regrexp:

    <DT> ? <JJ> * <NN.?>
- Chunk all matching subsequences:

  The /DT little /JJ cat /NN sat /VBD on /IN the /DT mat /NN

  [The /DT little /JJ cat /NN] sat /VBD on /IN [the /DT mat /NN]
- If matching subsequences overlap, the first one gets priority
- Regular expressions can be cascaded

Chinking

- A *chink* is a subsequence of the text that is not a chunk.
- Define a regular expression that matches the sequences of tags in a chink.
  - A simple chink regexp for finding NP chunks:

    (<VB.?> | <IN>)+
- Chunk anything that is *not* a matching subsequence:

```
the/DT little/JJ cat/NN  sat/VBD on /IN the /DT mat/NN
[the/DT little/JJ cat/NN]  sat/VBD on /IN [the /DT mat/NN]
          chunk                 chink            chunk
```

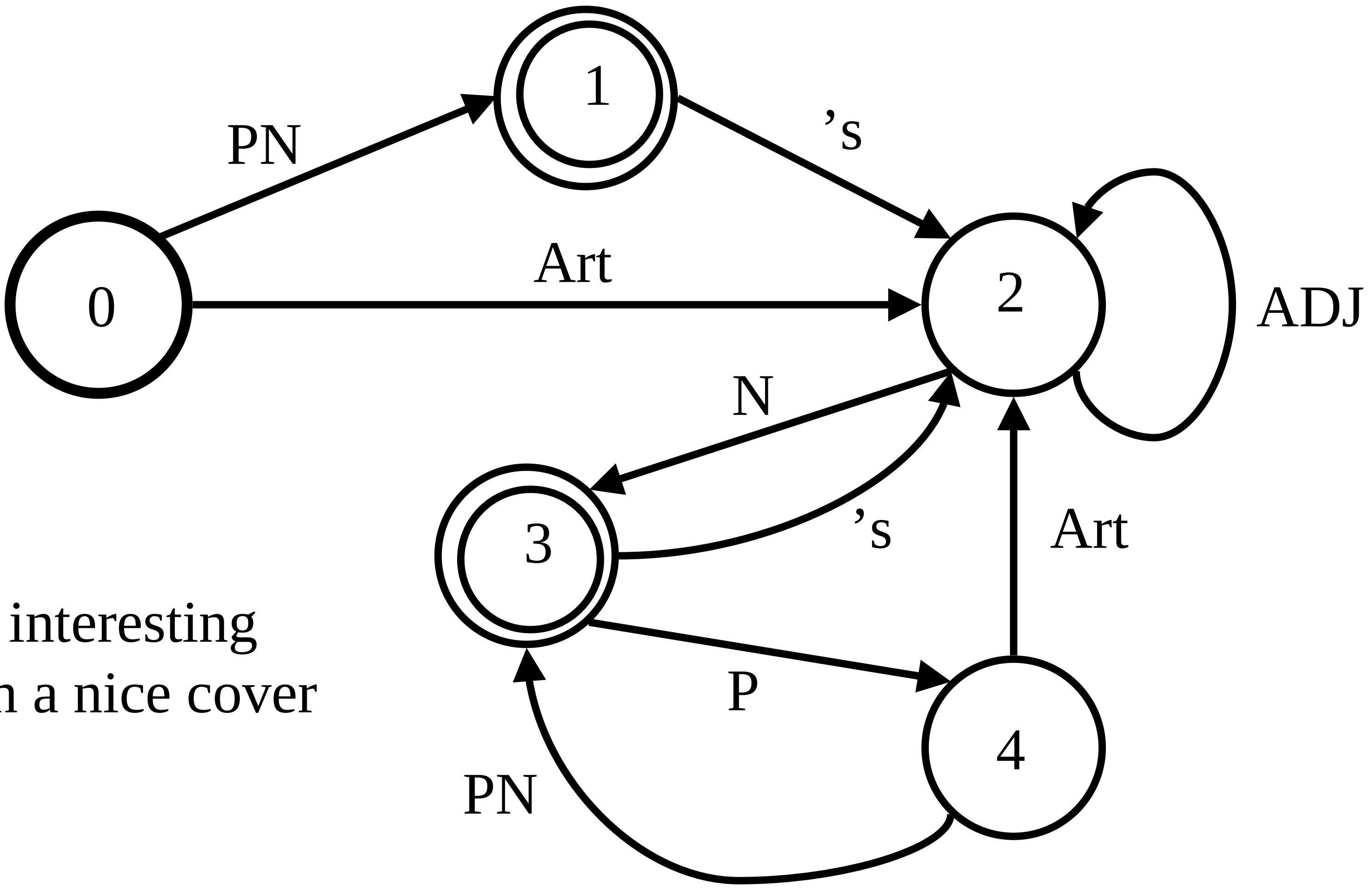
Chomsky Hierarchy of Grammar	Hierarchy of Automata
Regular Grammar	Finite State Automata
Context Free Grammar	Push Down Automata
Context Sensitive Grammar	Linear Bounded Automata
Type 0 Grammar	Turing Machine

Computationally more complex, Less Efficiency

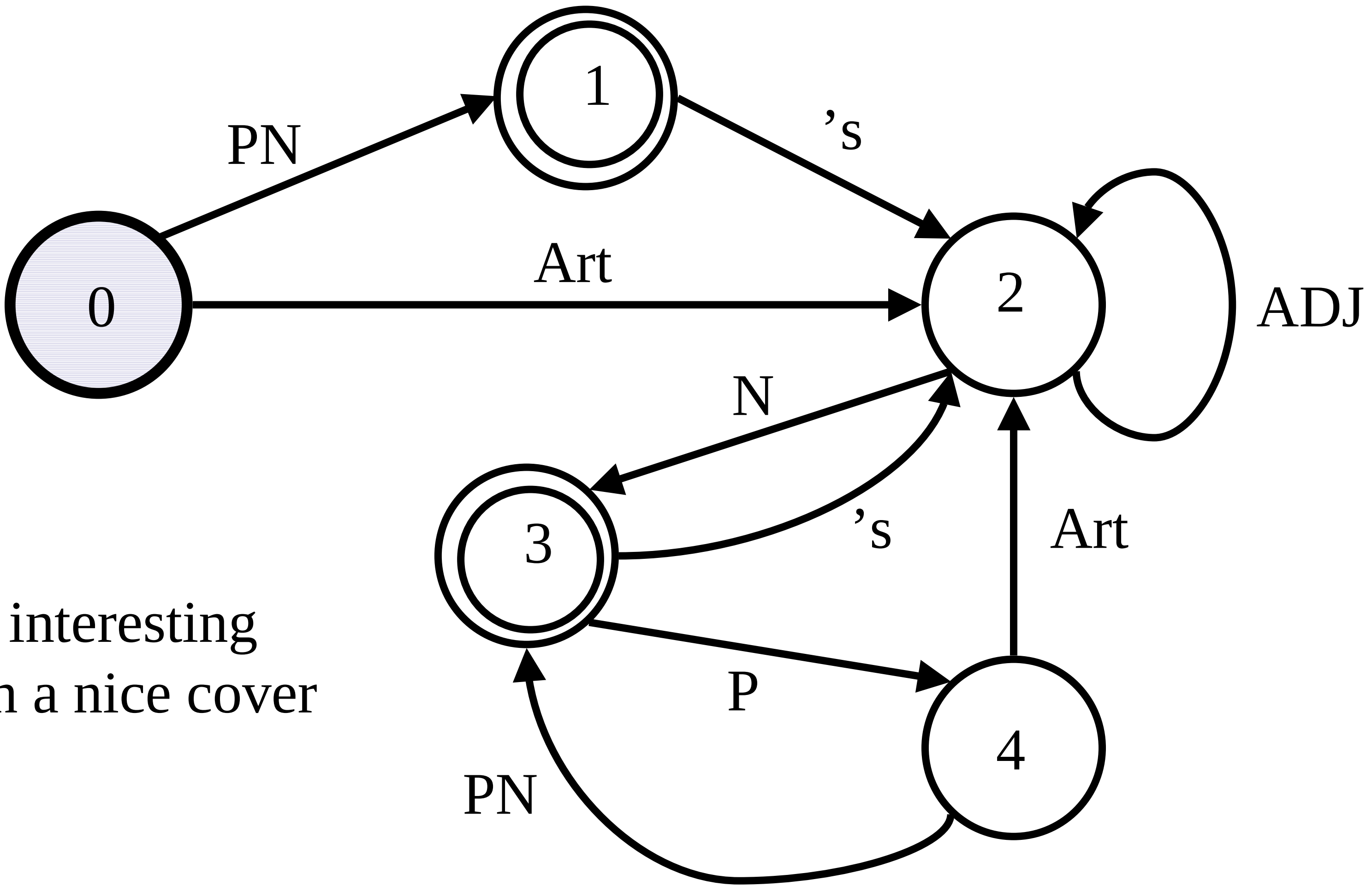
Chomsky Hierarchy of Grammar	Hierarchy of Automata
Regular Grammar $\boxed{A^nB^n}$	***Finite State Automata***
Context Free Grammar	Push Down Automata
Context Sensitive Grammar	Linear Bounded Automata
Type 0 Grammar	Turing Machine

$\downarrow$

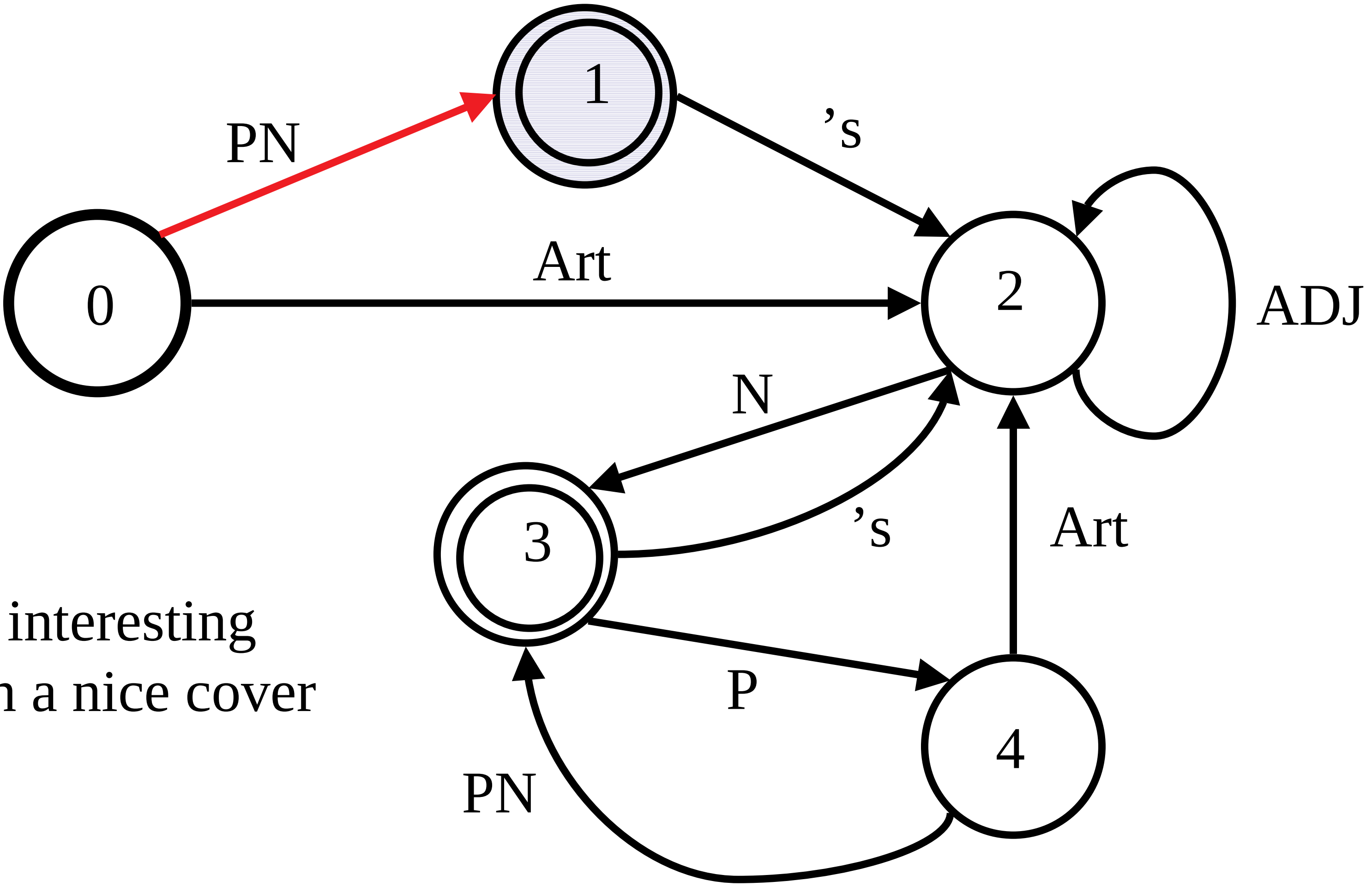
Computationally more complex, Less Efficiency



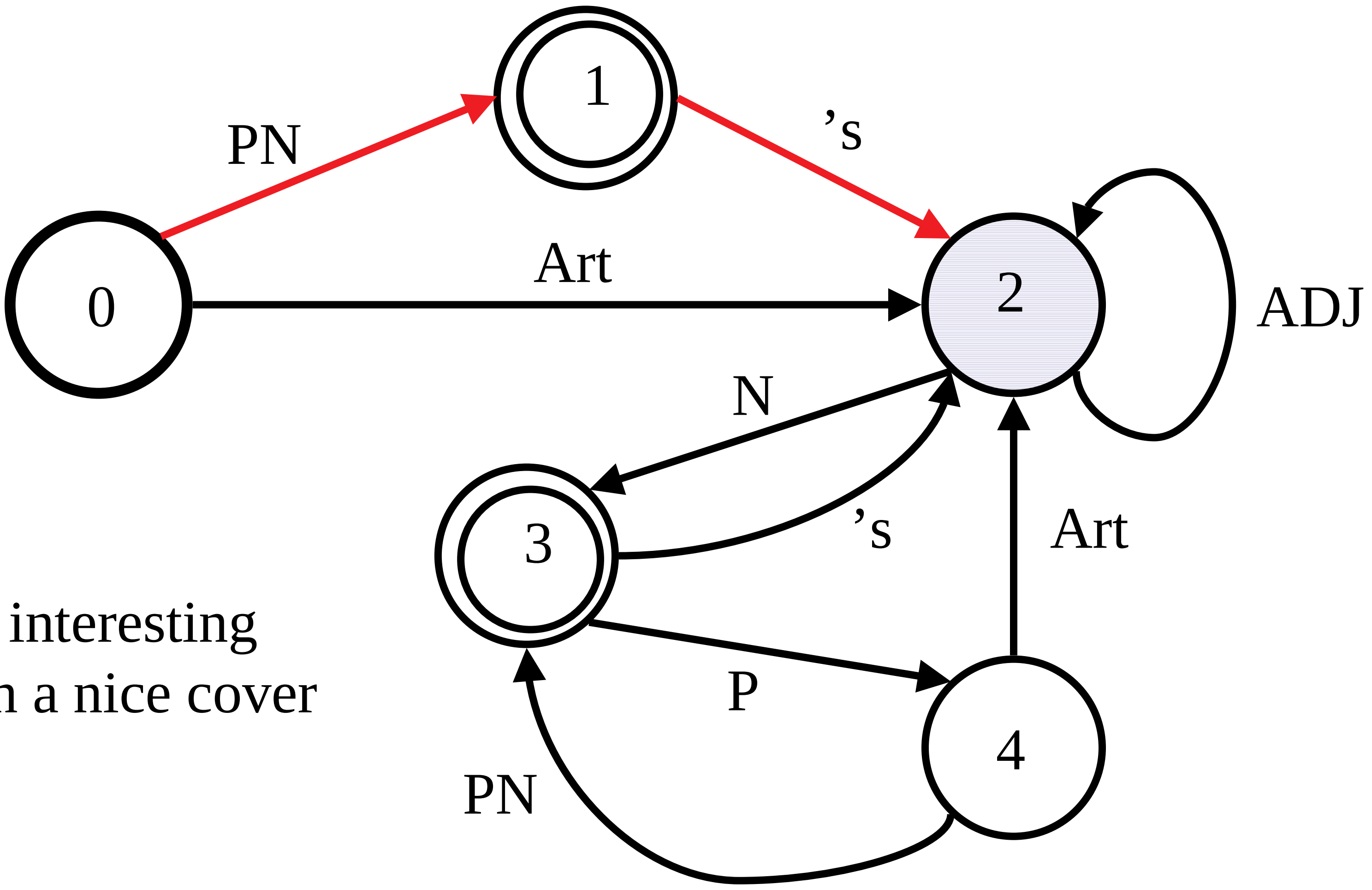
John’s interesting
book with a nice cover



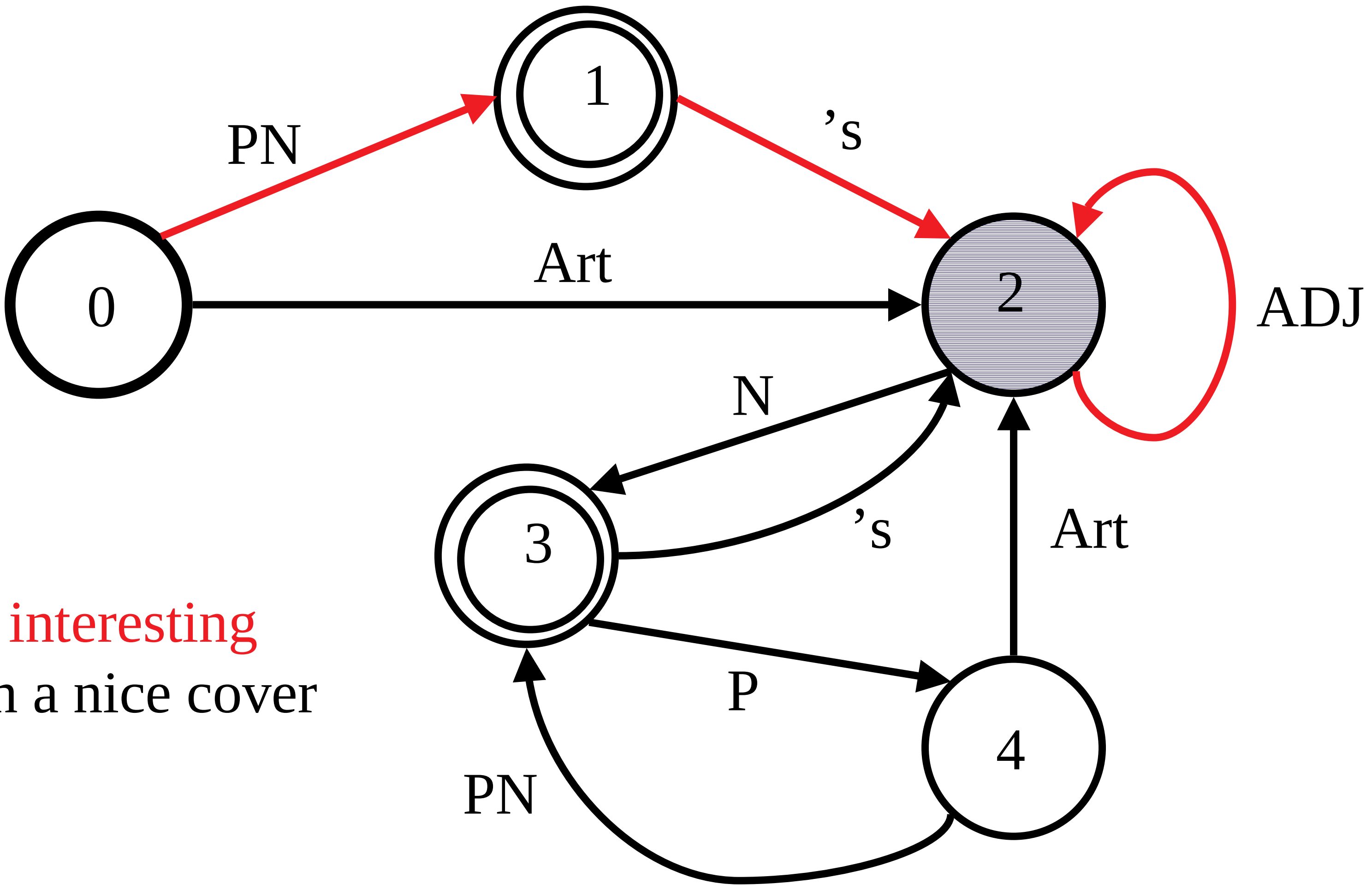
John's interesting
book with a nice cover



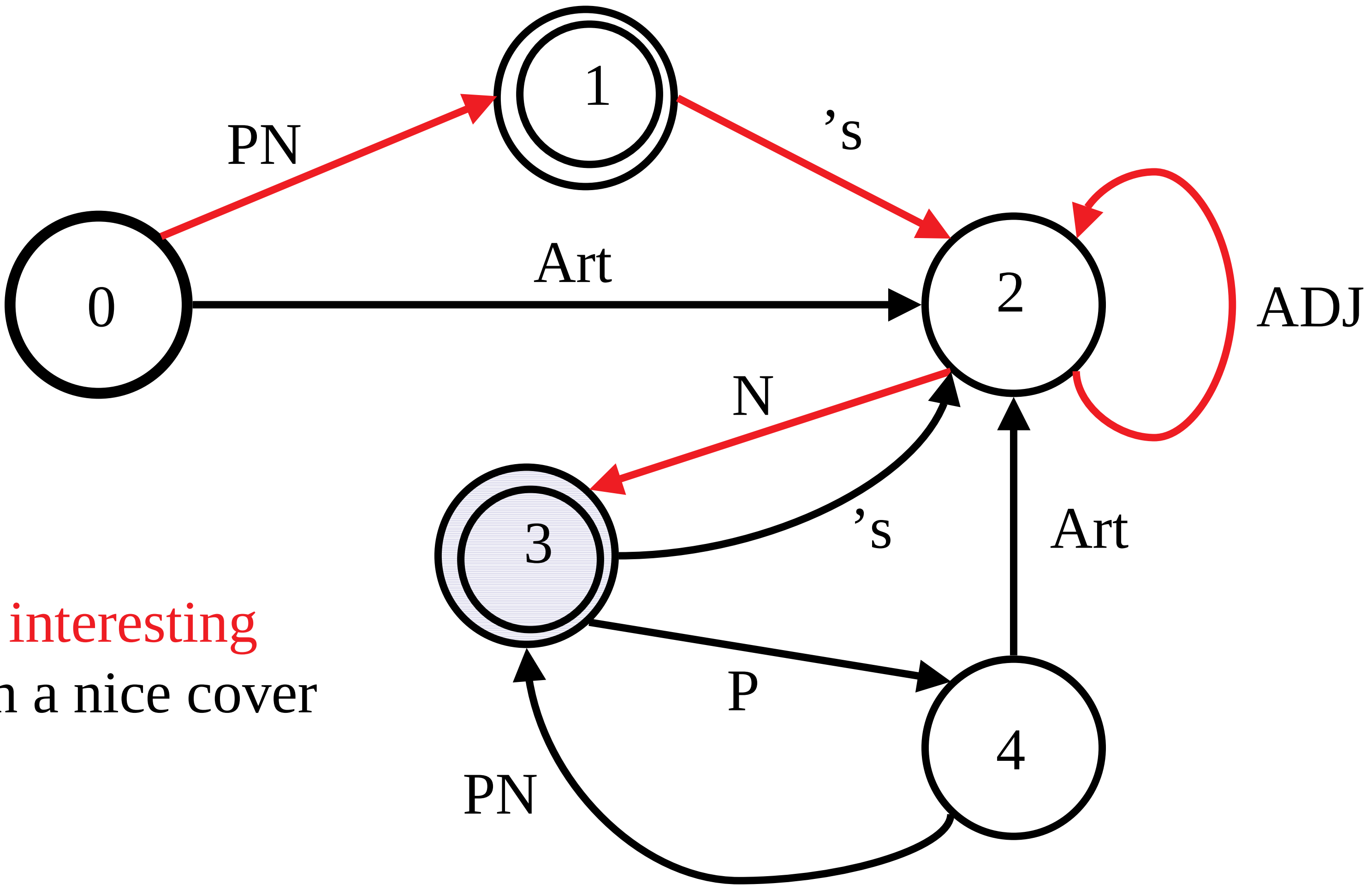
John's interesting book with a nice cover



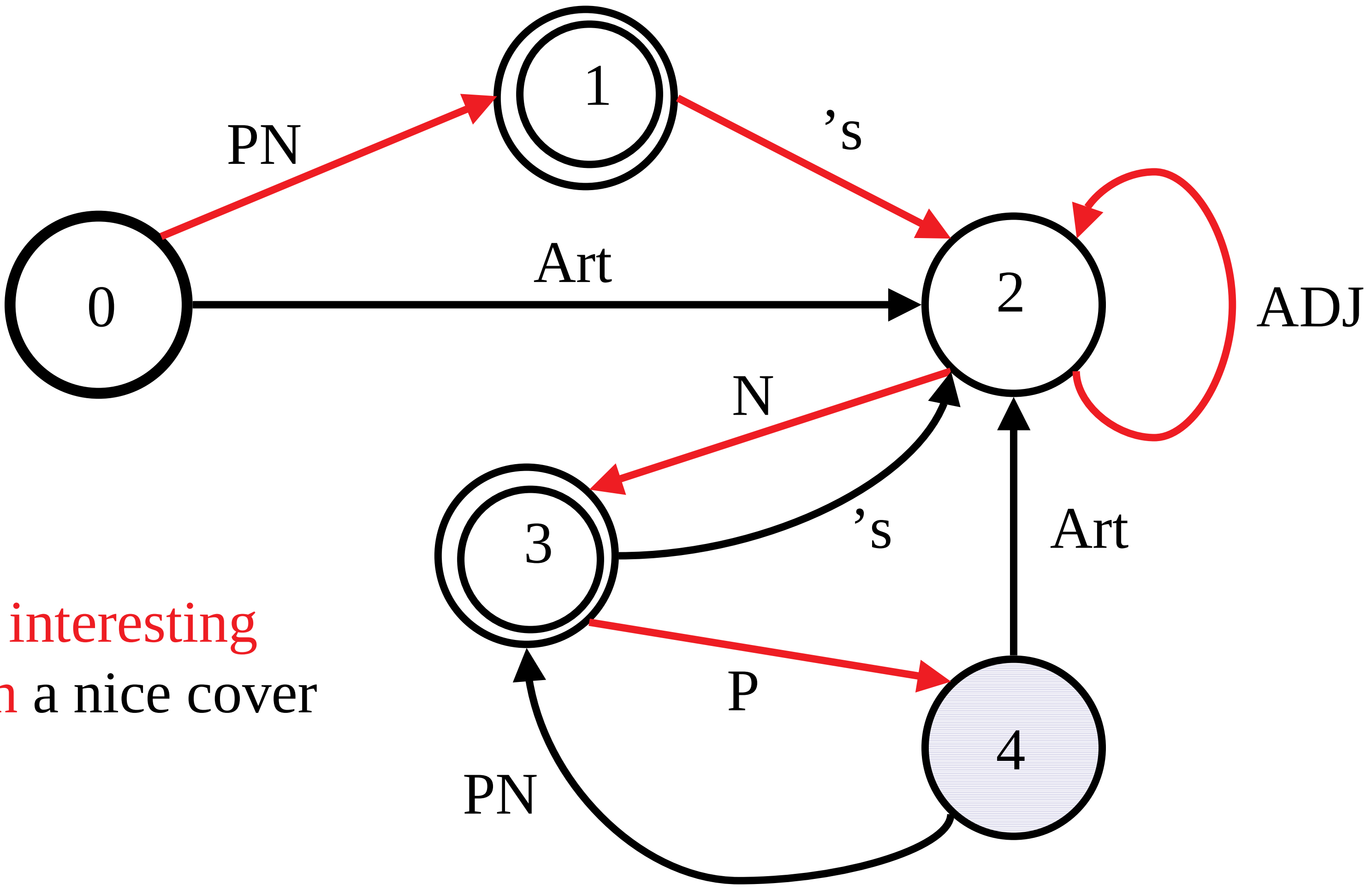
John's interesting
book with a nice cover



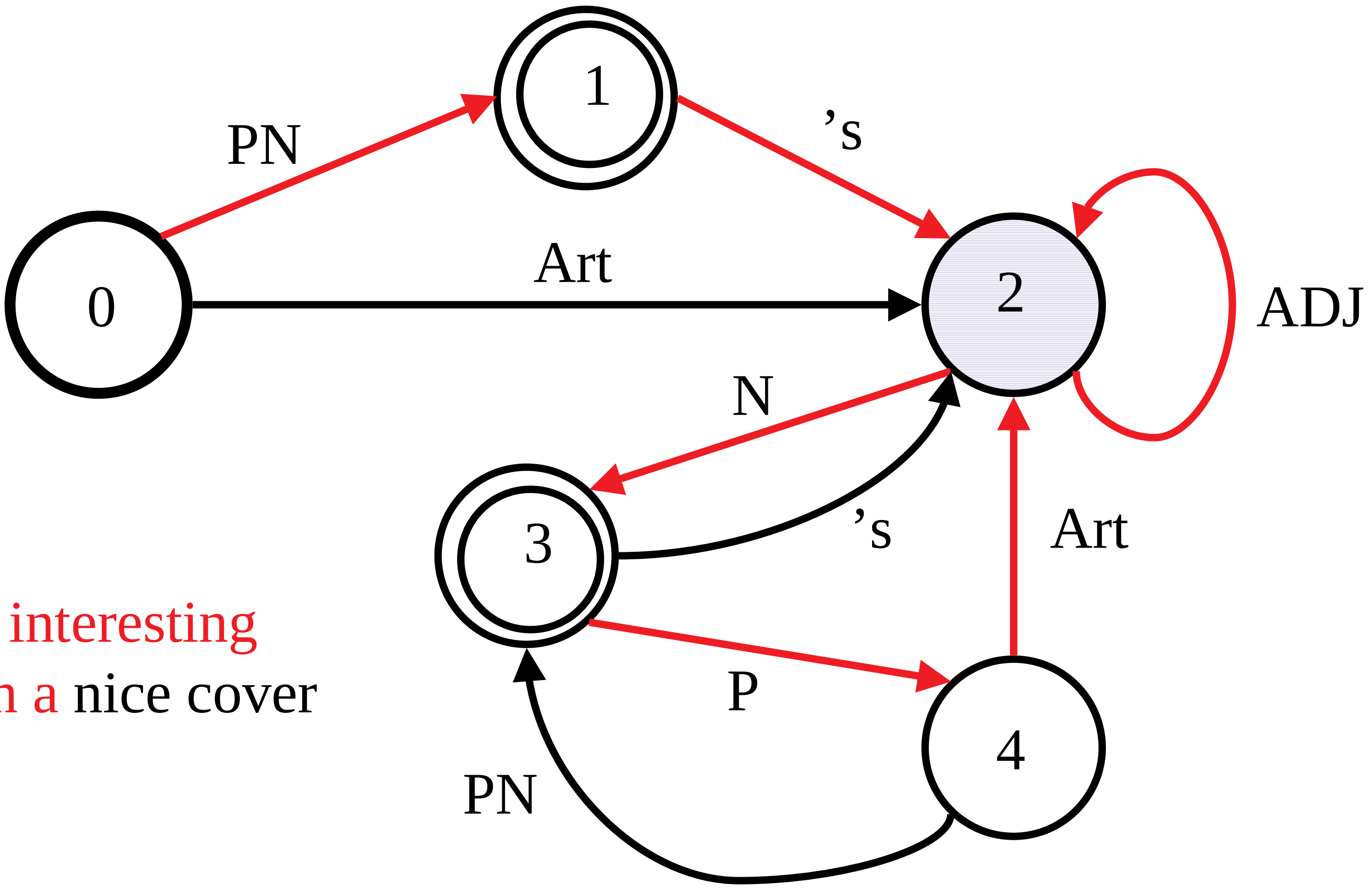
John's interesting
book with a nice cover



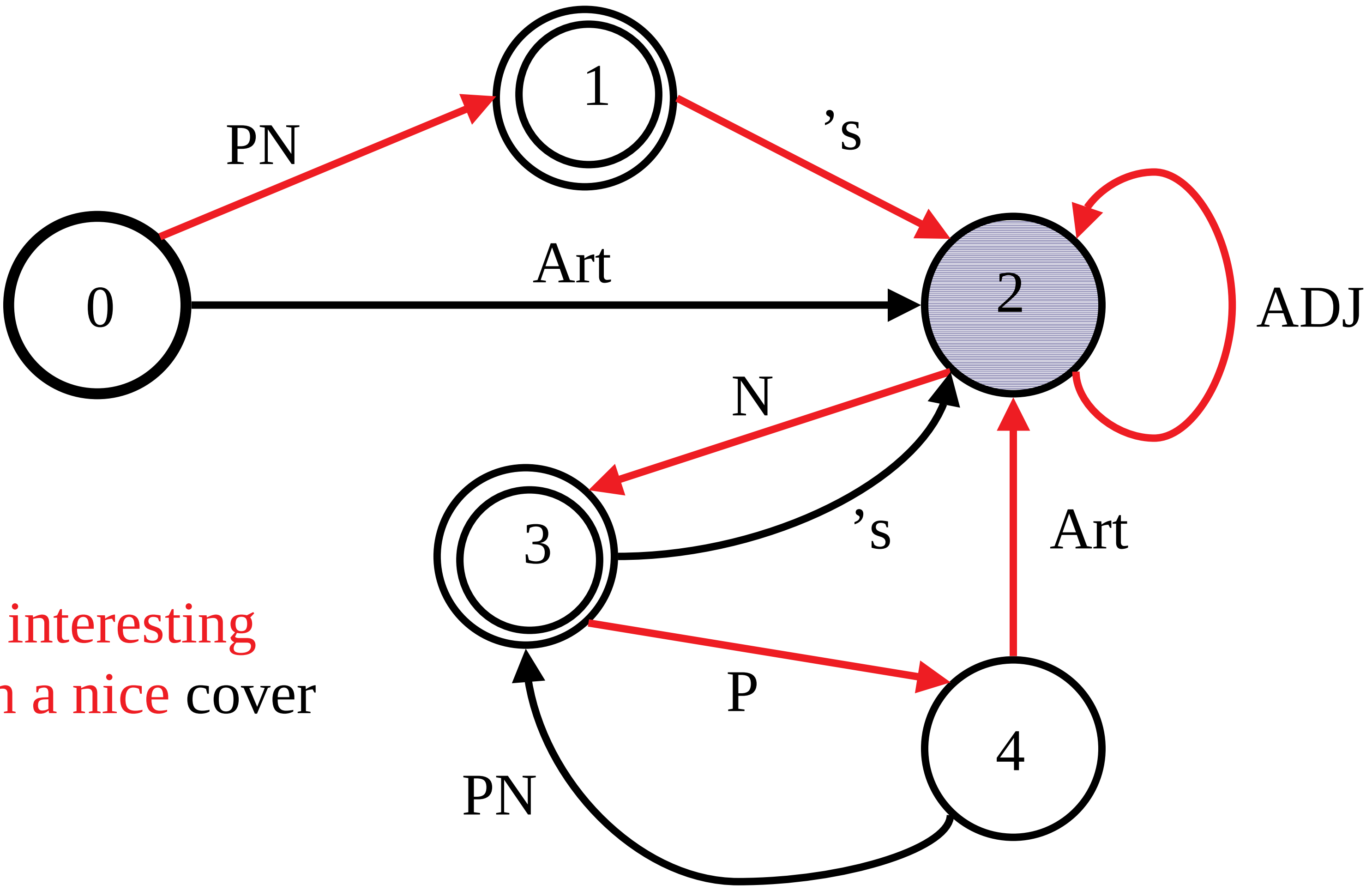
John's interesting book with a nice cover



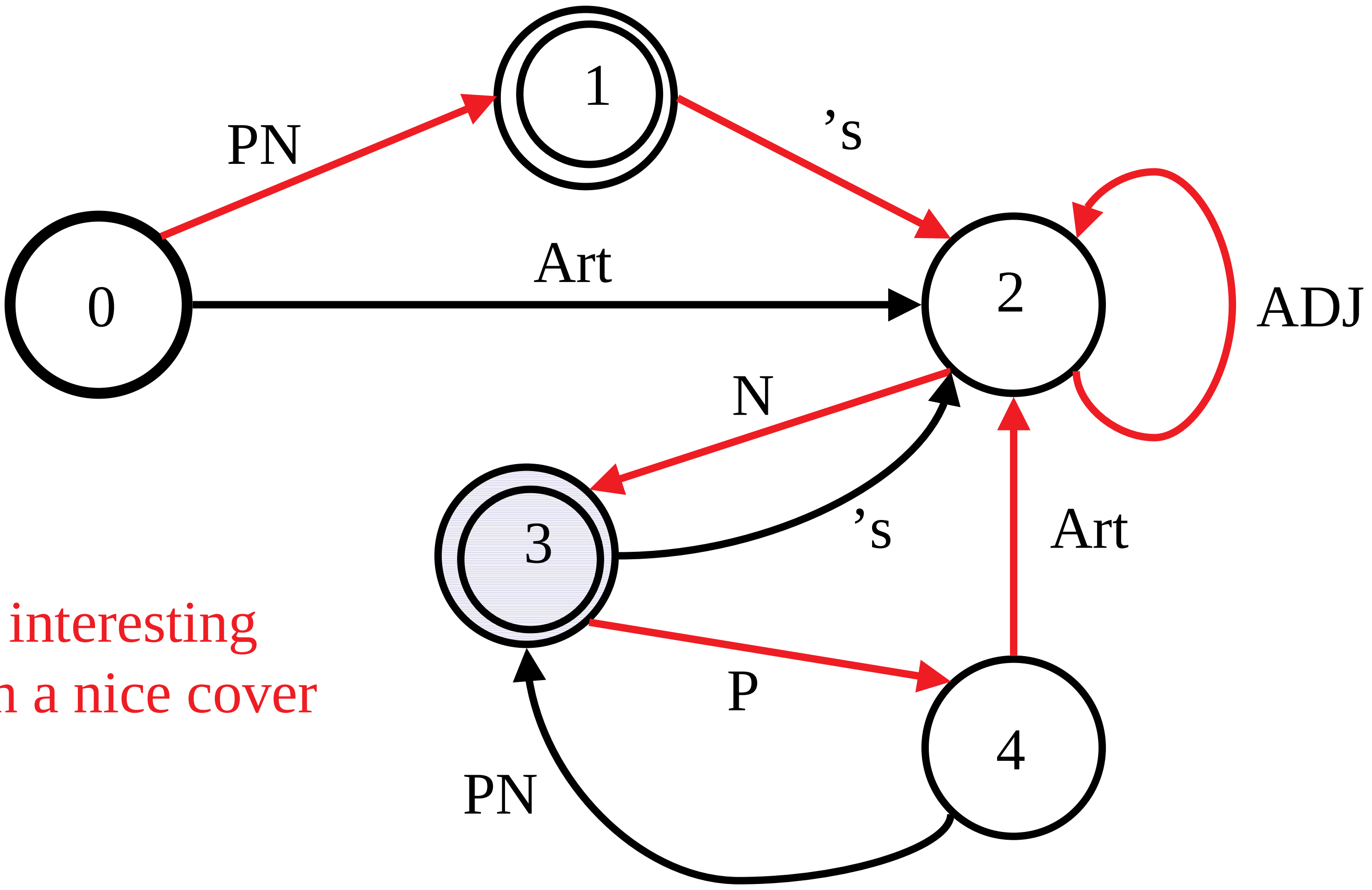
John's interesting book with a nice cover



John's interesting
book with a nice cover



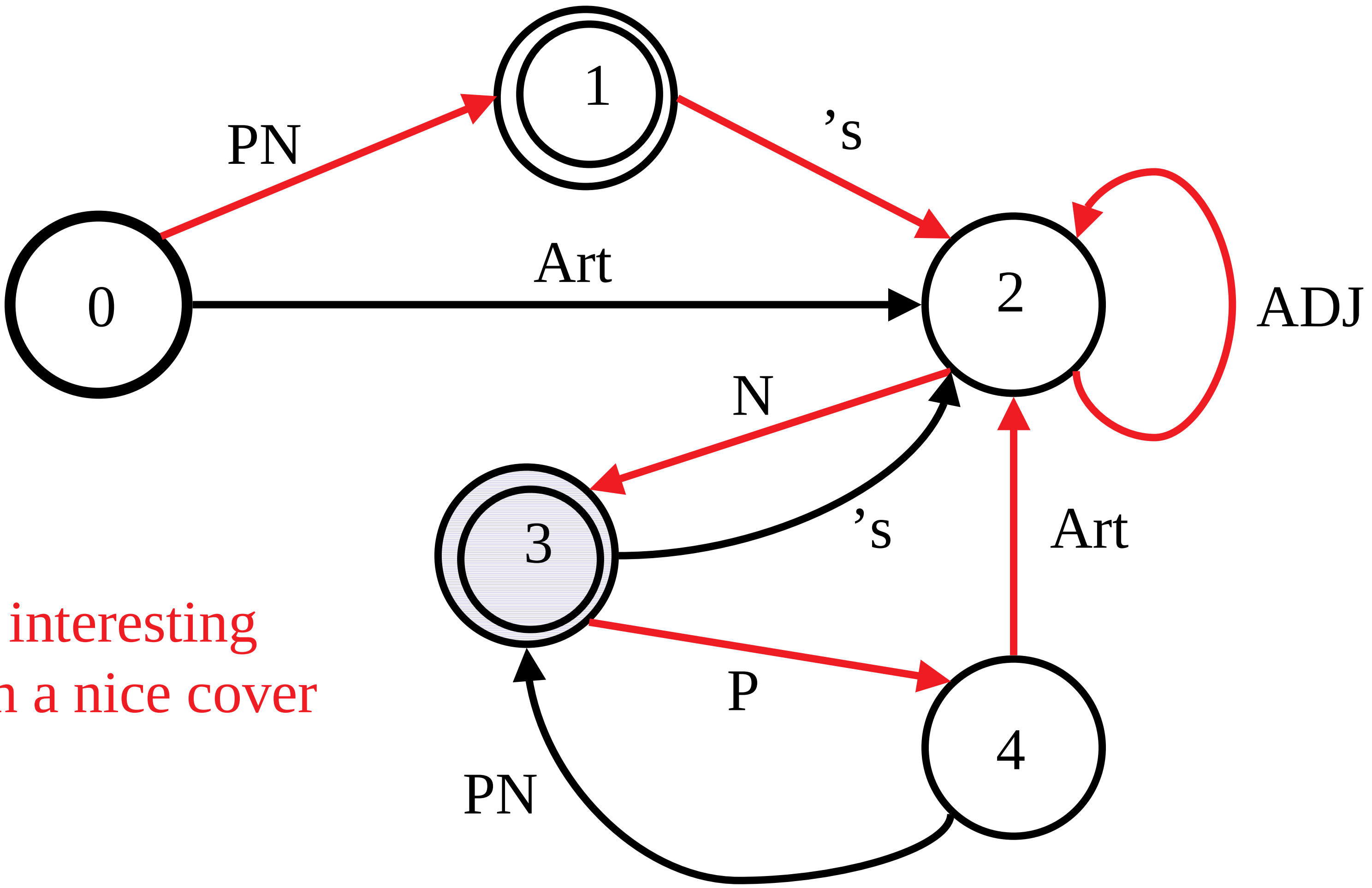
John's interesting
book with a nice cover



John’s interesting
book with a nice cover

Pattern-maching

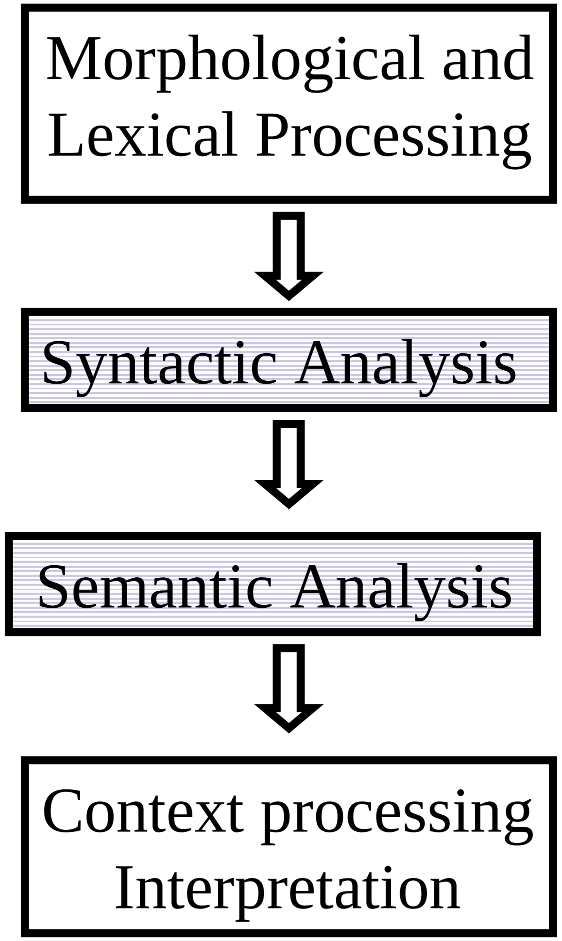
PN ’s (ADJ)* N P Art (ADJ)* N



John’s interesting
book with a nice cover

FASTUS

General Framework of NLP


Morphological and Lexical Processing
Syntactic Analysis
Semantic Analysis
Context processing Interpretation

Based on finite states automata (FSA)

1.Complex Words:
Recognition of multi-words and proper names

2.Basic Phrases:
Simple noun groups, verb groups and particles

3.Complex phrases:
Complex noun groups and verb groups

4.Domain Events:
Patterns for events of interest to the application

Basic templates are to be built.

5. Merging Structures:
Templates from different parts of the texts are merged if they provide information about the same entity or event.

Example of IE: FASTUS(1993)

Bridgestone Sports Co. said Friday it had set up a joint venture in Taiwan with a local concern and a Japanese trading house to produce golf clubs to be supplied to Japan.

The joint venture, Bridgestone Sports Taiwan Co., capitalized at 20 million new Taiwan dollars, will start production in January 1990 with production of 20,000 "metal wood" clubs a month.

1.Complex words

2.Basic Phrases:

Bridgestone Sports Co.	: Company name
said	: Verb Group
Friday	: Noun Group
it	: Noun Group
had set up	: Verb Group
a joint venture	: Noun Group
in	: Preposition
Taiwan	: Location

Attachment Ambiguities are not made explicit

Example of IE: FASTUS(1993)

Bridgestone Sports Co. said Friday it had set up a joint venture in Taiwan with a local concern and { a Japanese trading house } to produce golf clubs to be supplied to Japan.

The joint venture, Bridgestone Sports Taiwan Co., capitalized at 20 million new Taiwan dollars, will start production in January 1990 with production of 20,000 "metal wood" clubs a month.

1.Complex words

a Japanese trading house

a [Japanese trading] house
a Japanese [trading house]

2.Basic Phrases:

Bridgestone Sports Co.	: Company name
said	: Verb Group
Friday	: Noun Group
it	: Noun Group
had set up	: Verb Group
a joint venture	: Noun Group
in	: Preposition
Taiwan	: Location

Example of IE: FASTUS(1993)

Bridgestone Sports Co. said Friday it had set up a joint venture in Taiwan with a local concern and a Japanese trading house to produce golf clubs to be supplied to Japan.

The joint venture, Bridgestone Sports Taiwan Co., capitalized at 20 million new Taiwan dollars, will start production in January 1990 with production of 20,000 "metal wood" clubs a month.

1.Complex words

Structural Ambiguities of NP are ignored

2.Basic Phrases:

Bridgestone Sports Co.	: Company name
said	: Verb Group
Friday	: Noun Group
it	: Noun Group
had set up	: Verb Group
a joint venture	: Noun Group
in	: Preposition
Taiwan	: Location

Example of IE: FASTUS(1993)

Bridgestone Sports Co. said Friday it had set up a joint venture in Taiwan with a local concern and a Japanese trading house to produce golf clubs to be supplied to Japan.

The joint venture, Bridgestone Sports Taiwan Co., capitalized at 20 million new Taiwan dollars, will start production in January 1990 with production of 20,000 “metal wood” clubs a month.

2.Basic Phrases:

3.Complex Phrases

Bridgestone Sports Co.	: Company name
said	: Verb Group
Friday	: Noun Group
it	: Noun Group
had set up	: Verb Group
a joint venture	: Noun Group
in	: Preposition
Taiwan	: Location

Example of IE: FASTUS(1993)

[COMPNY] said Friday it [SET-UP] [JOINT-VENTURE] in [LOCATION] with [COMPANY] and [COMPNY] to produce [PRODUCT] to be supplied to [LOCATION].

[JOINT-VENTURE], [COMPNY], capitalized at 20 million [CURRENCY-UNIT] [START] production in [TIME] with production of 20,000 [PRODUCT] a month.

2.Basic Phrases:

Bridgestone Sports Co.	: Company name
said	: Verb Group
Friday	: Noun Group
it	: Noun Group
had set up	: Verb Group
a joint venture	: Noun Group
in	: Preposition
Taiwan	: Location

3.Complex Phrases

Some syntactic structures like …

Example of IE: FASTUS(1993)

[COMPNY] said Friday it [SET-UP] [JOINT-VENTURE] in [LOCATION] with [COMPANY] to produce [PRODUCT] to be supplied to [LOCATION].

[JOINT-VENTURE] capitalized at [CURRENCY] [START] production in [TIME]
with production of [PRODUCT] a month.

2.Basic Phrases:

Bridgestone Sports Co.	: Company name
said	: Verb Group
Friday	: Noun Group
it	: Noun Group
had set up	: Verb Group
a joint venture	: Noun Group
in	: Preposition
Taiwan	: Location

3.Complex Phrases

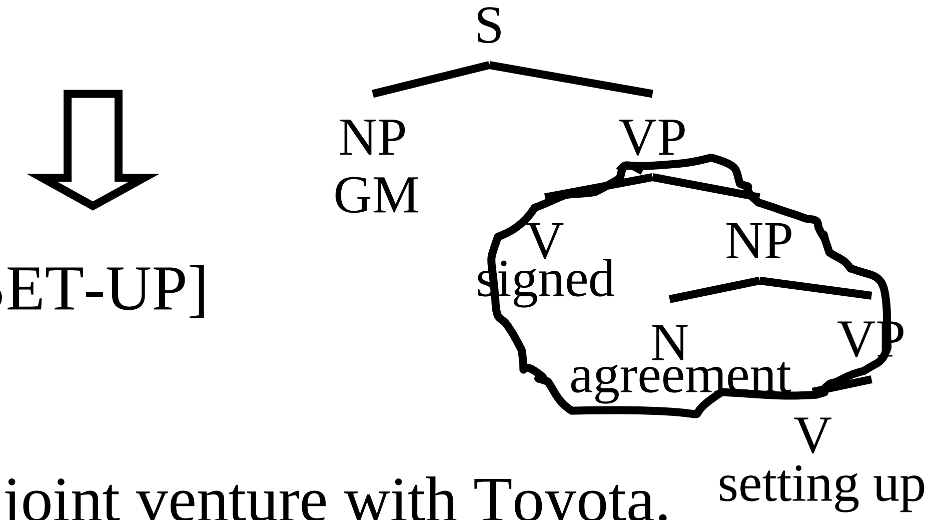
Syntactic structures relevant to information to be extracted are dealt with.

Syntactic variations

GM set up a joint venture with Toyota.
GM announced it was setting up a joint venture with Toyota.
GM signed an agreement setting up a joint venture with Toyota.
GM announced it was signing an agreement to set up a joint venture with Toyota.

Syntactic variations

GM set up a joint venture with Toyota.
GM announced it was setting up a joint venture with Toyota.
GM signed an agreement setting up a joint venture with Toyota.
GM announced it was signing an agreement to set up a joint venture with Toyota.



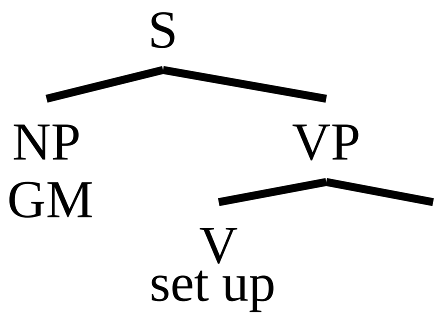
GM plans to set up a joint venture with Toyota.
GM expects to set up a joint venture with Toyota.

Syntactic variations

GM set up a joint venture with Toyota.
GM announced it was setting up a joint venture with Toyota.
GM signed an agreement setting up a joint venture with Toyota.
GM announced it was signing an agreement to set up a joint venture with Toyota.

⇩

[SET-UP]



GM plans to set up a joint venture with Toyota.
GM expects to set up a joint venture with Toyota.

Example of IE: FASTUS(1993)

[COMPNY] [SET-UP] [JOINT-VENTURE]
in [LOCATION] with [COMPANY] to
produce [PRODUCT] to be supplied to [LOCATION].

[JOINT-VENTURE] capitalized at [CURRENCY] [START]
production in [TIME]
with production of [PRODUCT] a month.

3.Complex Phrases
4.Domain Events

[COMPANY][SET-UP][JOINT-VENTURE]with[COMPNY]
[COMPANY][SET-UP][JOINT-VENTURE] (others)* with[COMPNY]

The attachment positions of PP are determined at this stage.
Irrelevant parts of sentences are ignored.

The majority of current information extraction systems perform a partial parsing approach following a bottom-up strategy

Major steps

lexical processing

- including morphological analysis, POS-tagging, Named Entity recognition

phrase recognition

- general nominal & prepositional phrases, verb groups

clause recognition via domain-specific templates

- templates triggered by domain-specific predicates attached to relevant verbs;
- expressing domain-specific selectional restrictions for possible argument fillers

Bottom-up chunk parsing

- perform clause recognition after phrase recognition is completed

However a bottom-up strategy showed to be problematic in case of German free text processing

Crucial properties of German

• highly ambiguous morphology (e.g., case for nouns, tense for verbs);

• free word/phrase order;

• splitting of verb groups into separated parts into which arbitrary phrases and clauses can be spliced in (e.g., *Der Termin findet morgen statt*. *The date takes place tomorrow.)*

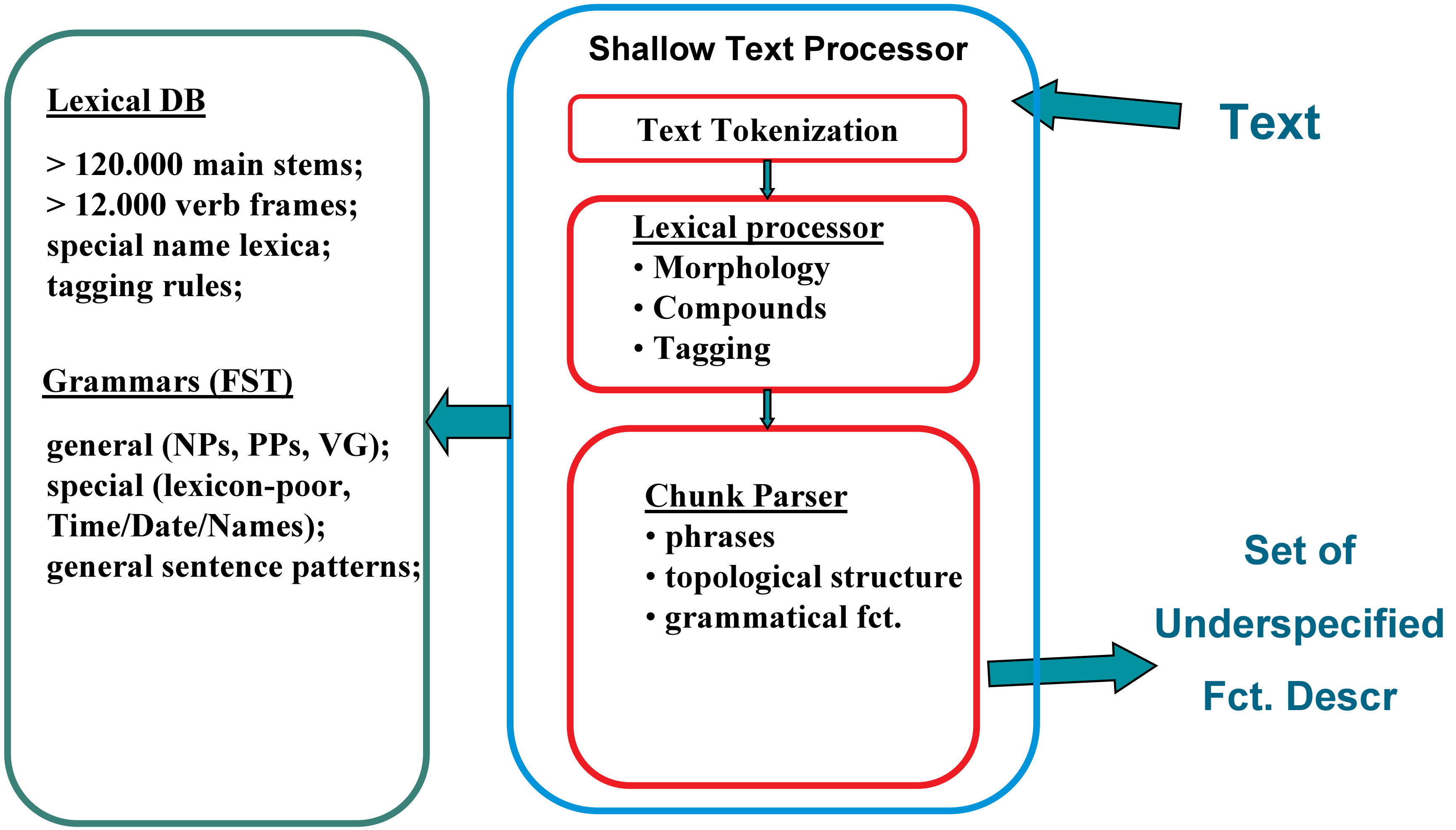
Main problem in case of a bottom-up parsing approach

even recognition of simple sentence structure depends heavily on performance of phrase recognition

NP is common practice

[NP Die vom Bundesgerichtshof und den Wettbewerbern als Verstoß gegen das Kartellverbot gegeisselte zentrale TV-Vermarktung] ist gängige Praxis.
[NP Central television marketing censured by the German Federal High Court and the guards against unfair competition as an infringement of anti-cartel legislation] is common practice.

A Robust Parser for unrestricted German Text

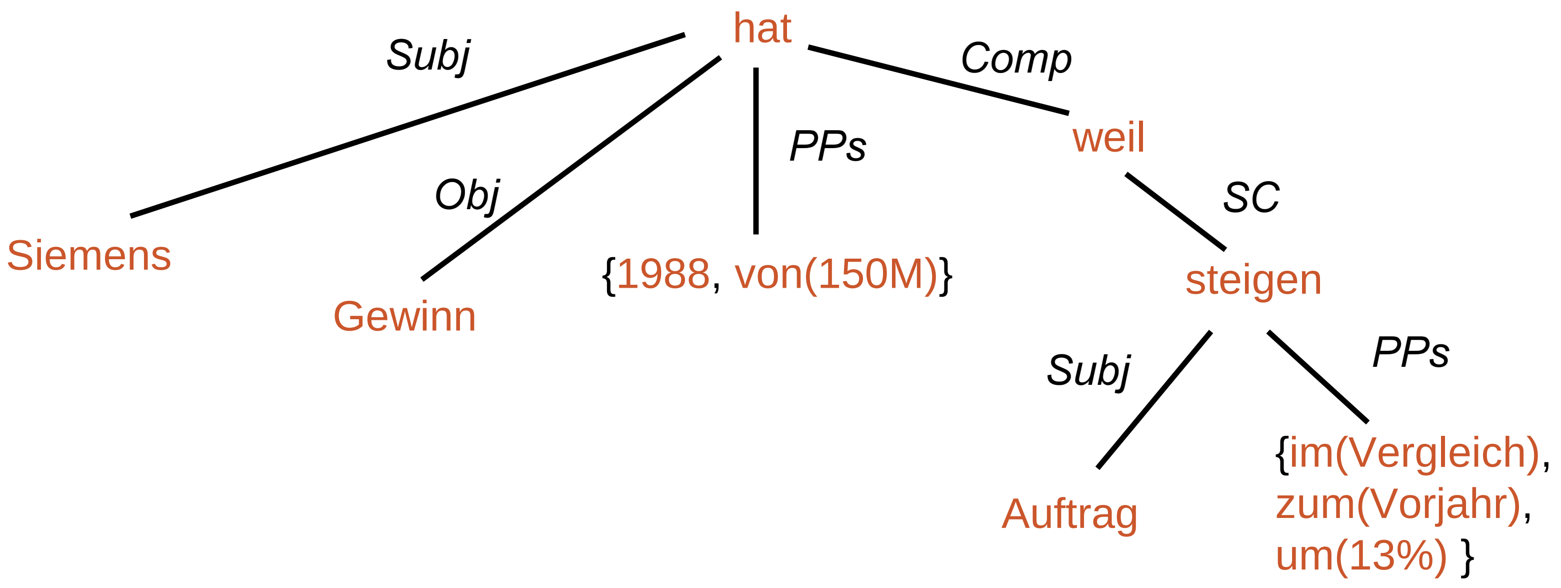


Underspecified (partial) functional descriptions UFDs

UFD: flat dependency-based structure, only upper bounds for attachment and scoping

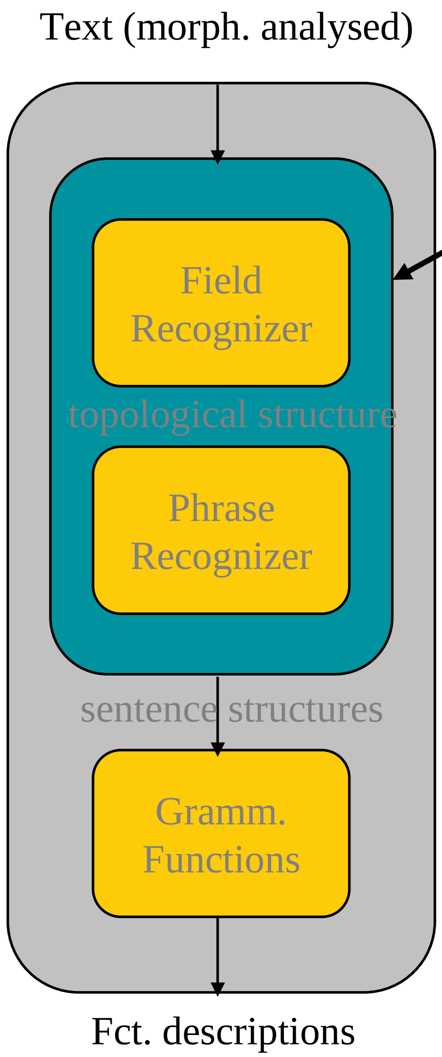
$[_{PN}$Die Siemens GmbH$]$ $[_{V}$hat$]$ $[_{year}$1988$][_{NP}$einen Gewinn$]$ $[_{PP}$von 150 Millionen DM$]$,
$[_{Comp}$weil$]$ $[_{NP}$die Auftraege$]$ $[_{PP}$im Vergleich$]$ $[_{PP}$zum Vorjahr$]$ $[_{Card}$um 13%$]$ $[_{V}$gestiegen sind$]$.

"The siemens company has made a revenue of 150 million marks in 1988, since the orders increased by 13% compared to last year."



Quelle: GNSNLP, GN

In order to overcome these problems we propose the following two phase divide-and-conquer strategy



Divide-and-conquer strategy

1. Recognize verb groups and topological structure (*fields*) of sentence domain-independently;

FrontField LeftVerb MiddleField RightVerb RestField

2. Apply general as well as domain-dependent phrasal grammars to the identified fields of the main and sub-clauses

[CoordS [CSent *Diese Angaben konnte der Bundesgrenzschutz aber nicht bestätigen*], [CSent *Kinkel sprach* von *Horrorzahlen,* [Relcl *denen er keinen Glauben schenke*]]].

This information couldn't be verified by the Border Police, Kinkel spoke of horrible figures that he didn't believe.

The divide-and-conquer approach offers several advantages

Improved robustness

- topological sentence structure determined on basis of simple indicators like verbgroups and conjunctions and their interplay;
- phrases need not be recognized completely

Resolution of some ambiguities

- relative pronouns vs. determiners
- subjunction vs. preposition
- clause vs. NP coordination

Modularity

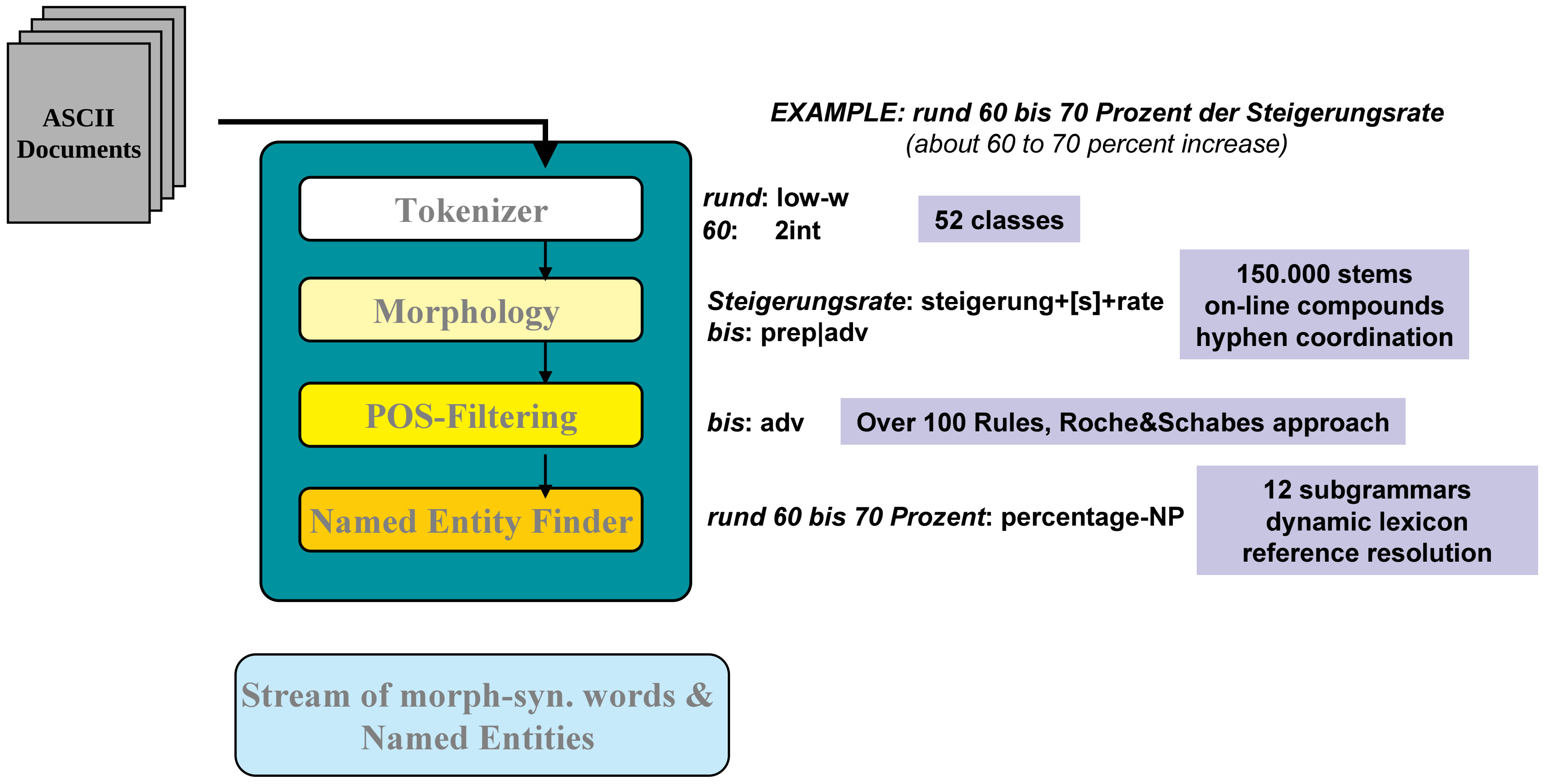
- easy exchange/extension of (domain-specific) phrase grammars

Some more examples (source text)

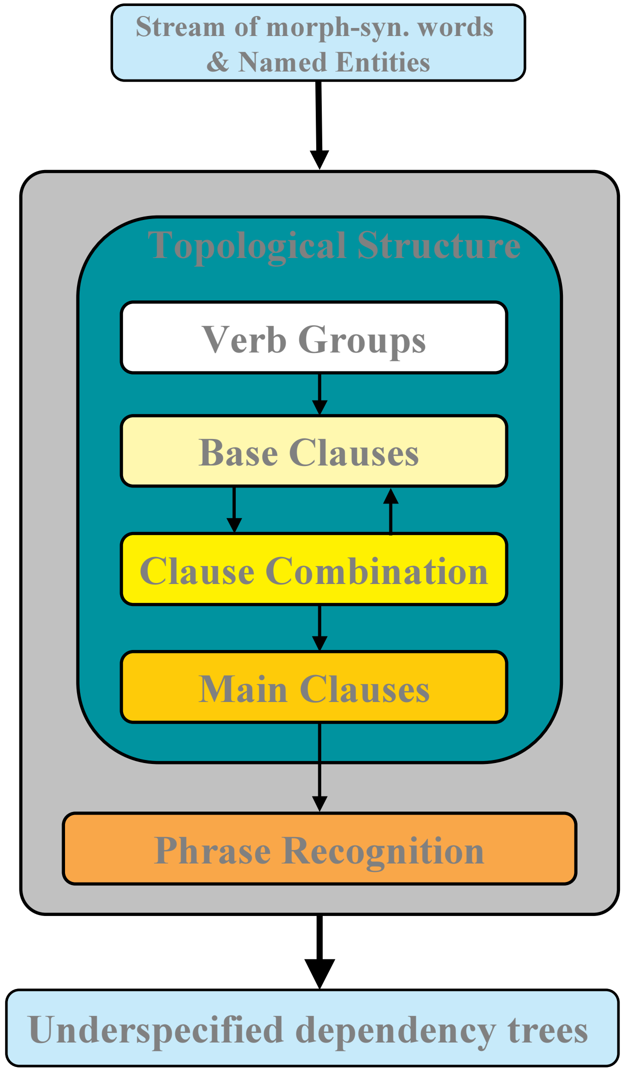
- topological structure
- plus expanded phrase structure

The divide-and-conquer parser benefits from a powerful lexical preprocessor

The lexical processor is realized on basis of state-of-the-art finite state technology, however taking care of German language specificities.


ASCII Documents
Tokenizer
Morphology
POS-Filtering
Named Entity Finder
Stream of morph-syn. words & Named Entities
EXAMPLE: rund 60 bis 70 Prozent der Steigerungsrate
(about 60 to 70 percent increase)
rund: low-w
60: 2int
52 classes
Steigerungsrate: steigerung+[s]+rate
bis: prep|adv
150.000 stems
on-line compounds
hyphen coordination
bis: adv
Over 100 Rules, Roche&Schabes approach
rund 60 bis 70 Prozent: percentage-NP
12 subgrammars
dynamic lexicon
reference resolution

The divide-and-conquer parser is realized by means of a series of finite state grammars



Weil die Siemens GmbH, die vom Export lebt, Verluste erlitt, mußte sie Aktien verkaufen.
Because the Siemens Corp which strongly depends on exports suffered from losses they had to sell some shares.

Weil die Siemens GmbH, die vom Export Verb-FIN, Verluste Verb-FIN, Modv-FIN sie Aktien FV-Inf.

Weil die Siemens GmbH, Rel-Clause Verluste Verb-FIN, Modv-FIN sie Aktien FV-Inf.

Subconj-Clause,
Modv-FIN sie Aktien FV-Inf.

Clause

The Shallow Text Processor has several Important Characteristics

Modularity:	each subcomponent can be used in isolation;
Declarativity:	lexicon and grammar specification tools;
High coverage:	more than 93 % lexical coverage of unseen text;
	high degree of subgrammars
Efficiency:	finite state technology in all components;
	specialized constrained solvers
	(e.g. agreement checks & grammatical functions);
Run-time:	4.5 msec real time per token (Standard PC environment)
Available for research:	
	http://www.dfki.de/~neumann/pd-smes/pd-smes.html

Morphological Processing

- Performed by the Morphix package
  http://www.dfki.de/~neumann/morphix/morphix.html

- Morphix performs:
  – Inflectional analysis
  – Compound analysis
  – Generation of word forms

Dynamic tries as basic data structure for lexical data

- Dynamic tries (letter tries)
  - sole storage device for all sorts of lexical information
  - Robust specialized regular matcher
  - Dynamic memory allocation (based on access frequency and access time)

$$\begin{tikzcd}
 & & T \arrow[r] & E \arrow[r] & L := N \\
H \arrow[r] & O \arrow[ur] \arrow[dr] & & & \\
 & & S \arrow[r] \arrow[dr] & E \arrow[r] & N := N \\
 & & & P \ldots &
\end{tikzcd}$$

Basic processing strategy of Morphix

- Recursive trie traversal of lexicon
- Application of finite state automata for handling inflectional regularities
- Preprocessing
  - Each word form is fristly transformed into a set of tripples <prefix, lemma, suffix>
    - Prefix: (complex) verb prefix or GE-
    - Lemma: possible lexical stem, where possible umlauts are reduced (e.g., Mädchen vs. Häusern)
    - Suffix: longest matching inflection ending (using a inflection lexicon)

Representation of results

- Set of tripple <stem, inflection, POS>
- Compound processing handles words with
  - nominal root (*Häuserblock* *“block of houses”*)
  - adjectival root (*tiefschwarz* *“deep black”*)
  - verbal root (*blaugefärbt* *“blue colored”*)
- Compound processing
  - a recursive trie traversal
  - Identification of allowable infixes

Quelle: GNSNLP, GN

Flexible output interface

Compute DNF for the compactly represented disjunctive morpho-syntactic output. User can choose different forms of DNF representation:

disjunctive output for the form “die Häuser” (“*the houses*”)

```
(“haus” (cat noun) (flexion ((ntr ((pl (nom gen acc)))))))
```

as symbol list (e.g., used in case of lexical tagging)

```
(“haus” (ntr-pl-nom ntr-pl-gen ntr-pl-acc) . :n)
```

as feature term (e.g., used in case of shallow parsing)

```
(“haus”
(((:tense . :no) (:person . :no) (:gender . :ntr) (:number . :pl) (:case . :nom))
((:tense . :no) (:person . :no) (:gender . :ntr) (:number . :pl) (:case . :gen))
((:tense . :no) (:person . :no) (:gender . :ntr) (:number . :pl) (:case . :acc)))
. :n)
```

Quelle: GNSNLP, GN

Morphix comes with a very flexible output interface

- Finite set of possible morpho-syntatic output structures
  - DNF computation can be done off-line and on-line using memorization techniques
- User can select interactively subset from possible morpho-syntactic feature set {:cat :mact :sym :comp :comp-f :det :tense :form :person :gender :number :case}

```
e.g.      ("haus"
              (((:number . :pl) (:case . :nom))
               ((:number . :pl) (:case . :gen))
               ((:number . :pl) (:case . :acc)))
              . :n)
```

  - supports lexical tagging (use of different tag sets)
  - supports feature relaxation (ignore uninteresting features)
    - Increased robustness

Quelle: GNSNLP, GN

Specialized Unifier

- Currently, constraints are mainly used to express morpho-syntactical agreement

- Feature checking performed by a simple but fast specialized unifier
  - Feature vector representation
  - Special symbol :no used as anonymous variable
  - Example

```
s1=(((:TENSE . :NO) (:FORM . :NO) (:NUMBER . :S) (:CASE . :N))
      ((:TENSE . :NO) (:FORM . :NO) (:NUMBER . :S) (:CASE . :A))
      ((:TENSE . :NO) (:FORM . :NO) (:NUMBER . :P) (:CASE . :N))
      ((:TENSE . :NO) (:FORM . :NO) (:NUMBER . :P) (:CASE . :A))))
s2=(((:TENSE . :NO) (:FORM . :XX) (:NUMBER . :S) (:CASE . :N))
     ((:TENSE . :NO) (:FORM . :NO) (:NUMBER . :S) (:CASE . :G))
     ((:TENSE . :NO) (:FORM . :NO) (:NUMBER . :S) (:CASE . :D)))
unify(s1,s2)=
   (((:TENSE . :NO) (:FORM . :XX) (:NUMBER . :S) (:CASE . :N)))
```

Quelle: GNSNLP, GN

Writing grammars with SMES

- Finite state transducers FST <identifier, recognition part, output description, compiler options>
- Recognition part is a regular expression where alphabet is implicitly expressed via basic edges
  - Predicate or a specific class of tokens, e.g. (:morphix-cat *partikel pre*)
  - :morphix-cat is a predicate which checks whether the current token‘s POS equals *partikel*, and if so, bound the token to the variable *pre*

Example of simple NP rule

```
(:conc
   (star<=n (:morphix-cat det det) 1)
   (:star (:morphix-cat adj adj))
   (:morphix-cat n noun))
```

Thus defined, a nominal phrase is the concatenation of one optional determiner (expressed by the loop operator :star<=n, where n starts from 0 and ends by 1), followed by zero or more adjectives followed by a noun.

NP with feature vector unification

```
(compile-regexp
 '(:conc
   (:current-pos start)
   (:alt
     (:star<=n (:morphix-unify :indef  NIL agr det) 1)
     (:star<=n (:morphix-unify :def  NIL agr det) 1))
   (:star<=n (:morphix-unify :a agr agr adj) 1)
   (:morphix-unify :n  agr agr noun)
   (:current-pos end))
 :output-desc
 '(:lisp (build-item
              :type :np :start start :end end :agr agr
             :det det :adj adj :noun noun))
 :name 'small-np)
```

Special basic edge

Empty feature vector

Output description (typed based)

Phrase recognition

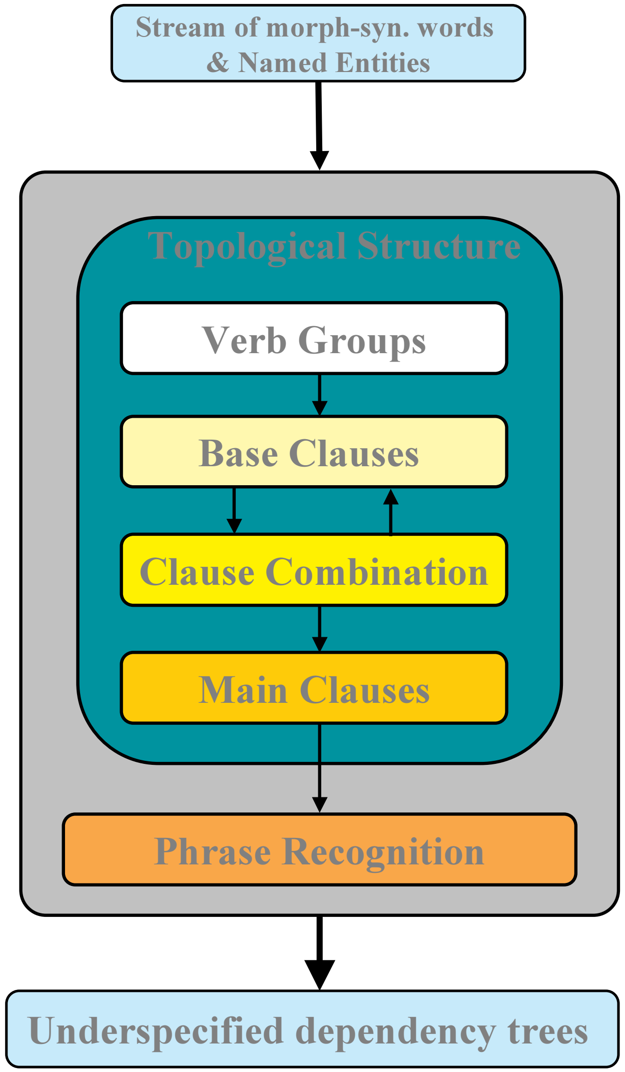
- Nominal phrases NP
  - *dem Fernrohr*
- Prepositional phrases PP
  - *mit dem Fernrohr*
- Verb groups VG
  - *glaubt mit dem Fernrohr sehen zu können*
- NE grammars
  - *Kanzler Schröder glaubt mit dem Fernrohr sehen zu können.*

Example

- Der Mann sieht die Frau mit dem Fernrohr.
  The man sees the woman with the telescope.

```
((:SEM (:HEAD "mann") (:QUANTIFIER "d-det"))
 (:AGR
  ((:TENSE . :NO) ... (:CASE . :NOM)))
 (:END . 2) (:START . 0) (:TYPE . :NP))
((:SEM (:HEAD "frau") (:QUANTIFIER "d-det"))
 (:AGR
  ((:TENSE . :NO) ... (:GENDER . :F) (:NUMBER . :S)
   (:CASE . :NOM))
  ((:TENSE . :NO) ... (:GENDER . :F) (:NUMBER . :S)
   (:CASE . :AKK)))
 (:END . 5) (:START . 3) (:TYPE . :NP))
((:SEM (:HEAD "mit")
      (:COMP (:QUANTIFIER "d-det") (:HEAD "fernrohr")))
 (:AGR
  ((:TENSE . :NO) ... (:GENDER . :NT) (:NUMBER . :S)
   (:CASE . :DAT)))
 (:END . 8) (:START . 5) (:TYPE . :PP)))
```

The divide-and-conquer parser is realized by means of a series of finite state grammars



Weil die Siemens GmbH, die vom Export lebt, Verluste erlitt, mußte sie Aktien verkaufen.
Because the Siemens Corp which strongly depends on exports suffered from losses they had to sell some shares.

Weil die Siemens GmbH, die vom Export Verb-FIN, Verluste Verb-FIN, Modv-FIN sie Aktien FV-Inf.

Weil die Siemens GmbH, Rel-Clause Verluste Verb-FIN, Modv-FIN sie Aktien FV-Inf.

Subconj-Clause,
Modv-FIN sie Aktien FV-Inf.

Clause

Verb grammar

- A verb grammar recognizes all
  - single occurrences of verbforms (in most cases corresponding to LeftVerb)
  - all closed verbgroups (in general RightVerb)
- Discontinuous verb groups (separated LeftVerb and RightVerb) are not put together
- Major problem here is not a structural one but the massive morphosyntactic ambiguity of verbs

Verb Grammars

- The verb rules solve most of these problems on the basis of feature value occurence (e.g., a rule is only triggered if the current verb form is finite).
- Feature checking is performed through term unification.
- The different rules assign to each recognized expression its type for example on the basis of time and active/passive information (e.g., whether it is final, modal perfect active).

Example output

- nicht gelobt haben kann
  could not have been praised

Type	VG-final
Subtype	Mod-Perf-Ak
Modal-stem	Koenn
Stem	Lob
Form	nicht gelobt haben kann
Neg	T
Agree	...

Base clauses

- Subclauses of type
  - Subjunctive (e.g., als, als ob, soweit, ...)
  - Subordinate (e.g., relative clauses)
- Simply be recognized on the basis
  - Commas
  - Initial elements (like complementizer)
  - Interrogative or relative item
- The different types of subclauses are described very compactly as finite state expressions

Snapshot of Base clause grammar

Base-clause ::=
  Inf-Cl|Subj-Cl|w-Cl|Rel-Cl|Parenthese

Sub-Cl ::=
  (,|Cl-Beg){funct-word} Subjunctor verb-final-cl

Subjunktor ::= als| als dass| sooft|...

Verb-final-cl ::= ...

In order to deal with embedded clauses, two sorts of recursions are identified

Middle-field recursion

embedded base clause is located in the middle field of the embedding sentence

…, weil die Firma, nachdem sie expandiert hatte, größere Kosten hatte.

(*…, *because the company, after it expanded had, increased costs had*.)

➢ …, weil die Firma [Subclause], größere Kosten hatte.

➢ … [Subclause].

Rest-field recursion

embedded clause follows the right verb part of the embedding sentence

…, weil die Firma größere Kosten hatte, nachdem sie expandiert hatte.

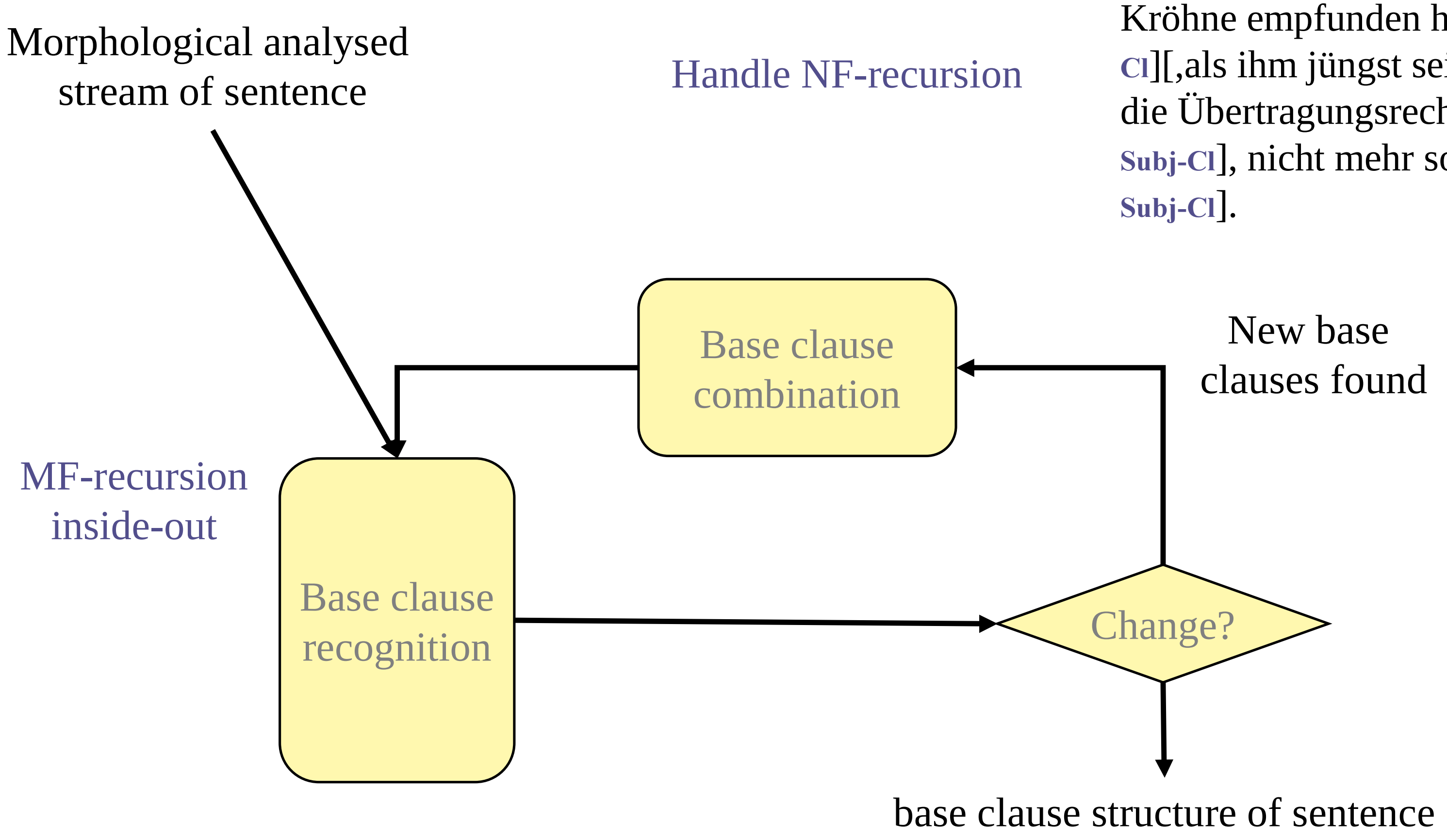
(*…, *because the company increased costs had, after it expanded had*.)

➢ … [Subclause] [Subclause].

➢ … [Subclause].

These recursions are treated as iterations which destructively substitute recognized embedded base clauses with their type

...*[daß das Glück [, das Jochen Kröhne empfunden haben soll Rel-Cl][,als ihm jüngst sein Großaktionär die Übertragungsrechte bescherte Subj-Cl], nicht mehr so recht erwärmt Subj-Cl].



Main clauses

- Builds the complete topological structure of the input sentence on the basis of
  - recognized (remaining) verb groups
  - base clauses
  - word form information (punctuations and coordinations)

Main clause grammar

Csent	::=	... LVP ... [RVP] ...
Ssent	::=	LVP [RVP] ...
CoordS	::=	CSent (, CSent)* Coord CSent | CSent (, SSent)* Coord SSent
AsyndSent	::=	CSent {,} CSent
ComplexCSent	:: =	CSent {,} SSent | CSent , CSent
AsyndCond	::=	SSent {,} SSent

Evaluation on unseen test data (press releases)

Lexical pre-processor (20.000 tokens)

	Recall %	Precision %	
compound analysis	99.01	99.29	
part-of-speech-filtering	74.50	97.90	
named entity (incl. dynamic lexicon)	85.00	95.77	
fragments (NPs, PPs):	76.11	91.94	

Divide-and-conquer parser (400 sentences, 6306 words)

verb module	98.10	98.43	
base-clause module	93.08 (94.61)	93.80 (93.89)	
main-clause module	89.00 (93.00)	94.42 (95.62)	
complete analysis	84.75	89.68	F=87.14

Preliminary summary

Divide-and-conquer parsing strategy

free German text processing

suited for free worder languages

high modularity

Main experience

full text processing necessary even if only some parts of a text are of interest;

application-oriented depth of text understanding;

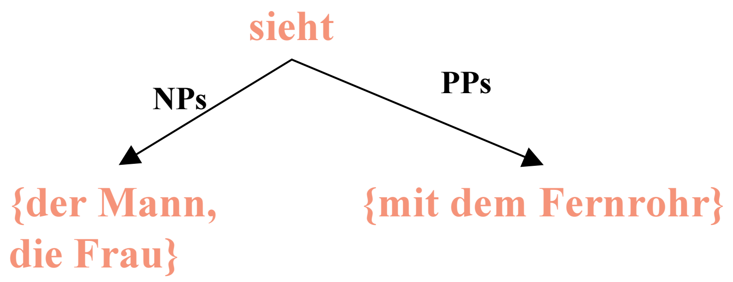
the difference between shallow and deep NLP seen as a continuum

Underspecified dependency tree

- After topological parsing, the phrase grammars are applied to the elements of the identified fields
- Then an underspecified dependency tree is computed by collecting
  - the elements from the verb groups which define the head of the tree
  - all NPs directly governed by the head into a set NP modifiers
  - all PPs directly governed by the head into a set PP modifiers
- This process is recursively applied to all embedded clauses
- The resulting structure is underspecified because only upper bounds for attachment are defined

Example dependency tree

Der Mann sieht die Frau
mit dem Fernrohr.



```
(((:PPS
   ((:SEM (:HEAD "mit")
         (:COMP (:QUANTIFIER "d-det") (:HEAD "fernrohr")))
    (:AGR
     ((:TENSE . :NO) ... (:CASE . :DAT)))
    (:END . 8) (:START . 5) (:TYPE . :PP)))
  (:NPS
   ((:SEM (:HEAD "mann") (:QUANTIFIER "d-det"))
    (:AGR
     ((:TENSE . :NO) ... (:CASE . :NOM)))
    (:END . 2) (:START . 0) (:TYPE . :NP))
   ((:SEM (:HEAD "frau") (:QUANTIFIER "d-det"))
    (:AGR
     ((:TENSE . :NO) ... (:CASE . :NOM))
     ((:TENSE . :NO) ... (:CASE . :AKK)))
    (:END . 5) (:START . 3) (:TYPE . :NP)))
  (:VERB
   (:COMPACT-MORPH
    ((:TEMPUS . :PRAES) ... (:PERSON . 3)
     (:GENUS . :AKTIV)))
   (:MORPH-INFO
    ((:TENSE . :PRES) (:FORM . :FIN) ... (:CASE . :NO)))
   (:ART . :FIN) (:STEM . "seh")
   (:FORM . "sieht") (:C-END . 3) (:C-START . 2)
   (:TYPE . :VERBCOMPLEX))
  (:END . 8) (:START . 0) (:TYPE . :VERB-NODE)))
```

Grammatical function recognition GFR

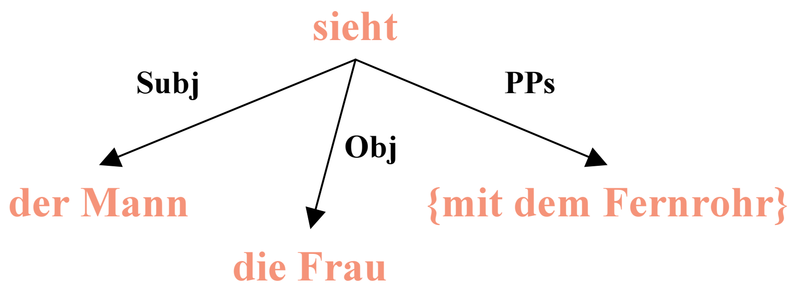
- In the final step of parsing process, the grammatical functions are determined for all subtrees of the dependency tree
- Main knowledge source is a huge subcategorization lexicon for verb
- During a recursive traversal of the dependency tree the longest matching subcat frame is checked to identify the head and modifier elements

Main steps of GFR

- Identification of possible *arguments* on the basis of the lexical subcategorization information available for the local head (the verb group)
- Marking of the other non-head elements of the dependence tree as *adjuncts*, possibly by applying a distinctive criterion for standard and specialized adjuncts.
- Adjuncts - opposed to arguments, for which an attachment resolution is attempted - have to be considered underspecified wrt. attachment, even after GFR
  - in other words, their dependency relation to the head counts as an *upper border* rather than an attachment

Example of GFR output

Der Mann sieht die Frau
mit dem Fernrohr.



```
(((:SYN
   (:SUBJ
    (:RANGE (:SEM (:HEAD "mann") (:QUANTIFIER "d-det"))
     (:AGR
      ((:PERSON . 3) (:GENDER . :M)
       (:NUMBER . :S) (:CASE . :NOM)))
     (:END . 2) (:START . 0) (:TYPE . :NP)))
   (:OBJ
    (:RANGE (:SEM (:HEAD "frau") (:QUANTIFIER "d-det"))
     (:AGR
      ((:PERSON . 3) (:GENDER . :F)
       (:NUMBER . :S) (:CASE . :NOM))
      ((:PERSON . 3) (:GENDER . :F)
       (:NUMBER . :S) (:CASE . :AKK)))
     (:END . 5) (:START . 3) (:TYPE . :NP)))
   (:NP-MODS)
   (:PP-MODS
    ((:SEM (:HEAD "mit")
          (:COMP (:QUANTIFIER "d-det") (:HEAD "fernrohr")))
     (:AGR ((:PERSON . 3) (:GENDER . :NT)
           (:NUMBER . :S) (:CASE . :DAT)))
     (:END . 8) (:START . 5) (:TYPE . :PP)))
   (:PROCESS
    (:COMPACT-MORPH
     ((:TEMPUS . :PRAES) ... (:GENUS . :AKTIV)))
    (:MORPH-INFO
     ((:TENSE . :PRES) ... (:NUMBER . :S) (:CASE . :NO)))
    (:ART . :FIN) (:STEM . "seh") (:FORM . "sieht")
    (:TYPE . :VERBCOMPLEX))
   (:SC-FRAME ((:NP . :NOM) (:NP . :AKK)))
   (:START . 0) (:END . 8)
   (:TYPE . :SUBJ-OBJ))))
```

The subcategorization lexicon

- more than 25500 entries for German verbs
- the information conveyed by the verb subcategorization lexicon we use, includes subcategorization patterns, like arity, case assigned to nominal arguments, preposition/ subconjunction form for other classes of complements
- Example subcat for the verb fahr (to drive):

1. {<np,nom>}
2. {<np,nom>, <pp, dat, mit>}
3. {<np,nom>, <np,acc>}

Shallow strategy

- Given a set of different subcategorization frames that the lexicon associates to a verbal stem, the structure chosen as the final (disambiguated) solution is the one corresponding to the *maximal subcategorization frame* available in the set, which is the frame mentioning the largest number of arguments that may be succesfully applied to the input dependence tree.

Deep grammatical functions

- Obliquity hierarchy (implicitly assuming an ordering of the subcat elements; but only used for assigning a deep case label)
  - SUBJ: deep subject;
  - OBJ: deep object;
  - OBJ1: indirect object;
  - P-OBJ: prepositional object;
  - XCOMP: subcategorized subclause
- The subject and object does not necessarily correspond to the surface subject and direct object in the sentence, e.g., in case of passivization

Processing strategy of GFR

1. Retrieve the subcategorization frames for the verbal head of the root node of the input dependency tree;
2. Apply lexical rules in order to determine deep case information depending on the verb diathesis; since frames are expressed for active sentences only, a passivation rule exists which transforms NP-nominative to NP-accusative, and NP-nominative to PP-accusative with preposition von and durch
3. For each subcat frame sc do:
   1. match sc with the dependent elements; if matching succeeds, then call sc a valid subcat frame; otherwise sc is discarded;
   2. if sc is a valid subcat frame and sc_p is the current active subcat frame compute in the previous step of the loop, then if $|sc| > |sc_p|$ select sc as the current active subcat frame;
   3. insert the domain-specific information found for the verbal head of the root (if available); this information can be retrieved from the domain lexicon using the stem entry of the head verb (template triggering)
4. the same method is recursively applied on all sub-clauses
5. finally return the new dependency tree marked for deep grammatical functions; we call such dependency tree an underspecified functional description

Unification of subcat elements

- Expand subcat frame element to corresponding feature vector and unify it with the feature structure found for verbal head
- Example: *Der Mann sieht die Frau.*
  - subcat frame for *seh (to see):* {<np,nom>, <np,acc>}.
  - Fvect from input:
    ((:tense . :pres) (:form . :fin) (:person . 3)
    (:gender . :no)(:number . :s) (:case . :no))
  - Expanded and unified fvec:
    {((:tense . :pres) (:form . :fin) (:person . 3)
    (:gender . :no) (:number . :s) (:case . :nom)),
  - ((:tense . :no) (:form . :no) (:person . :no)
    (:gender . :no) (:number . :no) (:case . :acc))}
- Expanded fvec now used for unification with elements from NPs to assign subject and object.

Adjuncts are further grouped into type compatible subsets

- All elements which are not assigned grammatical functions are considered as adjuncts
- All elements of same type (e.g., date-np, loc-pp) are collected into disjunctive subsets (actually based on NE recognition):
  - {LOC-PP, LOC-NP, RANGE-LOC-PP} maps to LOC-MODS
  - {DATE-PP, DATE-NP} maps to DATE-MODS
- All others retain in their respective generic phrasals sets
  - NPS
  - PPS
  - Sclause
- Evaluation by Lappata: 11 EACL,2003

Summary

- SMES is a *mildly* deep parsing system
  - Combining shallow approaches with generic linguistic resources
  - Finite state backbone with feature constraints
  - Topological structure for coarse-grained sentence structure
  - Identification of grammatical functions
- Web
  - System: http://www.dfki.de/~neumann/smes
  - References: http://www.dfki.de/~neumann/publications/neumann-ref.html
- It is now used as part of our Clef-2004 question-answering system