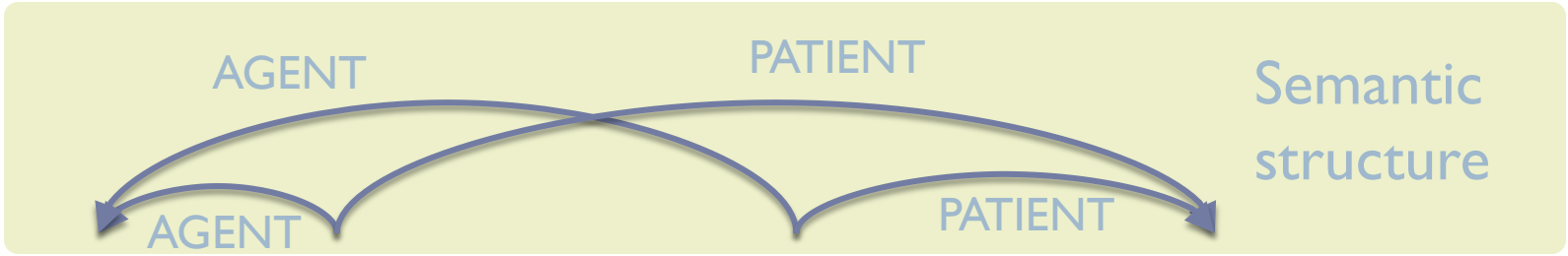
Online Graph Planarisation for Synchronous Parsing of Semantic and Syntactic Dependencies

Ivan Titov
University of Illinois at Urbana-Champaign

James Henderson, Paola Merlo, Gabriele Musillo
University of Geneva

Motivation / Problem Statement

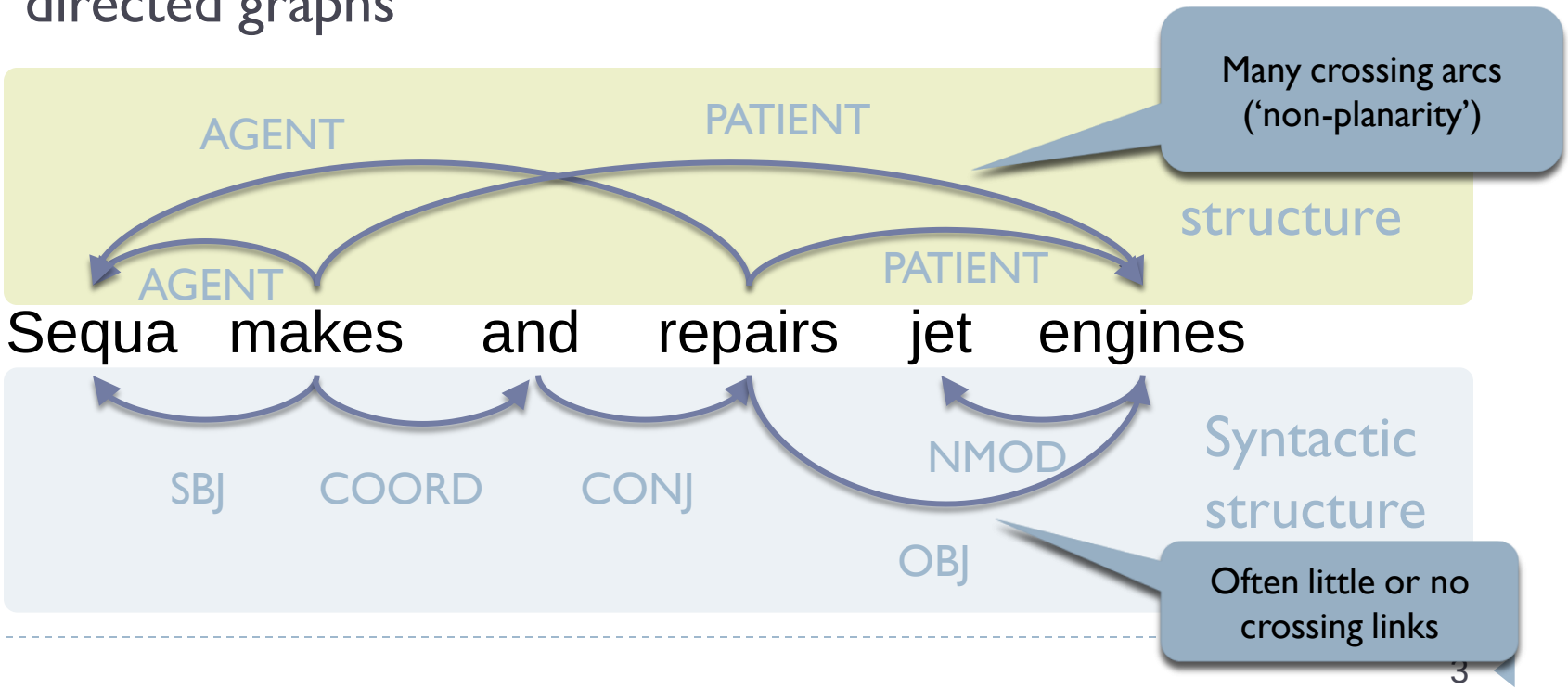
- NLP applications will require a shallow representation of meaning
  - Often shallow semantic structures can be regarded as labeled directed graphs



- Can we apply methods for syntactic dependency parsing to predict these trees?

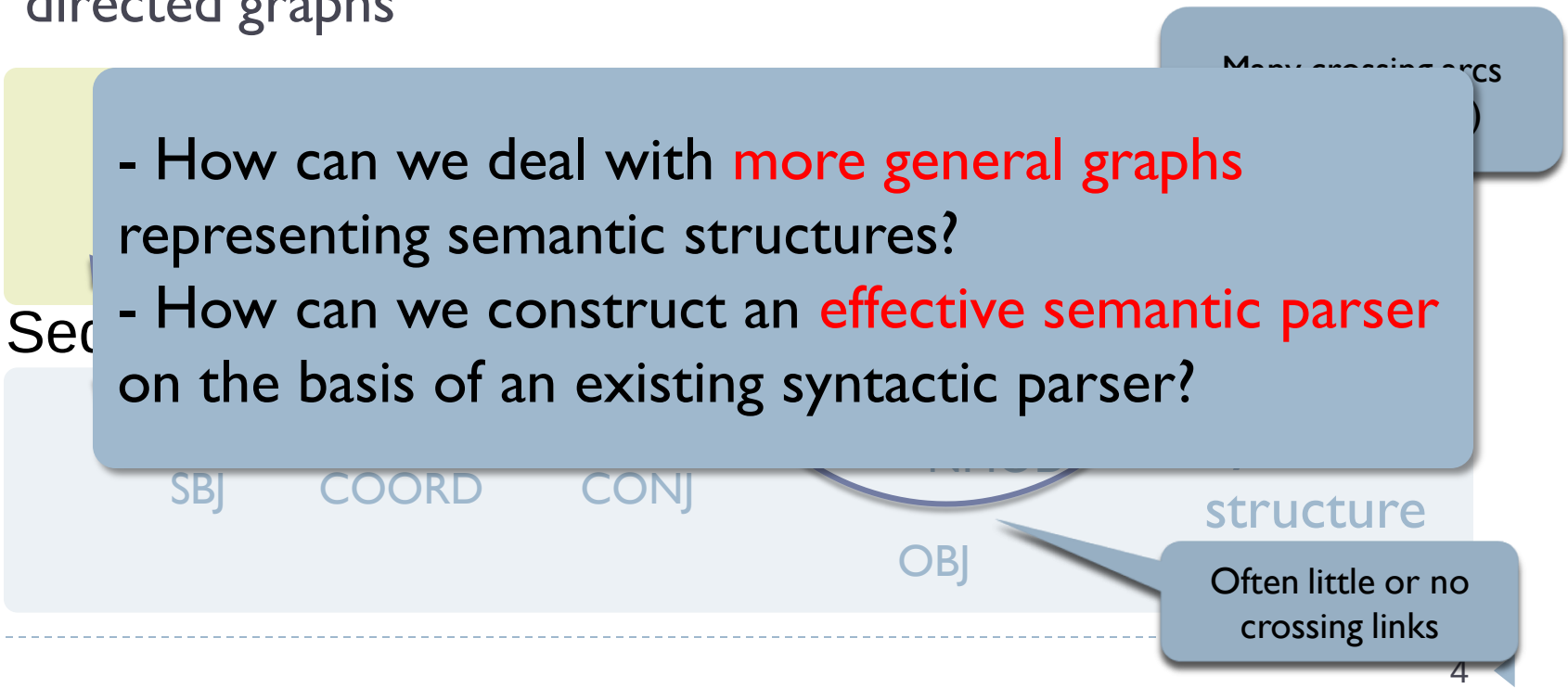
Motivation / Problem Statement

- NLP applications will require a shallow representation of meaning
  - Often shallow semantic structures can be regarded as labeled directed graphs



Motivation / Problem Statement

- NLP applications will require a shallow representation of meaning
  - Often shallow semantic structures can be regarded as labeled directed graphs



Outline

- Motivation / Problem Statement
- Background
  - Dependency parsing
  - Properties of dependency graphs
- Non-Planar Parsing using Swapping
- Synchronous Parsing of Semantic and Syntactic Dependencies
  - Synchronization
  - Statistical Model
- Experiments
- Conclusions and Future Directions

Dependency Parsing Problem

- Parsing: given sentence $x \in \mathcal{X}$ predict structure $y \in \mathcal{Y}$:

$$\hat{y} = \text{argmax}_y \, F(y, x|w)$$

Model estimated on a labeled dataset (treebank)

- Graph-based methods: assume that features factorize over subgraphs of y (e.g., edges) *[Eisner, 96; McDonald et al., 05]*

That will be the main focus of the talk

- Transition-based methods *[Yamada and Matsumoto, 03; Nivre et al., 04]*:
  - Define derivation order for structures, i.e. mapping from y to sequences of decisions $(d_1, \ldots, d_{n(y)})$
  - Learn to score individual decision given preceding decisions:

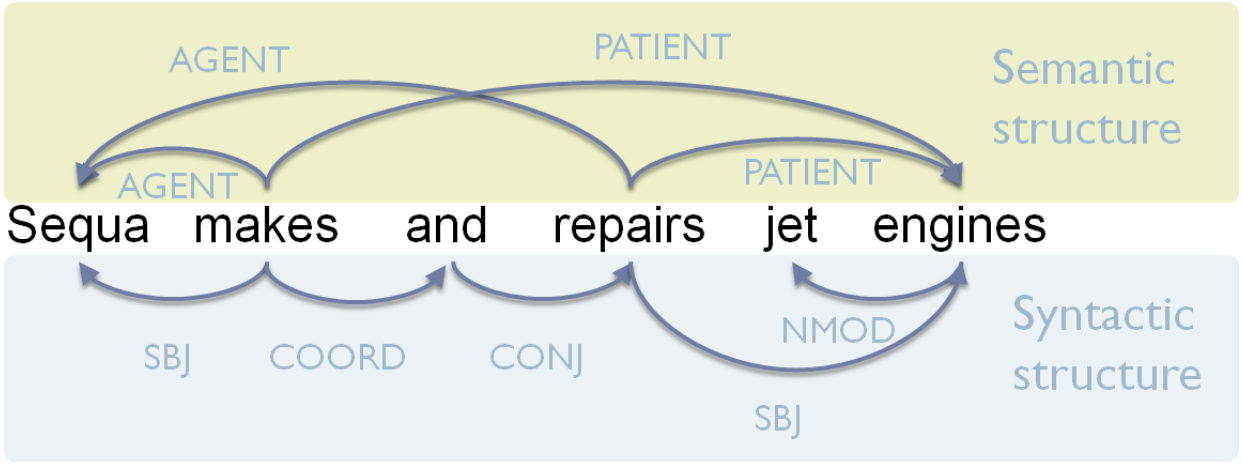
$$F(y, x|w) = \sum_{i=1}^{n(y)} f(d_i|d_1, \ldots, d_{i-1}, x, w)$$

  - Decode greedily: $\hat{d}_i = \text{argmax}_{d_i} \, f(d_i|\hat{d}_1, \ldots, \hat{d}_{i-1}, x, w)$

Beam-search can be used instead of greedy decoding

Properties of Semantic Structures

- Semantic structures are not trees
  - Graph-based (GB) methods based on maximum spanning tree algorithms are not directly applicable
- Semantic structures are not planar
  - *Definition:* planar graphs can be drawn in the semi-plane above the sentence without any two arcs crossing and without changing the order of words



Properties of Semantic Structures

- Semantic structures are not trees
  - Graph-based (GB) methods based on maximum spanning tree algorithms are not directly applicable
- Semantic structures are not planar
  - *Definition:* planar graphs can be drawn in the semi-plane above the sentence without any two arcs crossing and without changing the order of words
  - Most transition-based (TB) algorithms handle only planar graphs

- *Related work:*
  - *[Attardi, 06]: TB method with extended derivation order to handle non-planarity*
  - *[Nivre, 08]: Assumes that a structure can be made planar by changing order of words (not true for general non-tree graphs)*
  - ...

Properties of Semantic Structures

- Semantic structures are not trees
  - Graph-based (GB) methods based on maximum spanning tree algorithms are
- Semantic structu
  - *Definition:* plar e above the sentence with hanging the order of word
  - Most transition-based (TB) algorithms handle only planar graphs

We will propose a very simple technique to extend standard TB methods **to handle non-planar and not-tree structured graphs**

- *Related work:*
  - *[Attardi, 06]: TB method with extended derivation order to handle non-planarity*
  - *[Nivre, 08]: Assumes that a structure can be made planar by changing order of words (not true for general non-tree graphs)*
  - ...

Outline

- Motivation / Problem Statement
- Background
  - Dependency parsing
  - Properties of dependency graphs
- Non-Planar Parsing using Swapping
- Synchronous Parsing of Semantic and Syntactic Dependencies
  - Synchronization
  - Statistical Model
- Experiments
- Conclusions and Future Directions

Derivation order *[Nivre, 04]*

- State of the parser after steps $(d_1, \ldots, d_{i-1})$ is characterized by:
  - current stack *S* (w_j – word on top of the stack)
  - a queue *I* of remaining input words (w_k – next input word)
  - partial dependency structure defined by $(d_1, \ldots, d_{i-1})$
- New decision d_i can be
  - *LeftArc*$_r$ - adds a labeled dependency arc $w_j \overset{r}{\leftarrow} w_k$
  - *RightArc*$_r$ - adds a labeled dependency arc $w_j \overset{r}{\rightarrow} w_k$
  - *Shift* - moves w_k from queue to the stack
  - *Reduce* - remove w_j from the stack
- Terminates when the queue is empty

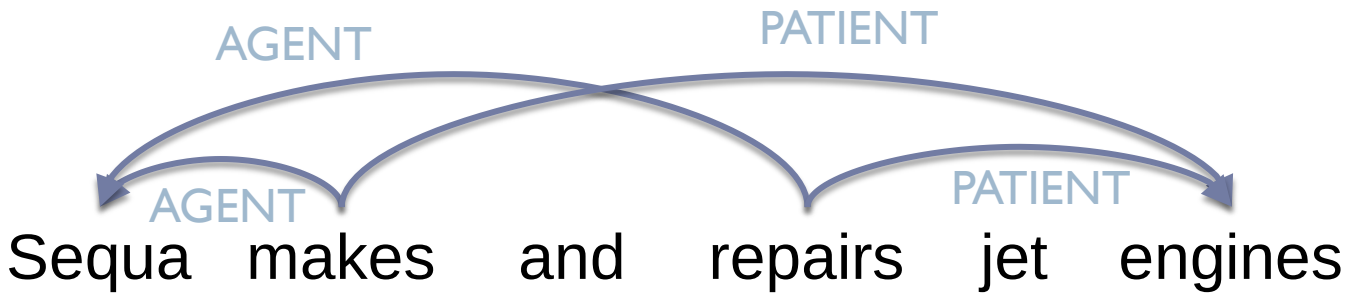
Handling non-planar structures

- *[Nivre, 04]* order cannot handle non-planar structures:
  - All the arcs are created between top of the stack and front of queue
  - Words are stored in the stack in the same order as they appear in the sentence
- A single new decision:
  - *Swap* - swaps 2 top words in the stack

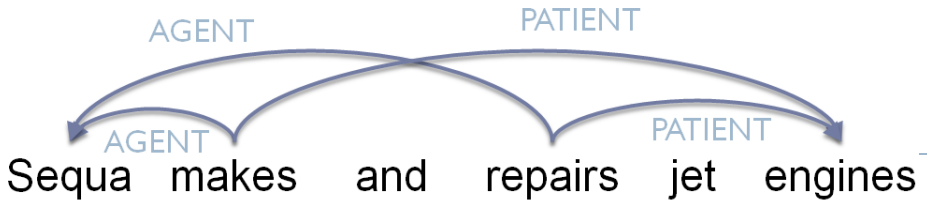
Stack before: $S = [\ldots, w_m, w_j]$

Stack after: $S = [\ldots, w_j, w_m]$

Example


AGENT
PATIENT
AGENT
PATIENT
Sequa makes and repairs jet engines

Example

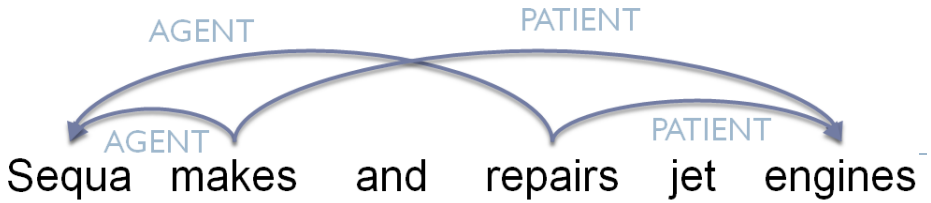


- Partial Structure:

Sequa makes and repairs jet engines

- S = [ ]
- I = [Sequa makes and ....]
- Next action: *Shift*

Example

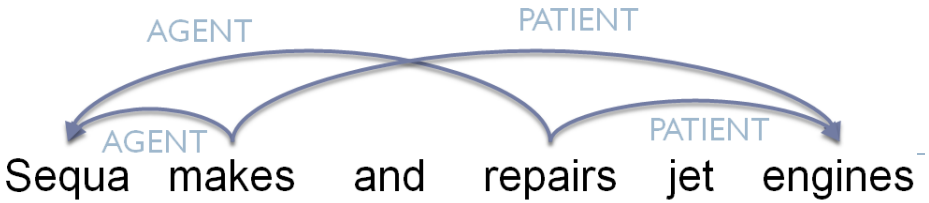


- Partial Structure:

Sequa makes and repairs jet engines

- S = [Sequa]
- I = [makes and repairs ....]
- Next action: $LeftArc_{AGENT}$

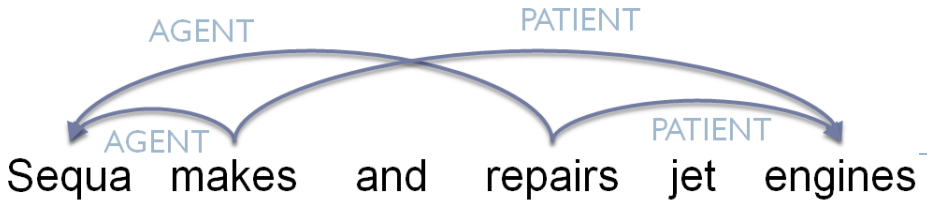
Example



- Partial Structure:

- S = [Sequa]
- I = [makes and repairs ....]
- Next action: *Shift*

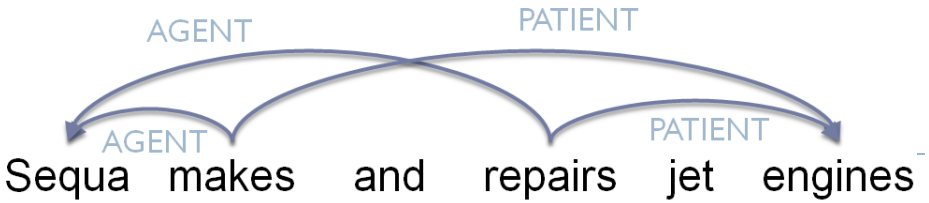
Example



- Partial Structure:

- S = [Sequa makes]
- I = [and repairs jet engines]
- Next action: *Shift*

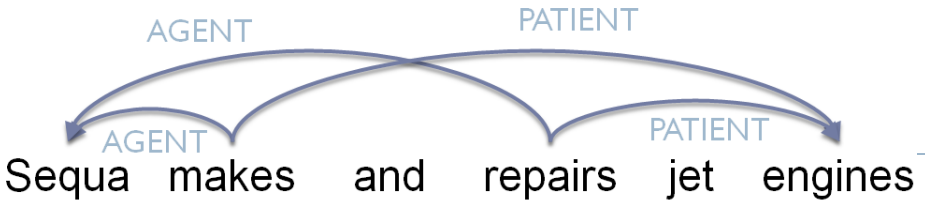
Example



- Partial Structure:

- S = [Sequa makes and]
- I = [repairs jet engines]
- Next action: *Reduce*

Example

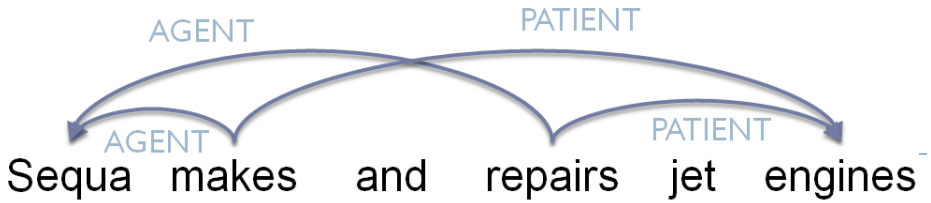


- Partial Structure:

- S = [Sequa makes]
- I = [repairs jet engines]
- Next action: *Swap*

Stalled if without swap:
- *repairs* needs an arc to *Sequa*
- but *makes* cannot be removed from stack

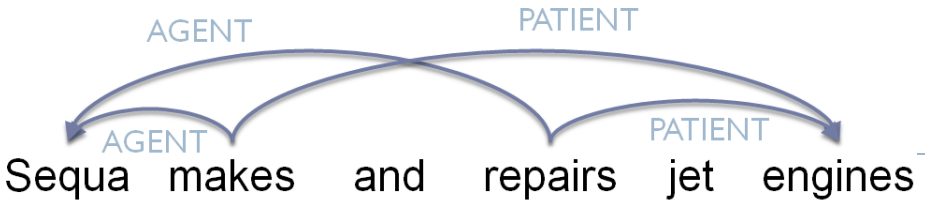
Example



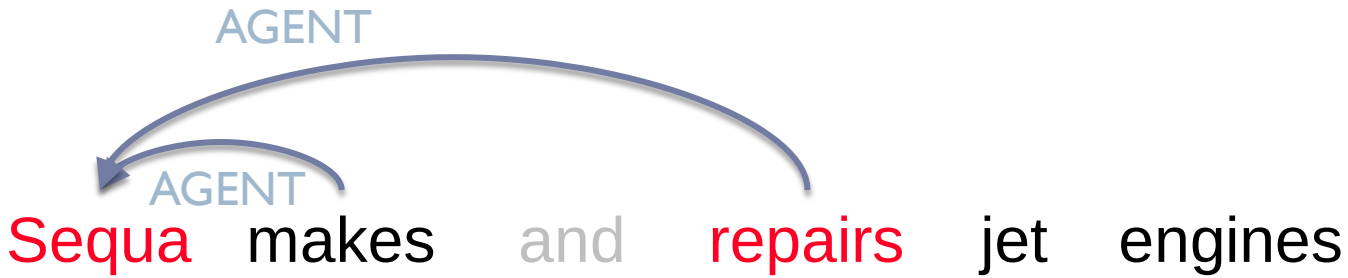
- Partial Structure:

- S = [makes Sequa]
- I = [repairs jet engines]
- Next action: *LeftArc*$_{AGENT}$

Example

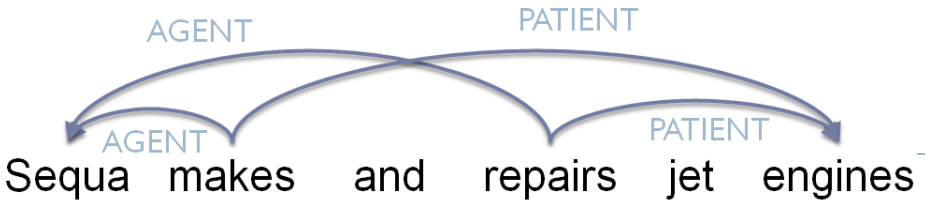


- Partial Structure:

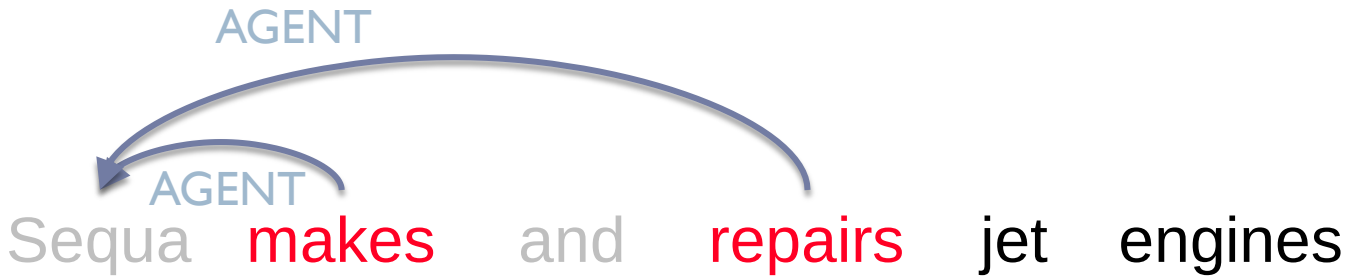


- S = [makes Sequa]
- I = [repairs jet engines]
- Next action: *Reduce*

Example

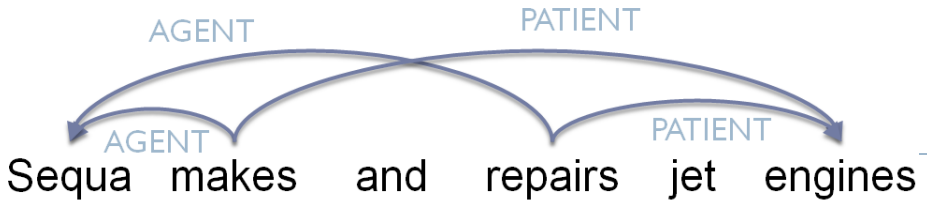


- Partial Structure:

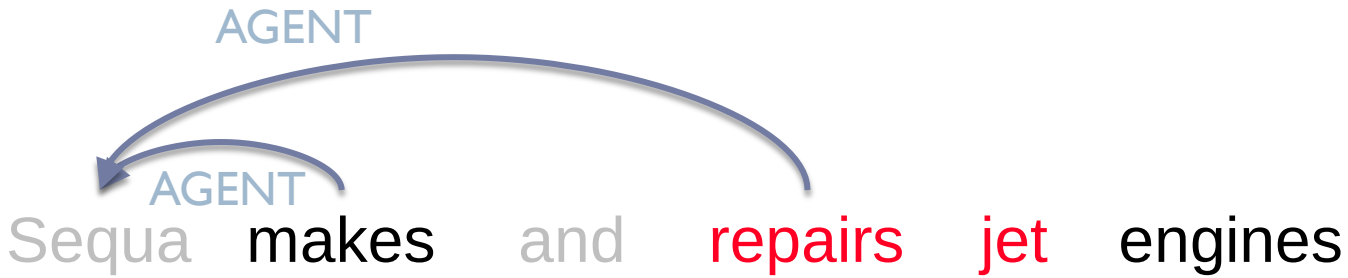


- S = [makes]
- I = [repairs jet engines]
- Next action: *Shift*

Example

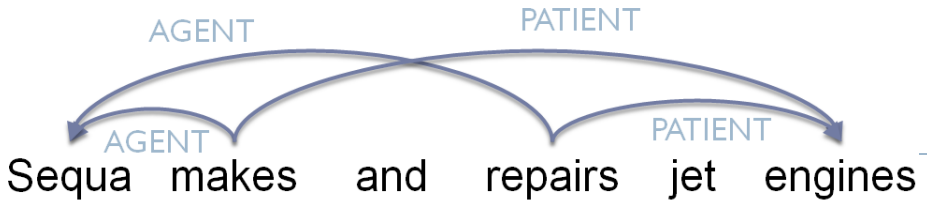


- Partial Structure:

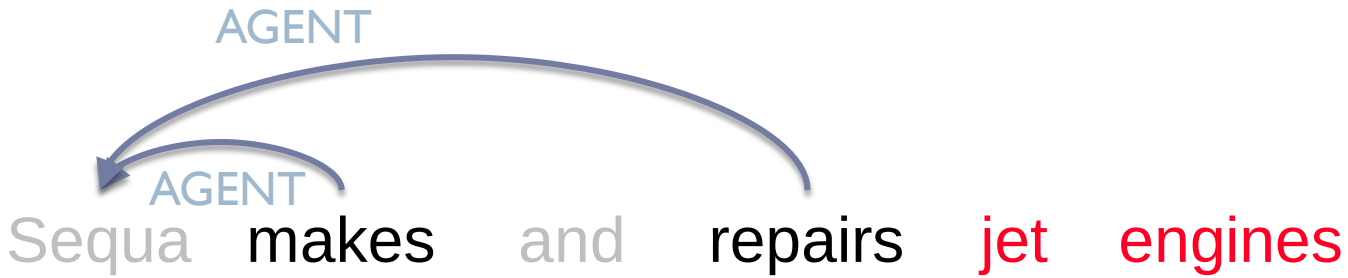


- S = [makes repairs]
- I = [jet engines]
- Next action: *Shift*

Example

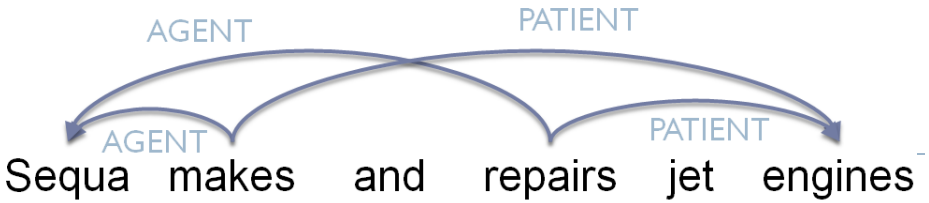


- Partial Structure:

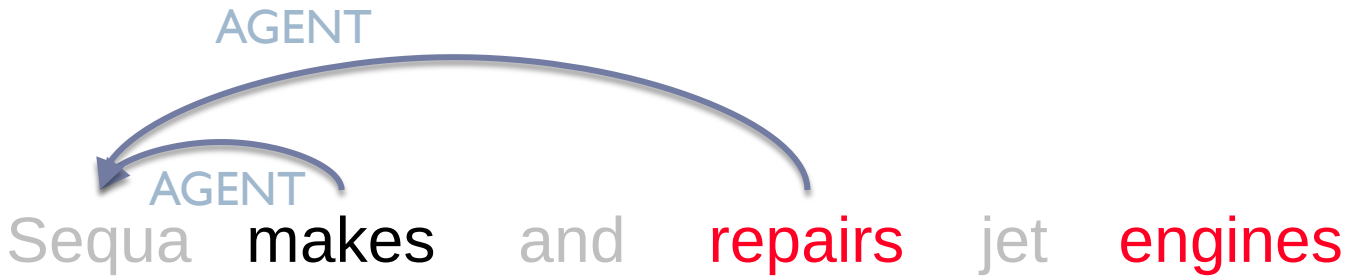


- S = [makes repairs jet]
- I = [engines]
- Next action: *Reduce*

Example

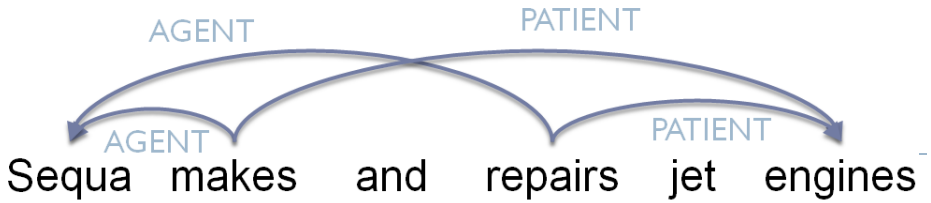


- Partial Structure:

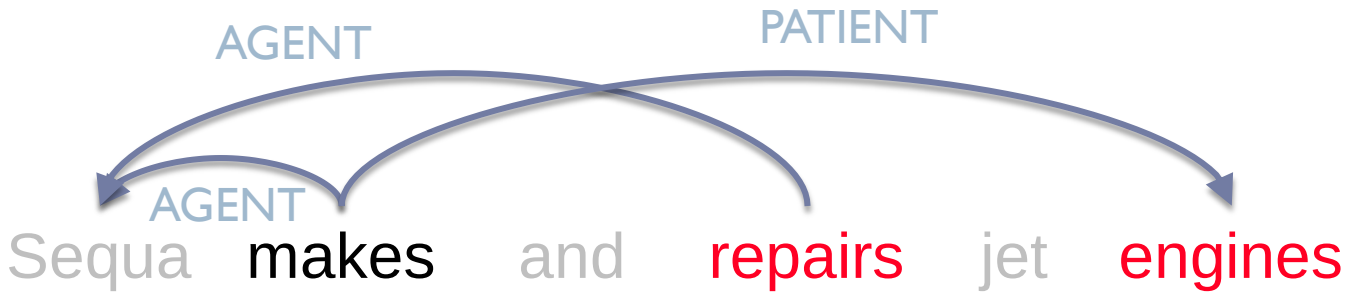


- S = [makes repairs]
- I = [engines]
- Next action: $RightArc_{PATIENT}$

Example

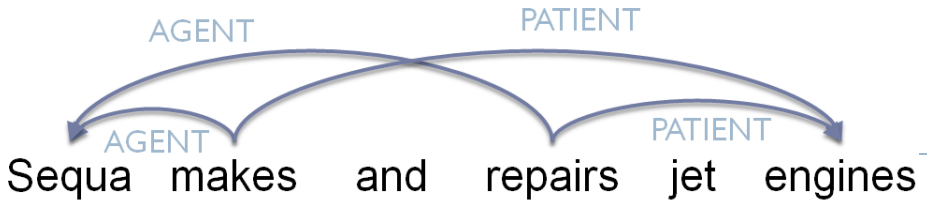


- Partial Structure:

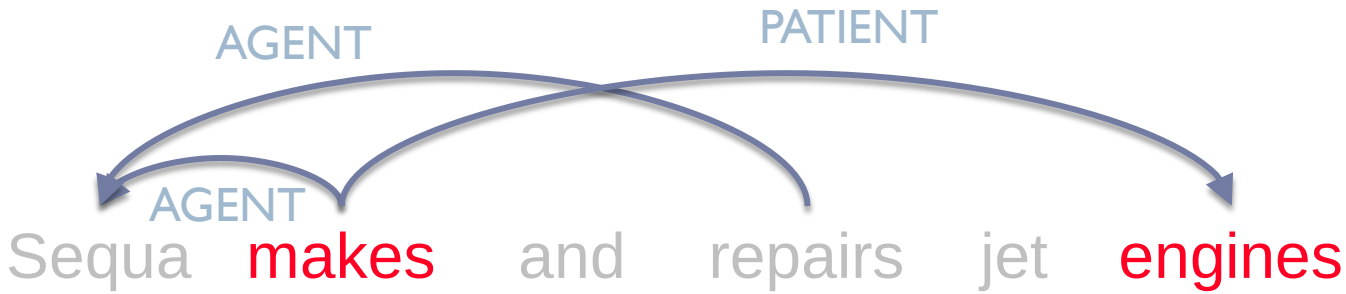


- S = [makes repairs]
- I = [engines]
- Next action: *Reduce*

Example

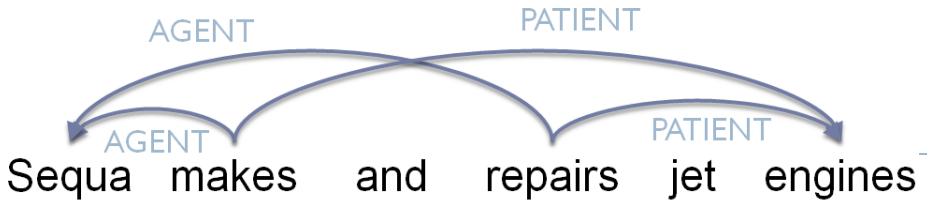


- Partial Structure:

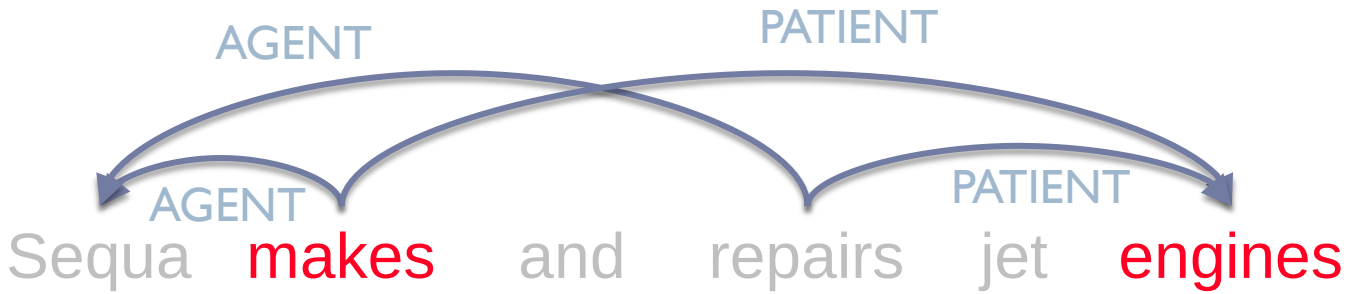


- S = [makes]
- I = [engines]
- Next action: *RightArc*$_{PATIENT}$

Example

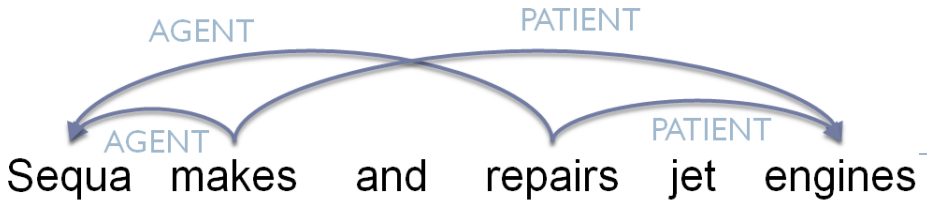


- Partial Structure:

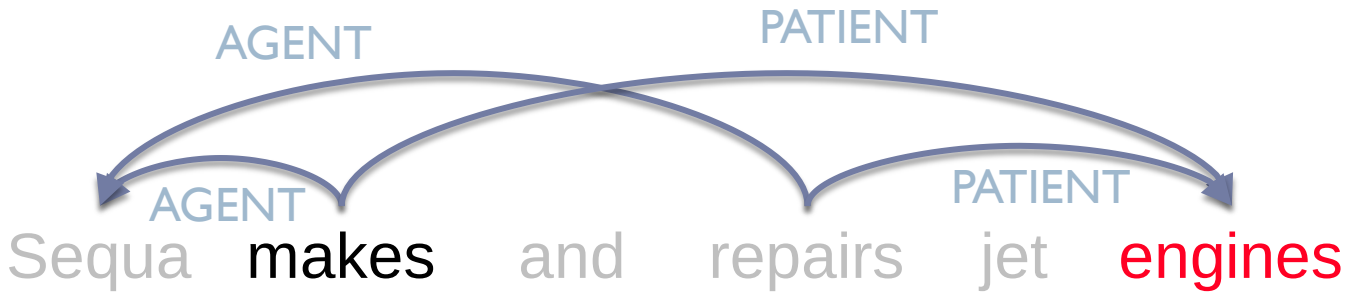


- S = [makes]
- I = [engines]
- Next action: *Shift*

Example



- Partial Structure:



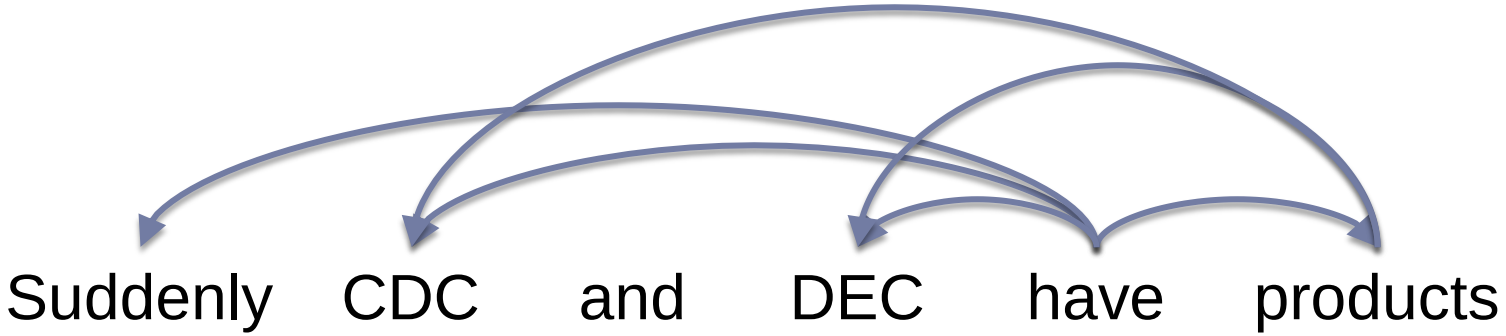
- S = [makes engines]
- I = [ ]
- Next action: *Stop*

Canonical orders

- The algorithm allows the same structure to be parsed multiple ways
  - Summing over all the possible derivation when parsing is not feasible
  - Instead, models are trained to produce derivations in a canonical way
- In other words, canonical orders define how derivations should be produced for the training set
- We consider two canonical derivations which differ in when swapping is done
  - last-resort: *Swap* is used as a last resort when no other operation is possible
  - exhaustive: *Swap* is used pre-emptively

Last-Resort ordering

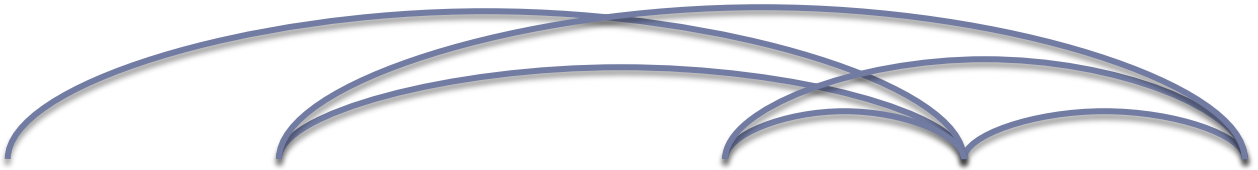
- Swap is used as a last resort when no operation is possible
- Drawback: not all the structures parsable with swapping have a last-resort derivation
  - in CoNLL-2008 dataset 2.8% fewer structures are parsable with last-resort ordering
  - An example of such a structure:



- Advantage: this canonical derivation is predictable and, therefore, supposedly easier to learn

Exhaustive ordering

- Algorithm for preemptive swapping:
  - Ordering follows standard planar parser ordering until no other operation except *Shift* and *Swap* are possible
  - Compute the ordered list of positions of words in the queue to which current top of the stack w_j will be connected
  - Compute a similar list for word w_m under the top of the stack
  - Swap if w_m's list precedes w_j's list in their lexicographical order



$Suddenly_1$ CDC_2 and_3 DEC_4 $have_5$ $products_6$

- S = [Suddenly CDC]
- I = [DEC ....]
- List for 'Suddenly': {5} $\prec$ List for 'CDC': {5, 6} $\Rightarrow$ *Swap*

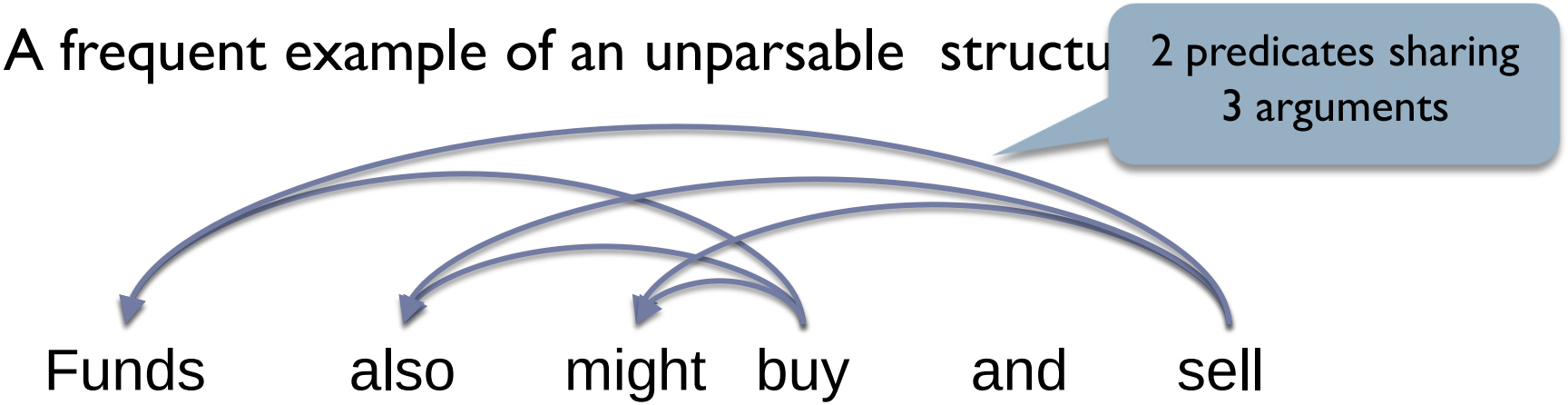
Exhaustive ordering

- Algorithm for preemptive swapping:
  - Ordering follows standard planar parser ordering until no other operation except *Shift* and *Swap* are possible
  - Compute the ordered list of positions of words in the queue to which current top of the stack w_j will be connected
  - Compute a similar list for word w_m under the top of the stack
  - Swap if w_m's list precedes w_j's list in their lexicographical order

- Theorem *If the graph is parsable with the defined set of operations then the exhaustive ordering is guaranteed to find a derivation*
  - See the paper for the proof sketch

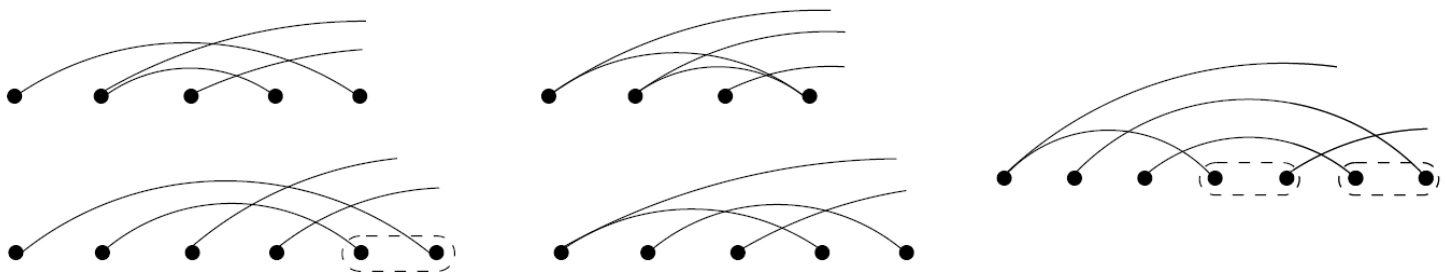
Structures Parsable with Swapping

- Not all the non-planar graphs are parsable
  - In CoNLL-2008 ST dataset only 1% of semantic structures are not parsable whereas 44% are not planar (i.e., require swapping)
- Among common linguistic structures requiring Swap are coordinations
  - E.g., *"Sequa makes, repairs and sells engines"*
- A frequent example of an unparsable structu



Structures Parsable with Swapping

- Any structures with isolated pairs of crossing arcs are parsable but they are more powerful than that
- Theorem *A graph cannot be parsed with the defined set of parsing operations iff the graph contains at least one of the subgraphs presented below:*
  - *the unspecified arc end points can be anywhere strictly following those specified*
  - *circled pairs of endpoints can be either a single word or two distinct words*



- See the paper for the proof sketch

Outline

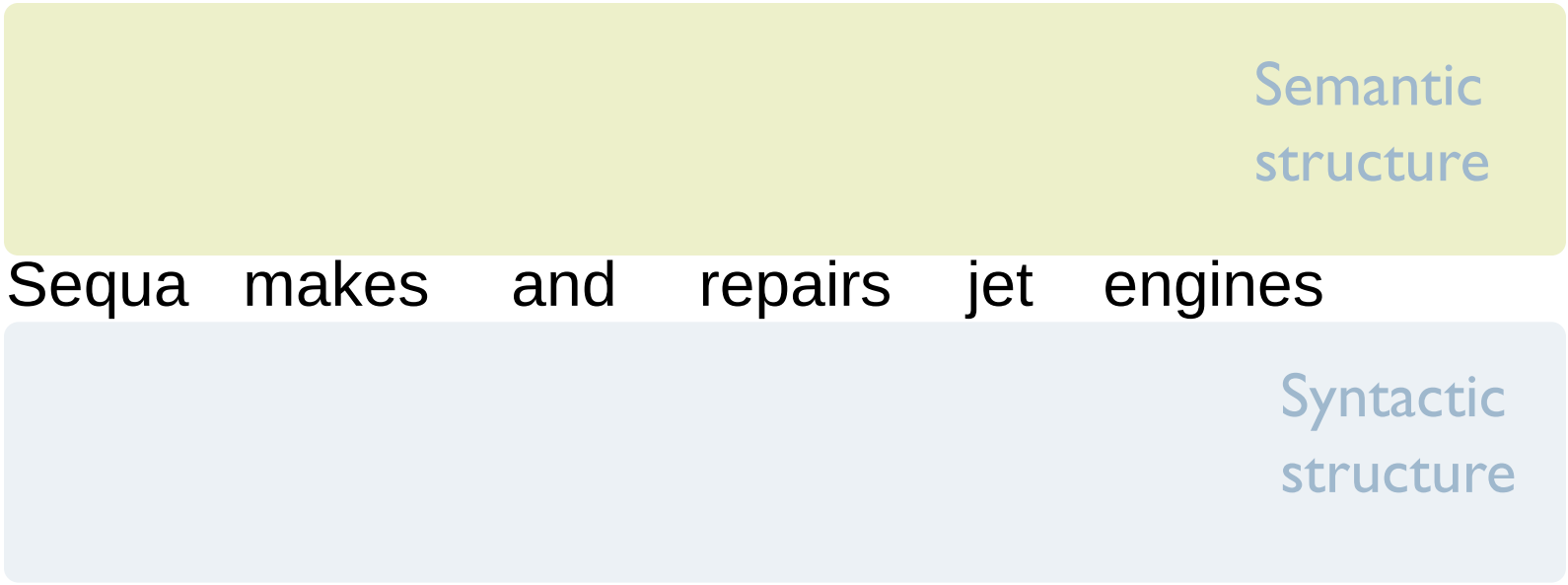
- Motivation / Problem Statement
- Background
  - Dependency parsing
  - Properties of dependency graphs
- Non-Planar Parsing using Swapping
- Synchronous Parsing of Semantic and Syntactic Dependencies
  - Synchronization
  - Statistical Model
- Experiments
- Conclusions and Future Directions

Synchronization [Henderson et al., 2008]

- We define two separate derivations: one for semantics, one for syntax
- Instead of using pipelines we synchronize these two derivations
  - joint learning and joint inference

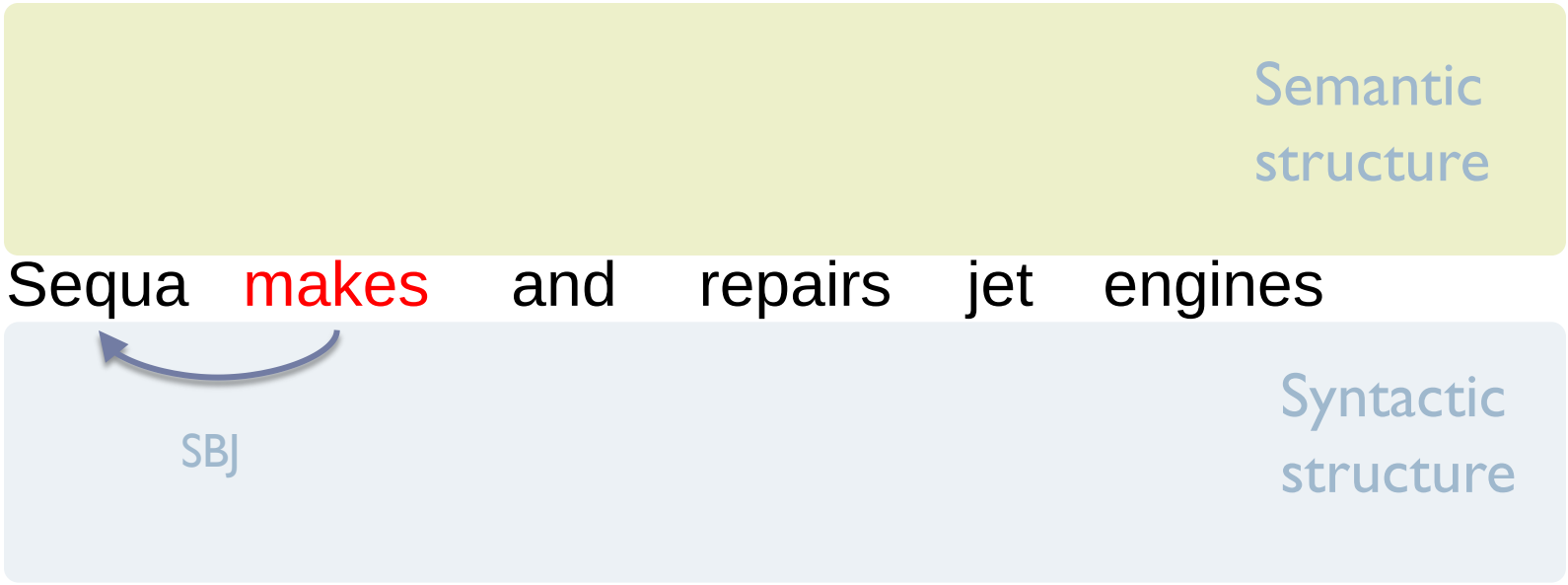
Synchronization

- Example:



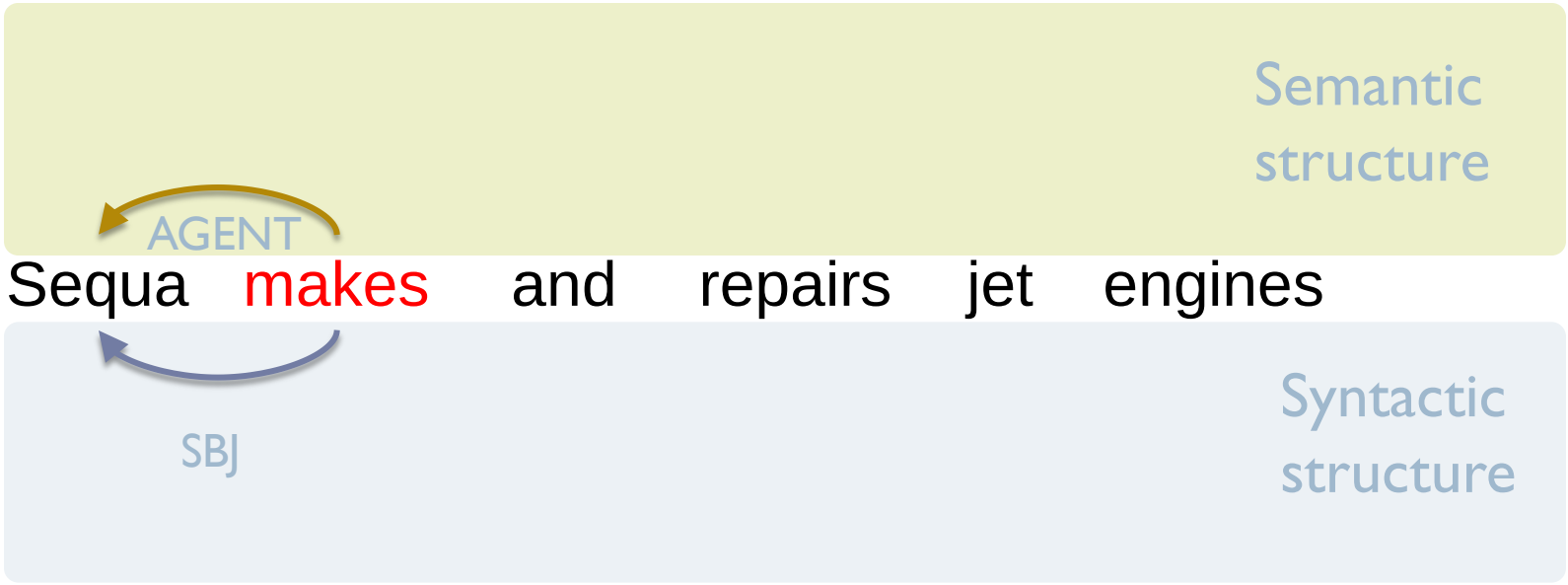
Synchronization

- Example:



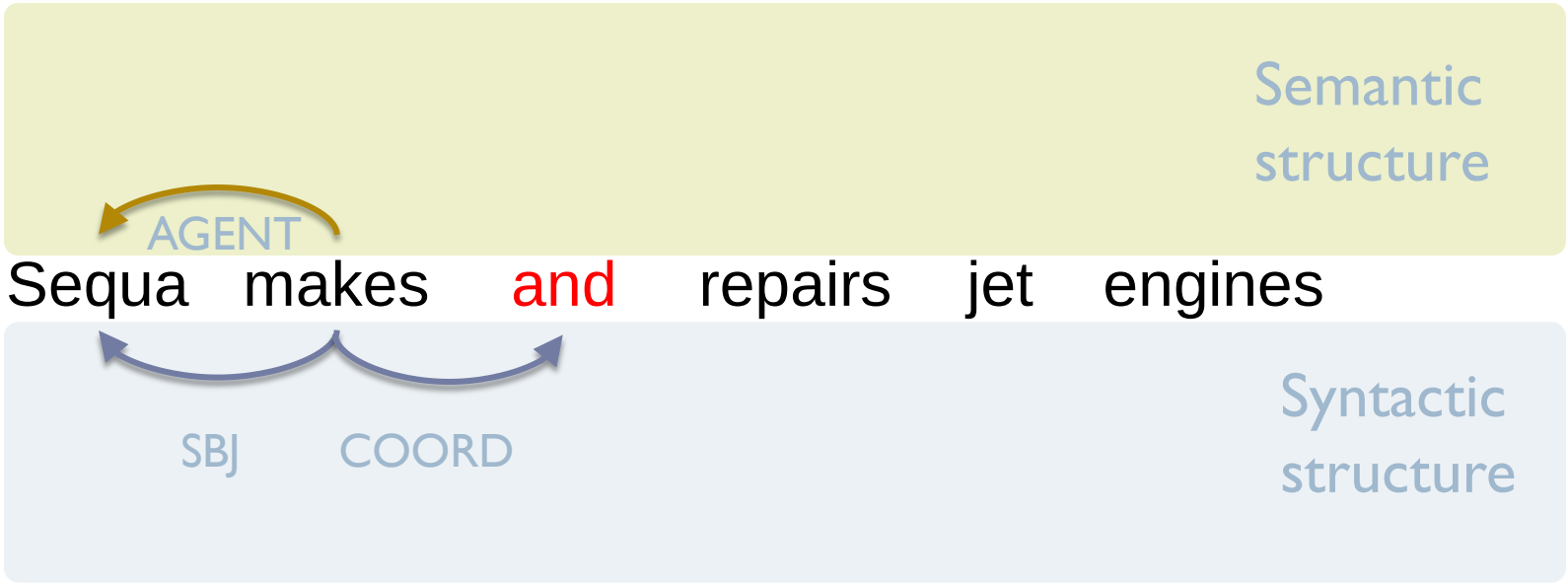
Synchronization

- Example:



Synchronization

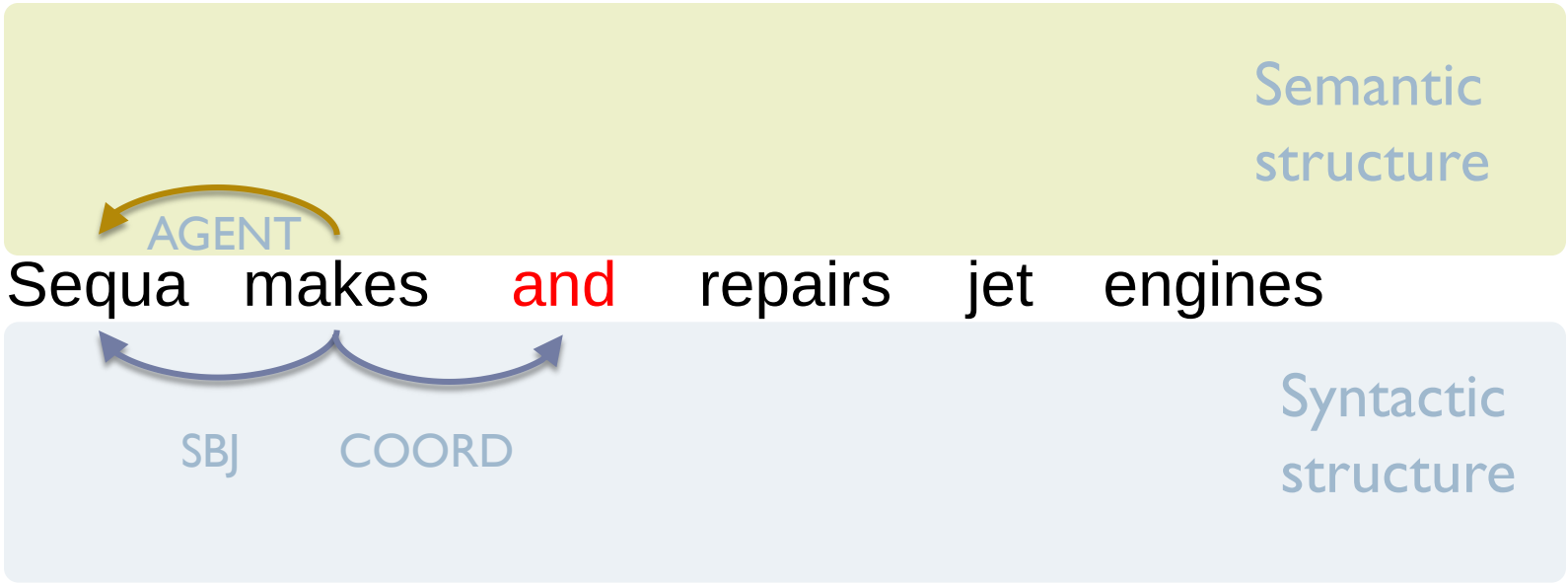
- Example:



$$D^2_{syn}$$

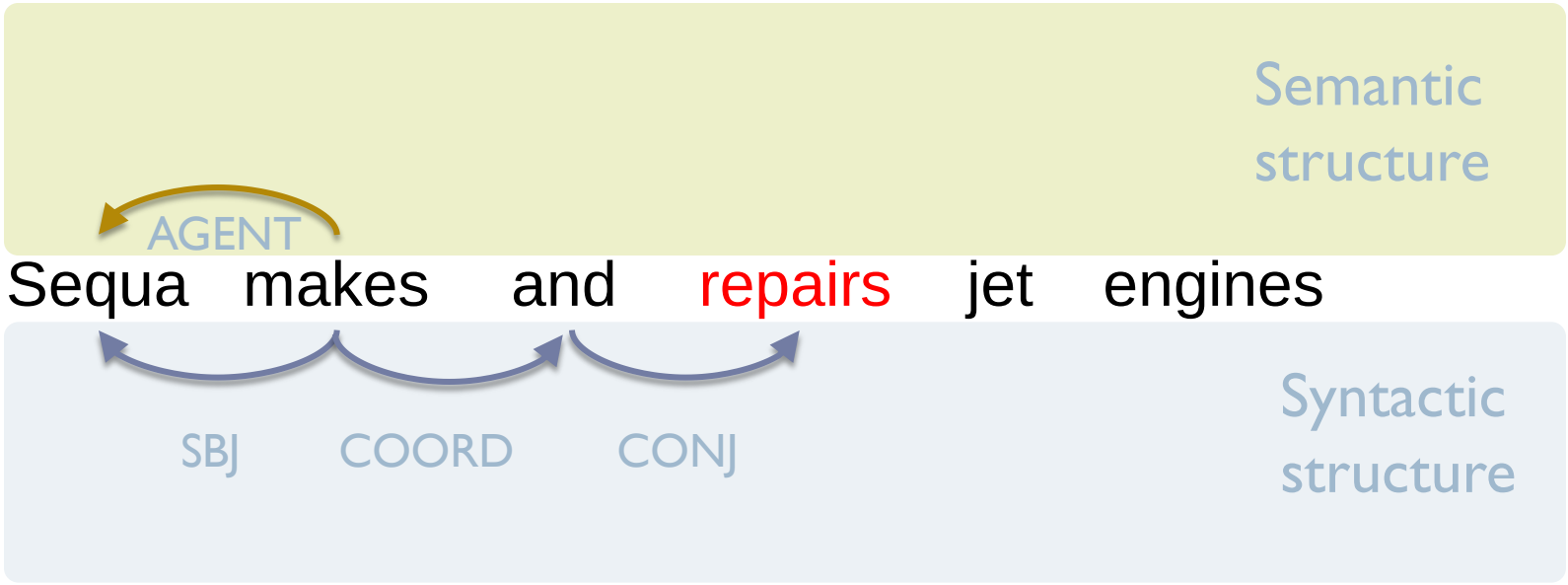
Synchronization

- Example:



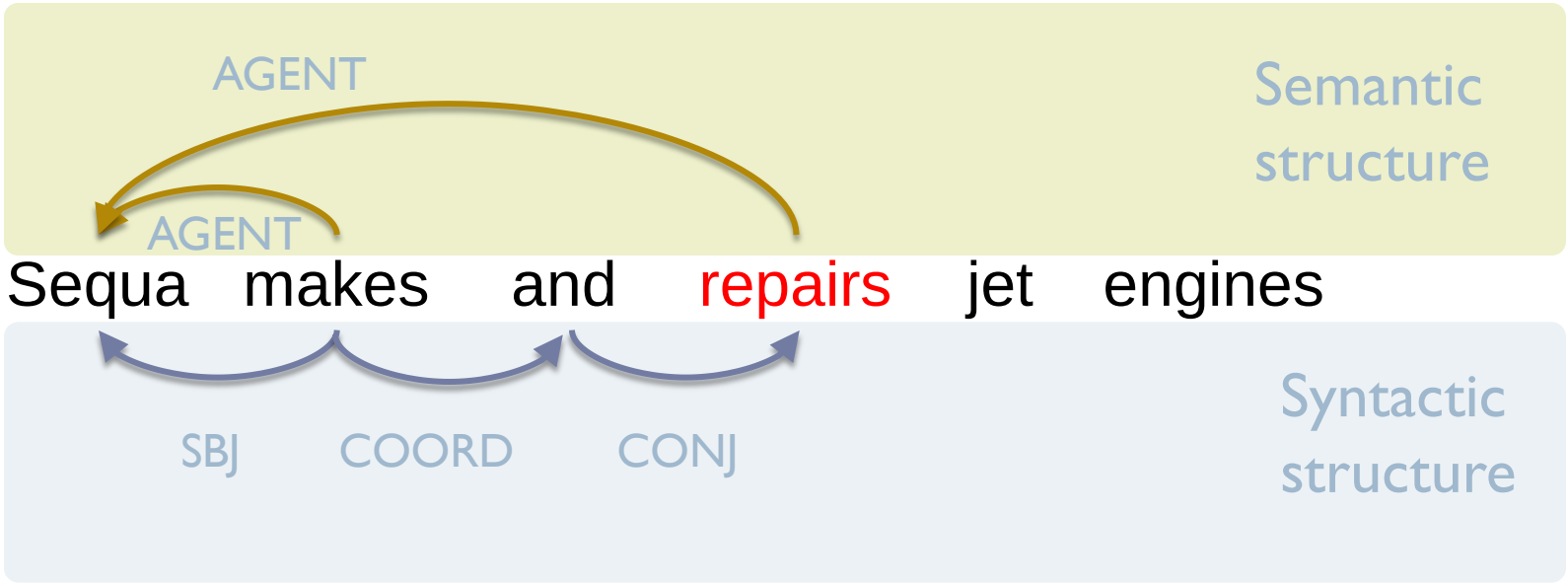
Synchronization

- Example:



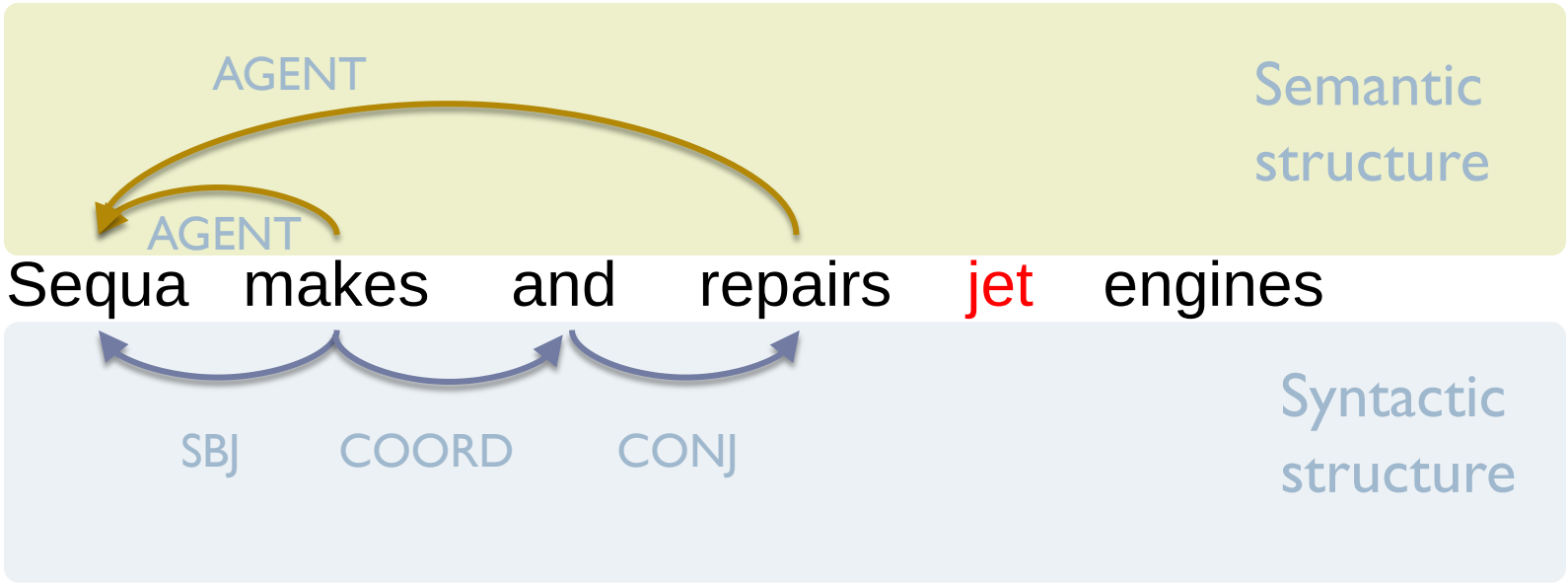
Synchronization

- Example:



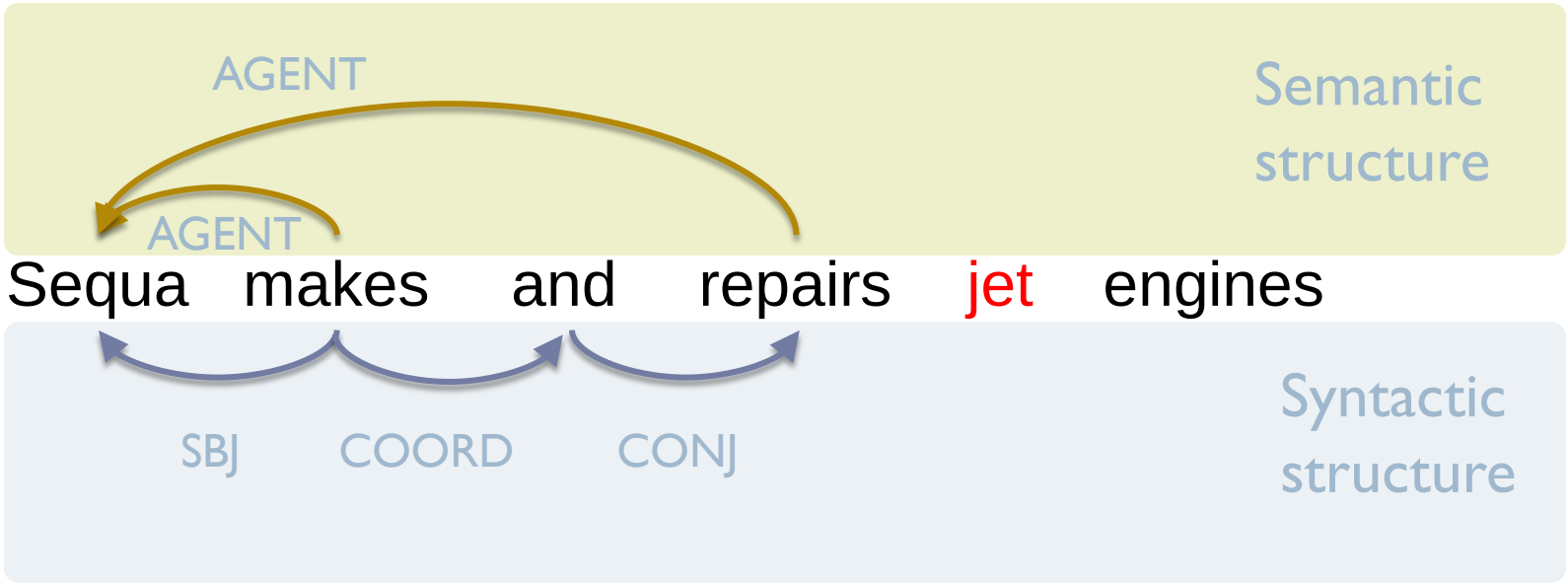
Synchronization

- Example:



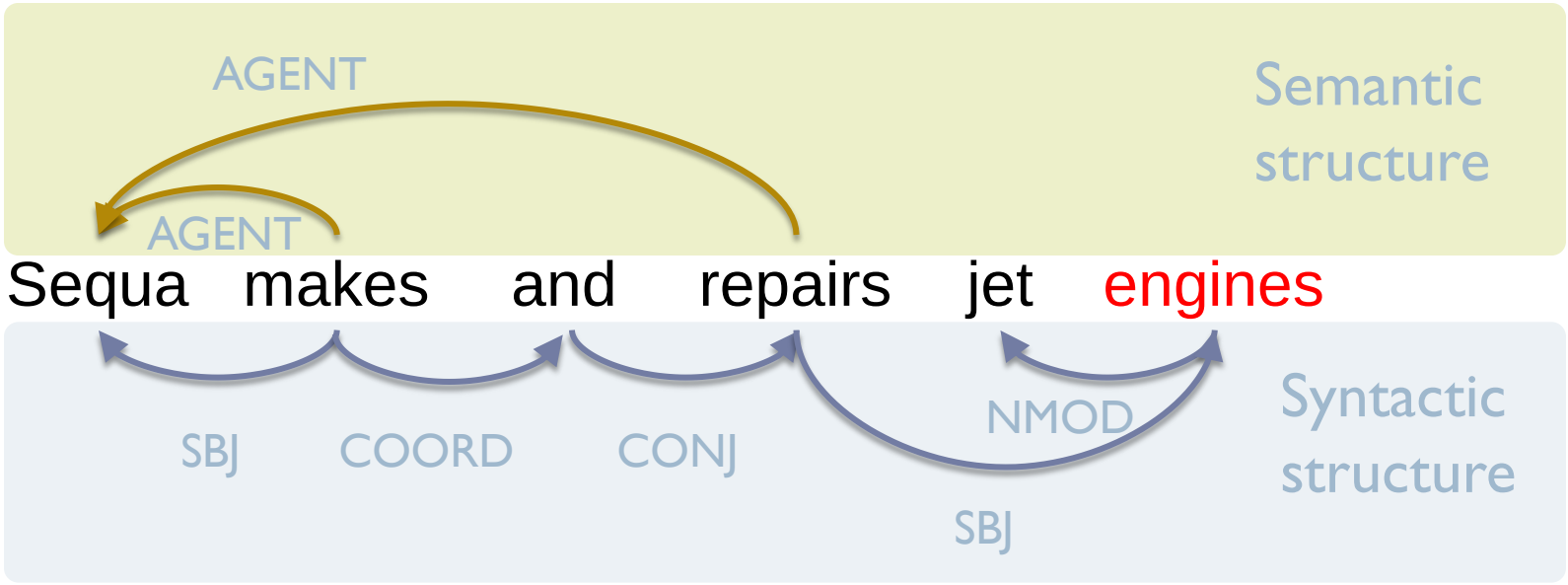
Synchronization

- Example:



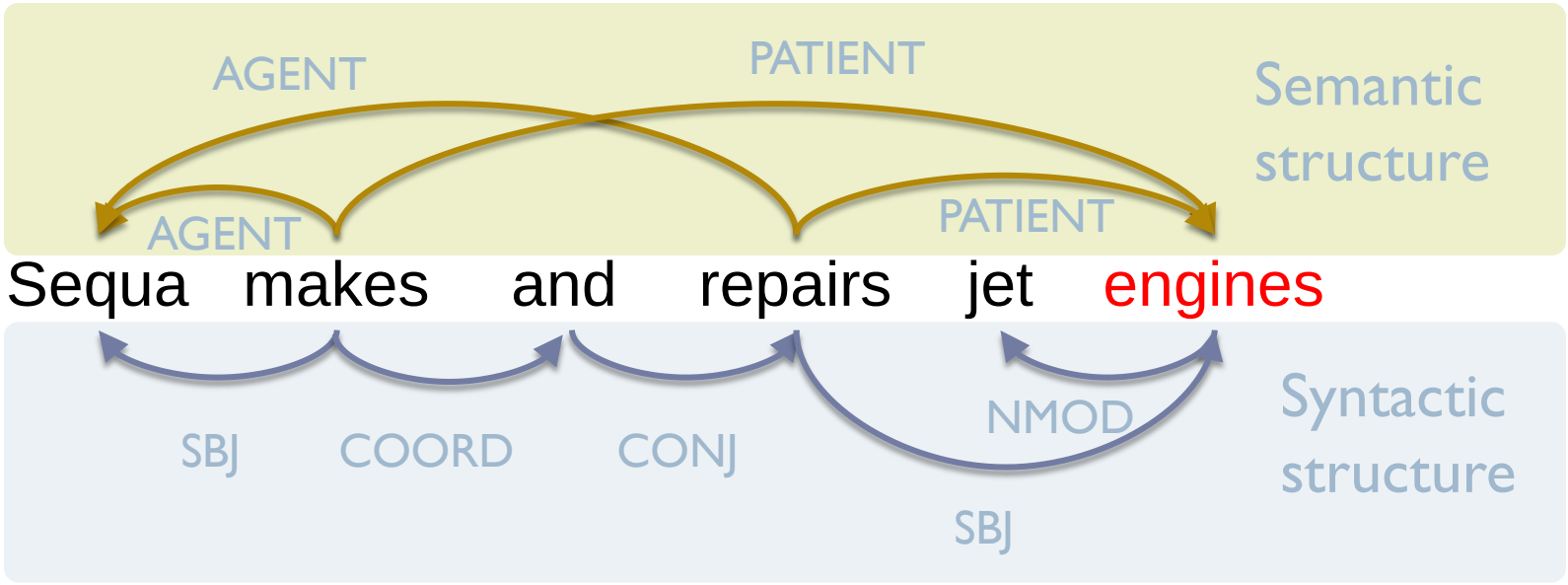
Synchronization

- Example:



Synchronization

- Example:



Statistical Model

- Following *[Henderson, et al.,08]*, the synchronous derivations are modeled with Incremental Sigmoid Belief Networks (ISBNs) *[Titov and Henderson, 07]*
  - previously successfully applied to constituent and dependency syntactic parsing
- Vectors of latent variables are associated with each parsing decision
- Each vector is connected with previous vectors by a pattern of interconnections determined by the previous decisions

Outline

- Motivation / Problem Statement
- Background
  - Dependency parsing
  - Properties of dependency graphs
- Non-Planar Parsing using Swapping
- Synchronous Parsing of Semantic and Syntactic Dependencies
  - Synchronization
  - Statistical Model
- Experiments
- Conclusions and Future Directions

Empirical Evaluation

- CoNLL-2008 Shared Task data *[Surdeanu et al., 08], merged*
  - *dependency transformation of Penn Treebank WSJ (syntax)*
  - *dependency representation of Propbank and Nombank (semantics)*
- Data: 39,279/1,334/2,824 sentences for training/development/testing
- Systems
  - Our models
    - exhaustive order
    - last-resort order
    - planar order (can only process projective parts of derivations)
  - *[Henderson et al., 08] planarisation: a modification of [Nivre and Nilsson, 05] method for semantic graphs (HEAD)*
    - *Crossing links are removed and encoded in labels of remaining arcs*
- Model structure and estimation methods are kept constant

Results on Development Set

Technique	CoNLL Measures			Crossing Paris (Sem)		
	Syn LAS	Sem F1	Aver F1	P	R	F1
Last resort	86.6	76.2	81.5	61.5	25.6	36.1
Exhaustive	86.8	76.0	81.4	59.7	23.5	33.8
HEAD	86.7	73.3	80.1	78.6	2.2	4.2
Planar	85.9	72.8	79.4	undef	0	undef

Results on Development Set

Technique	CoNLL Measures			Crossing Paris (Sem)		
	Syn LAS	Sem F1	Aver F1	P	R	F1
Last resort	86.6	76.2	81.5	61.5	25.6	36.1
Exhaustive	86.8	76.0	81.4	59.7	23.5	33.8
HEAD	86.7	73.3	80.1	78.6	2.2	4.2
Planar	85.9	72.8	79.4	undef	0	undef

Results on Development Set

Technique	CoNLL Measures			Crossing Pairs (Sem)		
	Syn LAS	Sem F1	Aver F1	P	R	F1
Last resort	86.6	76.2	81.5	61.5	25.6	36.1
Exhaustive	86.8	76.0	81.4	59.7	23.5	33.8
HEAD	86.7	73.3	80.1	78.6	2.2	4.2
Planar	85.9	72.8	79.4	undef	0	undef

Results on Development Set

Technique	CoNLL Measures			Crossing Pairs (Sem)		
	Syn LAS	Sem F1	Aver F1	P	R	F1
Last resort	86.6	76.2	81.5	61.5	25.6	36.1
Exhaustive	86.8	76.0	81.4	59.7	23.5	33.8
HEAD						4.2
Planar						undef

The model is generative and, therefore, decisions are not conditioned on future words – a likely reason why no improvement from using exhaustive strategy

Test Set Results

Model	CoNLL Measures			Crossing Pairs (Sem)		
	Syn LAS	Sem F1	Aver F1	P	R	F1
Johanssen	89.3	81.6	85.5	67.0	44.5	53.5
Ciaramita	87.4	78.0	82.7	59.9	34.2	43.5
Che	86.7	78.5	82.7	56.9	32.4	41.3
Zhao	87.7	76.7	82.2	58.5	36.1	44.6
This paper	87.5	76.1	81.8	62.1	29.4	39.9
Henderson+	87.6	73.1	80.5	72.6	1.7	3.3
Lluis	85.8	70.3	78.1	53.8	19.2	28.3

- 3% improvement over the baseline on semantics graphs
- However, does not outperform reranking or ensemble techniques

Test Set Results

Model	CoNLL Measures			Crossing Pairs (Sem)		
	Syn LAS	Sem F1	Aver F1	P	R	F1
Johanssen	89.3	81.6	85.5	67.0	44.5	53.5
Ciaramita	87.4	78.0	82.7	59.9	34.2	43.5
Che	86.7	78.5	82.7	56.9	32.4	41.3
Zhao	87.7	76.7	82.2	58.5	36.1	44.6
This paper	87.5	76.1	81.8	62.1	29.4	39.9
Henderson+	87.6	73.1	80.5	72.6	1.7	3.3
Lluis	85.8	70.3	78.1	53.8	19.2	28.3

- Recent results: 3rd result in CoNLL-2009 Shared task (7 languages)

Conclusions

- Proposed a simple modification to handle semantic graphs with transition-based parsers
  - though not powerful enough to process all the semantic graphs it is able to handle vast majority
  - proposed and analyzed two algorithms for canonical derivation induction
  - theoretically characterized the class of parsable structures
- Showed improvements over previous methods
- Demonstrated state-of-the-art results without ensemble techniques or pipelines
- Future directions: applying this approach to parsing of language with highly non-planar syntactic structures