Statistical Machine Translation by Parsing

final presentation

August 18, 2005

Andrea Burbank (Stanford), Marine Carpuat (HKUST), Stephen Clark (Oxford), Markus Dreyer (JHU), Pamela Fox (USC), Declan Groves (DCU), Keith Hall (JHU), Mary Hearne (DCU), Dan Melamed (NYU), Yihai Shen (HKUST), Andy Way (DCU), Ben Wellington (NYU), Dekai Wu (HKUST)

Statistical Machine Translation by Parsing

The Team

Andrea Burbank (Stanford), Marine Carpuat (HKUST),
Stephen Clark (Oxford), Markus Dreyer (JHU), Pamela Fox (USC), Declan Groves (DCU), Keith Hall (JHU), Mary Hearne (DCU),
Dan Melamed (NYU), Yihai Shen (HKUST), Andy Way (DCU),
Ben Wellington (NYU), Dekai Wu (HKUST)

The Supporting Team

Noah Smith, Josh Rosenblum, Svetlana Stenchikova, Wei Wang, Fred Jelinek, Sue Porterfield, Laura Graham, Eiwe Lingefors, Thomas Tornsey-Weir, Kristo Kirov, Victoria Fossum, Ali Argyle, Mona Diab, Nizar Habash, Chris Pike, Dan Bikel, Frank Keller, Abhishek Arun, Charles Schafer

Motivation for SMT by Parsing

- State-of-the-art SMT often produces word salad.
- Bolting trees onto FST-based (IBM-style) SMT doesn't seem to help.
- SMT is very compute-intensive (slow).
- SMT systems getting very complicated, making them hard to study and improve.

The Engineering Motivation for Syntax

- Need fewer parameters to express ordering preferences.
  - E.g.: Arabic adjectives always follow their nouns.
- Fewer parameters are easier to learn, given limited training data and/or computing resources.
- Less training data needed to reach a given level of accuracy.
- Better accuracy on fixed amount of data.
- All parameters interact during learning, so better estimates for syntactic parameters lead to better estimates for other types.

But isn't syntax too expensive?

- **Myth**: Translation models involving syntax are computationally too expensive to train.
- **Fact**: Finite-state models are *more* expensive! (more parameters)
- Of course, bolting syntax on top of a finite state model incurs the combined cost of both. (So we avoided that.)
- In machine learning with structured inference (most of NLP), better models should train *faster.*

Motivations for our team's work

- short-term
  - lower the entry barriers to the field
  - demonstrate feasibility of SMT by Parsing
- short- and long-term
  - answer fundamental scientific questions
  - educate the next generation
  - accelerate progress in MT
- long-term
  - help to reunite MT with NLP

following precedent

SMT @ WS'99	SMT @ WS'05
• EGYPT toolkit – incl. GIZA – no decoder	• GenPar toolkit – incl. "sandboxes" – no sep. decoder required
• Cairo	• MultiTreeViewer
• N. Smith & M. Jahr	• A. Burbank, P. Fox, and other educational achievements
• feasibility of SMT	• feasibility of SMT by Parsing

more precedent

SMT @ WS'99

- 1 week of data prep
- 3 weeks of software engineering
- 1 week of system integration
- 1 week of research

SMT @ WS'05

- 1 week of data prep
- 3 weeks of software engineering
- 1 week of system integration
- 1 week of research

Outline of the rest of the talk

- data preparation
- the GenPar toolkit
  - motivations
  - key algorithms
  - core system design
  - extensions
  - sandboxes: robust system integration
- feasibility of SMT by Parsing
- 2 proposals for follow-on research
- an empirical study

data preparation

1. obtain corpora (not easy)
2. tokenize
3. lemmatize
4. re-tokenize (re-lemmatize)
5. adapt text format to external taggers and parsers (Diab, Bikel)
6. tag & parse
7. partition into training/dev/test sets
8. filter out sentence pairs rejected by parser
9. induce word-to-word translation model (TM)
10. filter out sentence pairs rejected by TM
11. install improved taggers & parsers
12. repeat from step 2

The GenPar ToolKit

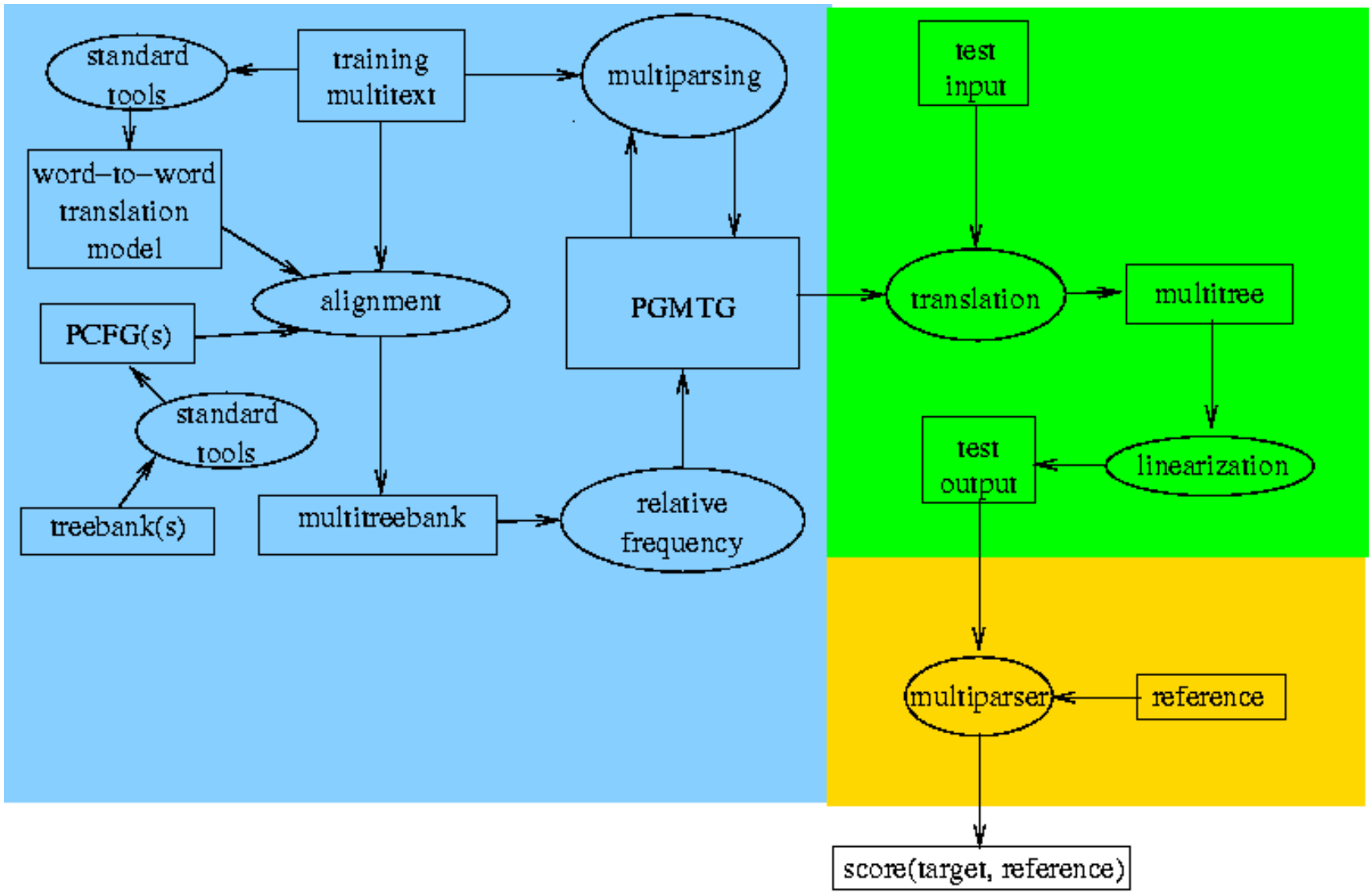
- a toolkit for generalized parsing
- integrated end-to-end system for translation by parsing, which is easy to obtain, understand, use, study, modify, extend, and improve
- relatively simple yet general architecture
- intuitive, flexible, object-oriented design
- 3 kinds of documentation: user, design, system
- dynamically configurable
- easily extendable
- freely downloadable!

http://www.clsp.jhu.edu/ws2005/groups/statistical/

The GenPar toolkit: outline

- main challenge
- key algorithm
- core system design
- design extensions
- robust system integration

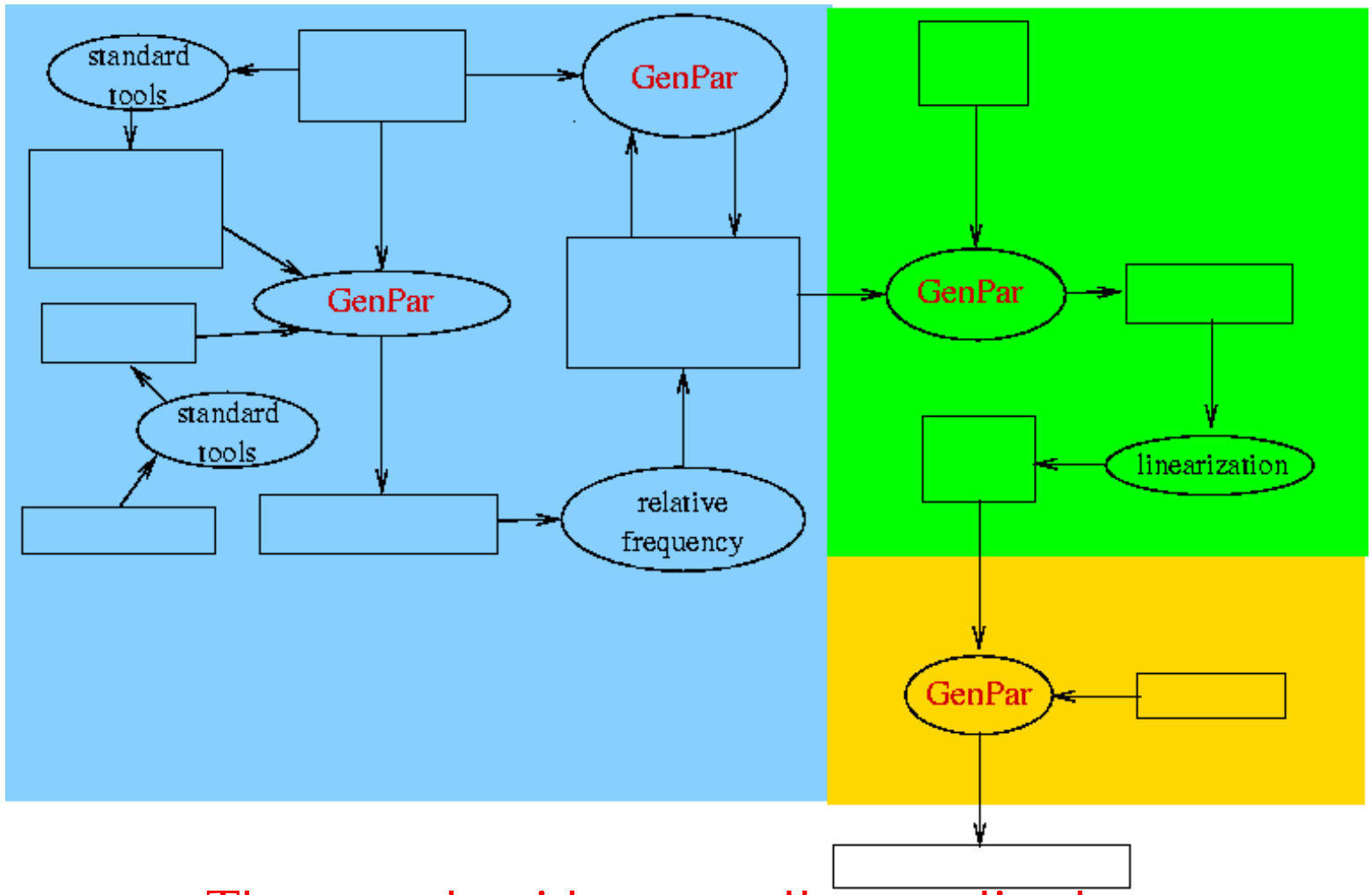
DFD for SMT by Parsing



Challenges of complex systems

- Hard to study
  - Difficult to do controlled experiments.
  - Difficult to assign credit/blame for changes in performance.
- Hard to modify/extend
  - Research prototypes are often not well-designed, with many features hard-coded.
- Hard to replicate
  - Most papers on syntax-driven SMT compare their results only to systems with no syntax.
- Hard for the community to make progress

Lowering entry barriers: Reducing system complexity

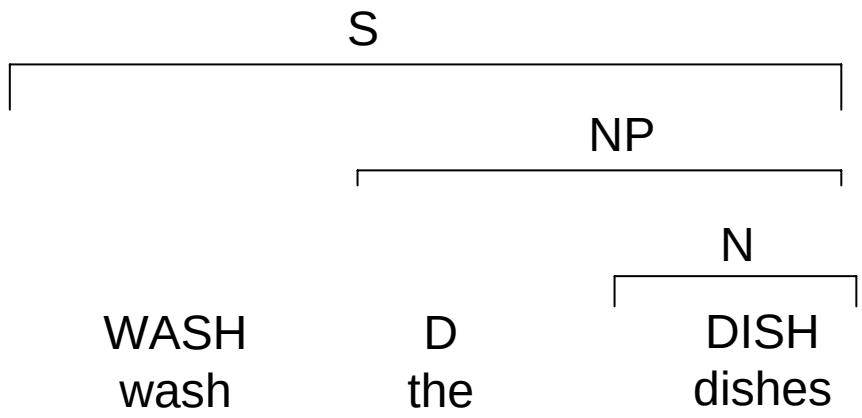


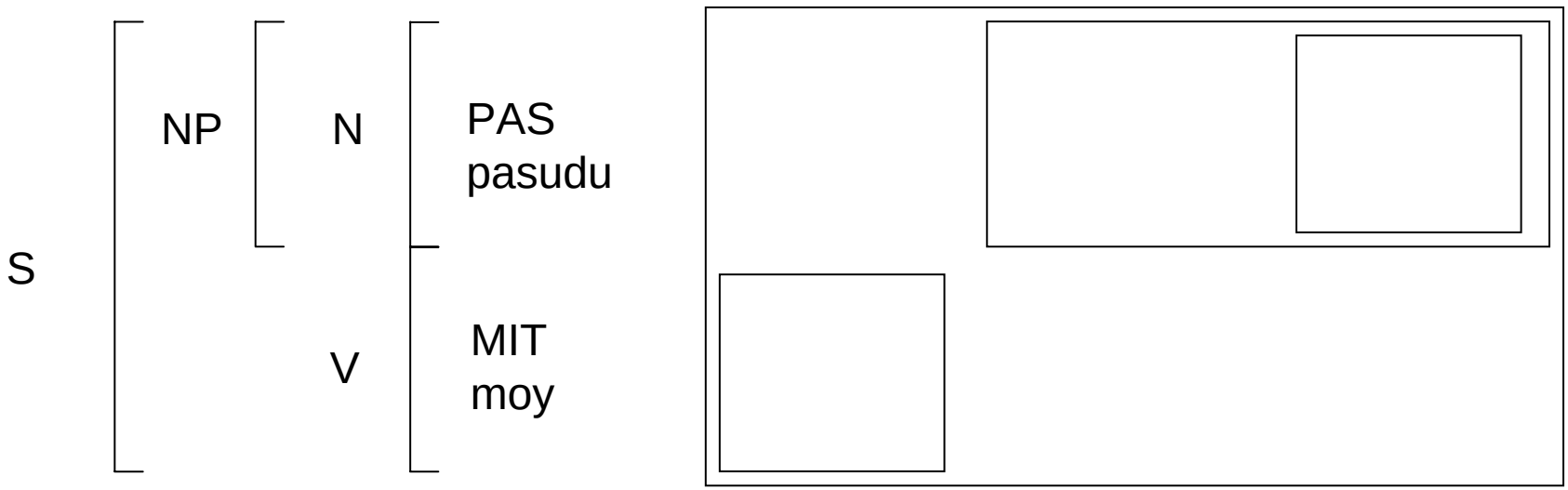
The core algorithms are all generalized parsers.

How to do everything by parsing?

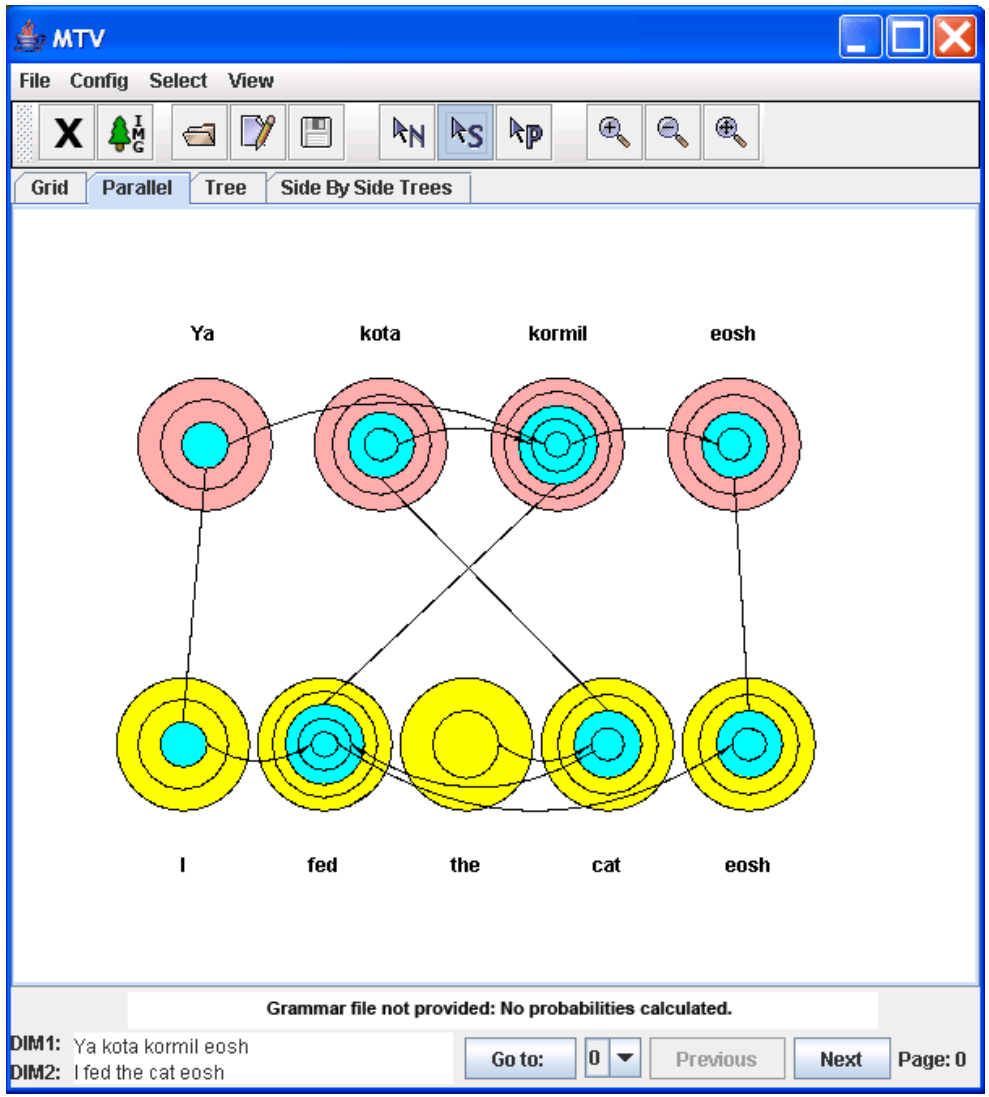
- multitrees
- multiparsing
- alignment by parsing
- translation by parsing
- later: MT evaluation by parsing

What's a multitree?

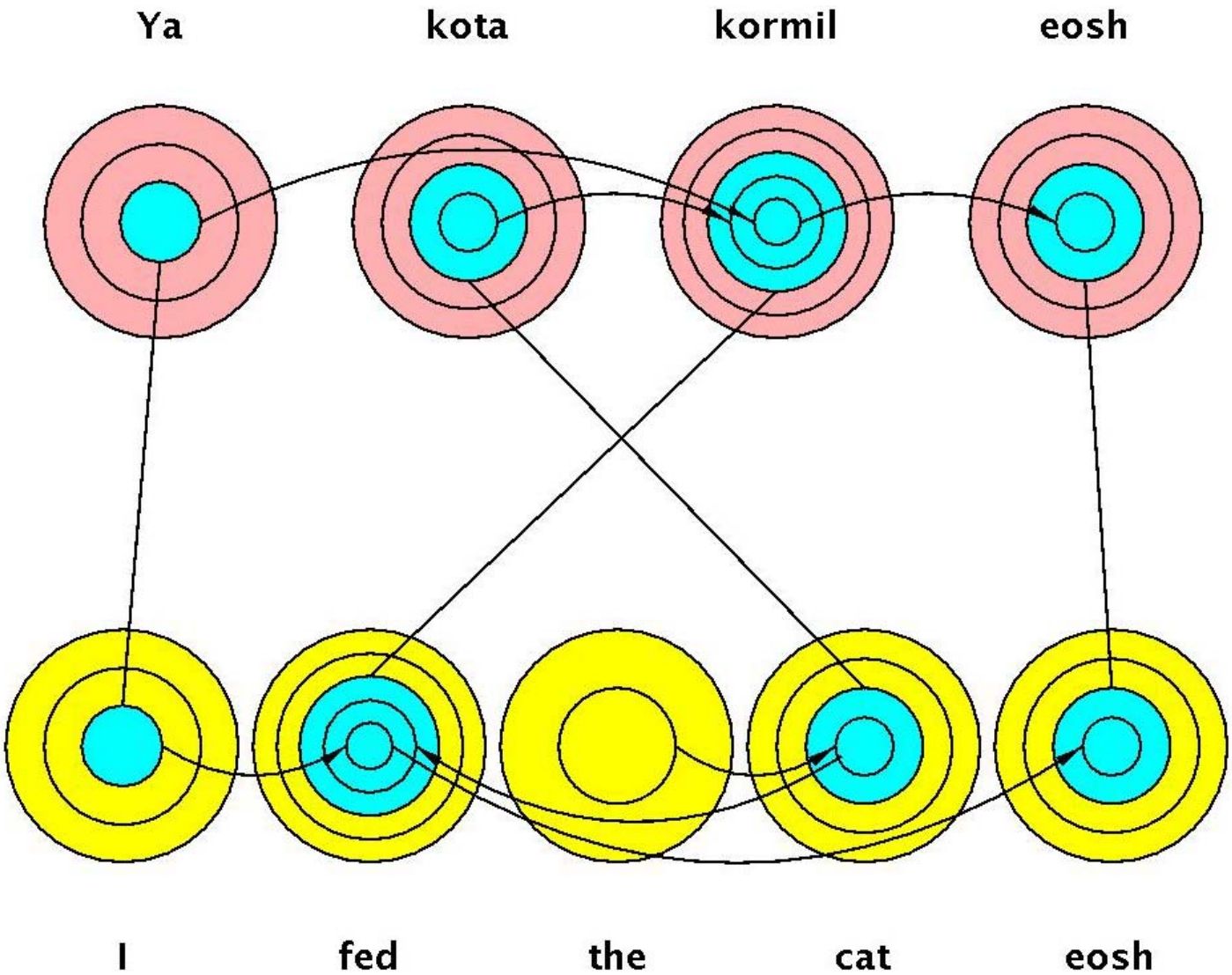




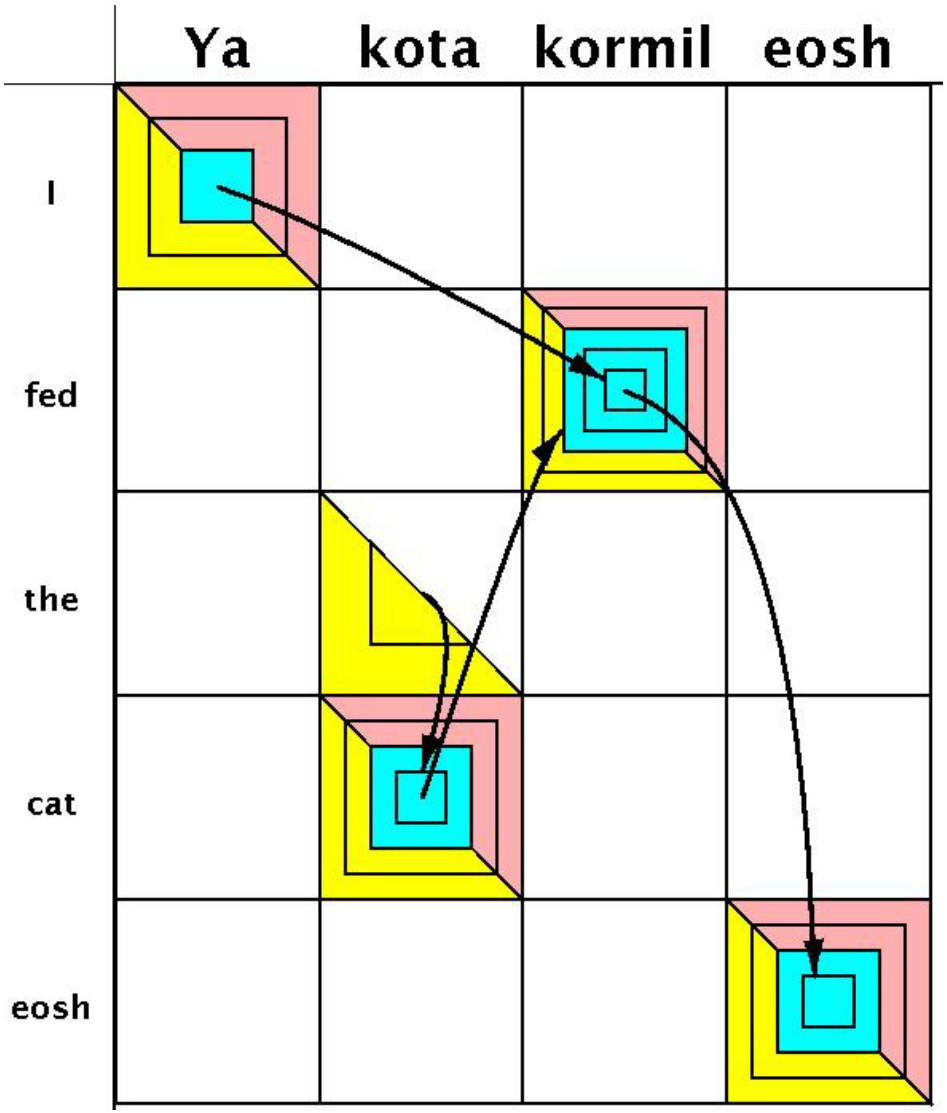
MultiTreeViewer (MTV)



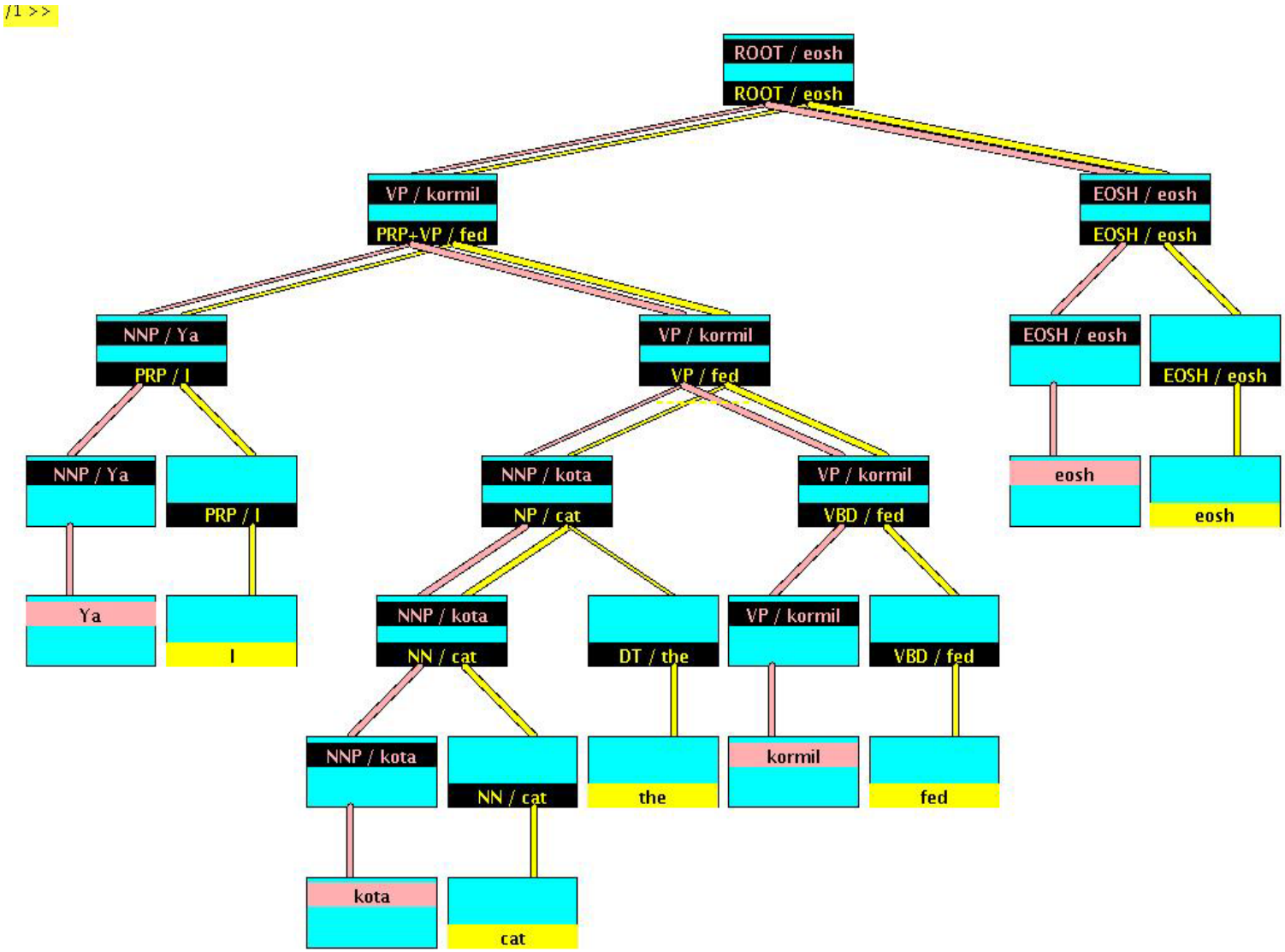
More perspectives on multitrees



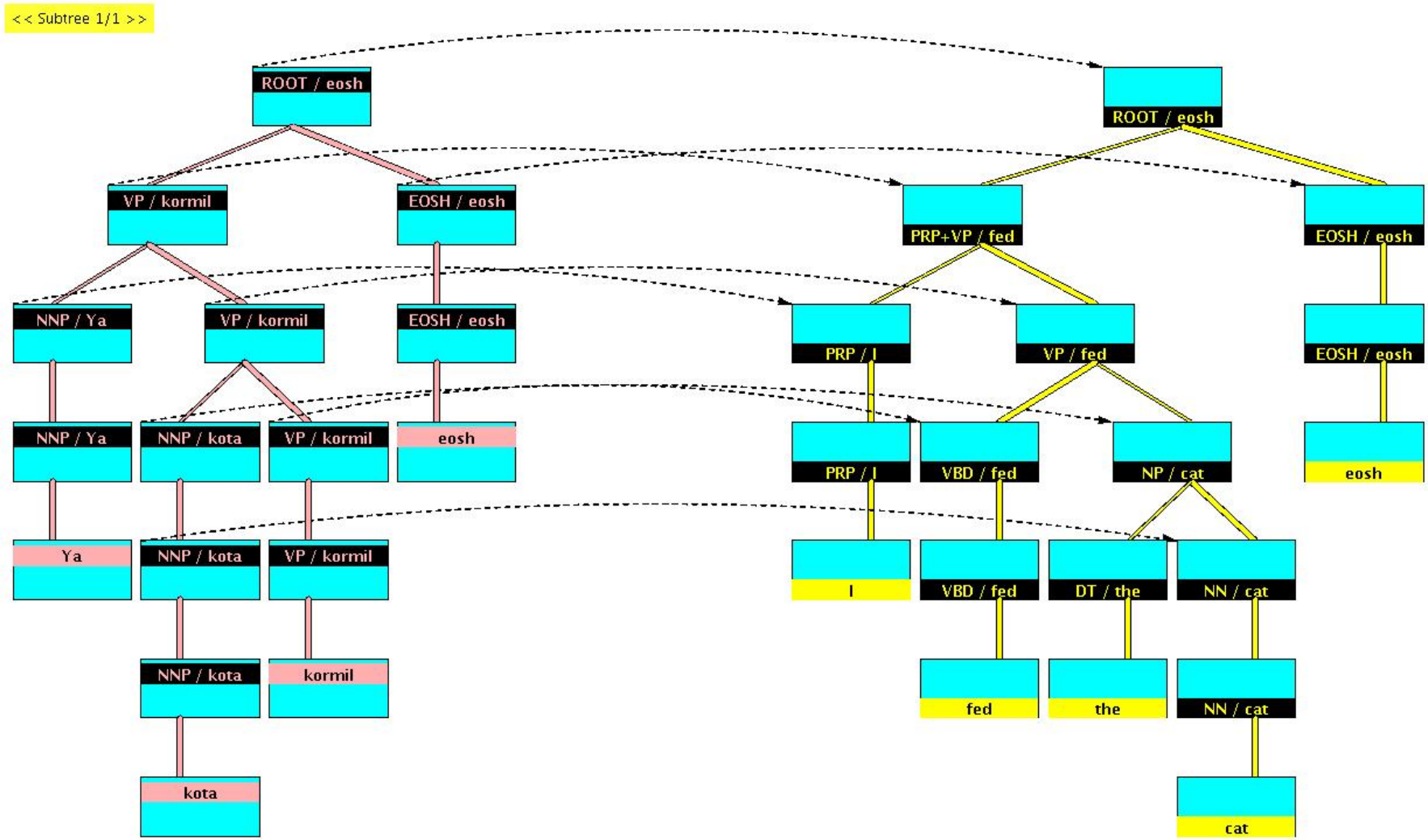
More perspectives on multitrees



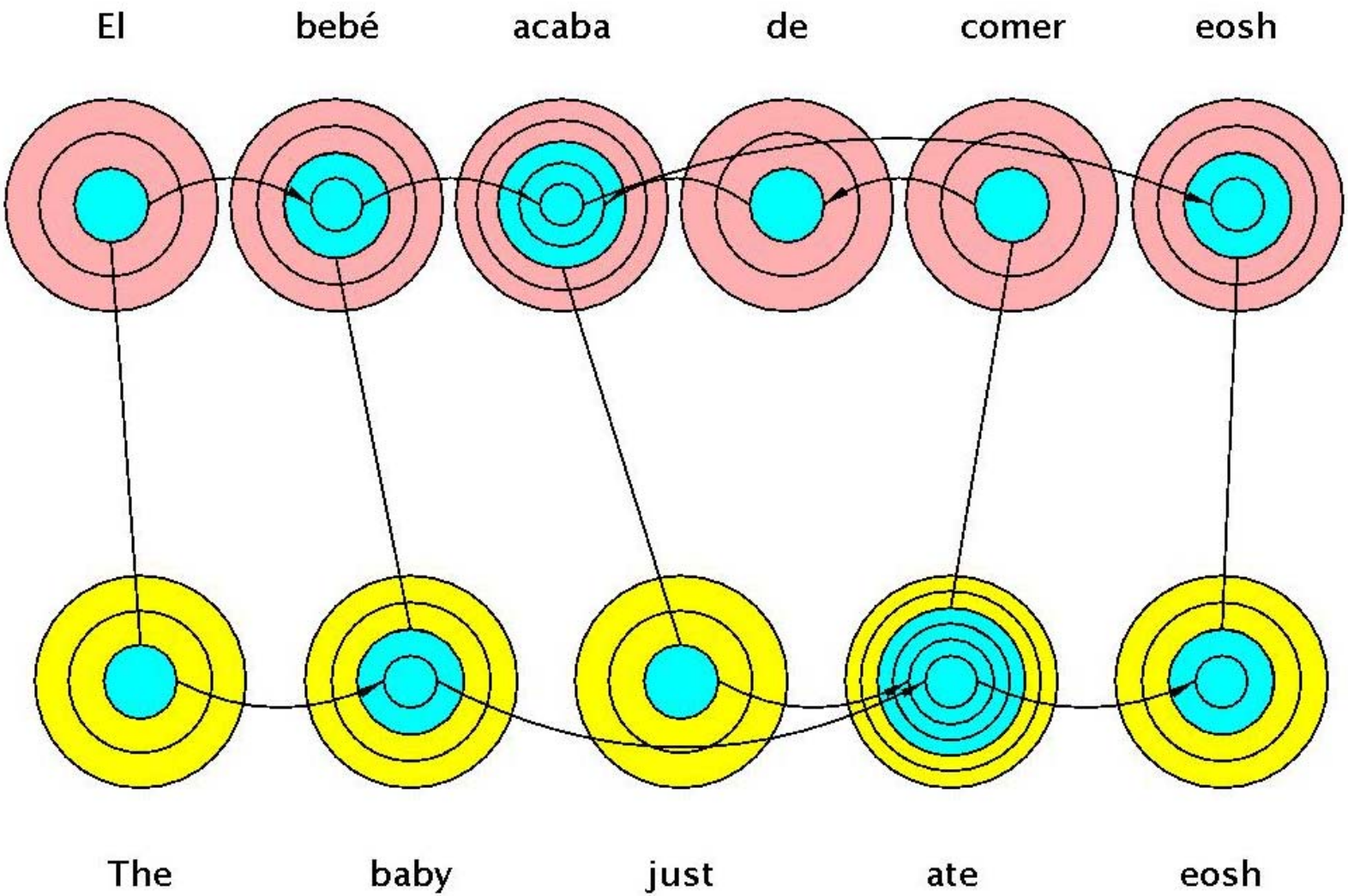
More perspectives on multitrees



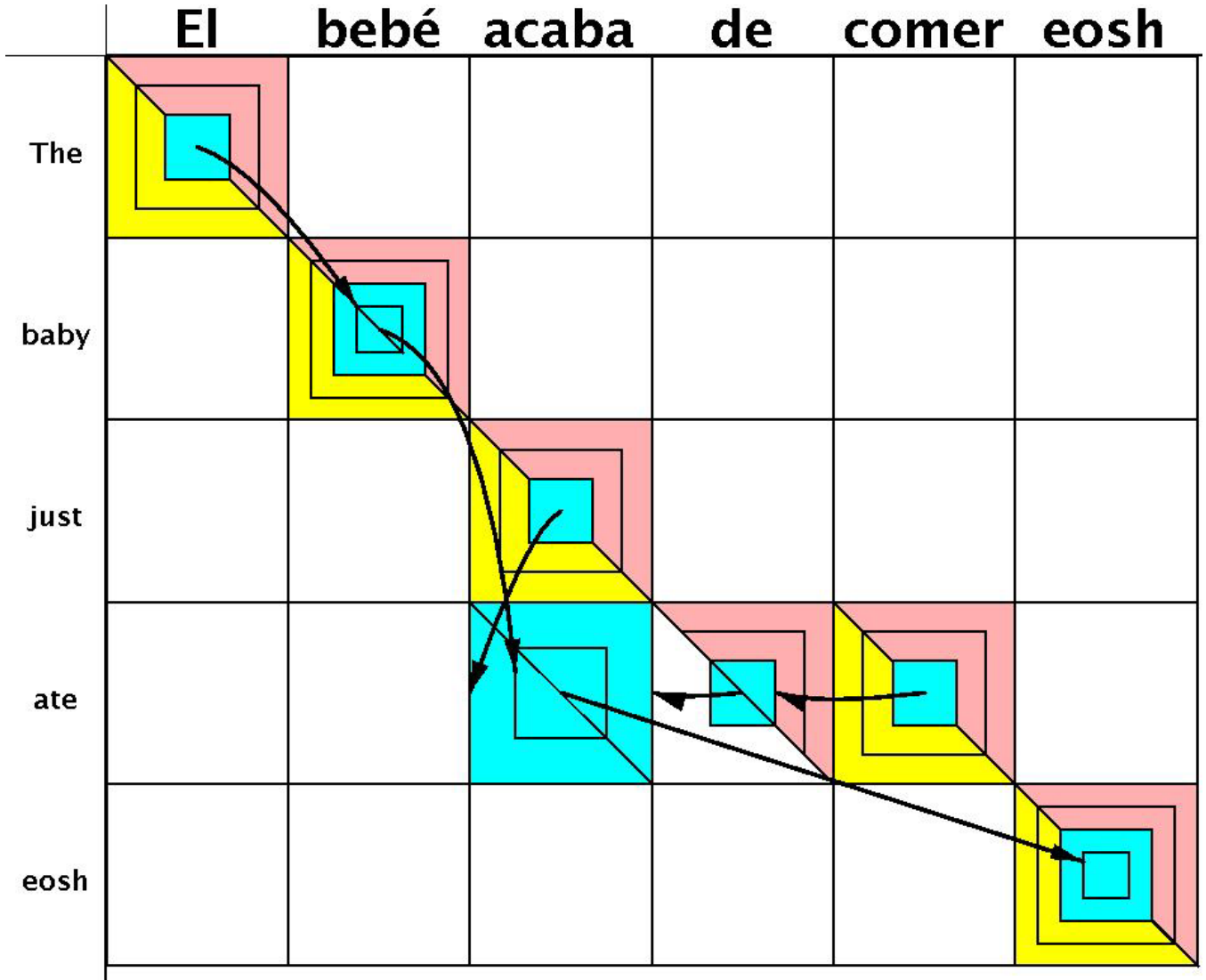
More perspectives on multitrees



head-switching multitrees

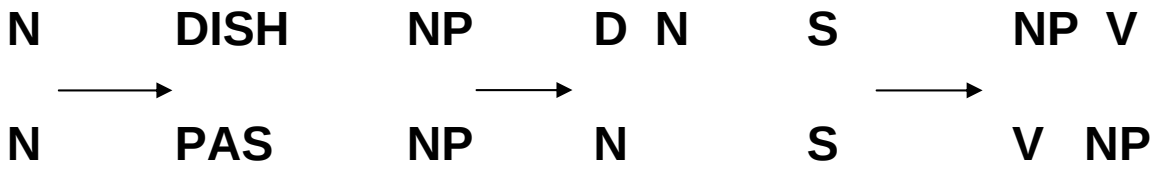


head-switching multitrees



How does a multiparser work?

Grammar:



Input:

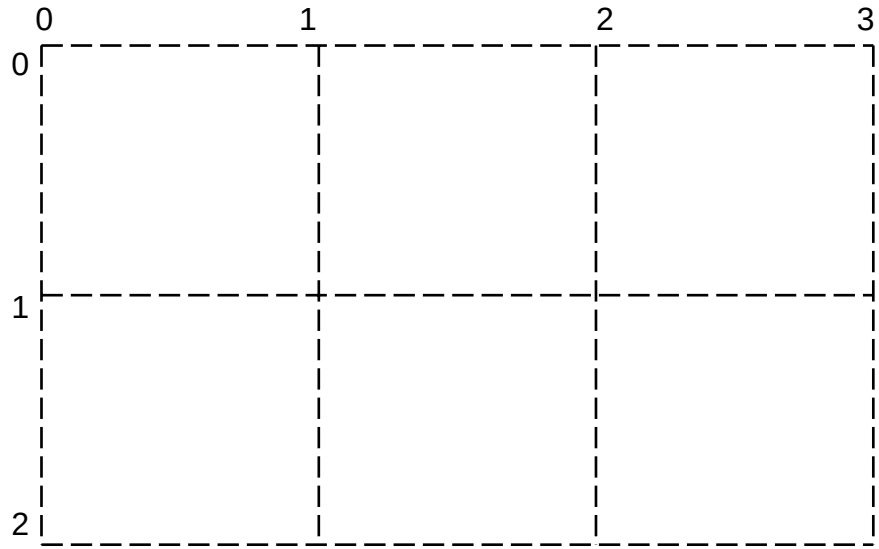
WASH wash

D the

DISH dishes

PAS pasudu

MIT moy

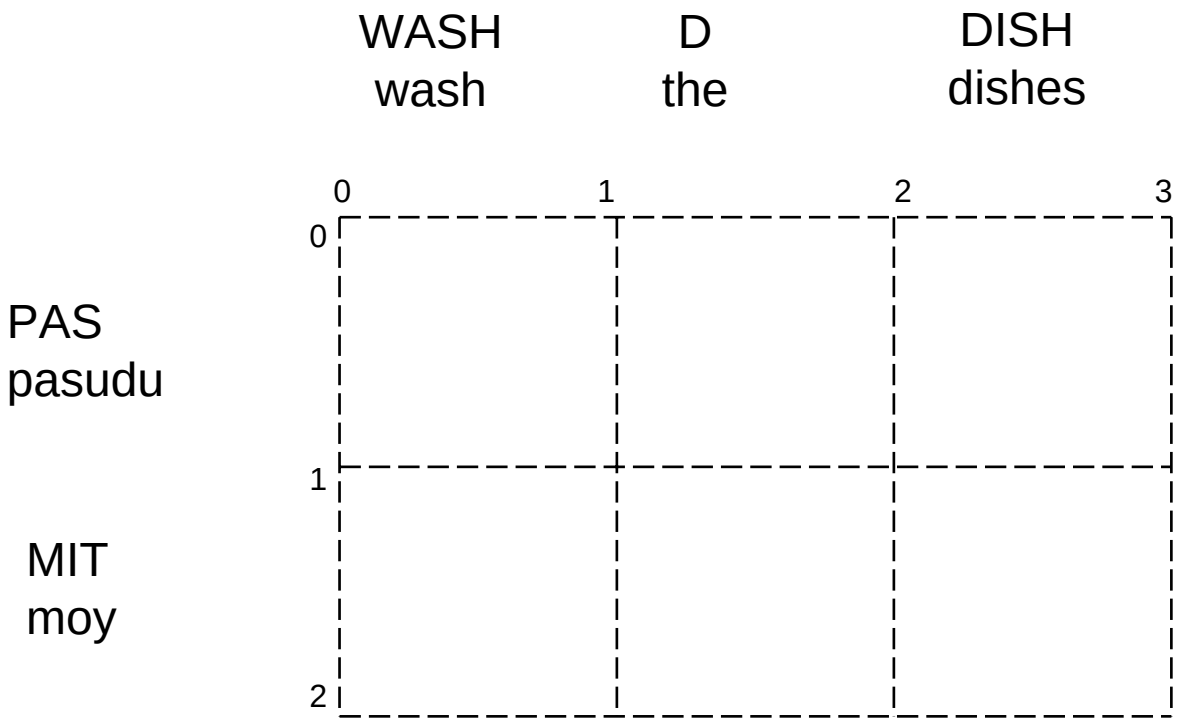


Multiparsing (1/4)

N **DISH**

→

N **PAS**

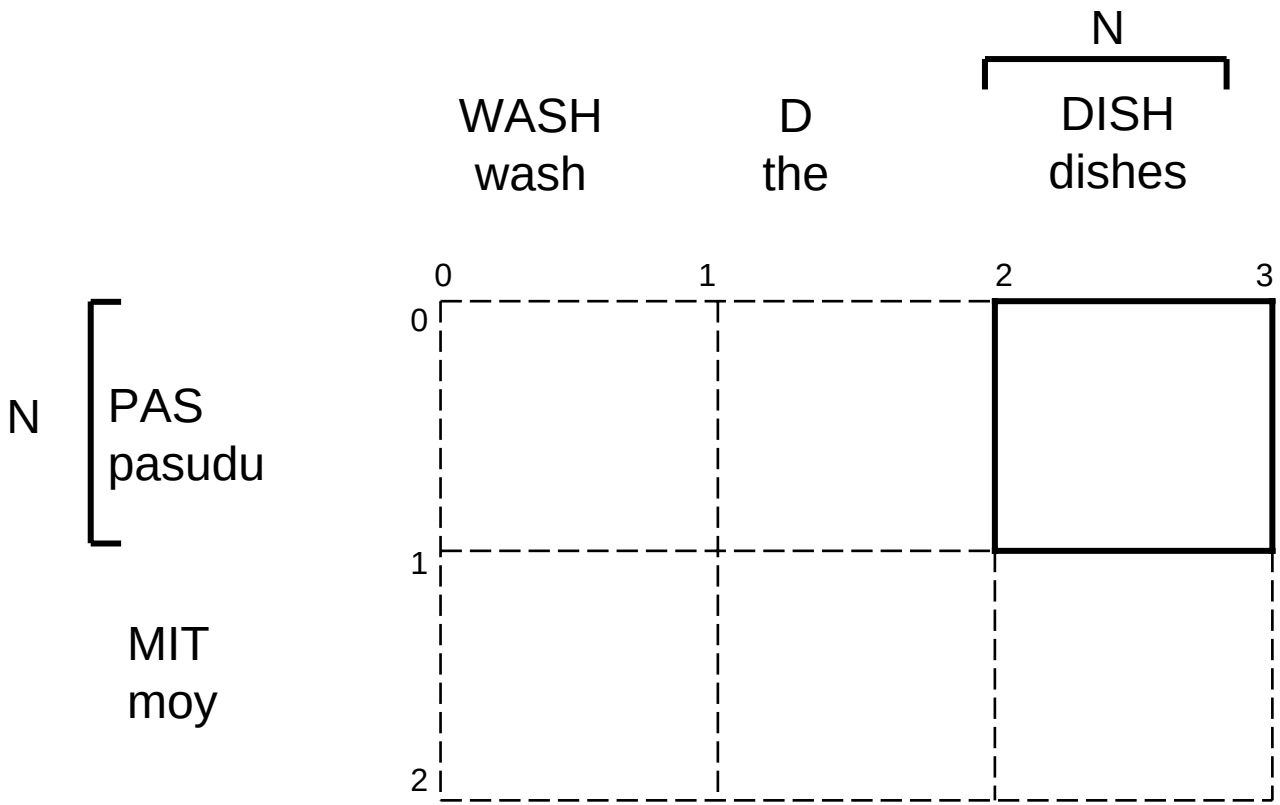


Multiparsing (1/4)

N **DISH**

→

N **PAS**

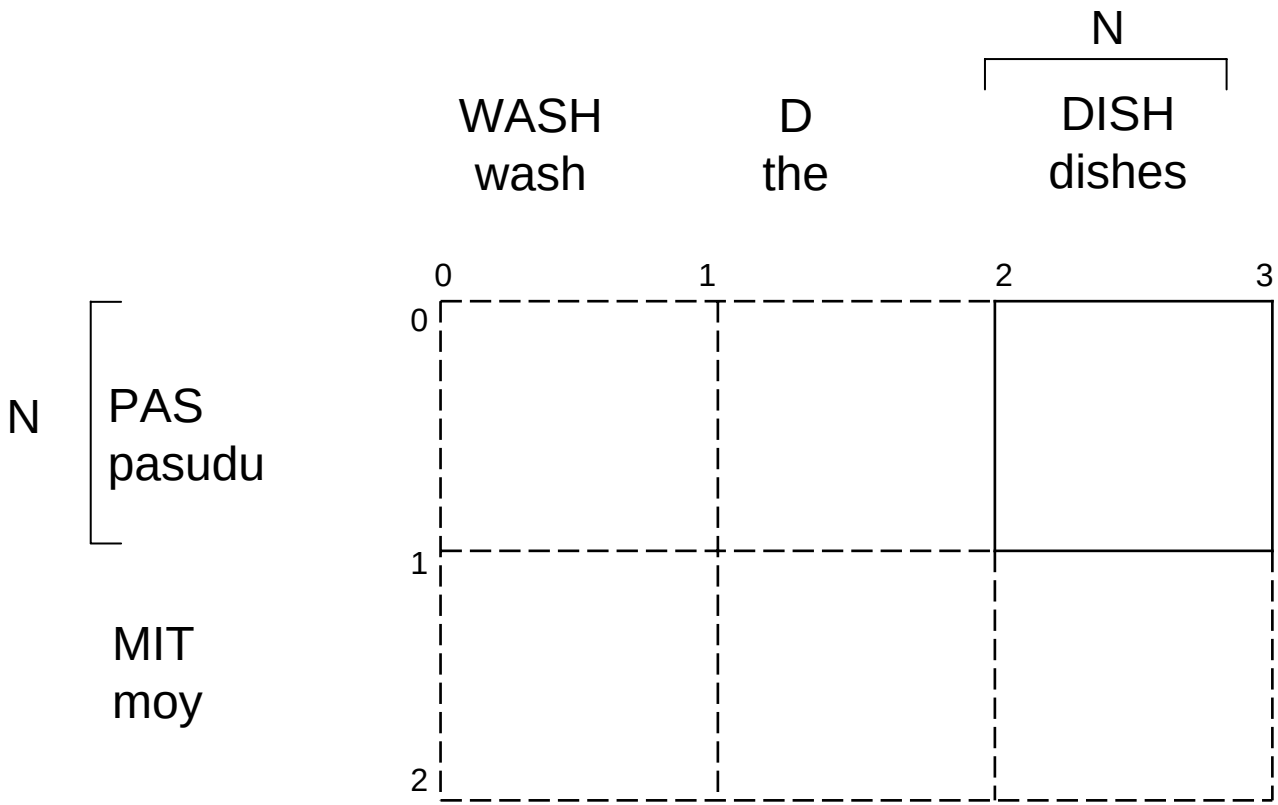


Multiparsing (2/4)

V **WASH**

→

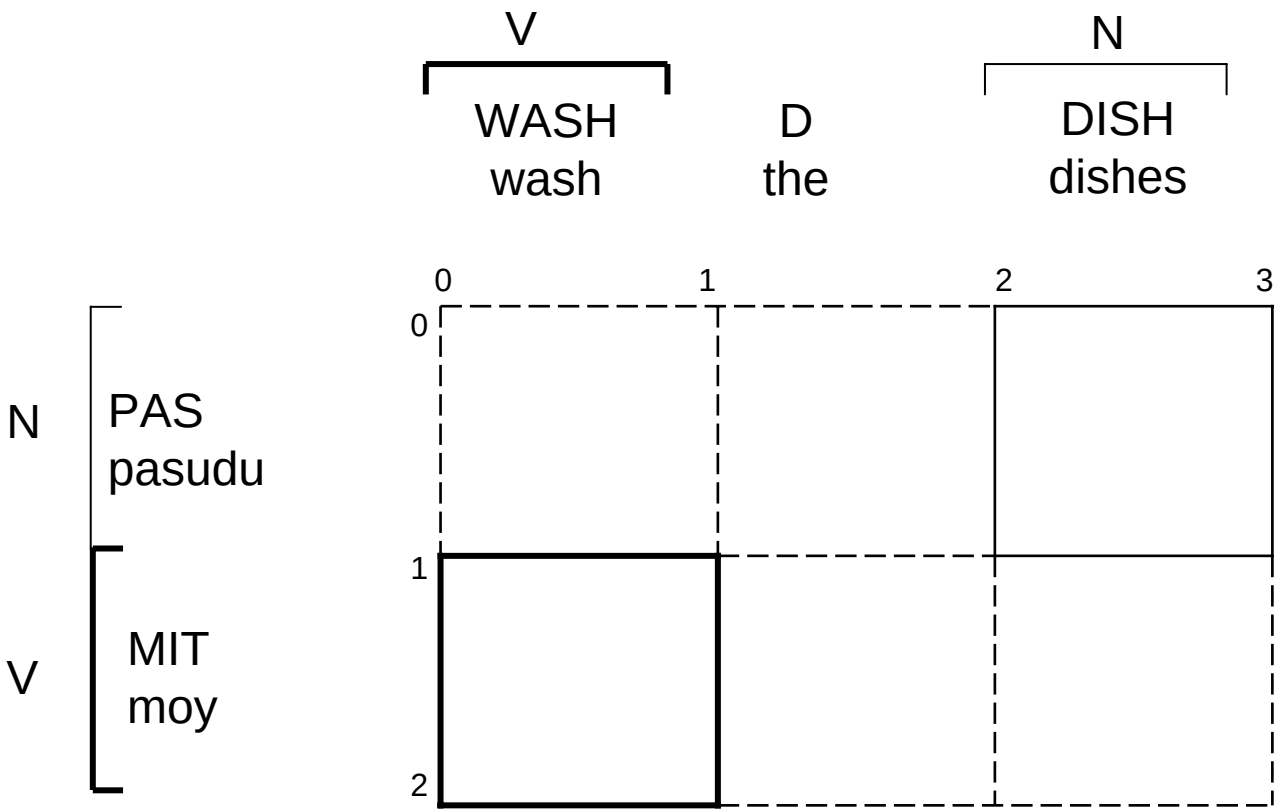
V **MIT**



Multiparsing (2/4)

V → **WASH**

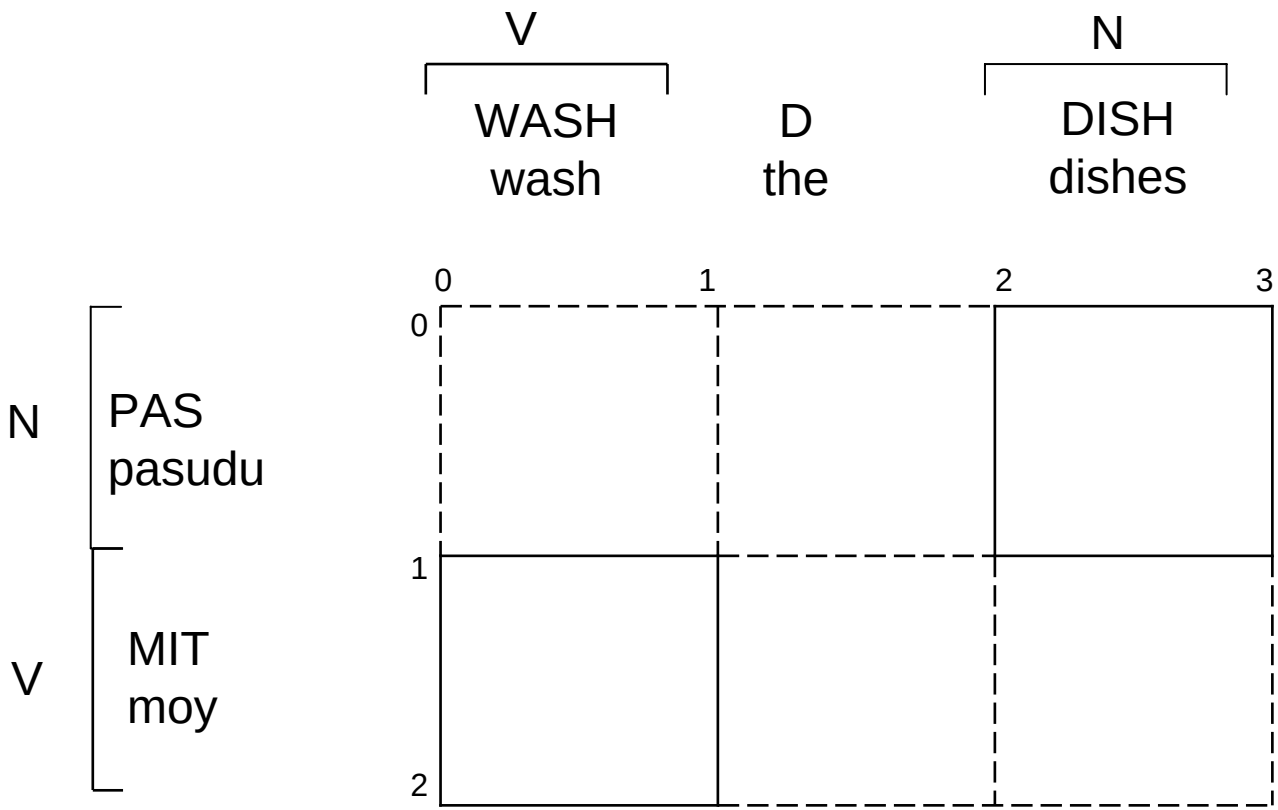
V → **MIT**



Multiparsing (3/4)

NP → **D N**

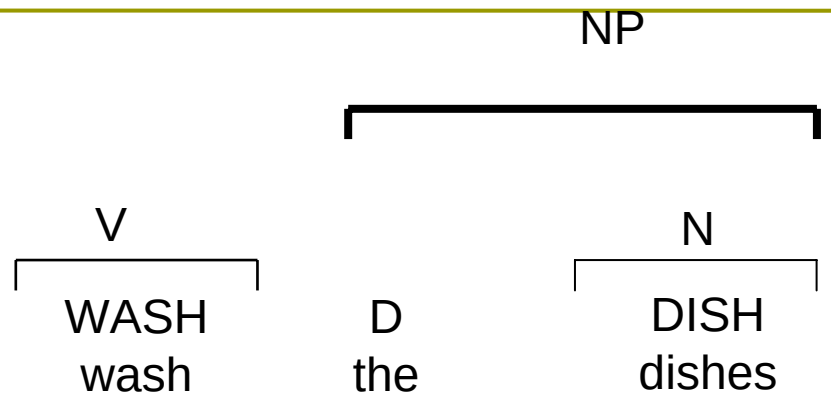
NP → **N**

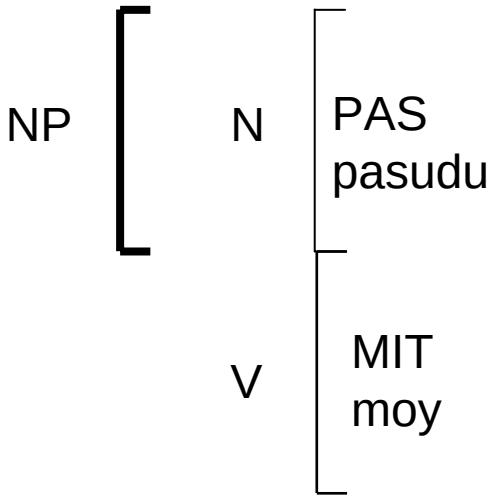


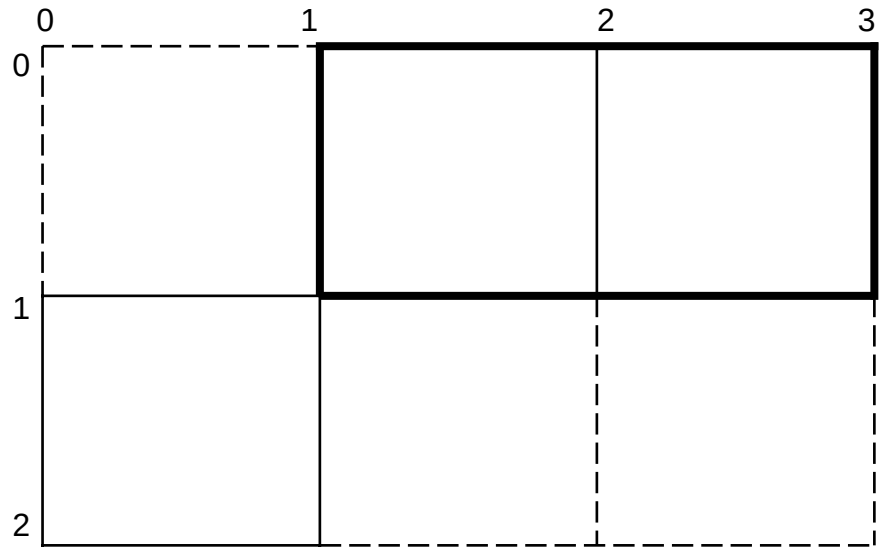
Multiparsing (3/4)

NP → **D N**

NP → **N**



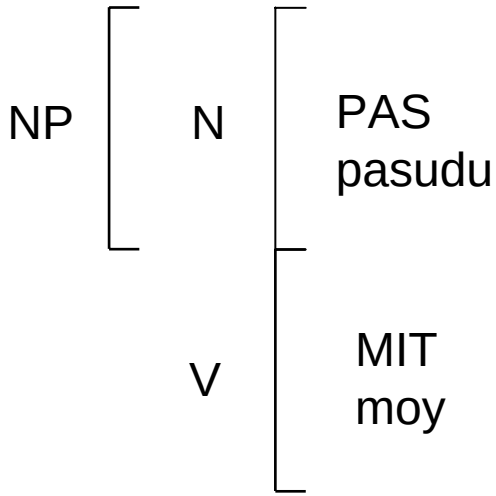


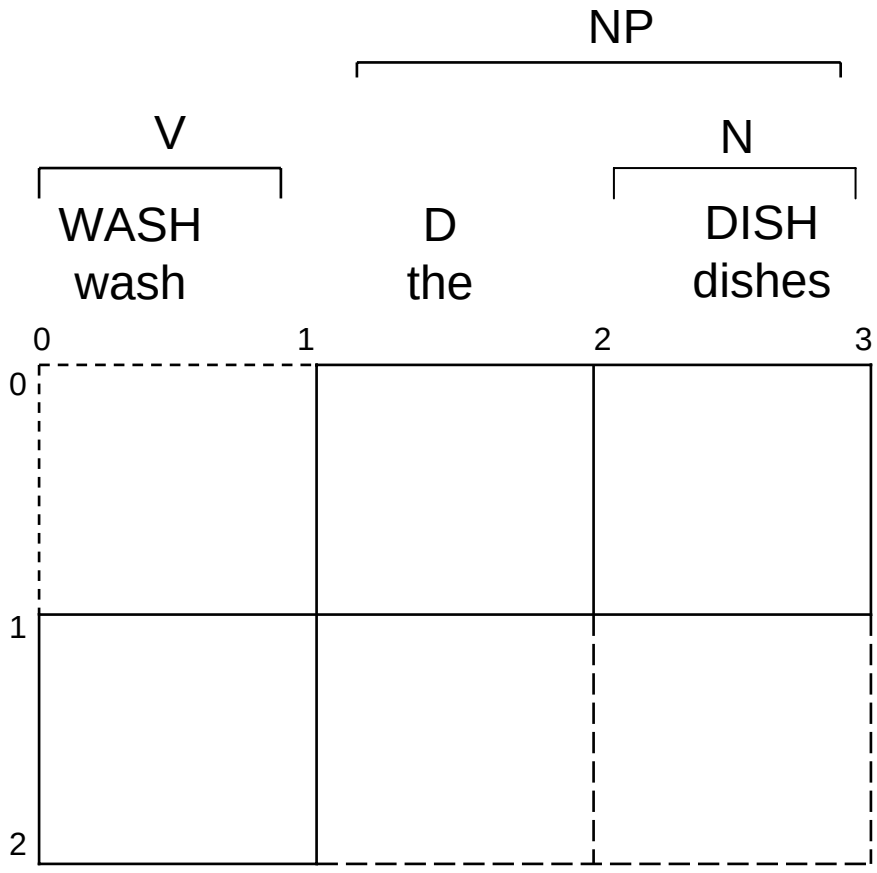


Multiparsing (4/4)

S → **NP V**

S → **V NP**

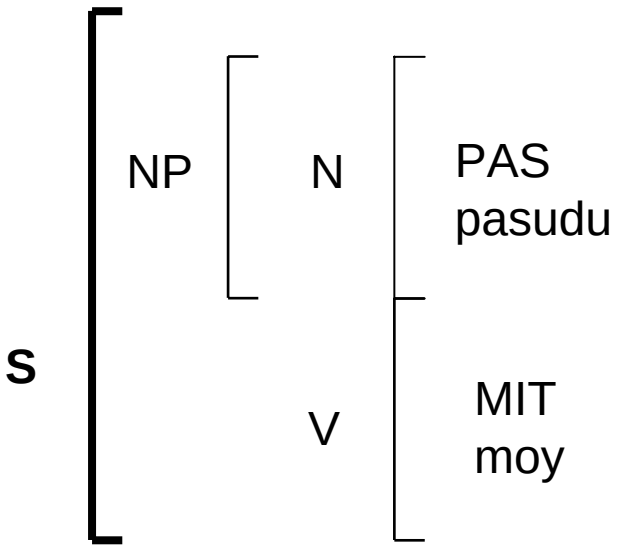


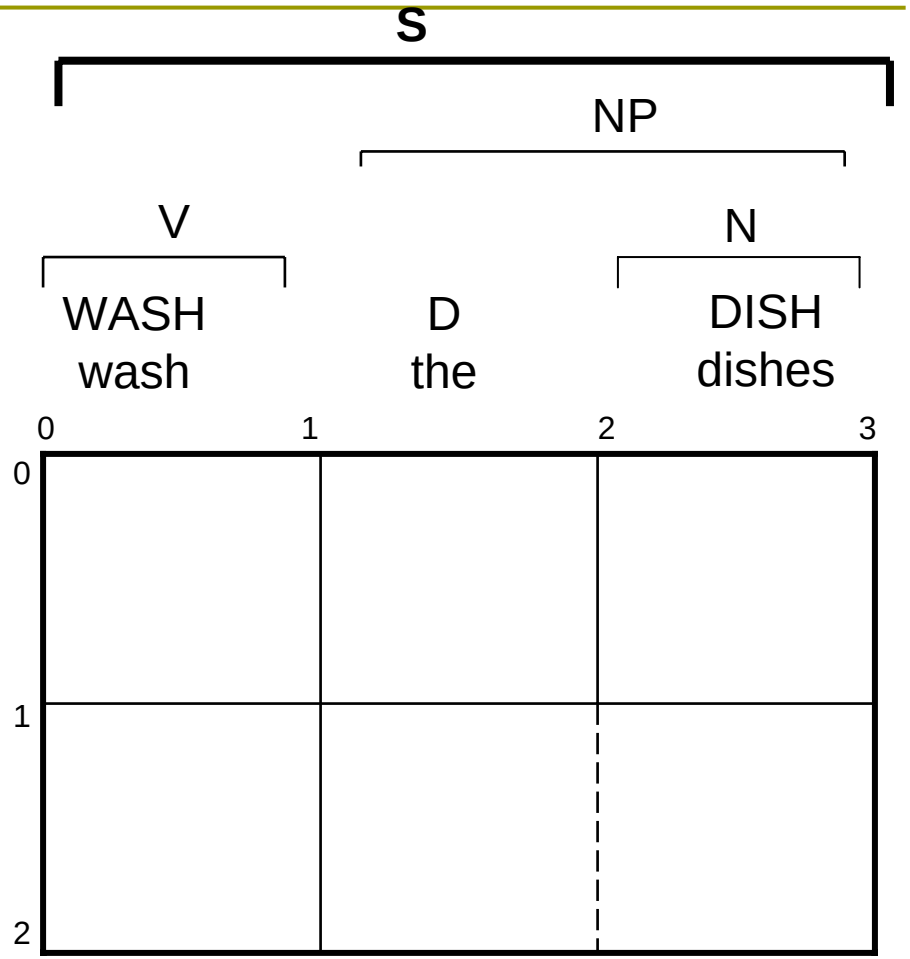


Multiparsing (4/4)

S → **NP V**

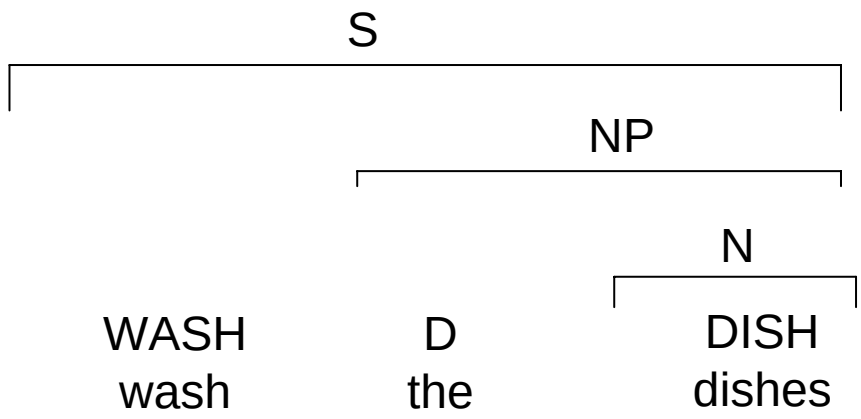
S → **V NP**

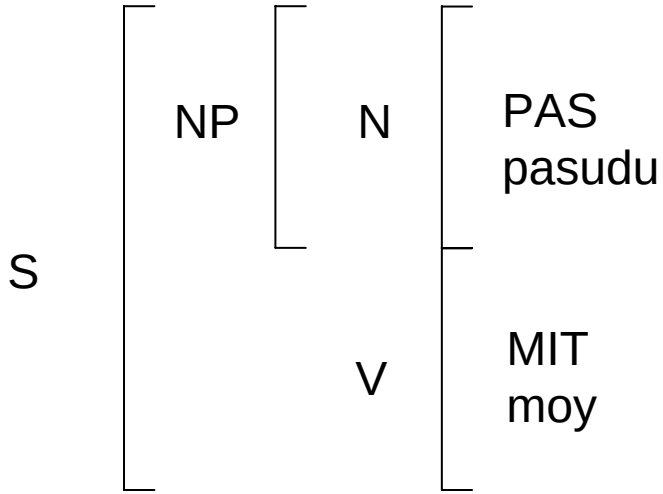


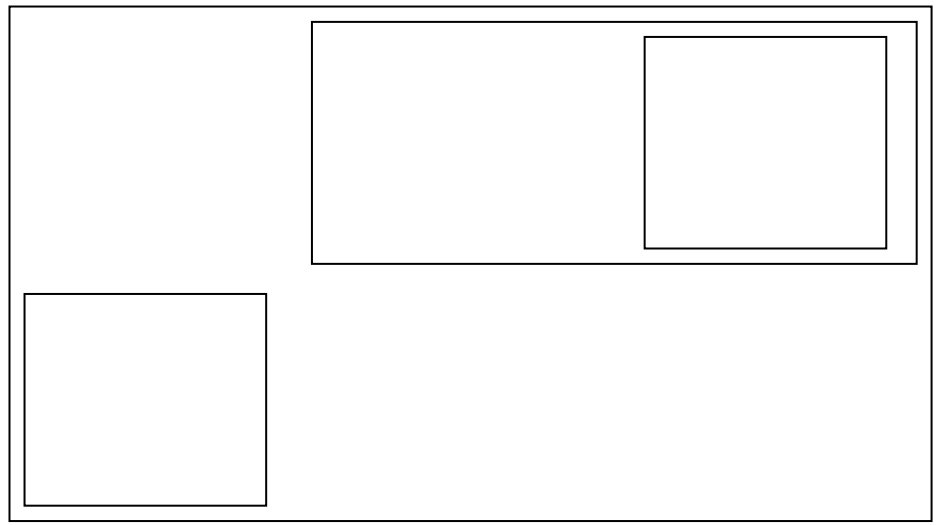


Alignment

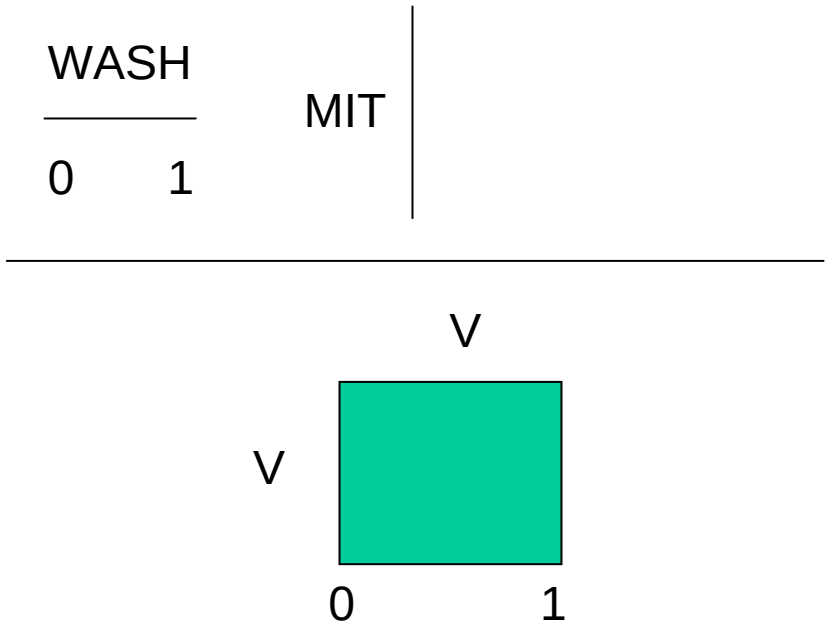
word-to-word model:
dishes = pasudu
wash = moy
the = ∅



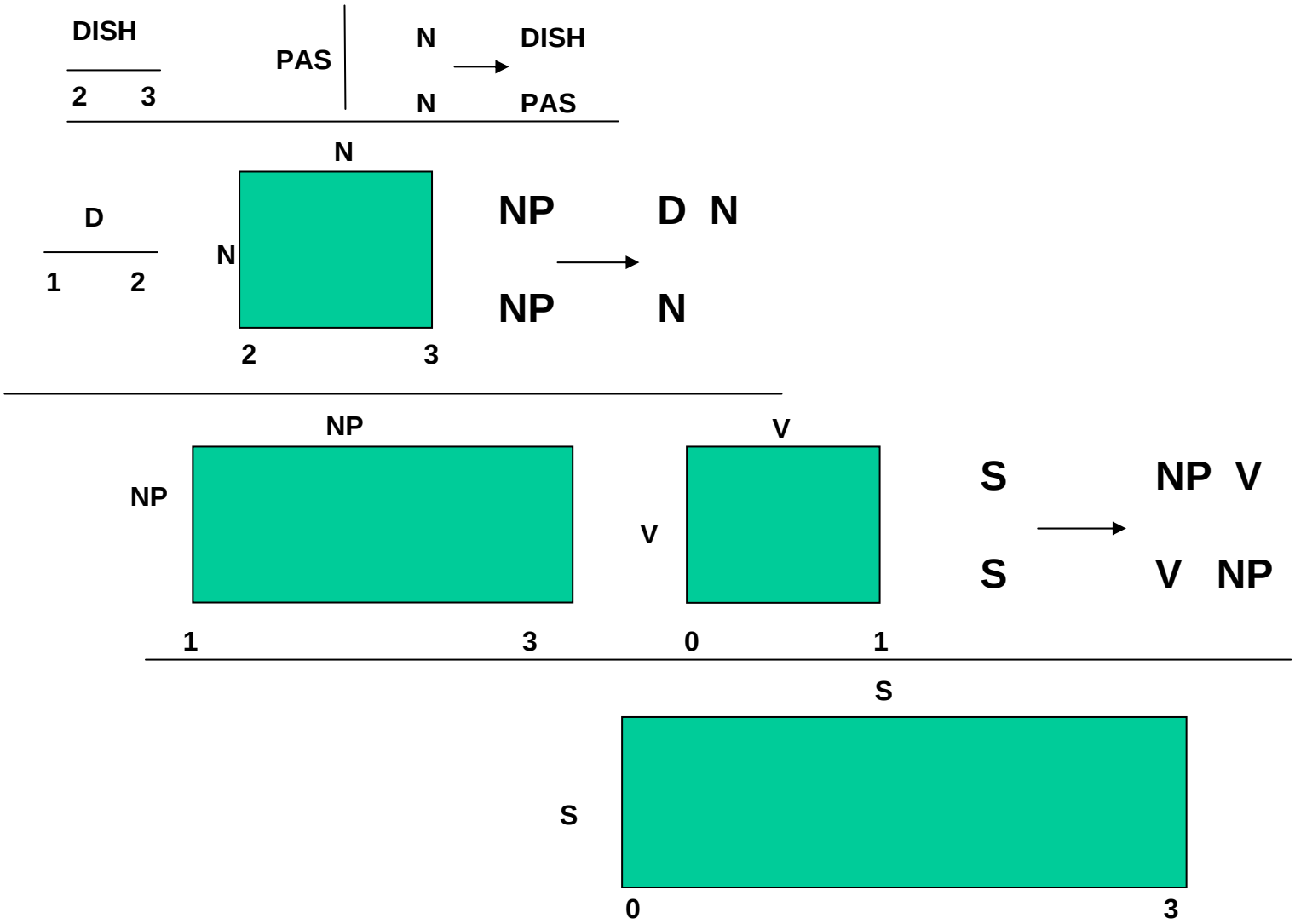




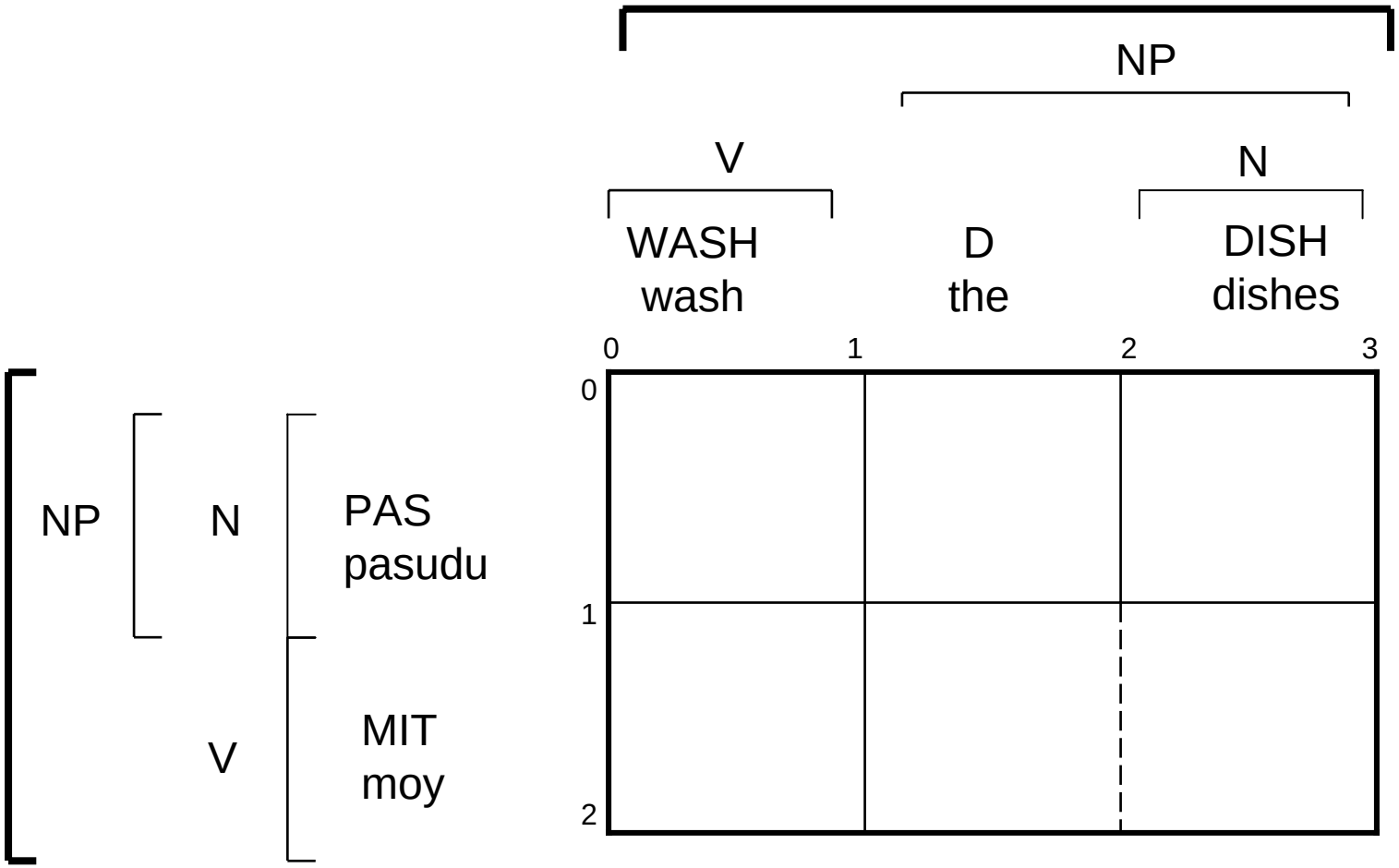
Translation (1)



Translation (2, 3, 4)



Result of translation is a multitree



Result of translation is a multitree?

- But we want a string!
- Trivial: Output string can be read off the multitree leaves by a trivial postprocess.
- Information about relative order of constituents is inferred as part of the parsing process.
- No separate “decoder” required.

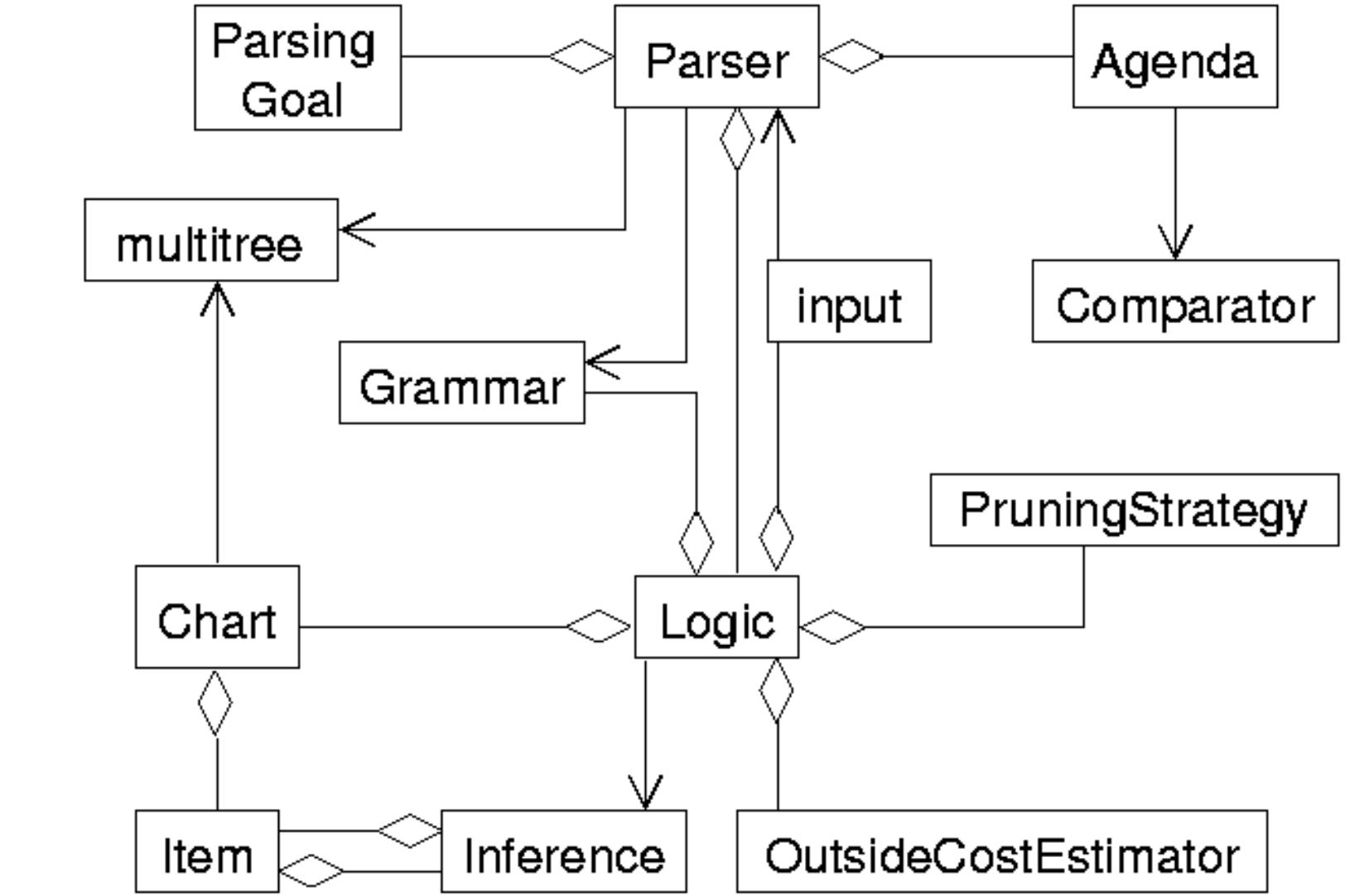
How to do everything by parsing

- single inference algorithm
- varying constraints:
  - production rules
  - pre-existing monolingual tree(s)
  - input string
  - etc.
- easier said than done...

GenPar design goals

1. powerful and configurable
   - can instantiate many kinds of SMTbyP systems
   - abstract classes interact in a fixed but generic way
   - concrete variants chosen at runtime via config files:
     - parsing algorithm
     - type of grammar
     - pruning strategy
     - etc., etc., etc.
2. easy to understand
3. easy to extend
   - new functionality added by subclassing abstract components -- surprisingly flexible!
   - OO design facilitates concurrent development of multiple components

The core of the design



Generic agenda-based parsing algorithm

Input: logic L (with grammar inside), pruning strategy P, sentence tuple, parsing goal, etc.

1. *Item I = null;*
2. *repeat*
3. *if (not agenda.empty()) then*
4. *I = agenda.pop();*
5. *set<Item> E = L.expand(G, I) // initialize if I is null*
6. *for (J ∈ E) do*
7. *if (not P(J)) // check if pruning*
8. *agenda.push(J);*
9. *until (agenda.empty() or parsing goal reached)*

Output: L.result() // multitree(s) with cost(s)

Key abstraction: Parsing Logics

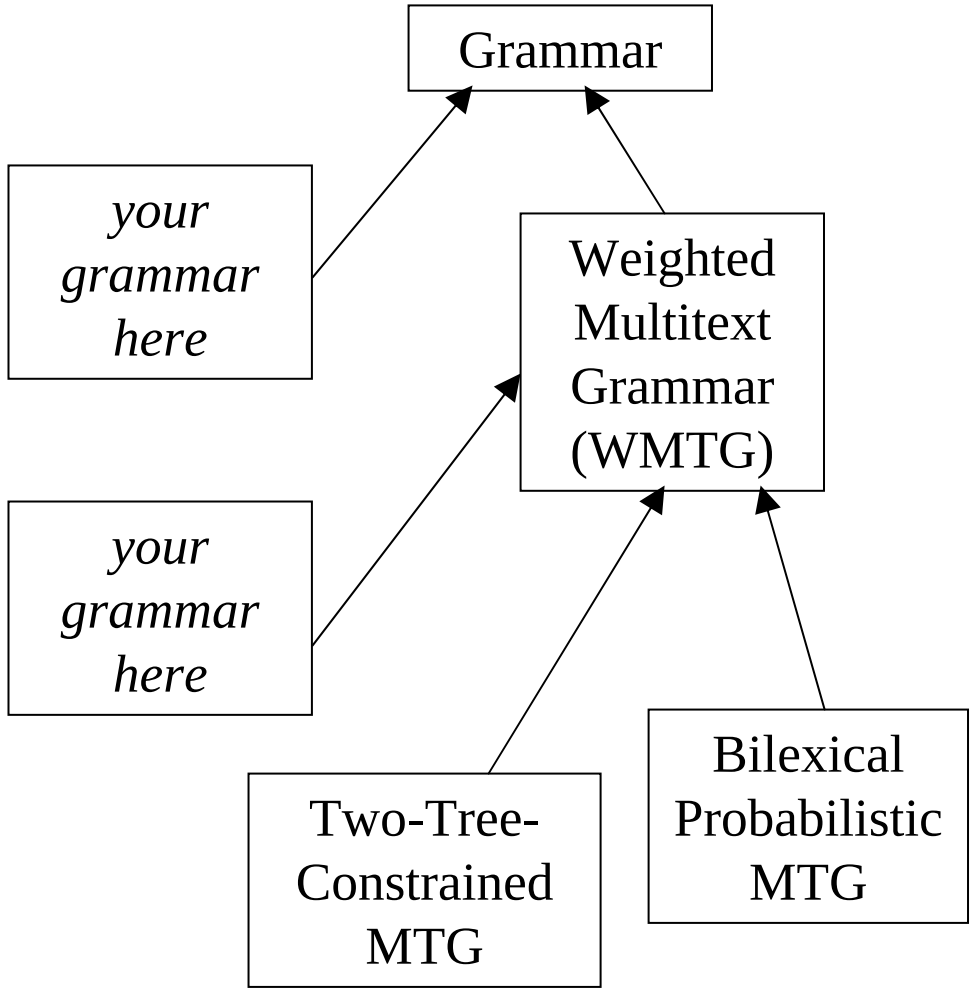
- nondeterministic parsing algorithm
- specifies which items can compose with which other items into which other items
- does not fully specify the order of compositions -- a separate search strategy can do that
- search strategy expressed by agenda's comparator
- degree of nondeterminism can vary from a lot to none

Key abstraction: Parsing Logics

- partially based on Shieber et al. (1995)
- E.g., CKY algorithm:
  - logic = bottom up
  - search strategy = shortest span first
- Different item comparator gives different parsing algorithms with same logic:
  - best first
  - left-to-right
  - random
- The comparator is a relatively tiny piece of code, easy to write.

Key Abstraction: Grammar encapsulation

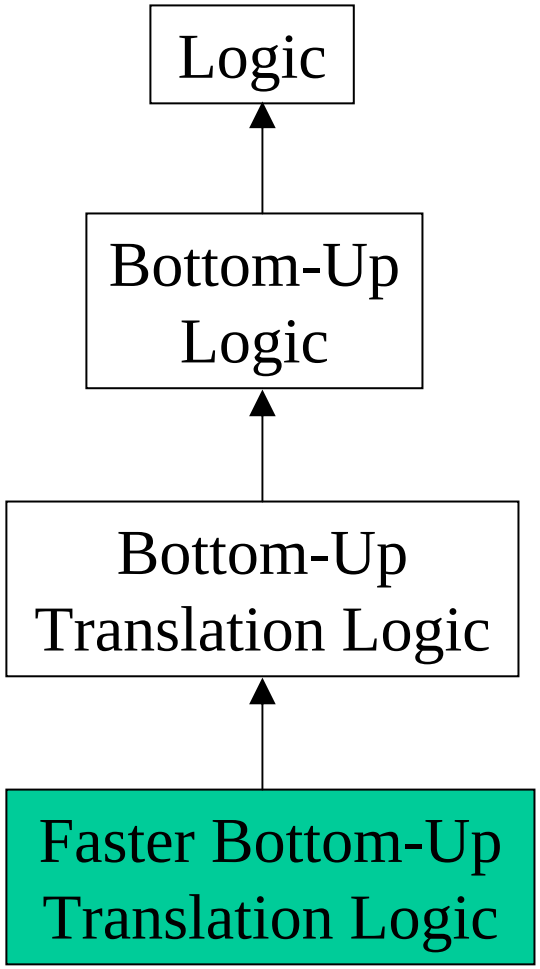
- grammars evaluate partial parses
  - *getPossibleCons(item1, item2)*: decide if two items can compose
  - *getCost(inference)*: compute inference cost
- can be based on production rules
- can be based on some completely different system of constraints



Extensions

- New variants are easy to implement, sometimes surprisingly easy!
- new logics for:
  - faster translation
  - multiparsing with no agenda
- new grammars for
  - alignment with one constraining tree
  - “phrases”

New functionality: faster translation



abstract

implements expand() with genScans() and genComposes()

extends expand() with genLoads()

overrides genLoads() to filter loads based on input sentence

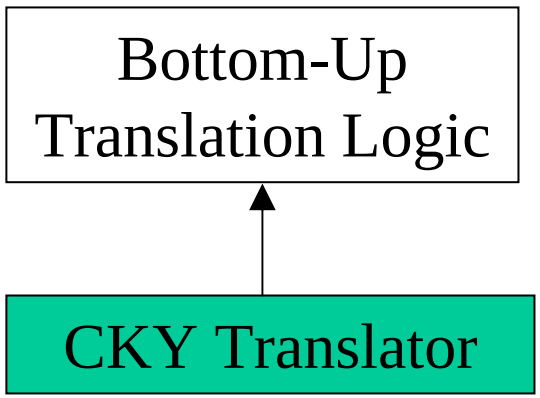
Effects of a single method override

inference counts for translation of 10 sentences using tiny grammar

language pair	inferences under naïve logic	inferences under faster logic	reduction factor
English to English	998	646	1.55
French to English	1092	974	1.12
Arabic to English	2341	996	2.35

On larger grammar: reduced by 2 orders of magnitude.

New functionality: translation without agenda



- overrides expand() to implement deterministic CKY translation algorithm
- after translation, expand() returns the empty set, so agenda never used
- still uses superclass's methods for genScans(), genLoads(), genComposes()

Generic ~~agenda based~~ parsing algorithm

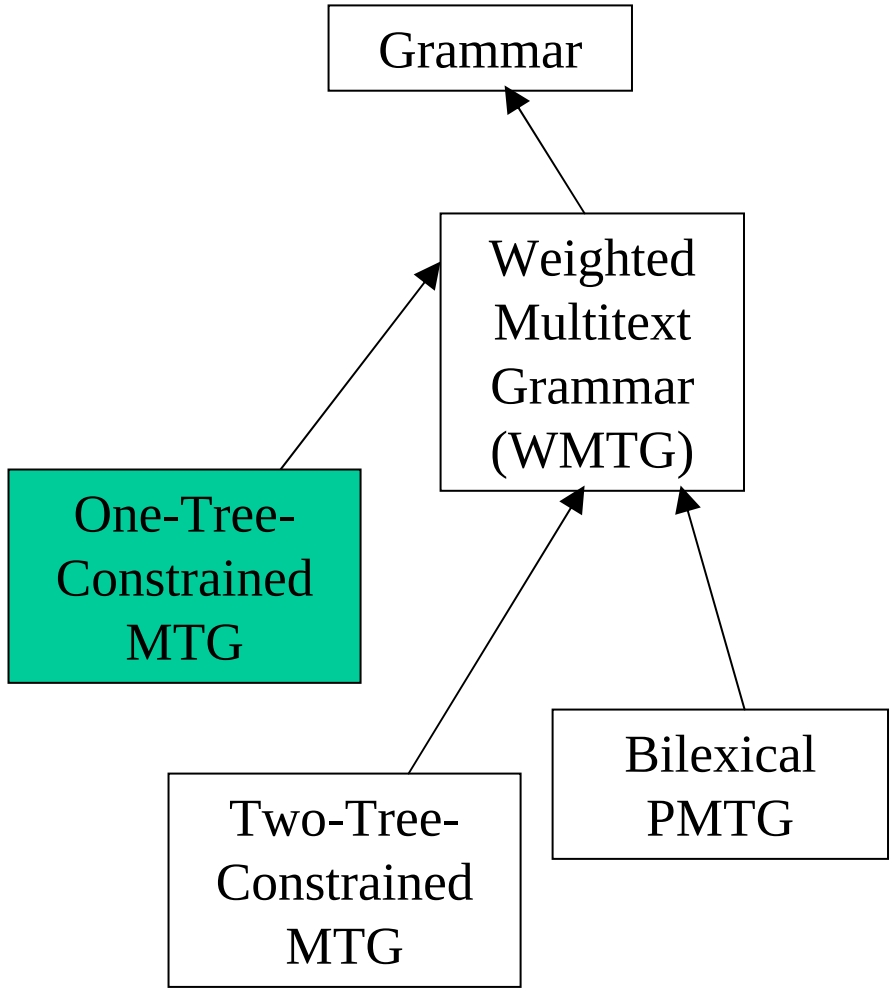
Input: logic L (with grammar inside), pruning strategy P, sentence tuple, parsing goal, etc.

```
1.  Item I = null;
2.  repeat
3.      if (not agenda.empty()) then
4.          I = agenda.pop();
5.      set<Item> E = L.expand(G, I)   // initialize if I is null
6.      for (J ∈ E) do
7.          if (not P(J))              // check if pruning
8.              agenda.push(J);
9.  until (agenda.empty() or parsing goal reached)
```

Output: L.result() // multitree(s) with cost(s)

New functionality: alignment with one constraining tree

- overrides getPossibleCons() to consult only one constraining tree
- completely transparent to logic



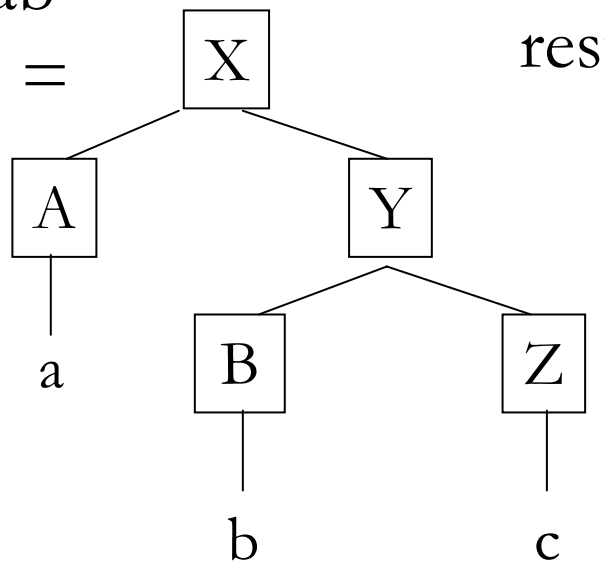
Possible new functionality: “phrases”

- assume we have a phrase list for one or both languages
- easy case: contiguous n-grams
- (treegrams a bit trickier, but not much)
- e.g., “kick the bucket”, “there is”, “account for”
- solution:
  - treat each phrase as a possible constituent, with unique nonterminal label
  - attach to or identify with lowest subsuming node in constraining tree during alignment
  - slash sibling nonterminal labels

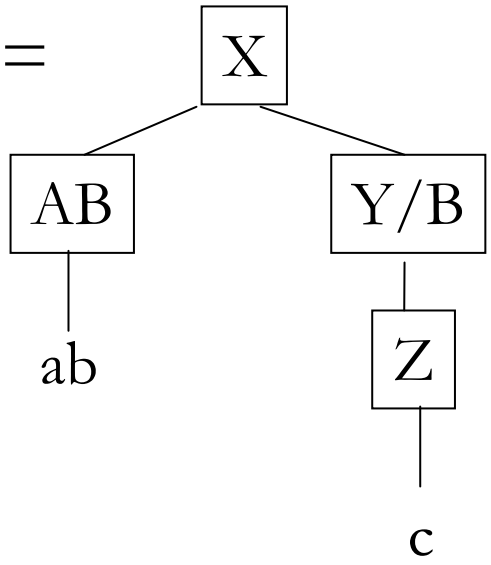
Possible new functionality: "phrases"

E.g., phrase = "ab"

constraining tree =

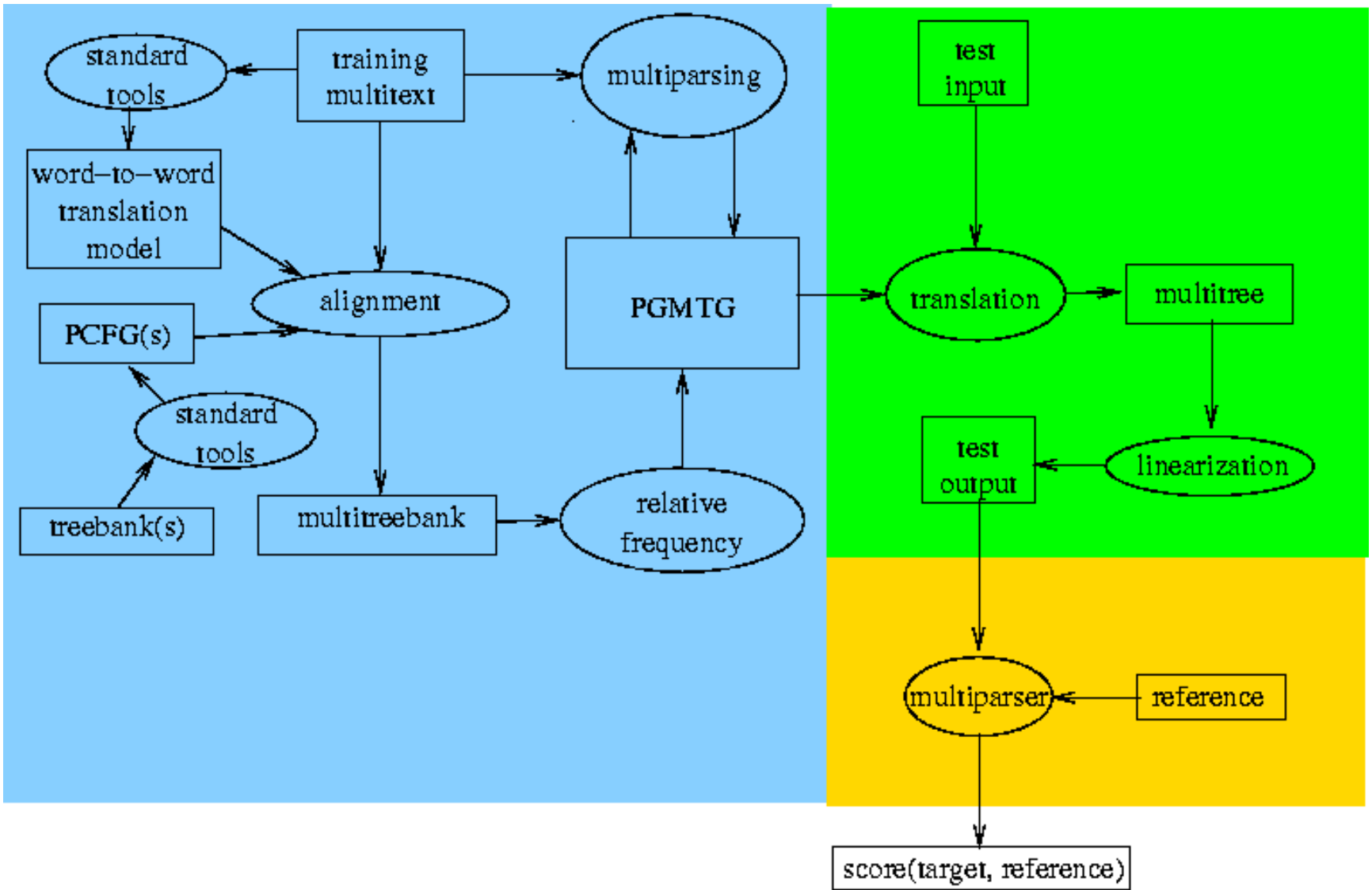


result =



- use trie for efficient phrase recognition during scanning of the input
- span of scanned word can be wider than 1
- no other changes for retraining and translation
- all changes encapsulated in grammar

System integration



Lowering entry barriers: Sandboxes

A sandbox is…

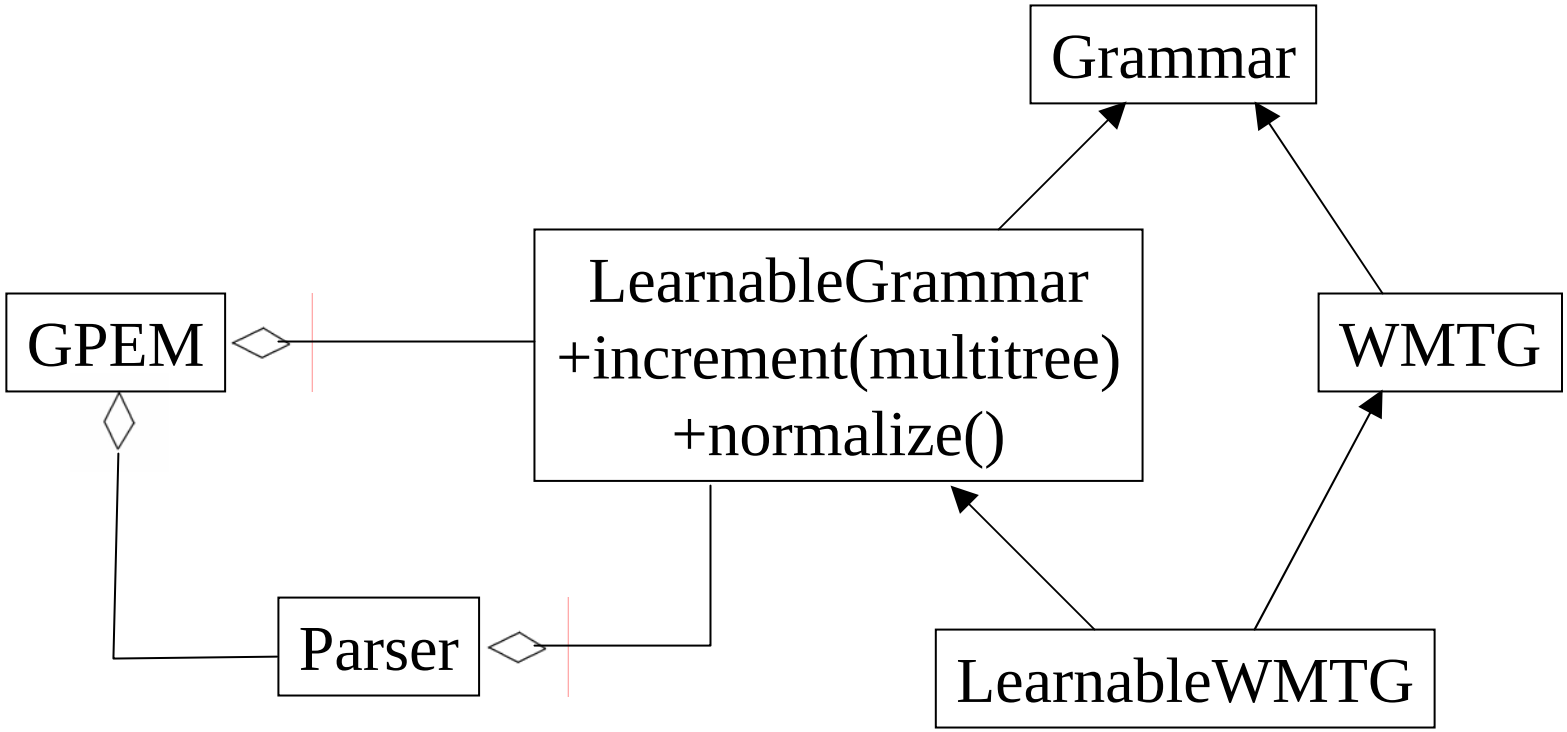
- a directory structure where a single command will run the end-to-end system on a toy-sized corpus
  - simply go to base directory and type “make”
- a validation suite for developers
  - after each change of code, make sure nothing broke
- an educational tool for SMT
  - seeing some data run through the pipeline is a good way to get familiar with the system
- a blueprint for experiments
  - change config files, and run your own data through it
- toolkit includes sandboxes for 3 language pairs

Feasibility of SMT by Parsing

- parameter estimation in GenPar
- highlights of configuration used
- the data
- automatic evaluation method
- preliminary results

parameter estimation in GenPar

So far, very primitive: Viterbi estimation.



Machine learning toolkits can be integrated.

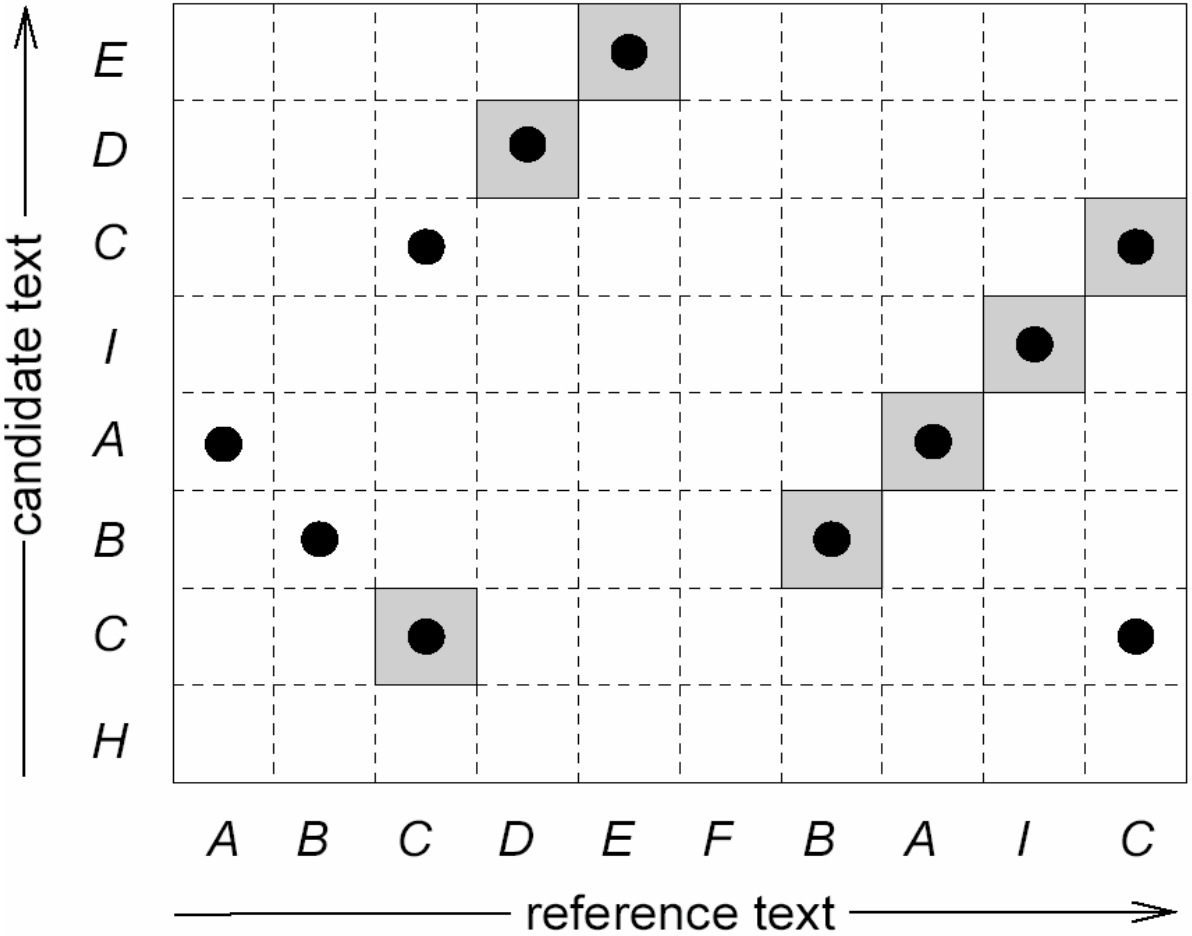
configuration used for experiments

- Logic: Bottom Up
- Search Strategy: Best First
- Grammar:
  - for alignment: tree-constrained MTG
  - for retraining & translation:
    - bilexical (headed) probabilistic MTG
    - fine-grained generative process with strong independence assumptions but no smoothing
- N.B: first-cut model and training method

data used for experiments

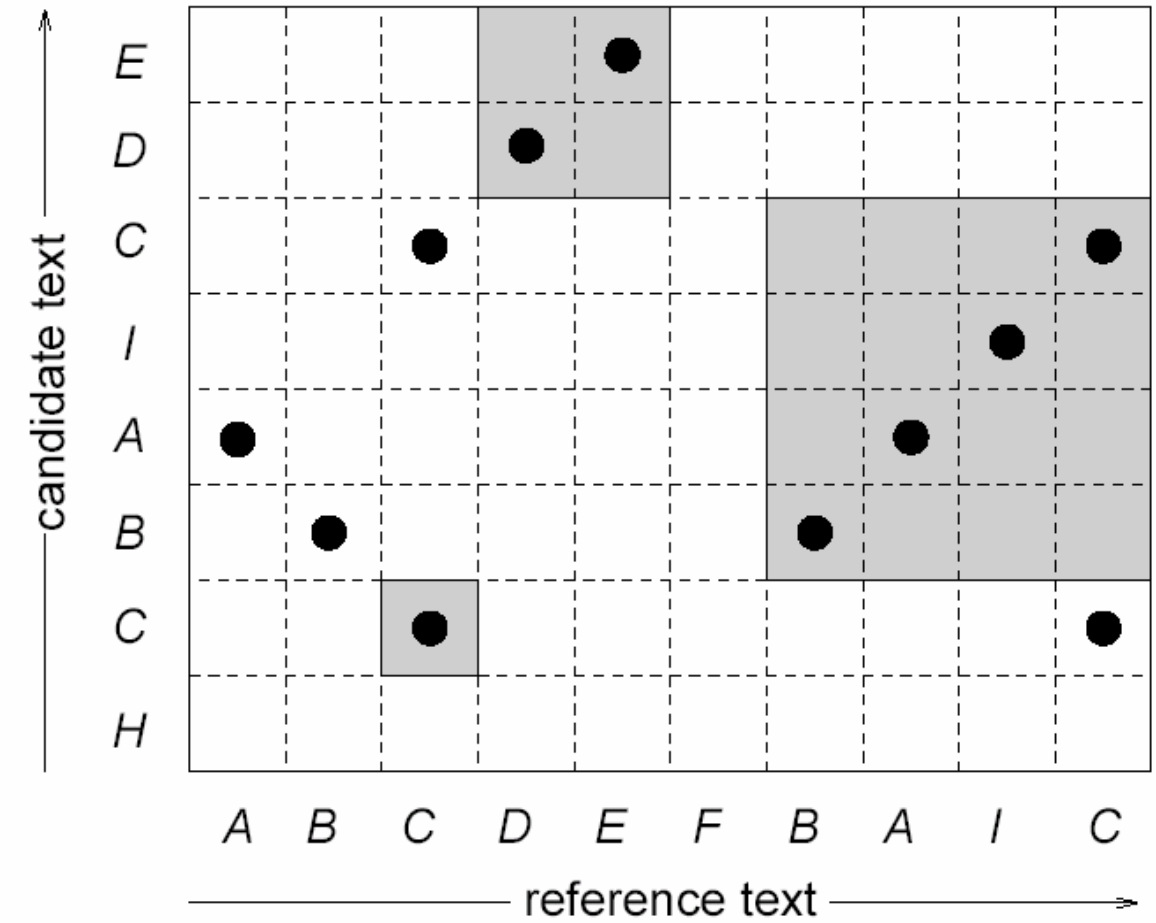
- English tagger (Ratnaparkhi) & parser (Bikel) trained on PTB
- French to English
  - subset of EuroParl corpus (Koehn'02)
  - pretokenized, except for some missing periods
  - stop-lists off the web
  - train/dev set from standard training partition
- Arabic to English
  - tagger from Diab@HLT'04
  - Bikel parser, trained on part of Arabic treebank (ATB)
  - training data: A/E parallel news corpus
  - test data: NIST MTEval'03 test set

MT evaluation: measuring text overlap



To avoid double-counting, the overlap is a maximum matching.

Rewards for longer matches



Reward diagonal runs more than linearly in their length.

E.g., run weight is the area of its minimum enclosing square.

MT evaluation: the standard measures

- maximum match size (MMS) = maximum combined area of non-overlapping squares
- take square root to linearize
- normalize by lengths of the candidate (C) and reference (R) to get a score between 0 and 1:
  - P = precision = MMS / |C|
  - R = recall = MMS / |R|
  - F-measure = harmonic mean of P and R

    ≈ the fraction of the grid covered by matches
- measure not developed here, but the software is part of toolkit

preliminary evidence for empirical feasibility

- approx speed (without much optimization)
  - alignment, retraining: about 1 sentence pair per second
  - translation: 1 sentence per minute with no LM
- learning curve (Declan)

New functionality: Target language models

A proposal for follow-on research

(Markus)

MT Evaluation by Parsing

A proposal for follow-on research (Ben)

The ITG Hypothesis for Arabic/English

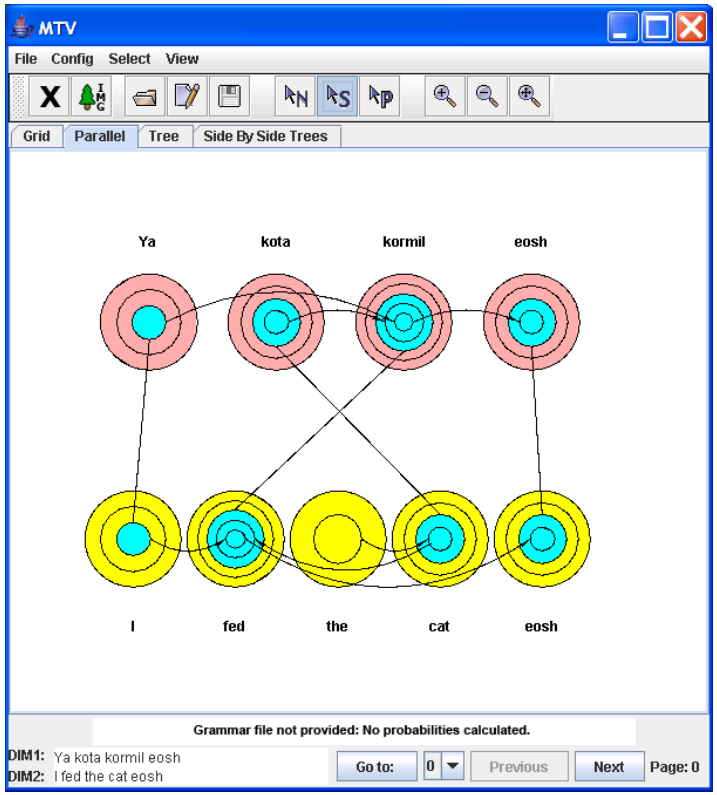
(Dekai)

Take-home messages

- Findings:
  - Multiparsing can be fast.
  - Typical models fit in the memory of typical machines.
  - Our first-cut system is comparable in accuracy to an off-the-shelf FST-based SMT system.
    - On the only corpus on which we could make a direct comparison, which was small.
  - Much low-hanging fruit ready to be picked, to improve speed and accuracy.
- Conjectures:
  - All processes are easily parallelizable at the sentence level.
  - Processing typical bitext sizes is no more difficult than for FST-based SMT.

Take-home software

- On our team's homepage
  http://www.clsp.jhu.edu/ws2005/groups/statistical
- MTV



- GenPar
  - sign onto genpar-announce@cs.nyu.edu to be notified of updates

(near) future directions

- target language models
- MT evaluation by parsing
- phrases
- other grammar formalisms (CCG, TSG, ...)
- smoothing methods for generative transduction grammars
- machine learning beyond EM
- more sophisticated pruning
- beyond single-best translations

Review

- short-term
  - lower the entry barriers to the field
  - demonstrate feasibility of SMT by Parsing
- short- and long-term
  - answer fundamental scientific questions
  - educate the next generation
  - accelerate progress in MT
- long-term
  - help to reunite MT with NLP

Helping to reunite MT with NLP

- Recently, MT research has been borrowing more from machine learning than from NLP.
- Strong connection between MT and parsing should make both subfields pay more attention to each other.
- Then, MT can improve by parsing.
- Parsing can claim another application.
- A Good Thing, long term.

Looking forward

	SMT	SMT by Parsing
first proposed	1988	1995
publicly available toolkit	+11 years = 1999	+10 years = 2005
dominant approach	+3 years = 2002	??