Macau, March 12th

XI UNL School

Day #2

Day #2

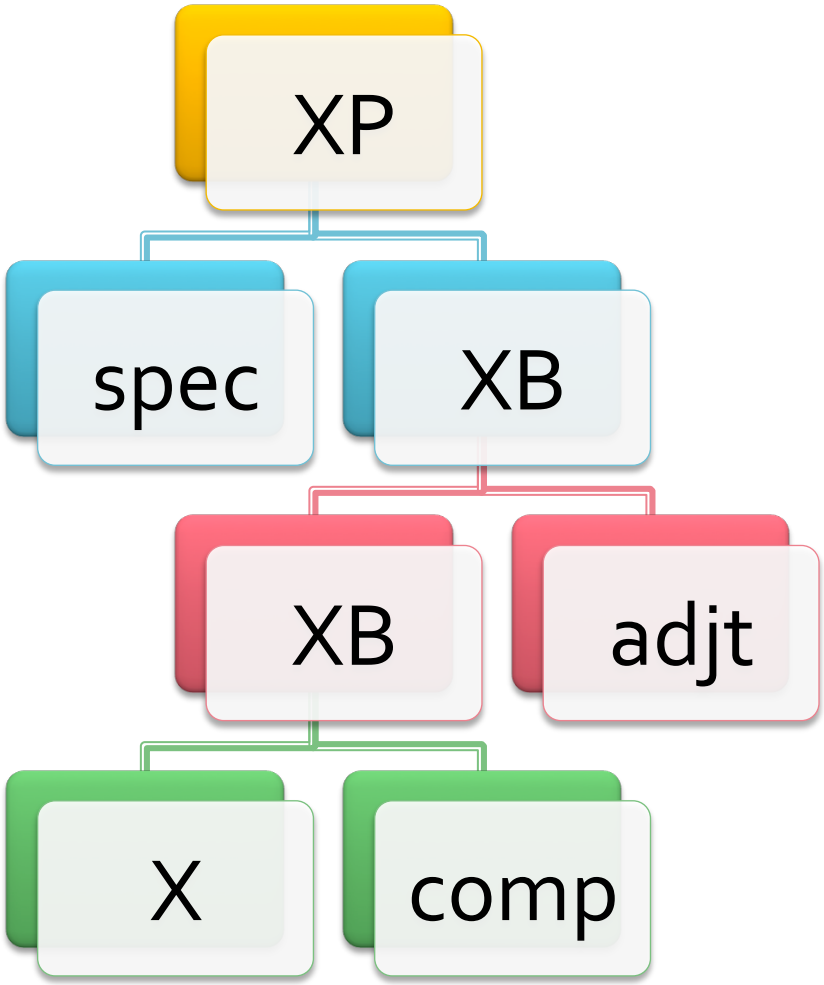
- X-bar
- Grammars
  - Grammar Specs
    - Basic symbols
    - Nodes
    - Relations
  - Grammar Strucutre
    - Normalization Grammar
    - Language-Specific Grammar
    - Default Grammar
  - Grammar Types
    - T-grammar
    - D-grammar

X-bar Theory

Grammar frameworks

- DCG (Definite Clause Grammar)
- GPSG (Generalized Phrase Structure Grammar)
- HPSG (Head-driven Phrase-Structure Grammar)
- TAG (Tree Adjoining Grammar)
- UG (Unification Grammar)
- CG (Categorial Grammar)
- FUG (Functional Unification Grammar)
- SFG (Systemic functional grammar (SFG)
- LFG (Lexical-functional Grammar)
- Generative Grammar
  - ST (Standard Theory)
  - EST (Extended Standard Theory)
  - X-bar
  - GB (Government and Binding)
  - PP (Principles and Parameters)

X-bar structure

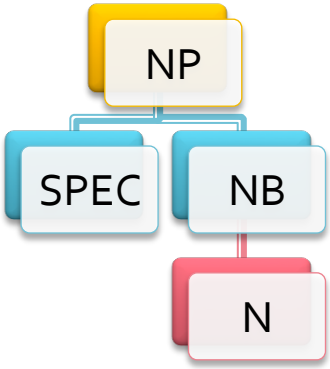


- Where:
  - XP = maximal projection
  - XB = intermediate projections
  - spec = specifier
  - adjt = adjunct
  - comp = complement
  - X = head
    - N (noun)
    - V (verb)
    - J (adjective)
    - A (adverb)
    - D (determiner)
    - P (preposition)
    - C (conjunction)

Possible configurations of a XP (I)

[construction]

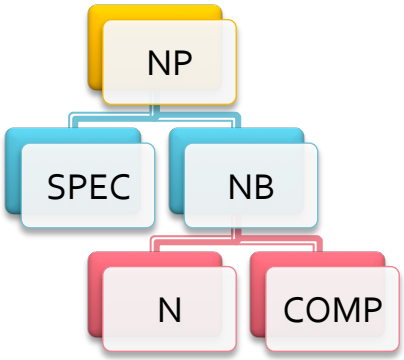
[the] [construction]



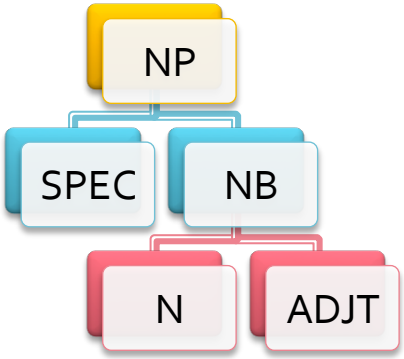
[construction] [of the tower]

[fateful] [construction]

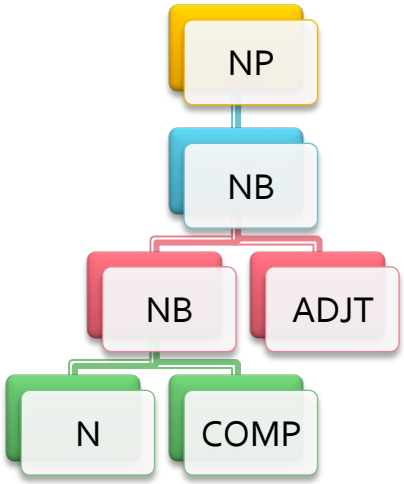
[the] [construction] [of the tower]



[the] [fateful] [construction]

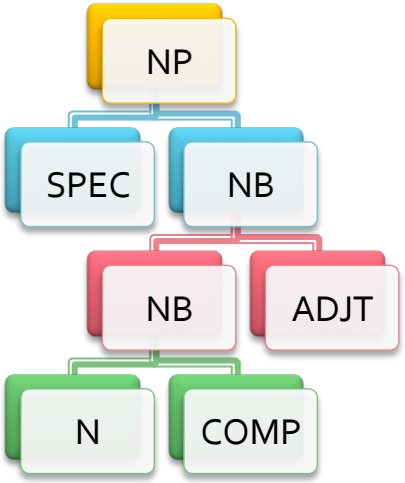


[fateful] [construction] [of the tower]

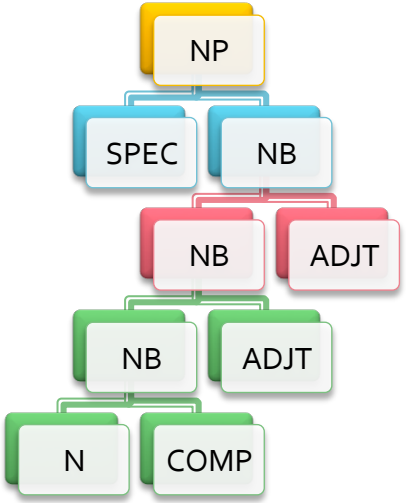


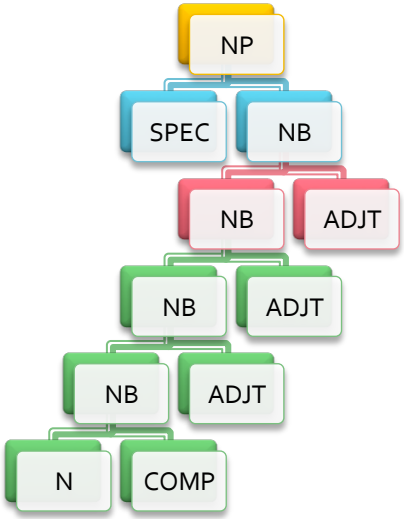
Possible configurations of a XP (II)

[the] [fateful] [construction] [of the tower]



[the] [long] [fateful] [construction] [of the tower]



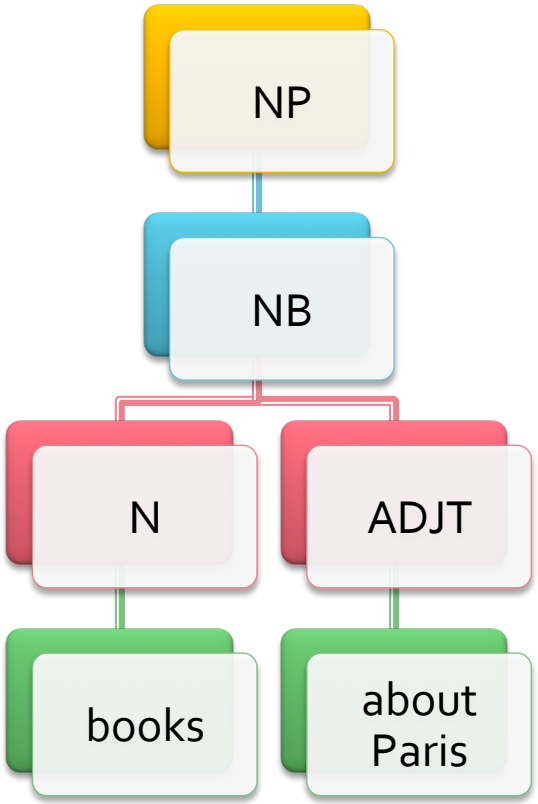


[the] [long] [fateful] [expensive] [construction] [of the tower]

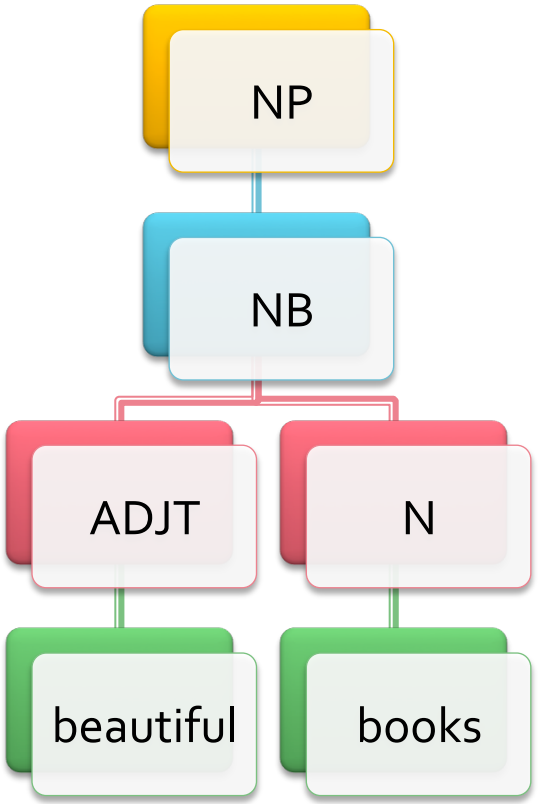
etc.

Order

RIGHT ADJUNCTION

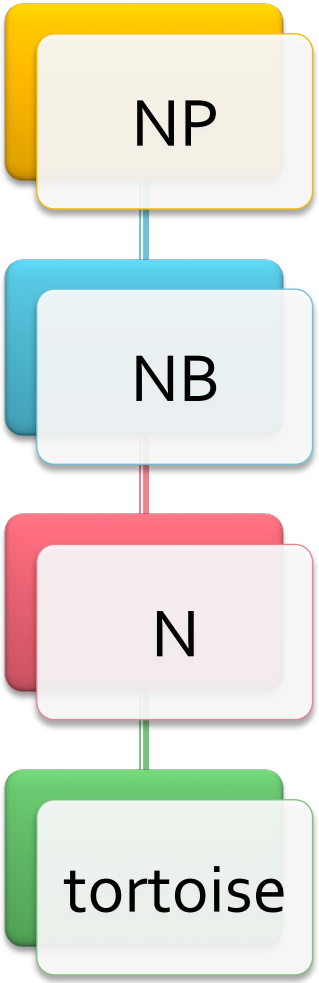


LEFT ADJUNCTION

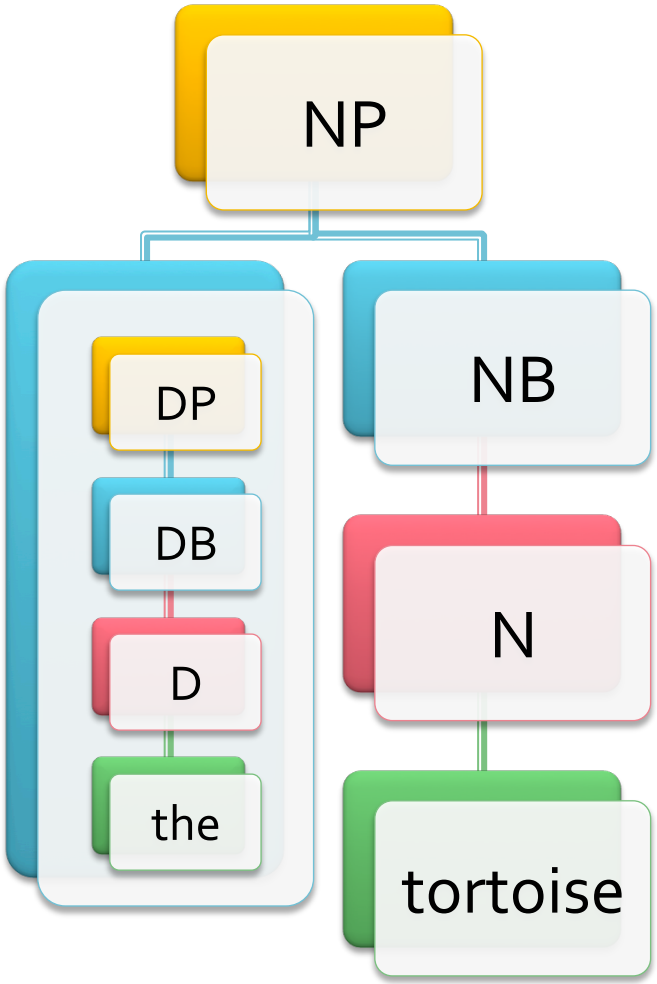


Examples

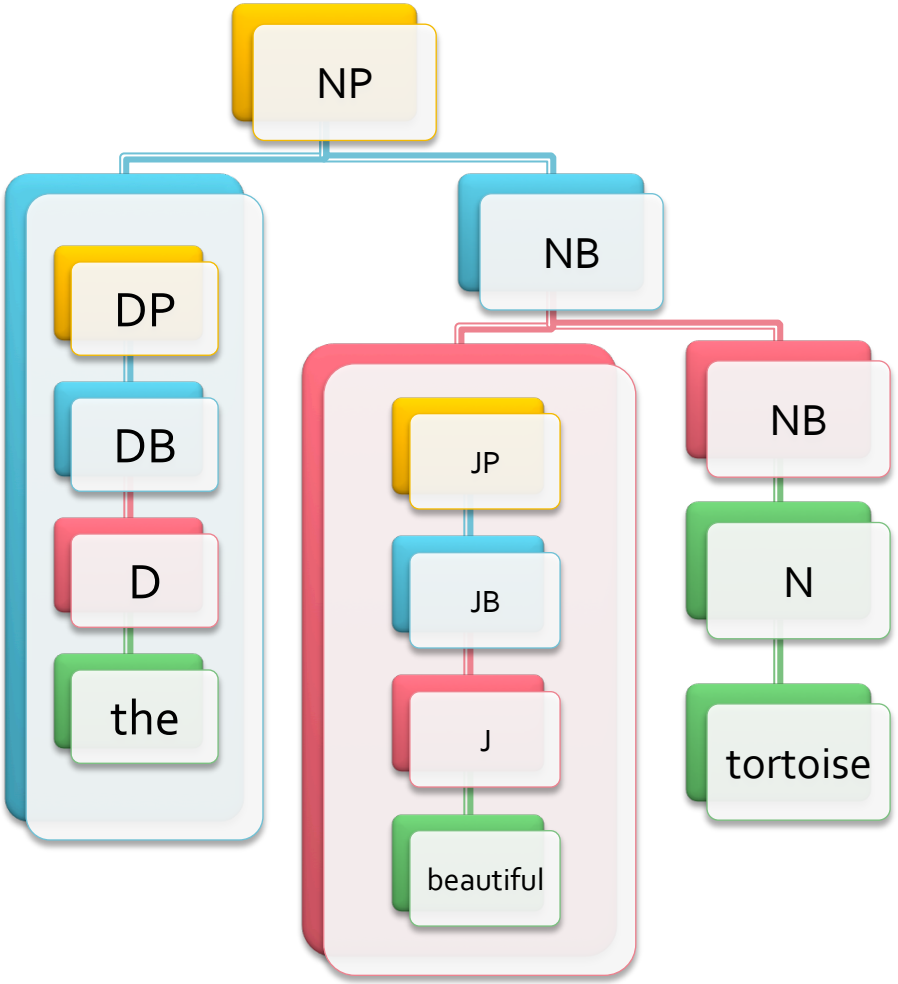
Tortoise



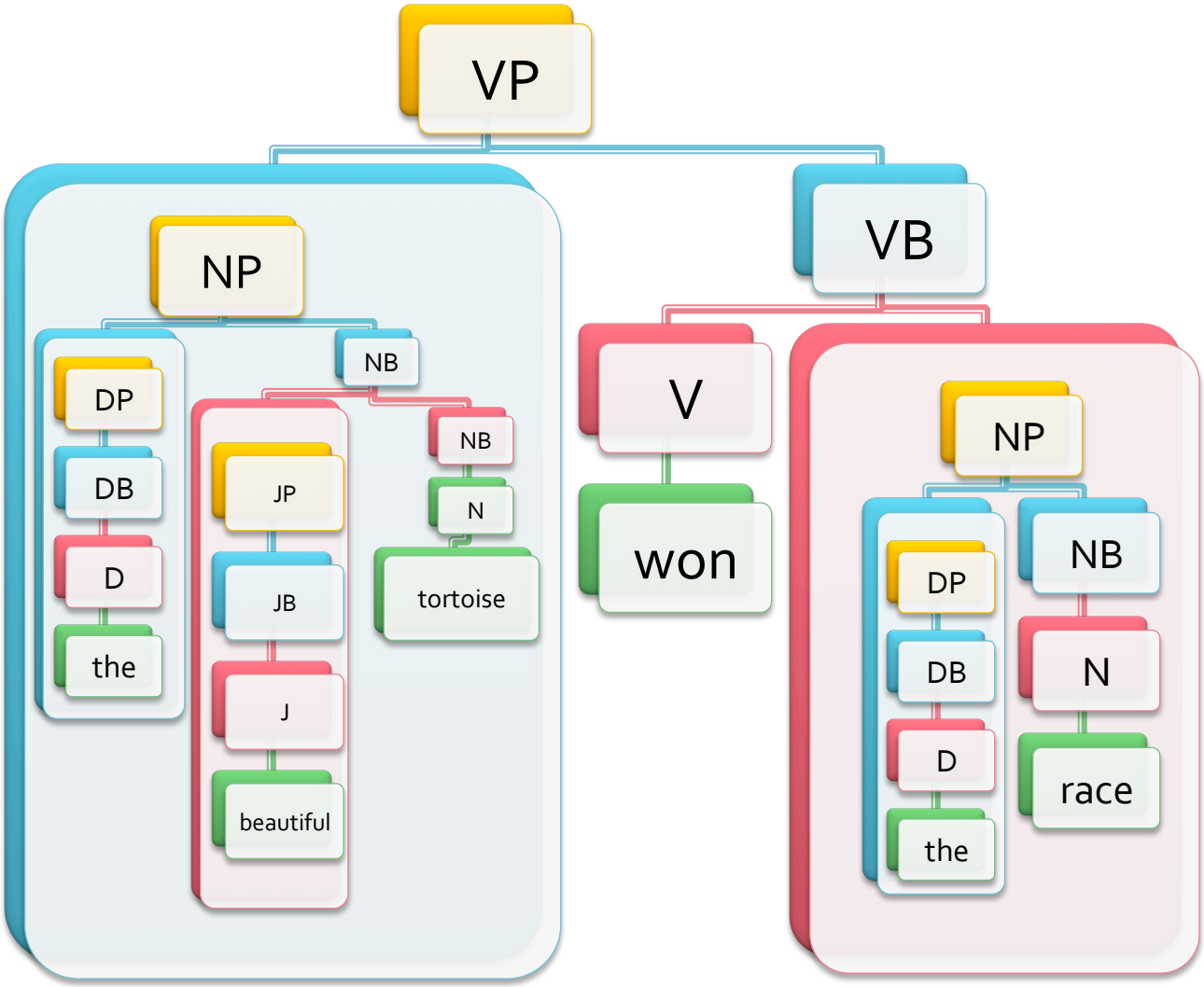
The tortoise



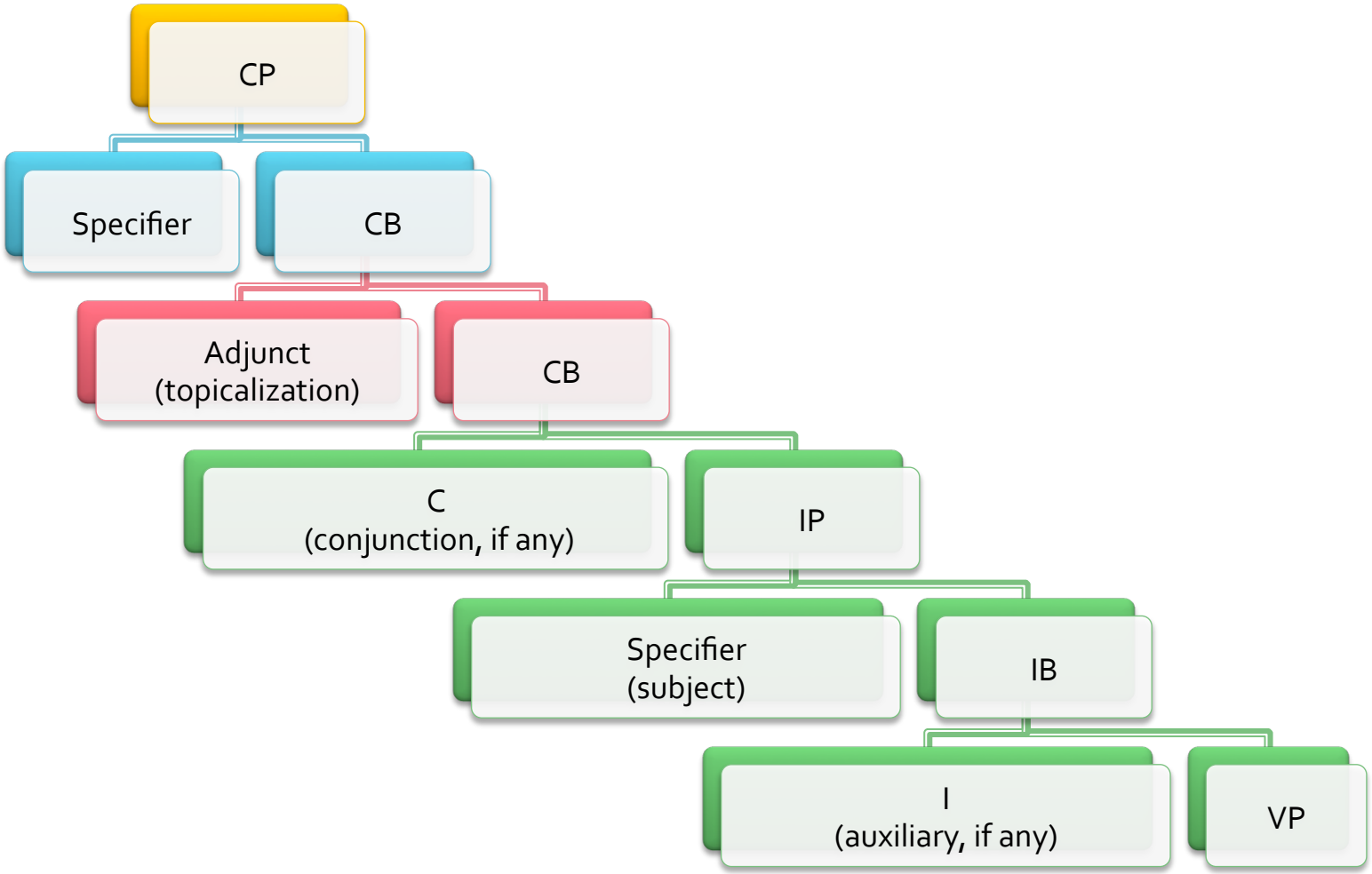
The beautiful tortoise



The beautiful tortoise won the race



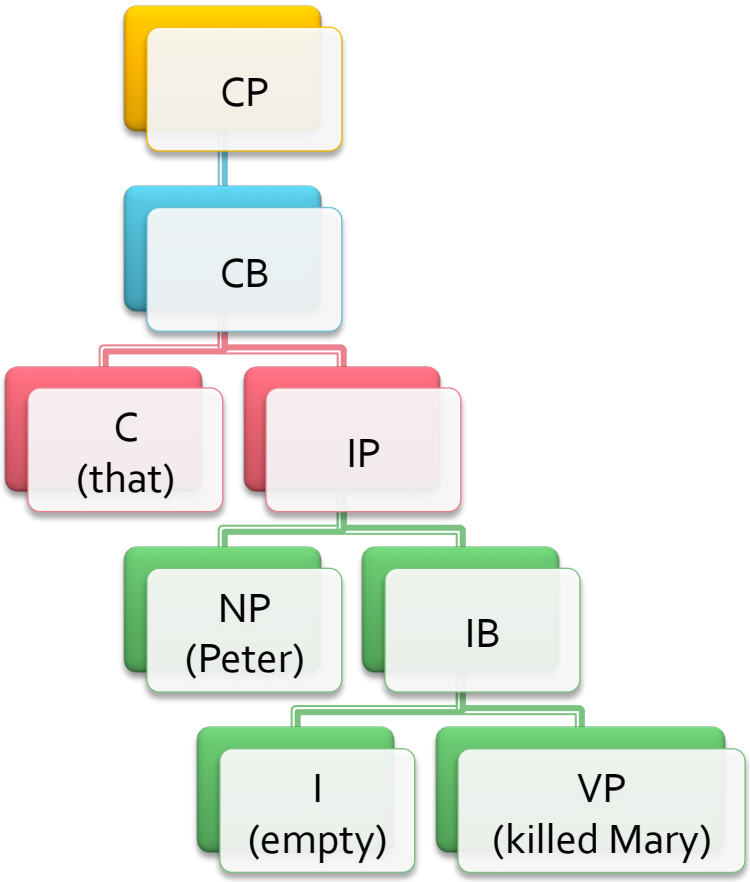
Topmost levels



CP
when to use

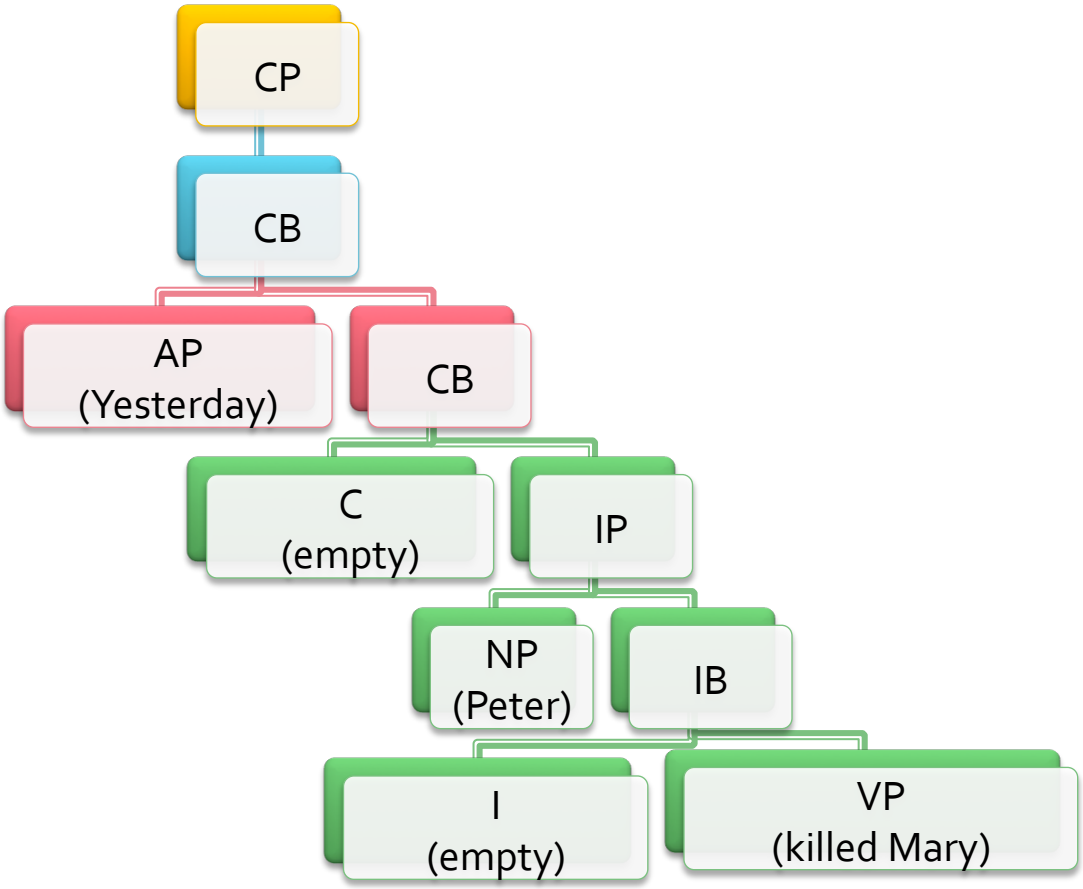
(I know that) Peter killed Mary

subordinating conjunction ("that")



Yesterday Peter killed Mary

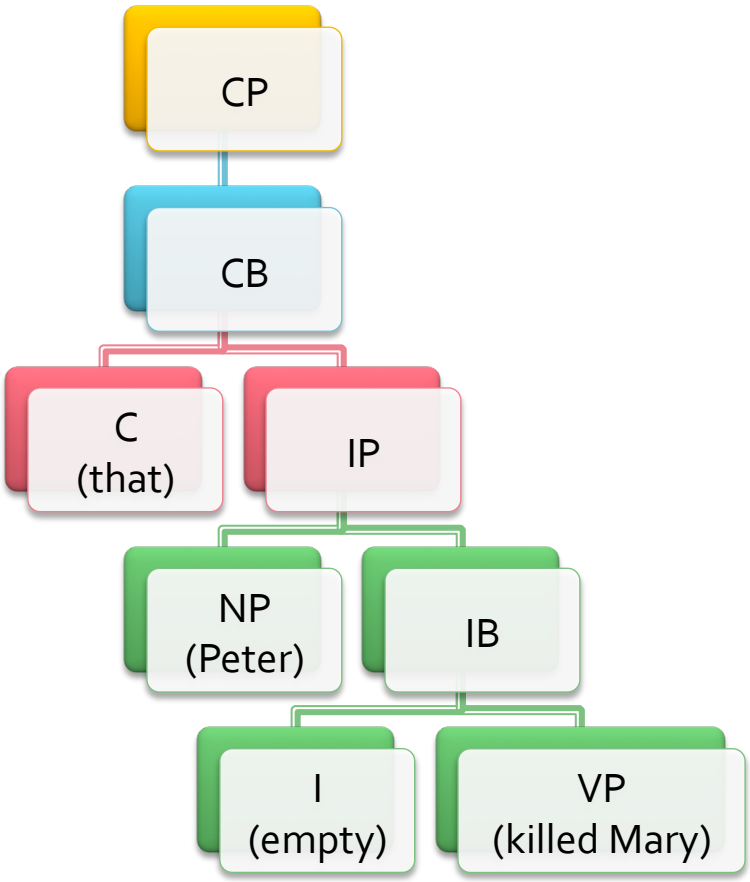
topicalization (of "yesterday")



IP
when to use

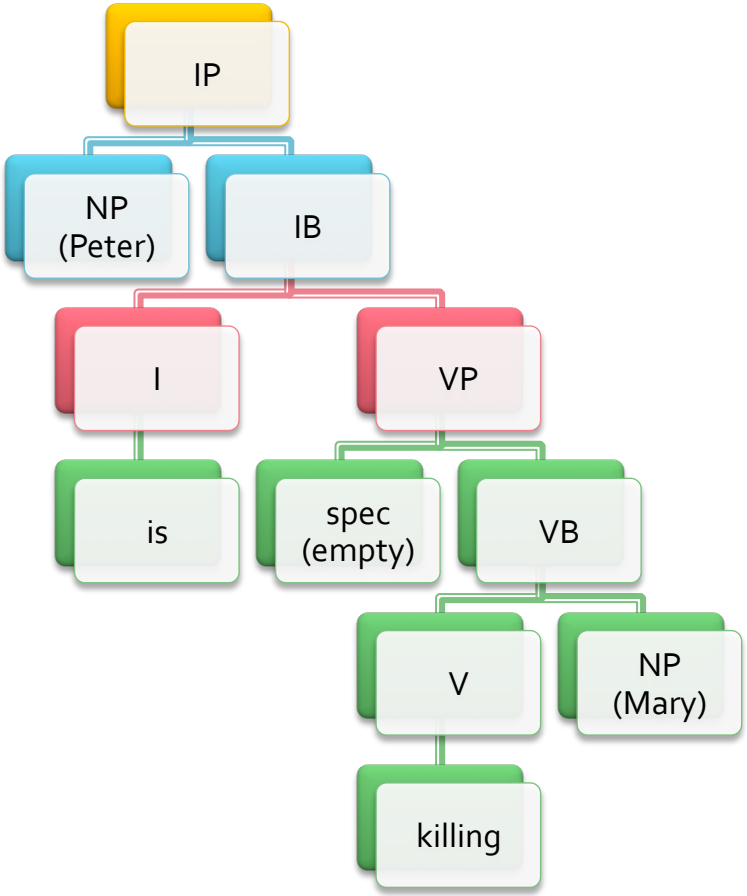
(I know that) Peter killed Mary

CP exists

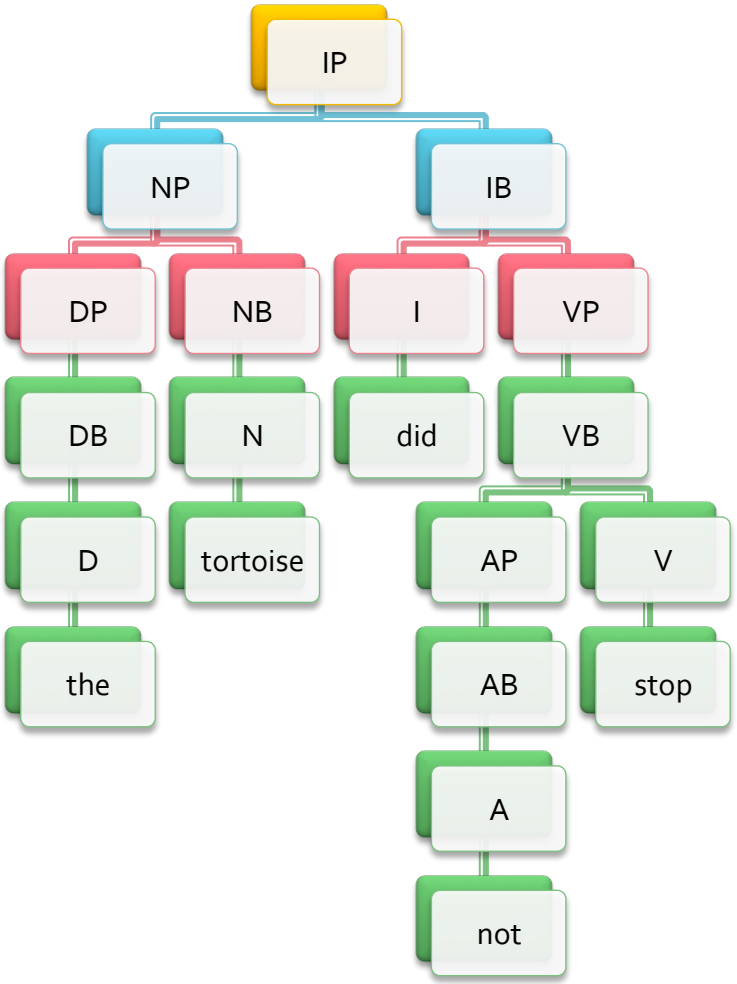


Peter is killing Mary

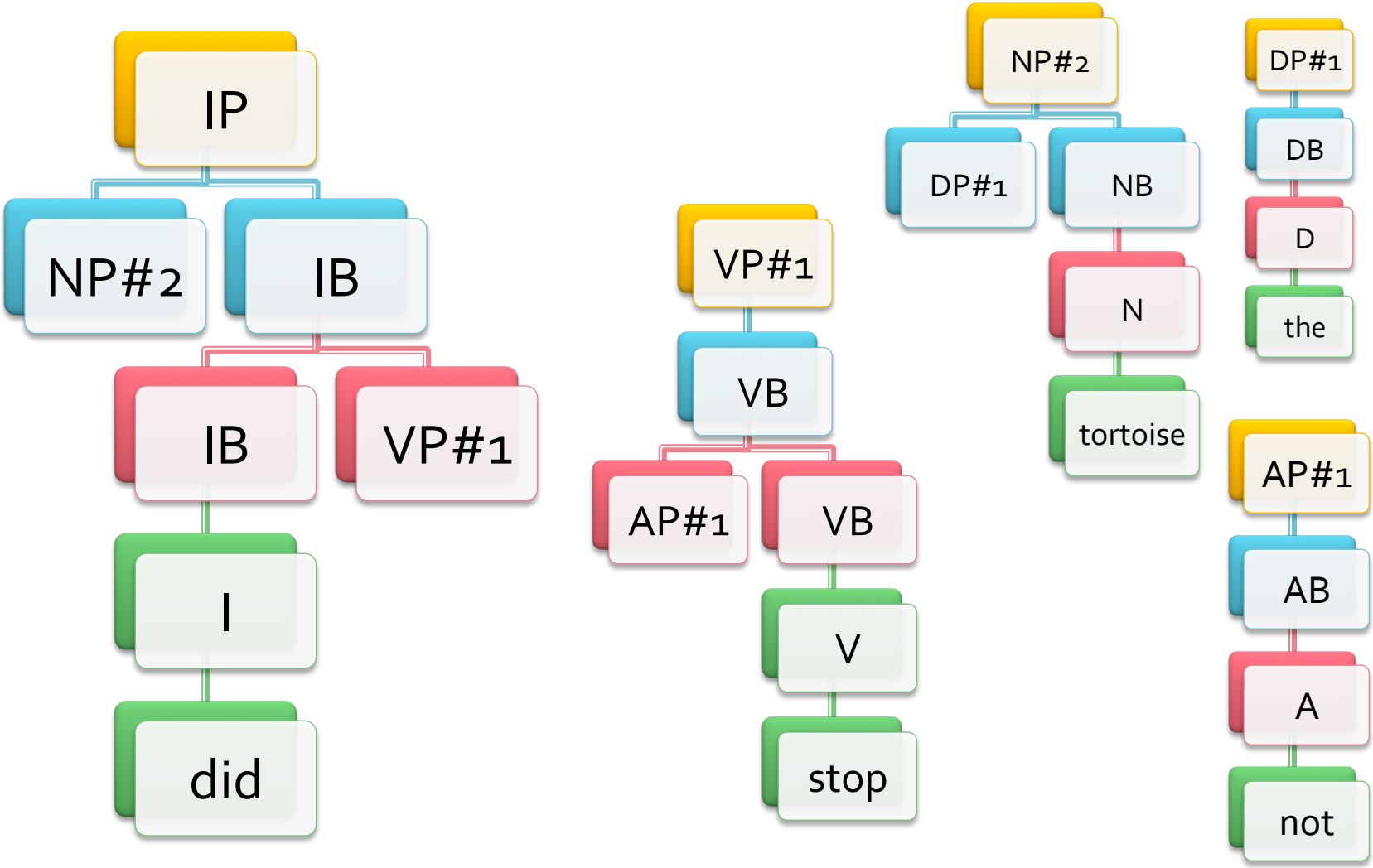
Auxiliary verbs



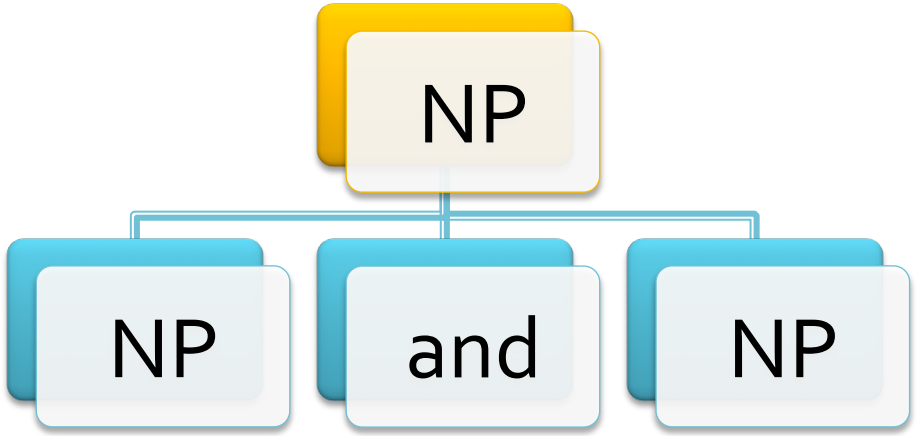
The Tortoise did not stop (I)

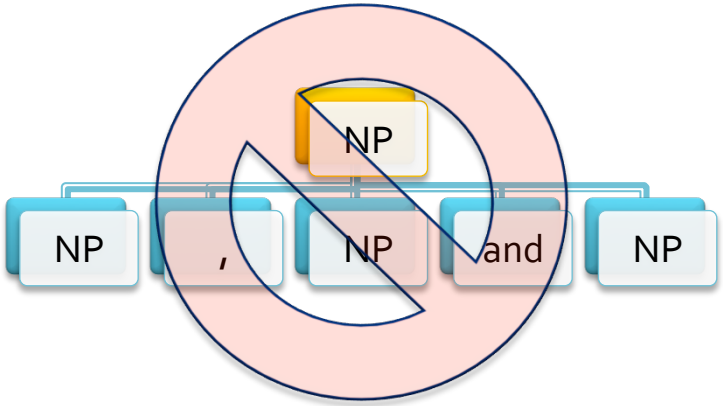


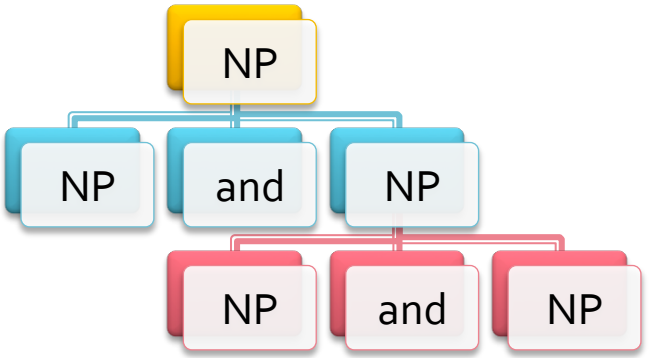
The Tortoise did not stop (II)



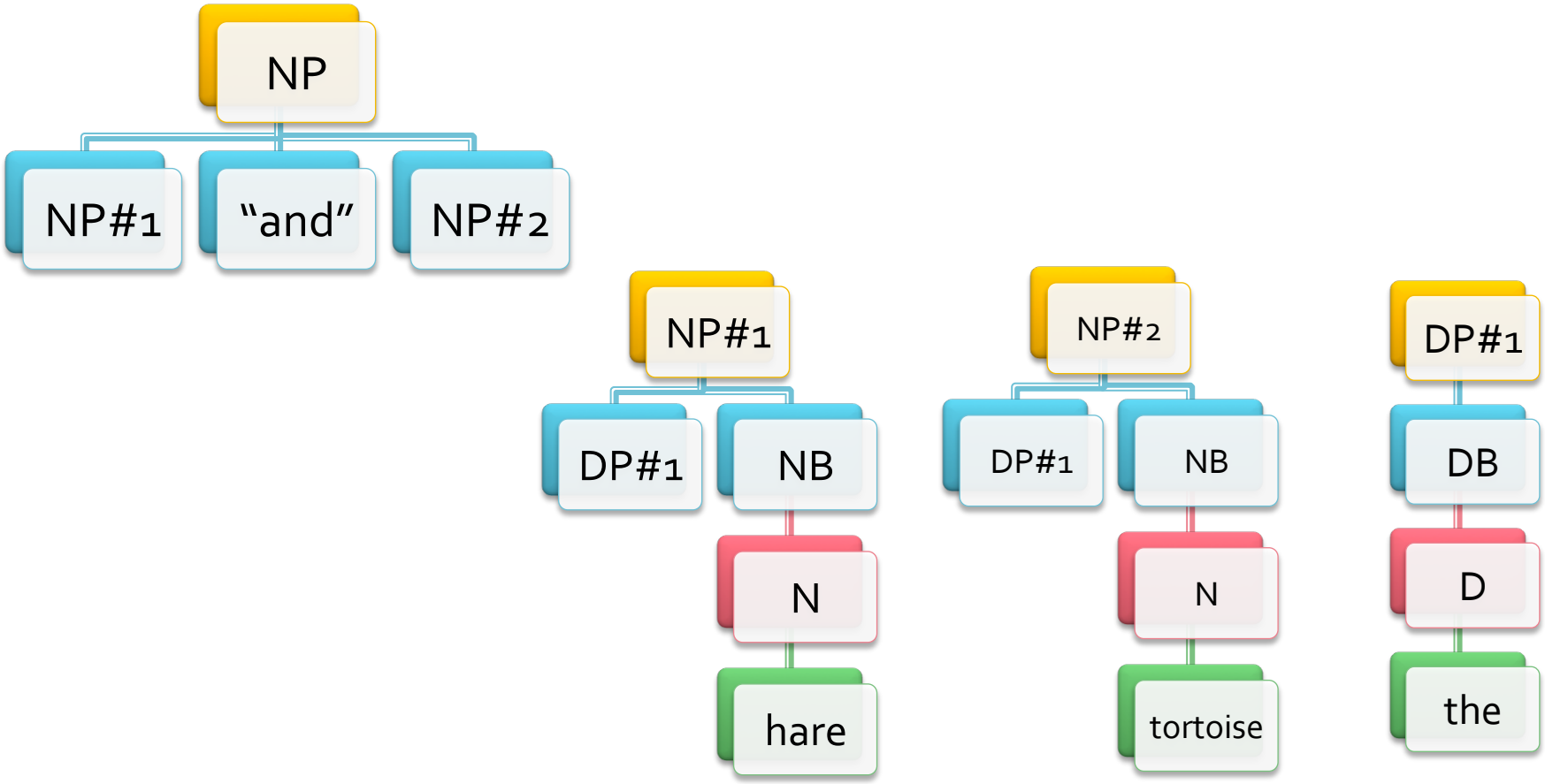
Coordination







The Hare and the Tortoise



Exercise #6

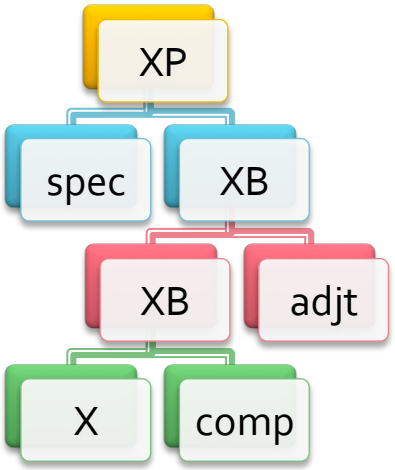
- Build the x-bar tree (modified) for the sentences of the corpus UCB1 (available at www.unlweb.net/wiki/UCB1)

1. ~~The Hare and the Tortoise~~
2. The Hare one day ridiculed the short feet and slow pace of the Tortoise.
3. ~~The Tortoise replied:~~
4. "Though you be swift as the wind, I will beat you in a race."
5. The Hare believed her assertion to be simply impossible and assented to the proposal.
6. They agreed that the Fox should choose the course and fix the goal.
7. On the day appointed for the race the two started together.
8. ~~The Tortoise did not stop.~~
9. She went on with a slow but steady pace straight to the end of the course.
10. The Hare laid down by the wayside and took a nap under a tree.
11. At last, he woke up and ran as fast as he could, but it was too late.
12. The Tortoise had already won the race.
13. Slow but steady progress wins the race.

Trees x Networks

Trees x Networks

TREE

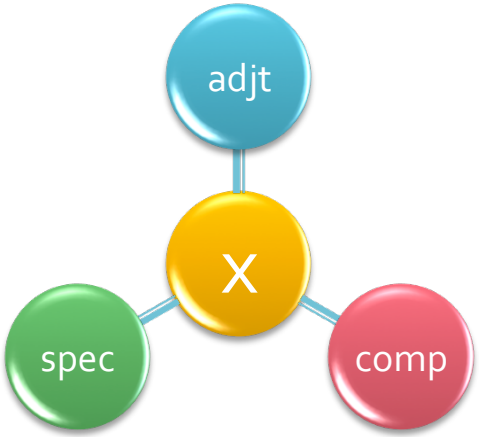


XP(XB(XB(X;comp);adjt);spec)

DEARBORISATION

ARBORISATION

NETWORK



XS(X;spec)
XA(X;adjt)
XC(X;comp)

Grammar Specs

www.unlweb.net/wiki/Grammar_Specs

Basic Symbols

Symbol	Definition	Example
^	not	^a = not a
{ | }	or	{a|b} = a or b
%	index for nodes, attributes and values	%x (see below)
#	index for sub-NLWs	#01 (see below)
=	attribute-value assignment	POS=NOU
!	rule trigger	!PLR
&	merge operator	%x&%y
?	dictionary lookup operator	?[a]
“ “	string	"went"
[ ]	natural language entry (headword)	[go]
[[ ]]	UW	[[to go(icl>to move)]]
( )	node	(a)
//	regular expression	/a{2,3}/ = aa,aaa

Pre-defined values

- SCOPE - Scope
- SHEAD - Sentence head (the beginning of a sentence)
- STAIL - Sentence tail (the end of a sentence)
- CHEAD - Scope head (the beginning of a scope)
- CTAIL - Scope tail (the end of a scope)
- TEMP - Temporary entry (entry not found in the dictionary)
- DIGIT - Any sequence of digits (i.e.: 0,1,2,3,4,5,6,7,8,9)

Nodes

(NODES)

- "string" {1}
- [headword] {1}
- [[UW]] {1}
- %index {1}
- attribute=value {0,}
- feature {0,}

Examples of Nodes

- ("ing")
  - (a node making reference only to its actual string value)
- ([book])
  - (a node making reference only to its headword,i.e., its original state in the dictionary)
- ([[book(icl>document)]])
  - (a node making reference only to its UW value)
- (SNG)
  - (a node making reference only to one of its features)
- (POS=NOU)
  - (a node making reference only to one of its features in the attribute-value pair format)
- (%x)
  - (a node making reference only to its unique index)
- ("string",[headword], [[UW]],feature1,feature2,…,attribute1=value1,attribute2=value2,…,%x)
  - (complete node)

Properties of Nodes (I)

- **Nodes are enclosed between (parentheses)**
  - ("a") is a node
  - "a" is not a note
- **The elements of a node are separated by comma**
  - ("a",[a],[[a]],A,B,A=C,%a)
- **The order of elements inside a node is not relevant.**
  - ("a",[a],[[a]],A,B,A=C,%a) is the same as ([[a]],B,A,"a",[a],A=C,%a)
- **Nodes may have one single string, headword, UW and index, but may have as many features as necessary**
  - ~~("a","b")~~ (a node may not contain more than one string)
  - ~~([a],[b])~~ (a node may not contain more than one headword)
  - ~~([[a]],[[b]])~~ (a node may not contain more than one UW)
  - ~~(%a,%b)~~ (a node may not contain more than one index)
  - (A,B,C,D,…,Z) (a node may contain as many features as necessary)

Properties of Nodes (II)

- **A node may be referred by any of its elements**
  - ("a") refers to all nodes where actual string = "a"
  - ([a]) refers to all nodes where headword = [a]
  - ([[a]]) refers to all nodes where UW = [[a]]
  - (%a) refers to all nodes having the feature A
  - ("a",[a],[[a]],A) refers to all nodes having the feature A where string = "a" and headword = [a] and UW = [[a]]
- **Nodes are automatically indexed according to a position-based system if no explicit index is provided**
  - ("a")("b") is actually ("a",%01)("b",%02)
- **Regular expressions may be used to make reference to any element of the node, except the index**
  - ("/a{2,3}/") refers to all nodes where string is a sequence of 2 to 3 characters "a"
  - ([/a{2,3}/]) refers to all nodes where headword is a sequence of 2 to 3 characters "a"
  - ([[/a{2,3}/]]) refers to all nodes where UW is a sequence of 2 to 3 characters "a"
  - (/a{2,3}/) refers to all nodes having a feature that is a sequence of 2 to 3 characters "a"

Properties of Nodes (III)

- **Nodes may contain disjoint features enclosed between {braces} and separated by the vertical bar |**
  - ({A|B}) refers to all nodes having the feature A OR B
- **Node features may be expressed as simple attributes, or attribute-value pairs**
  - (MCL) - feature as an attribute: refers to all nodes having the feature MCL
  - (GEN=MCL) - feature as an attribute-value pair, which is the same as (GEN,MCL): refers to all nodes having the features GEN and MCL.
- Attribute-value pairs may be used to create co-reference between different nodes (as in agreement):
  - (%x,GEN)(%y,GEN=%x) - the value of the attribute GEN of the node %x is the same of the attribute GEN of the node %y

Relations

- REL(ARG#1;ARG#2;…;ARG#n)
  - Relations **do not** have any feature:
    - string
    - headword
    - attribute
    - values
    - indexes
    - etc.
  - Relations may have scopes
    - REL(ARG#1;:01)
    - REL:01(ARG#2;ARG#3)

Examples of Relations

- L(%x;%y) - linear relation
- agt(%x;%y) - semantic relation
- VH(%x) - unary syntactic relation
- VC(%x;%y) - binary syntactic relation
- XX(%x;%y;%z) - possible ternary syntactic relation

Properties of Relations (I)

- Arguments of relations are not commutative
  - relation(%x;%y) ≠ relation(%y;%x)
- Inside each relation, nodes are isolated by semicolon (;)
  - relation(%x;%y)
- Inside each node, features are isolated by comma (,)
  - relation("string1",[headword1], [[UW1]],feature1,attribute=value,…,%x; "string2", [headword2],[[UW2],feature2,attribute=value,…,%y)

Properties of Relations (II)

- **Relations may be disjoined through {braces}**
  - {("a")|("b")}("c") - either ("a")("c") or ("b")("c")
  - {agt(%x;%y)|exp(%x;%y)}obj(%x;%z) - either agt(%x;%y)obj(%x;%z) or exp(%x;%y)obj(%x;%z)
- **Relations may be replaced by regular expressions**
  - /.{2,3}/(%x;%y) - any relation made of two or three characters between %x and %y

Types of Grammar

Types of Grammar

T-GRAMMAR

- Mandatory
- Transformation rules
- CONDITION:=ACTION;
- Examples:
  - (A,%x):=(%x,-A,+B);
  - (A,%x)(B,%y):=;
  - (A,%x)(B,%y):=(%x);
  - (A,%x)(B,%y):=(%y);
  - (A,%x)(B,%y):=(%y)(%x);
  - (A,%x)(B,%y):=(%x)(C,%z)(%y);
  - (A,%x)(B,%y):=rel(%x;%y);

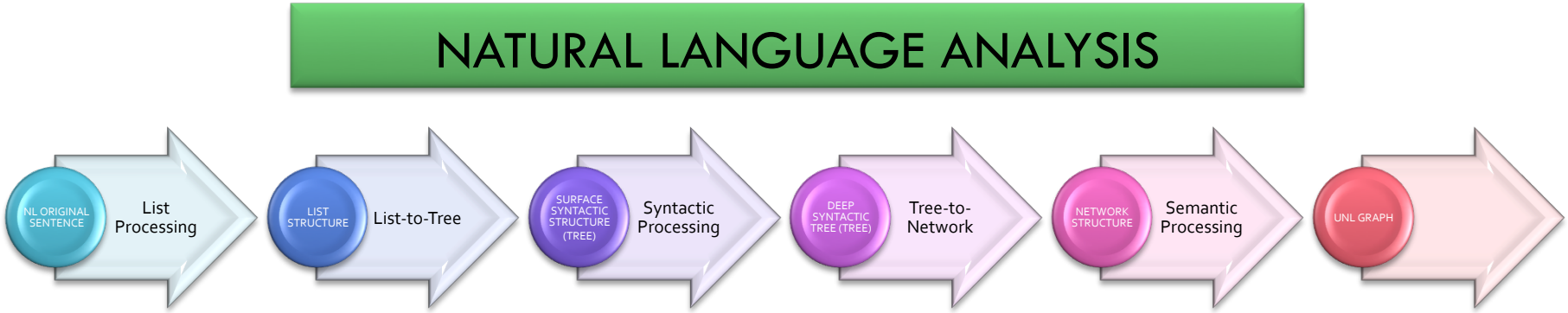
D-GRAMMAR

- Optional
- Disambiguation rules
- CONDITION=PROBABILITY;
- Examples:
  - (%a)=0;
  - (%b)(%a)=0;
  - (%a)(%b)=1;

UNDL FOUNDATION

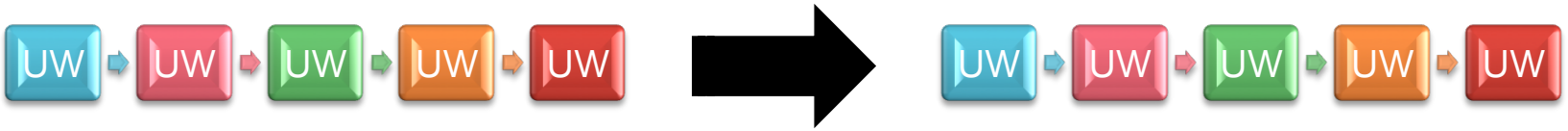
Transformation Rules

Analysis


NATURAL LANGUAGE ANALYSIS
NL ORIGINAL SENTENCE
List Processing
LIST STRUCTURE
List-to-Tree
SURFACE SYNTACTIC STRUCTURE (TREE)
Syntactic Processing
DEEP SYNTACTIC TREE (TREE)
Tree-to-Network
NETWORK STRUCTURE
Semantic Processing
UNL GRAPH

LIST-TO-LIST (LL)

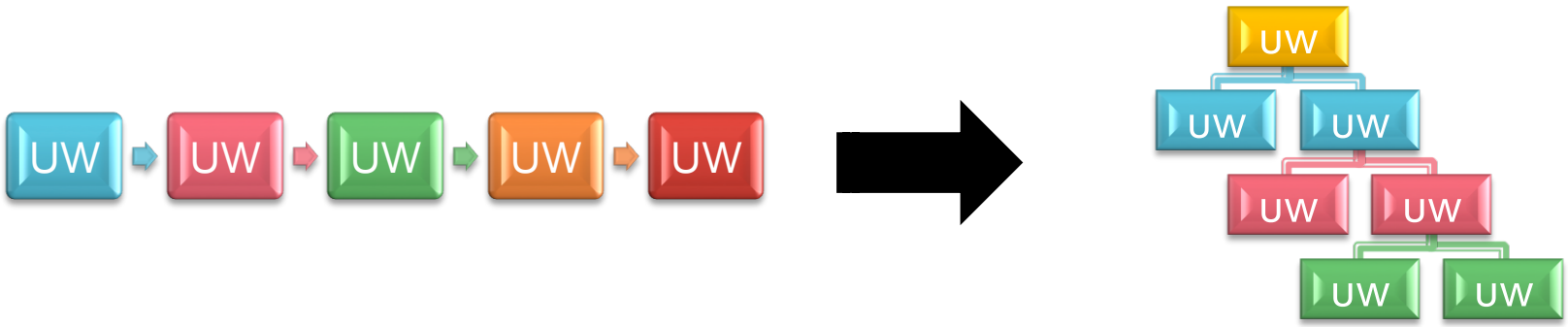
Transformation Rules



ACTION	RULE
ADD	(%a):=(%a)(%b); or (%a):=(%b)(%a);
DELETE	(%a):=-(%a);
REPLACE	(%a):=(%b);
MERGE	(%a)(%b):=(%c);
DIVIDE	(%a):=(%b)(%c);

LIST-TO-TREE (LT)

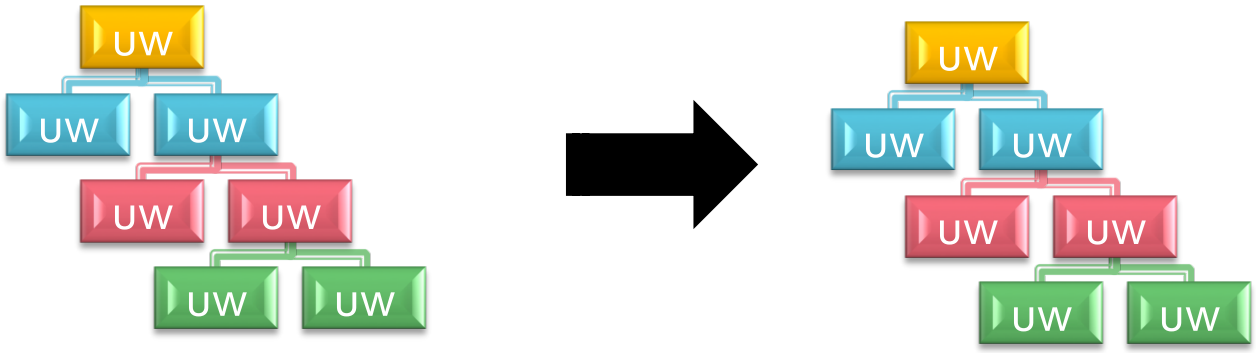
Transformation Rules



ACTION	RULE
REPLACE	(%a)(%b):=SYN(%a;%b);

TREE-TO-TREE (TT)

Transformation Rules

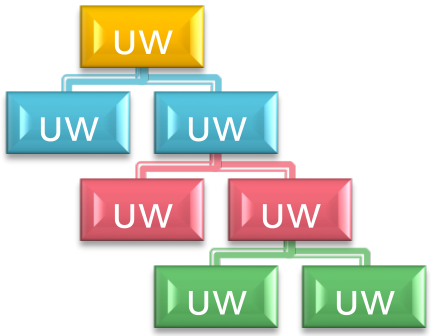


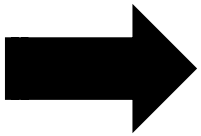
ACTION	RULE
ADD RELATION	SYN(%a;%b):=+SYN(%c;%d);
DELETE RELATION	SYN(%a;%b):=-SYN(%a;%b);
REPLACE RELATION	SYN(%a;%b):=SYN(%c;%d);
MERGE RELATION	SYN(%a;%b)SYN(%c;%d):=SYN(%e;%f);
DIVIDE RELATION	SYN(%a;%b):=SYN(%c;%d)SYN(%e;%f);

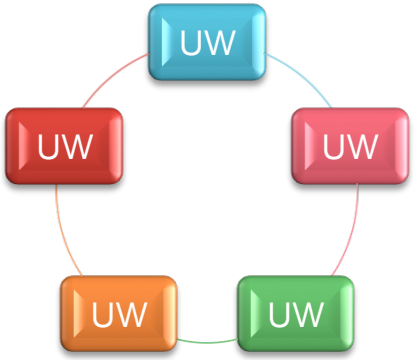
ACTION	RULE
ADD NODE	SYN(%a;%b):=SYN(%a;%b;%c);
DELETE NODE	SYN(%a;%b):=SYN(%a);

TREE-TO-NETWORK (TN)

Transformation Rules



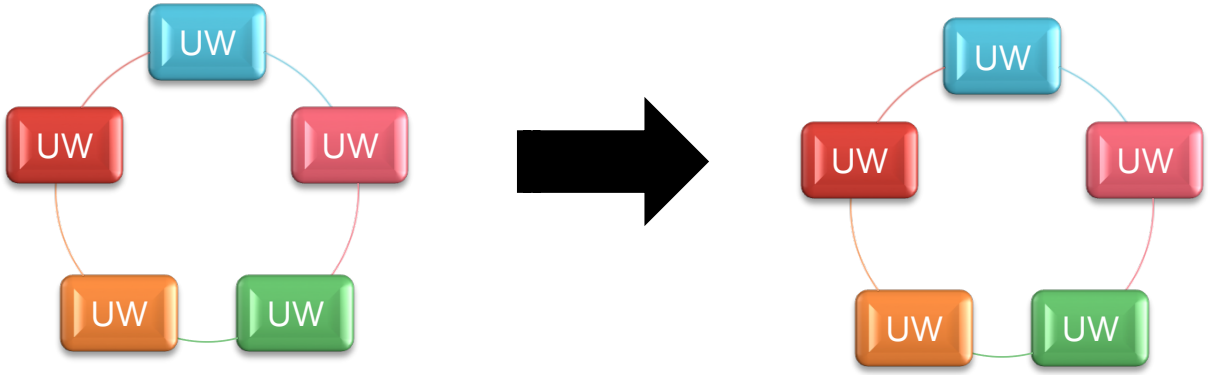




ACTION	RULE
REPLACE	SYN(%c;%d):=SEM(%a;%b);:

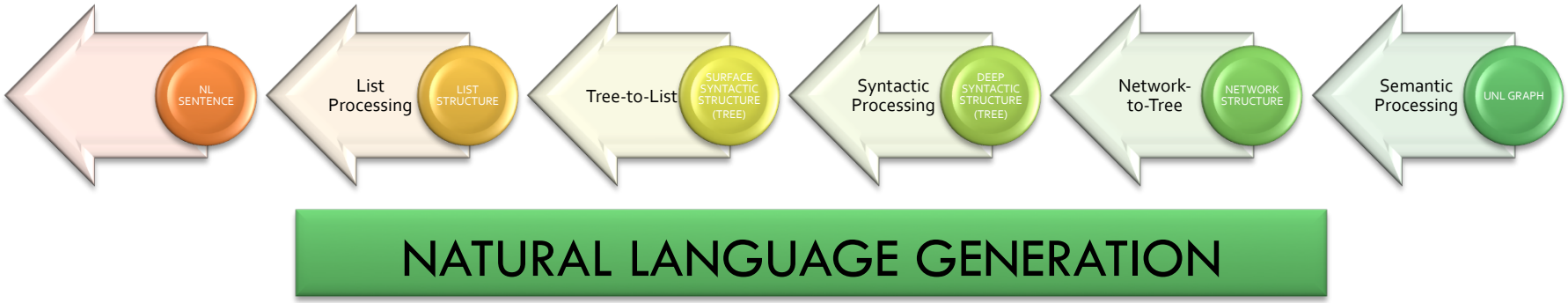
NETWORK-TO-NETWORK (NN)

Transformation Rules



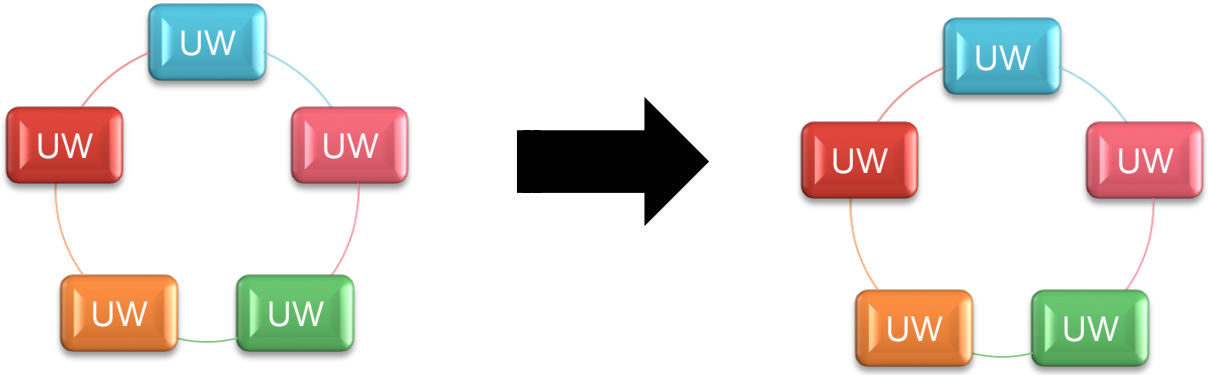
ACTION	RULE
ADD RELATION	SEM(%a;%b):=+SEM(%c;%d);
DELETE RELATION	SEM(%a;%b):=-SEM(%a;%b);
REPLACE RELATION	SEM(%a;%b):=SEM(%c;%d);
MERGE RELATION	SEM(%a;%b)SEM(%c;%d):=SEM(%e;%f);
DIVIDE RELATION	SEM(%a;%b):=SEM(%c;%d)SEM(%e;%f);

Generation


NL SENTENCE
List Processing
LIST STRUCTURE
Tree-to-List
SURFACE SYNTACTIC STRUCTURE (TREE)
Syntactic Processing
DEEP SYNTACTIC STRUCTURE (TREE)
Network-to-Tree
NETWORK STRUCTURE
Semantic Processing
UNL GRAPH
NATURAL LANGUAGE GENERATION

NETWORK-TO-NETWORK (NN)

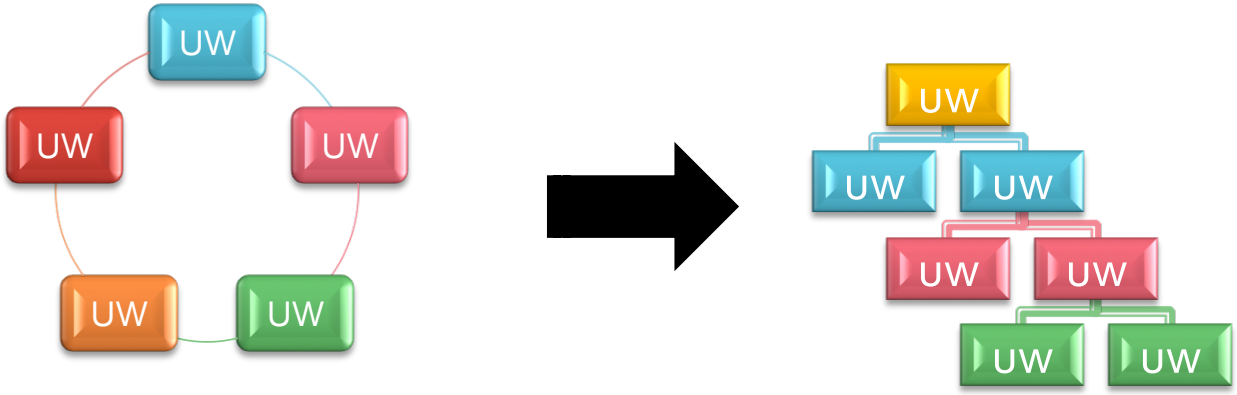
Transformation Rules



ACTION	RULE
ADD RELATION	SEM(%a;%b):=+SEM(%c;%d);
DELETE RELATION	SEM(%a;%b):=-SEM(%a;%b);
REPLACE RELATION	SEM(%a;%b):=SEM(%c;%d);
MERGE RELATION	SEM(%a;%b)SEM(%c;%d):=SEM(%e;%f);
DIVIDE RELATION	SEM(%a;%b):=SEM(%c;%d)SEM(%e;%f);

NETWORK-TO-TREE (NT)

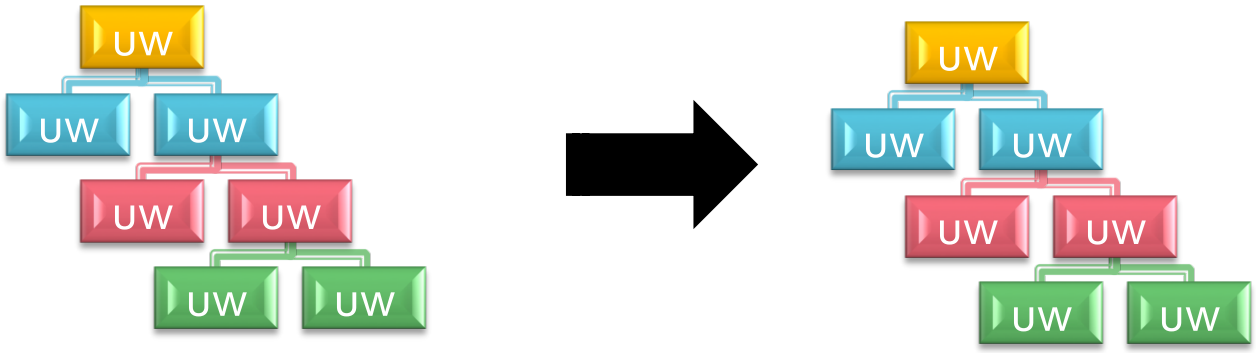
Transformation Rules



ACTION	RULE
REPLACE	SEM(%a;%b):=SYN(%c;%d);

TREE-TO-TREE (TT)

Transformation Rules

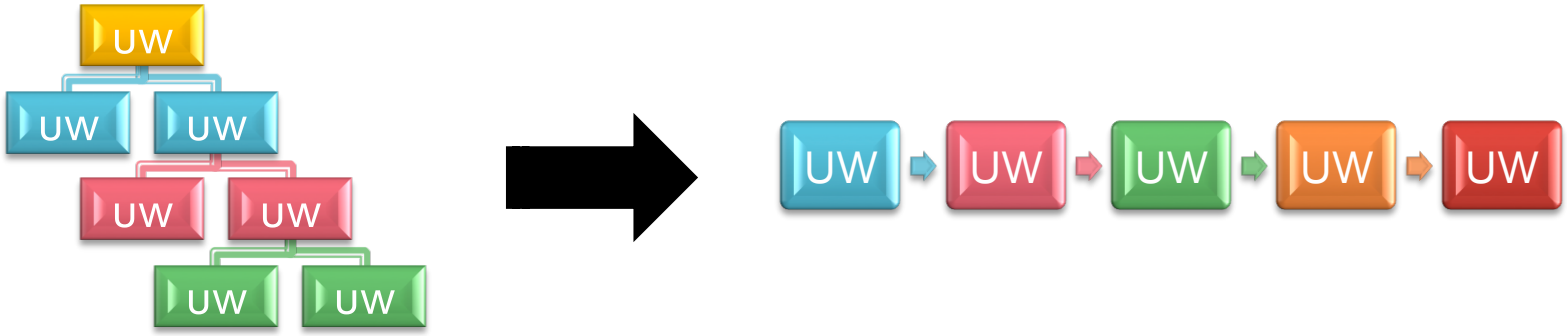


ACTION	RULE
ADD RELATION	SYN(%a;%b):=+SYN(%c;%d);
DELETE RELATION	SYN(%a;%b):=-SYN(%a;%b);
REPLACE RELATION	SYN(%a;%b):=SYN(%c;%d);
MERGE RELATION	SYN(%a;%b)SYN(%c;%d):=SYN(%e;%f);
DIVIDE RELATION	SYN(%a;%b):=SYN(%c;%d)SYN(%e;%f);

ACTION	RULE
ADD NODE	SYN(%a;%b):=SYN(A;B;C);
DELETE NODE	SYN(%a;%b):=SYN(%a);

TREE-TO-LIST (TL)

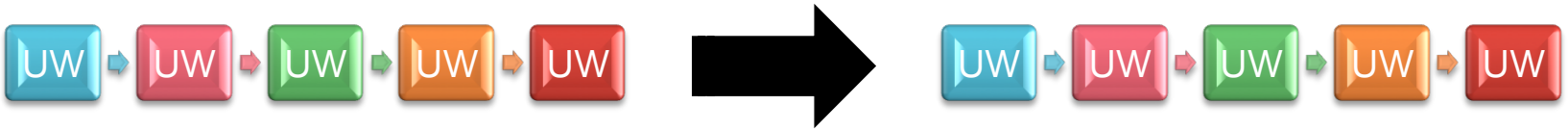
Transformation Rules



ACTION	RULE
REPLACE	SYN(%a;%b):=(%c);

LIST-TO-LIST (LL)

Transformation Rules



ACTION	RULE
ADD	(%a):=(%a)(%b); or (%a):=(%b)(%a);
DELETE	(%a):=-(%a);
REPLACE	(%a):=(%b);
MERGE	(%a)(%b):=(%c);
DIVIDE	(%a):=(%b)(%c);

General Properties of
Transformation Rules

PRIORITY
Rules are applied serially, according to the order defined in the grammar. The first rule will be the first to be applied, the second will the second, and so on.

RECURSIVENESS
Rules are applied recursively as long as their conditions are true.

COMPREHENSIVENESS
Grammars are applied comprehensively as long as there is at least one applicable rule.

ACTION
The rules may add or delete values to the source and the target nodes, but only in the right side items:

agt(a;b):=agt(+c;);
agt(a;b):=agt(;-b);

General Properties of
Transformation Rules

CONSERVATION

Rules affect only the information clearly specified. No relation, node or feature is deleted unless explicitly informed.

For instance, in the examples below, the source node of the "agt" relation preserves, in all cases, the value "a". The only change concerns the feature "c", which is added to the source node of the "agt" in the first two cases; and the feature "b", which is deleted from the target node in the third case.

agt(a;b):=agt(c;);
agt(a;b):=agt(+c;);
agt(a;b):=agt(;-b);

In any case, the ADD and DELETE rules (i.e., when the right side starts with "+"or "-") **preserve** the items in the left side, except for the explicitly deleted ones:

General Properties of
Transformation Rules

SCOPE

The REPLACE, MERGE and DIVIDE rules affect only their designated scopes.

NN may only replace, merge or divide semantic relations; TT may only replace, merge or divide syntactic relations; and LL may only replace, merge or divide list nodes. All other information is preserved, unless explicitly informed.

INPUT:	agt(a;b) cob(a;c)
RULE:	cob(;):=obj(;);
OUTPUT:	agt(a;b) obj(a;c)

INPUT:	agt(a;b) cob(a;c)
RULE:	cob(a;):=obj(-a,+d;);
OUTPUT:	agt(a;b) obj(d;c)

General Properties of
Transformation Rules

CONJUNCTION

Both the left and the right side of the rule may have as many items as necessary.
The items must be juxtaposed.

SEM(%a;%b)SEM(%c;%d)SEM(%e;%f):=SEM(G;H)SEM(I;J)SEM(K;L);

DISJUNCTION

The left side of the rules may bring disjuncts, but not the right side.

{SEM(%a;%b)|SEM(%c;%d)}^SEM(%e;%f):=+SEM(%e;%f);
SEM(%a;%b){SEM(%c;%d)|SEM(%e;%f)}:=-SEM(%a;%b);
agt(VER,{V01|V02};NOU,^SNG}):=;

General Properties of
Transformation Rules

COMMUTATIVITY

Inside the same side of the rule, the order of the factors does not affect the end result, except for list-processing rules (LL, LT and TL).

SEM(%a;%b):=SEM(%c;%d)SEM(%e;%f); = SEM(%a;%b):= SEM(%e;%f)SEM(%c;%d);
SYN(%a;%b):=SYN(%c;%d)SYN(%e;%f); = SYN(%a;%b):= SYN(%e;%f)SYN(%c;%d);

But:

(%a):=(%b)(%c); ≠ (%a):=(%c)(%b);
SYN(%a;%b):=(%c)(%d); ≠ SYN(%a;%b):=(%d)(%c);
(%c)(%d):=SYN(%a;%b); ≠ (%d)(%c):=SYN(%a;%b);

Additionally, the order of the features inside a relation does not affect the end result, but the order of the nodes is non-commutative.

SEM(VER,TRA ; NOU,MCL) = SEM(TRA,VER ; MCL,NOU)

But:

SEM(VER,TRA ; NOU,MCL) ≠ SEM(NOU,MCL ; VER,TRA)

General Properties of
Transformation Rules

INDEXATION

- Default indexation
  - If omitted, indexes are assigned by default
  - In default indexation, left-side nodes are automatically co-indexed with right-side nodes **if and only if** their position and number are the same:
    - X(A;B):=Y(C;D); is the same as X(**%01**,A;**%02**,B):=Y(**%01**,C;**%02**,D);
  - Non-co-indexed nodes in the right side means ADDITION, whereas left-side nodes that are not referred to in the right side means DELETION
    - X(%a;%b):=Y(%a;X;%b); is the same as X(%a;%b):=Y(%a;**%02**,X,;%b);
  - Indexes may also be used to transfer attribute values expressed in the format ATTRIBUTE=VALUE
    - X(A,%a,ATT1=VAL1;B,%b):=X(%a;%b,ATT1=%a);

Examples of T-rules

- (BLK):=;
  - deletes the blank space
- (N,PLR,^@pl,^@multal,^@paucal,^@all):=(+att=@pl);
  - books > book.@pl
- ([not])([to],%x)(V,%y):=(%x)(+att=@not,%y);
  - not to do > to do.@not
- (D,%d)([all],%all):=(%all)(%d);
  - the all books > all the books, my all books > all my books
- (TEMP,%x)(BLK,%y)(TEMP,%z):=(%x&%y&%z,-BLK);
  - merges temporary words
- agt(%x,COP;%y)obj(%x;%z):=aoj(%z,+att=%x;%y);
  - eliminates the copula

Exercise #7

- Write the following rules:
  1. Delete all the punctuation signs (PUT)
  2. Add a blank space (BLK) between two nouns (N)
  3. Reverse the order of (J)(N)
  4. Remove the feature MCL from the nouns (N)
  5. Add the feature VER to the verbs (V) that do not have this feature yet
  6. Copy the value of the attribute gender (GEN) from the noun (N) to the immediately adjacent adjectives (J), to the left and to the right
  7. Create a relation REL between a noun (N) and the verb (V) that comes immediately at its left side
  8. Delete a relation REL between an adverb (%a) and a preposition (P)
  9. Reverse the order of the arguments inside a relation REL between a verb (V) and a pronoun (R)
  10. Replace the relation REL between two verbs by the relation REL2 between the same verbs

Disambiguation rules

Disambiguation rules

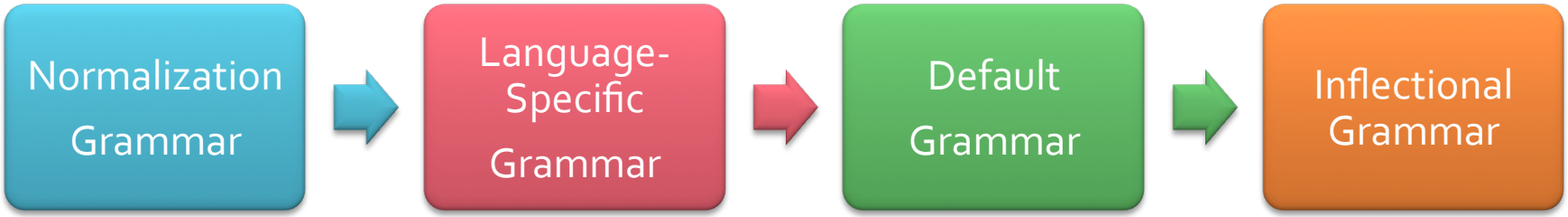
- Negative rules (prohibition)
  - (DET)(V)=0;
    - Determiners cannot precede verbs
- Positive rules (induction)
  - (SHEAD)(DET)=1;
    - Determiners normally come at the beginning of the sentence

Exercise #8

1. Prevent the string « a » from occurring before the string « b »
2. Prevent a noun from occurring before a verb
3. Prevent a preposition from occurring at the beginning of the sentence
4. Prevent any word, except verbs, from occurring at the end of the sentence

Grammar Structure

Grammar structure



Normalization Grammar

- Standardization
  - Puts the features in the structure
    - attribute=value
  - Examples

    (SNG,^NUM):=(-SNG,+NUM=SNG);

    (PLR,^NUM):=(-PLR,+NUM=PLR);
- Propagation
  - Propagates the values of features

    (SNGT,^SNG):=(-NUM,-SGNT,+NUM=SNG,+NUM=SNGT);

    (PLRT,^PLR):=(-NUM,-PLRT,+NUM=PLR,+NUM=PLRT);

    (INV,^SNG,^PLR):=(-NUM,-INV,+NUM=SNG,+NUM=PLR,+NUM=INV);

Language-Specific Grammar

- ([not])([to],%x)(V,%y):=(%x)(+att=@not,%y);
  - not to do > to do.@not
- ({[not]|[n't]})({V,^AUX|J|N|A|D},%x):=(+att=@not,%x);
  - not kill > kill.@not

Default Grammar

- (TEMP,%x)(BLK,%y)(TEMP,%z):=(%x&%y&%z,-BLK);
  - merges temporary words
- (PPN,%x)(BLK,%y)(PPN,%z):=(%x&%y&%z,+TEMP,-BLK);
  - merges sequences of proper names
- (BLK):=;
  - deletes the blank space (this rule applies only if the blank space is in the dictionary)

Exercise #9 (IAN)

- Download the normalization and the default grammar from the wiki and upload them to IAN.
- Test the grammars with the dictionary and the corpus UCA1_<your locale>.txt
- Analyze the results

The grammars are available at:
www.unlweb.net/wiki/UCA1

Exercise #10 (EUGENE)

- Download the normalization and the default grammar from the wiki and upload them to EUGENE.
- Test the grammars with the dictionary and the corpus UCA1_unl.txt

The grammars are available at:
www.unlweb.net/wiki/UCA1