Macau, March 13th

XI UNL School

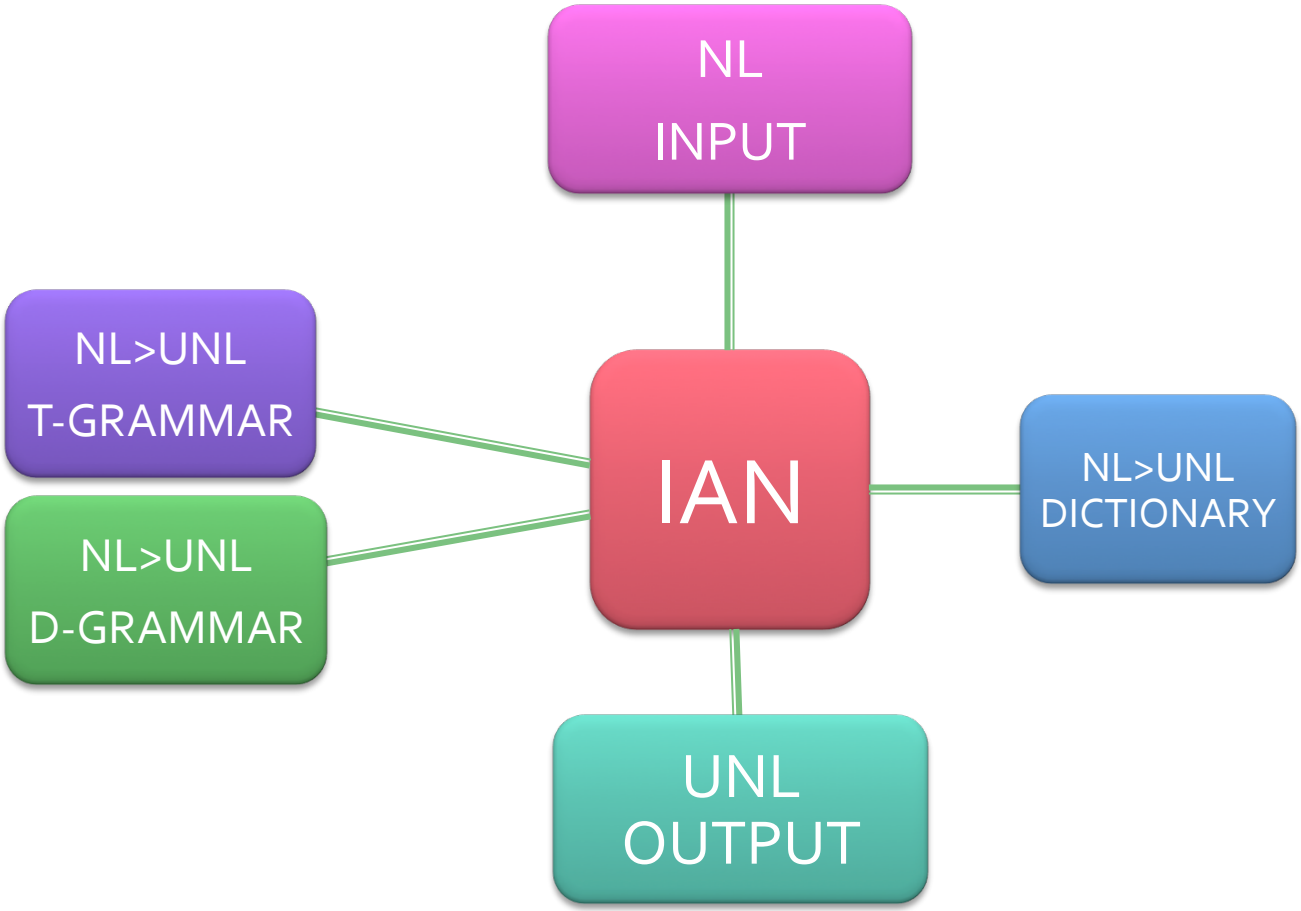
Day #3

Day #3

- UNLization
  - IAN
  - UNLization steps
  - Examples

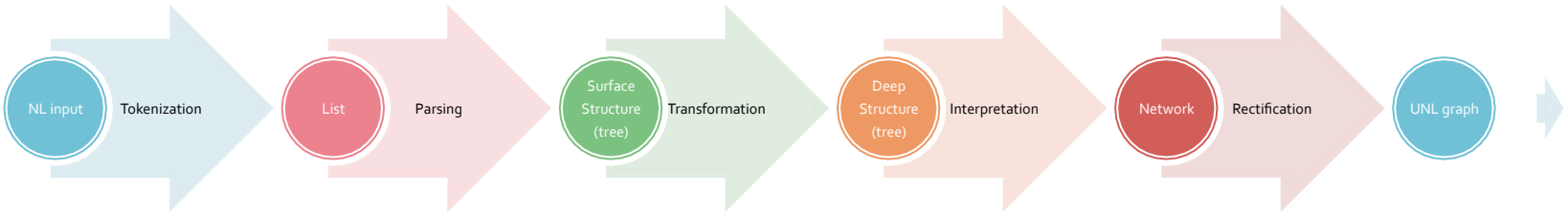
IAN

IAN (Interactive ANalizer)



UNLization Steps

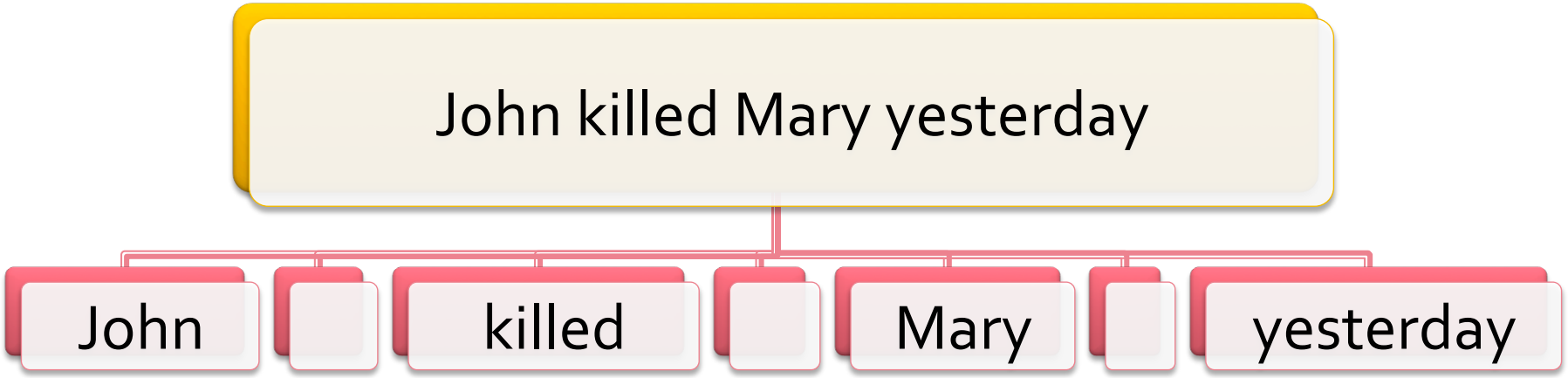
UNLization Steps



Tokenization

Tokenization

- Longest first
- From left to right
- Can be controlled by d-rules

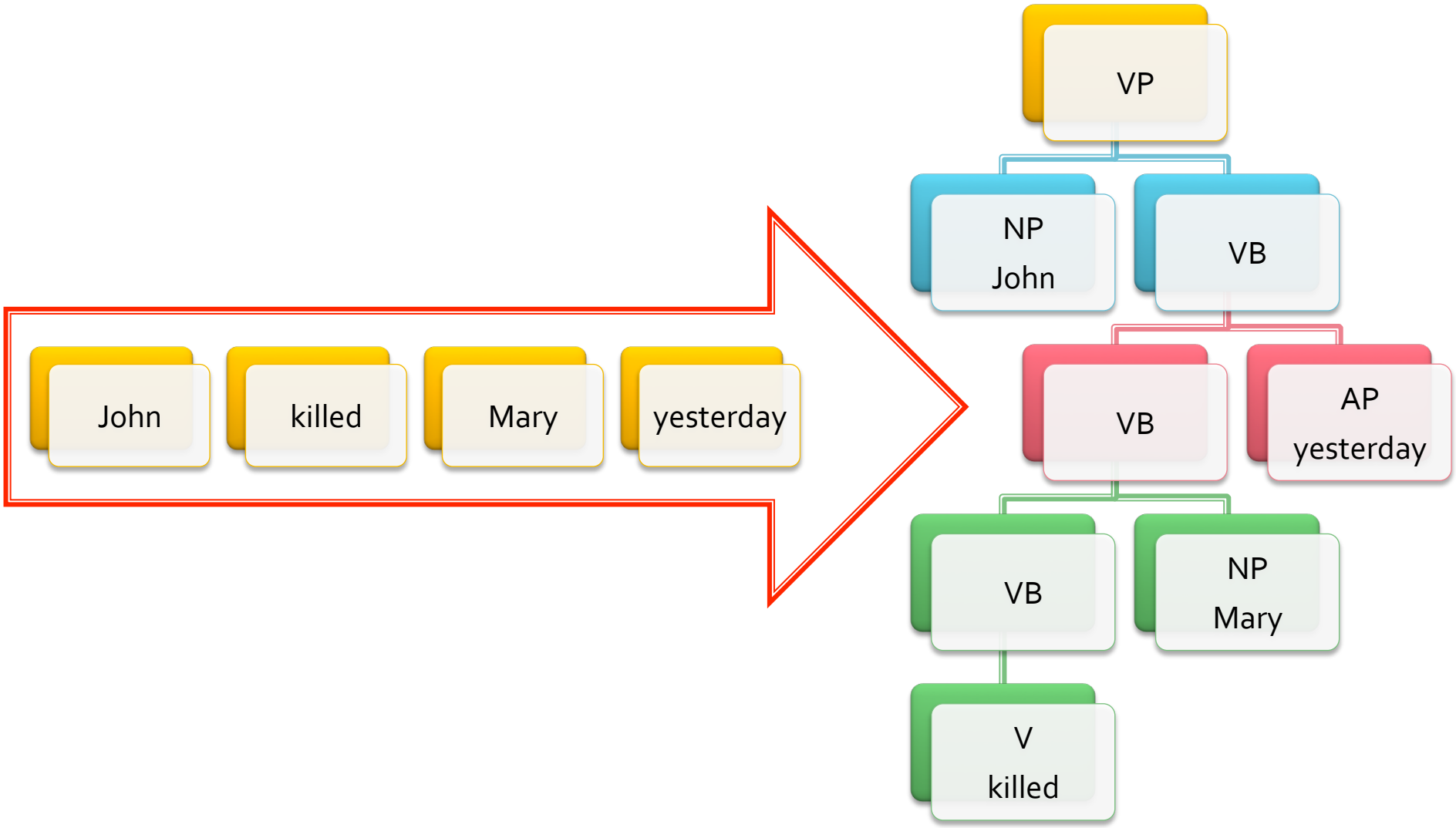


Example

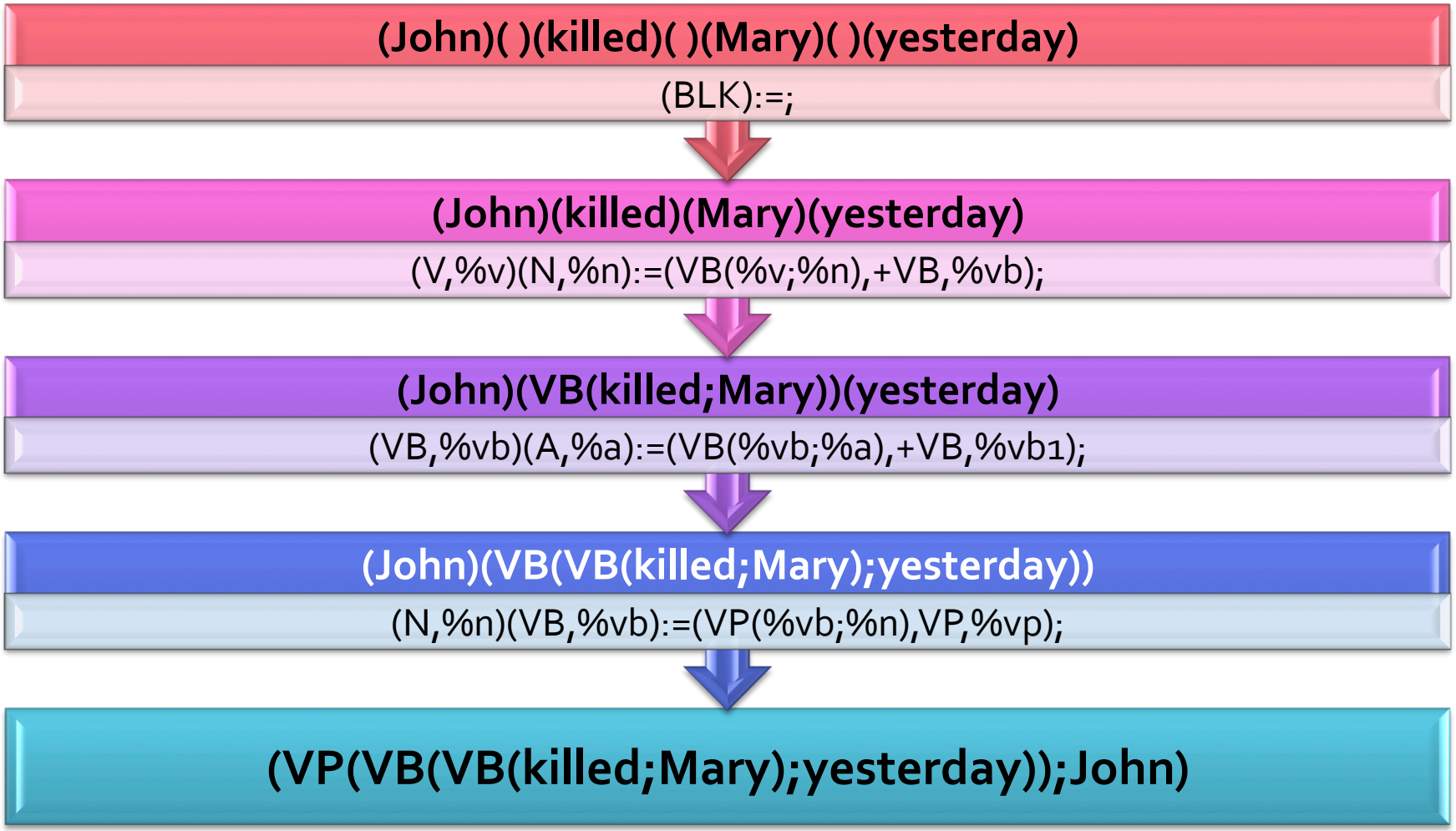
- Dictionary
  - [a] {} "1" (ATT=A) <eng,100,0>;
  - [aa] {} "2" (ATT=B) <eng,50,0>;
  - [aaa] {} "3" (ATT=C) <eng,10,0>;
- Input
  - aaaaaaaa
- Result #1
  - [aaa][aaa][aa]
- D-Grammar
  - (C)(C)=0;
- Result #2
  - [aaa][aa][aaa]

Parsing

Parsing
list > tree

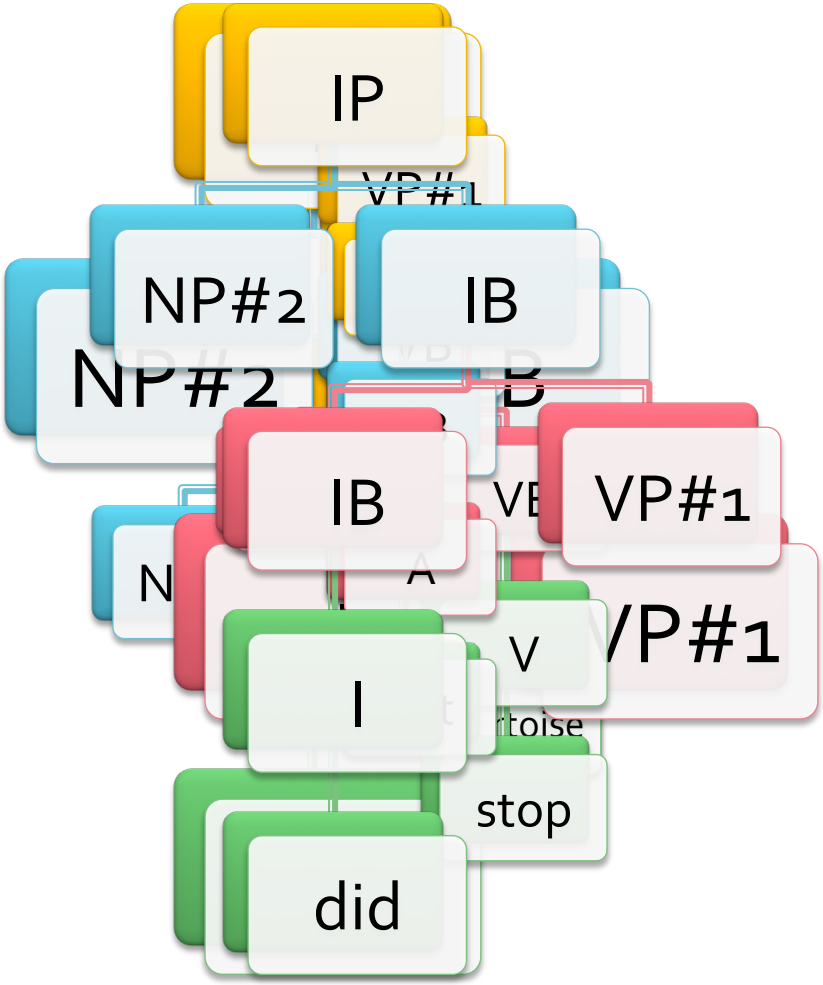


Example


(John)( )(killed)( )(Mary)( )(yesterday)
(BLK):=;
(John)(killed)(Mary)(yesterday)
(V,%v)(N,%n):=(VB(%v;%n),+VB,%vb);
(John)(VB(killed;Mary))(yesterday)
(VB,%vb)(A,%a):=(VB(%vb;%a),+VB,%vb1);
(John)(VB(VB(killed;Mary);yesterday))
(N,%n)(VB,%vb):=(VP(%vb;%n),VP,%vp);
(VP(VB(VB(killed;Mary);yesterday));John)

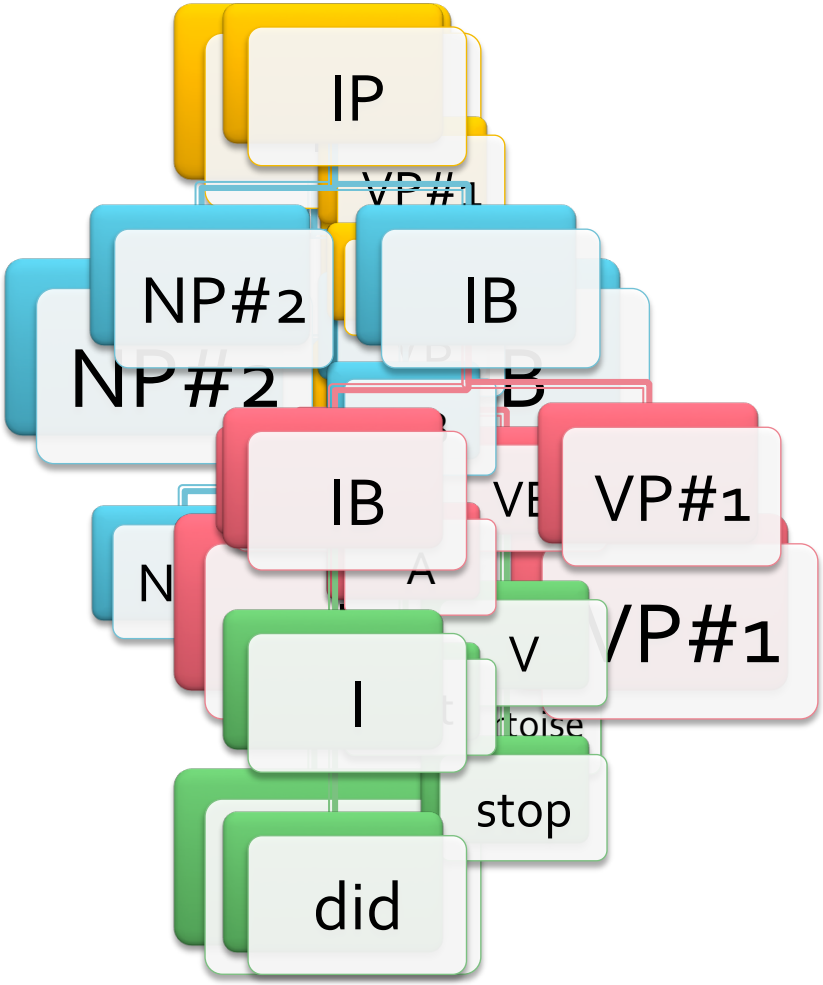
Building a grammar (I)

building a tree bank



Building a grammar (II)

extracting the rules from the tree bank



(NP):=(NP)(“and”)(NP);
(NP):=(DP)(NB);
(NB):=(N);
(DP):=(DB);
(DB):=(D);
(IP):=(NP)(IB);
(IB):=(VP);
(VP):=(VB);
(VB):=(V);
(IB):=(IB)(VP);
(IB):=(I);
(VB):=(AP)(VB);
(AP):=(AB);
(AB):=(A);

Building a grammar (III)

inverting the rules

(NP):=(NP)(“and”)(NP);
(NP):=(DP)(NB);
(NB):=(N);
(DP):=(DB);
(DB):=(D);
(IP):=(NP)(IB);
(IB):=(VP);
(VP):=(VB);
(VB):=(V);
(IB):=(IB)(VP);
(IB):=(I);
(VB):=(AP)(VB);
(AP):=(AB);
(AB):=(A);

(NP)(“and”)(NP):=(NP);
(DP)(NB):=(NP);
(N):=(NB);
(DB):=(DP);
(D):=(DB);
(NP)(IB):=(IP);
(VP):=(IB);
(VB):=(VP);
(V):=(VB);
(IB)(VP):=(IB);
(I):=(IB);
(AP)(VB):=(VB);
(AB):=(AP);
(A):=(AB);

Building a grammar (IV)

ordering the rules (longest first)

(NP)(“and”)(NP):=(NP);
(DP)(NB):=(NP);
(N):=(NB);
(DB):=(DP);
(D):=(DB);
(NP)(IB):=(IP);
(VP):=(IB);
(VB):=(VP);
(V):=(VB);
(IB)(VP):=(IB);
(I):=(IB);
(AP)(VB):=(VB);
(AB):=(AP);
(A):=(AB);

(NP)(“and”)(NP):=(NP);

(AP)(VB):=(VB);
(DP)(NB):=(NP);
(IB)(VP):=(IB);
(NP)(IB):=(IP);

(AB):=(AP);
(DB):=(DP);
(VB):=(VP);
(VP):=(IB);

(A):=(AB);
(D):=(DB);
(I):=(IB);
(V):=(VB);
(N):=(NB);

Building the grammar (V)

Creating the parsing module

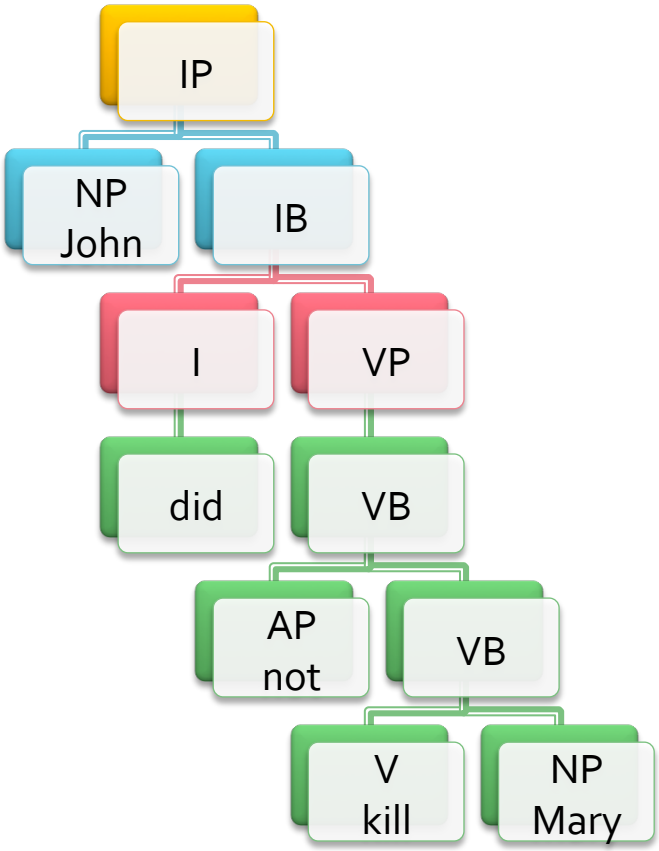
SOURCE	RULE
(NP)(“and”)(NP):=(NP);	(NP,%np1)(“and”)(NP,%np2):=(and(%np2;%np1),+XP=NP,+LEX=N);
(AP)(VB):=(VB);	(AP,%a)(VB,%v):=(VB(%v;%a,+adjt),+XB=VB, +LEX=V);
(DP)(NB):=(NP);	(DP,%d)(NB,%n):=(NP(%n;%d,+spec),+XP=NP, +LEX=N);
(IB)(VP):=(IB);	(IB,%i)(VP,%v):=(IB(%i;%v,+comp),+XB=IB, +LEX=I);
(NP)(IB):=(IP);	(NP,%n)(IB,%i):=(IP(%i;%n,+spec),+XP=IP, +LEX=I);
(AB):=(AP);	(AB,^AP):=(+XP=AP);
(DB):=(DP);	(DB,^DP):=(+XP=DP);
(VB):=(VP);	(VB,^VP):=(+XP=VP);
(VP):=(IB);	(VP,^IB):=(+XB=IB);
(A):=(AB);	(A,^AB):=(+XB=AB);
(D):=(DB);	(D,^DB):=(+XB=DB);
(I):=(IB);	(I,^IB):=(+XB=IB);
(V):=(VB);	(V,^VB):=(+XB=VB);
(N):=(NB);	(N,^NB):=(+XB=NB);

Transformation

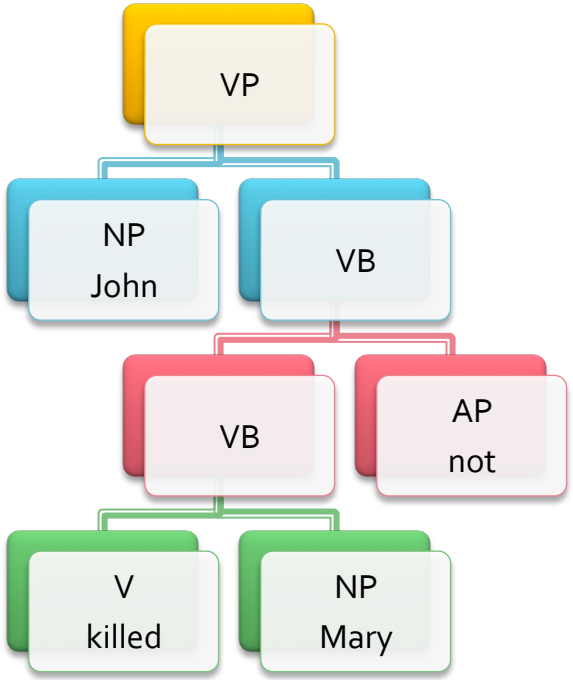
Transformation

syntactic tree (surface) > syntactic tree (deep)

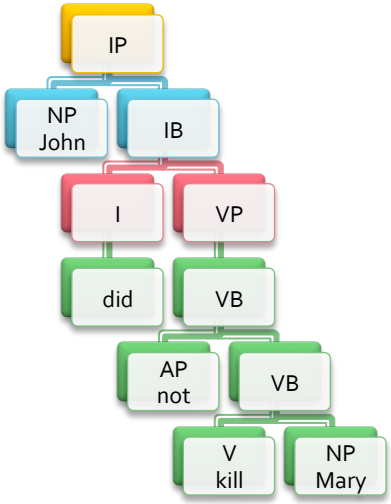
SURFACE STRUCTURE

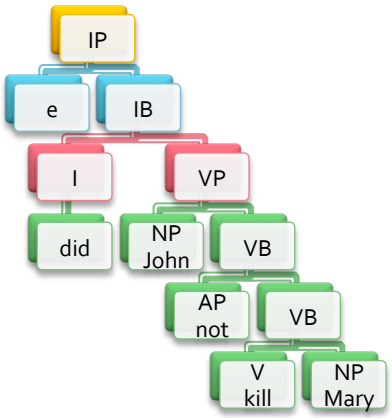


DEEP STRUCTURE



Example





IP(IB(did;VP(VB(VB(kill;Mary);AP(not));)));John)

IP(IB(;VP(;e));%s,^e):=IP(IB(;VP(;%s));%e,+e);

IP(IB(did;VP(VB(VB(kill;Mary);AP(not));John));e)

When is this necessary?

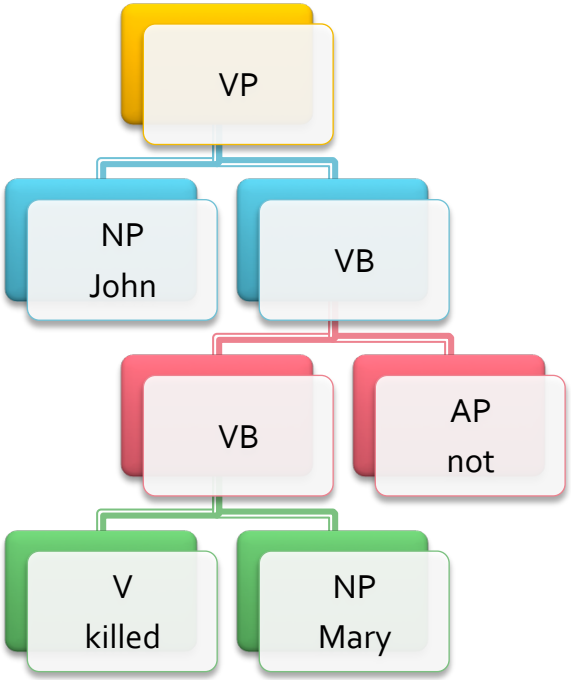
- To restore the dependency relations between constituents that are isolated in the surface structure
  - SUBJECT: from the specifier of IP to specifier of VP
  - ADJUNCTS: from the specifier of CP to adjuncts of VP
  - COORDINATION: filling in the missing branches

Dearborisation

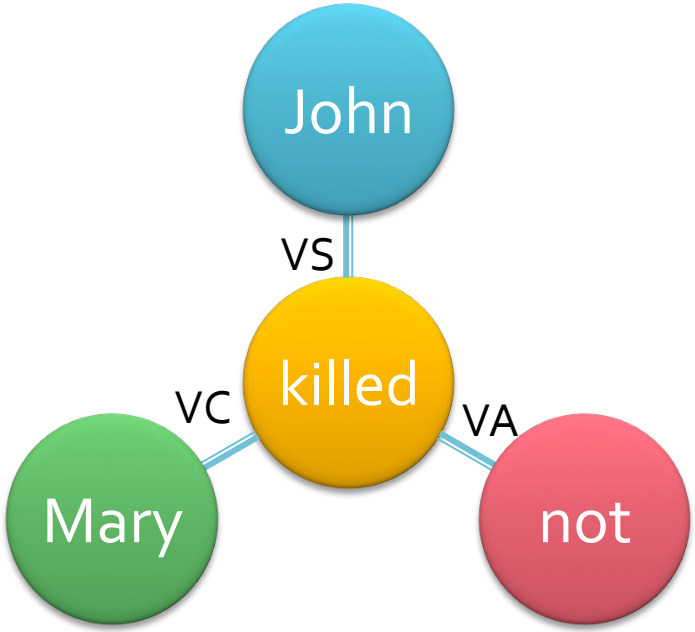
Dearborisation

syntactic tree > syntactic network

TREE STRUCTURE

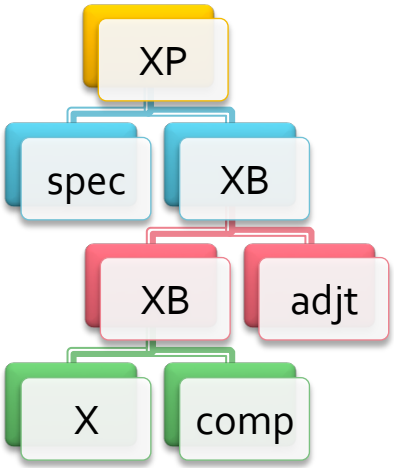


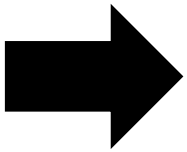
NETWORK STRUCTURE

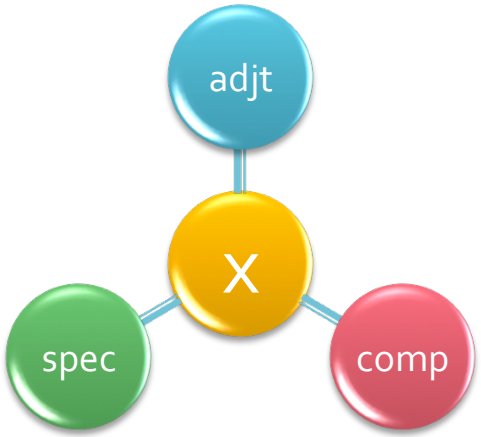


Dearborisation

tree > network





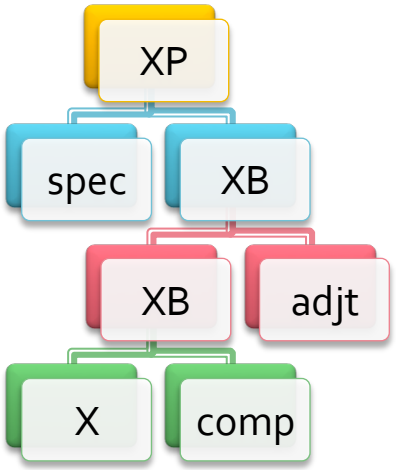


XP(XB(XB(X;comp);adjt);spec)

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

How is it done?

- Dearborisation (conversion of tree structures into head-driven structures)

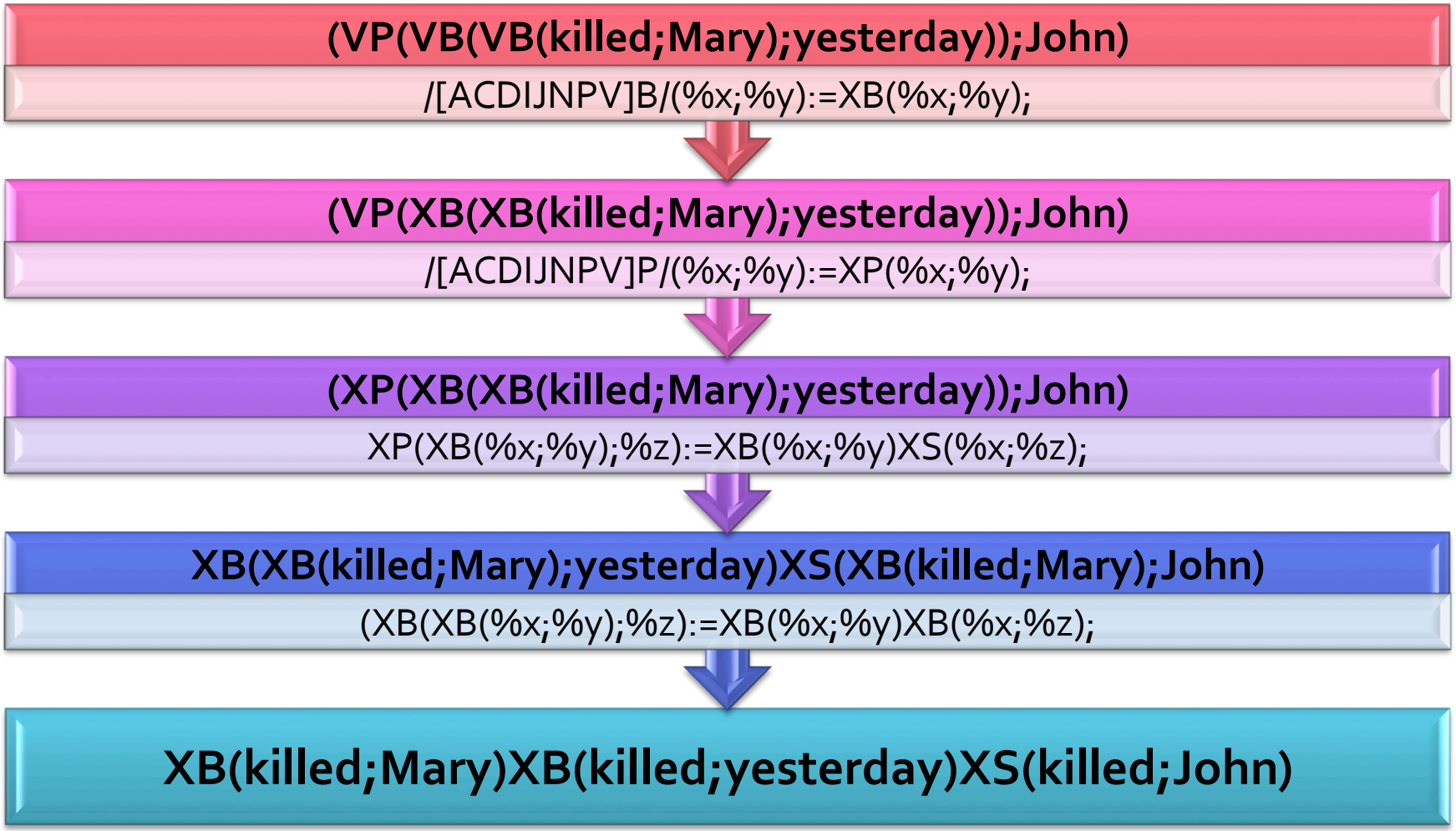


=

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

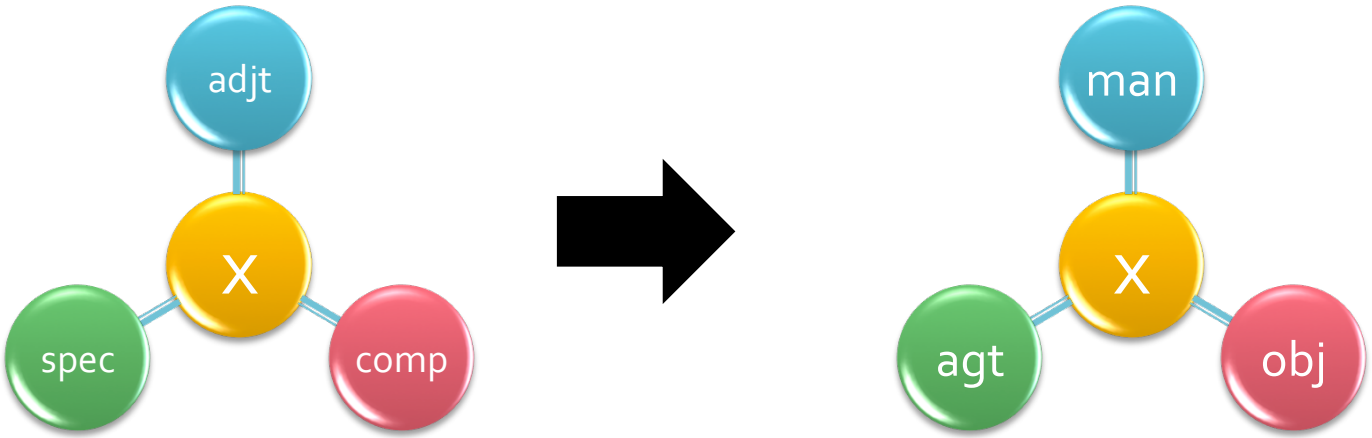
- XB(XB(%x;%y);%z):=XB(%x;%y)XB(%x;%z);
- XP(XB(%x;%y);%z):=XB(%x;%y)XS(%x;%z);

Example

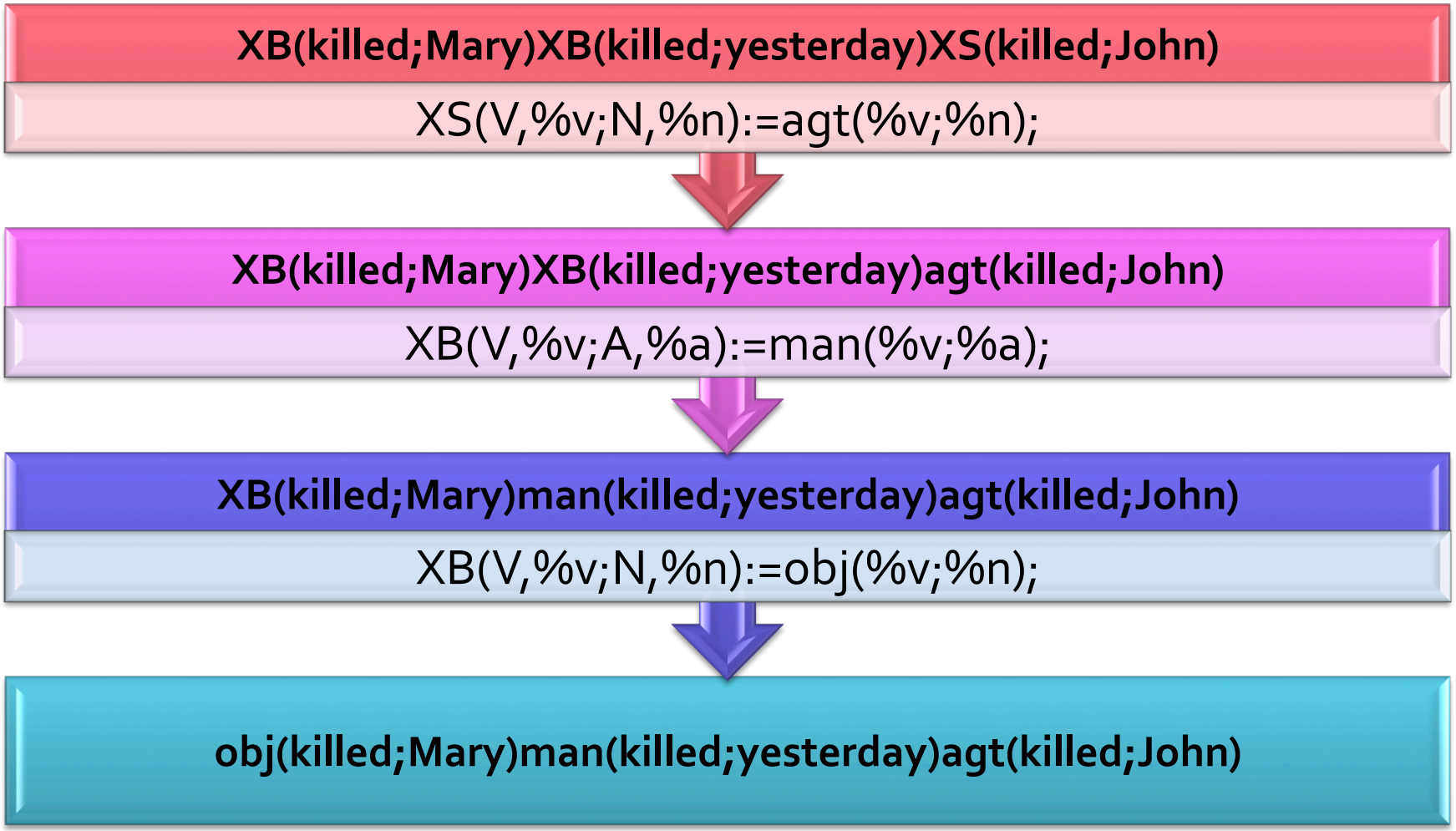

(VP(VB(VB(killed;Mary);yesterday));John)
/[ACDIJNPV]B/(%x;%y):=XB(%x;%y);
(VP(XB(XB(killed;Mary);yesterday));John)
/[ACDIJNPV]P/(%x;%y):=XP(%x;%y);
(XP(XB(XB(killed;Mary);yesterday));John)
XP(XB(%x;%y);%z):=XB(%x;%y)XS(%x;%z);
XB(XB(killed;Mary);yesterday)XS(XB(killed;Mary);John)
(XB(XB(%x;%y);%z):=XB(%x;%y)XB(%x;%z);
XB(killed;Mary)XB(killed;yesterday)XS(killed;John)

Interpretation

Interpretation



Example


XB(killed;Mary)XB(killed;yesterday)XS(killed;John)
XS(V,%v;N,%n):=agt(%v;%n);
XB(killed;Mary)XB(killed;yesterday)agt(killed;John)
XB(V,%v;A,%a):=man(%v;%a);
XB(killed;Mary)man(killed;yesterday)agt(killed;John)
XB(V,%v;N,%n):=obj(%v;%n);
obj(killed;Mary)man(killed;yesterday)agt(killed;John)

Rectification

Rectification (post processing

- agt(%x,COP;%y)obj(%x;%z):=aoj(%z,+att=%x;%y);
  - eliminates the copula
- agt(%x;%y,N)obj(%x;%z,J):=agt(%x;%y)aoj(%z;%y);
  - transforms the predicate of the object into the predicate of the subject
- or(%a;and(%b1;%b2),%b):=or(%a;or(%b1;%b2),%b);
  - propagates or to its internal conjunctions

Exercises

Subcorpus

- Sentences ending in -6
  - 6
  - 16
  - 26
  - …
  - 86
  - 96

Exercise #11 (Tokenization)
30 min

- Check whether the tokenization is correct for the sentences of the subcorpus*.
- In case it is not, provide disambiguation rules that may solve the problem.

*Use only the dictionary and select the option « tokenization only » in IAN

Exercise #12 (Parsing)
60 min

- Build the x-bar trees for the sentences of the subcorpus.
- Extract the rules from the trees.
- Invert and order the rules.
- Create the parsing module.
- Implement the parsing module in IAN.
- Test the results*

*In order to test the results, do not use the default grammar. Use only the normalization grammar and your language-specific rules. Use your dictionary as well.

Exercise #13 (Dearborization and Interpretation) 30 min

- Dearborize and interpret the subcorpus*.
- Analyze the results and provide the necessary changes.

*In order to dearborize and interpret the subcorpus, enable the default grammar.