A Tree Transducer Model for Synchronous Tree-Adjoining Grammars

Andreas Maletti

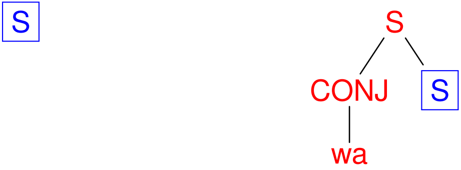
Universitat Rovira i Virgili
Tarragona, Spain

andreas.maletti@urv.cat

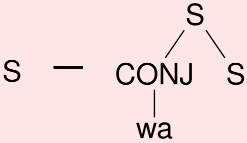
Uppsala, Sweden — July 13, 2010

Synchronous Tree Substitution Grammar

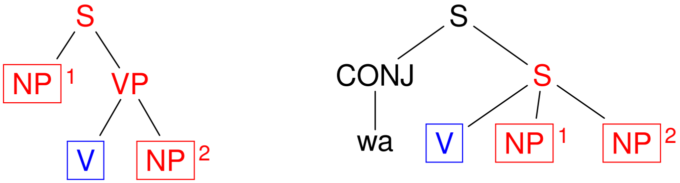
Synchronous Tree Substitution Grammar



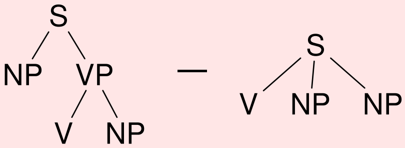
Used rule



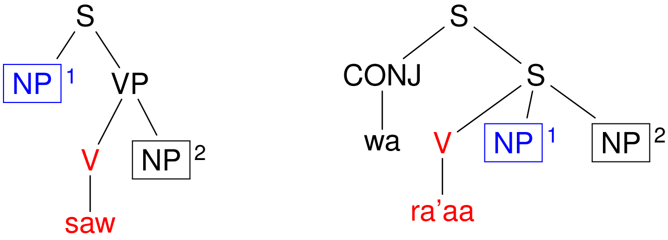
Synchronous Tree Substitution Grammar



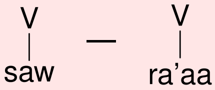
Used rule



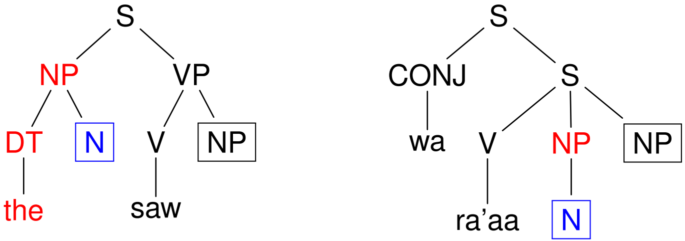
Synchronous Tree Substitution Grammar



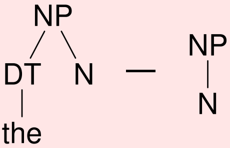
Used rule



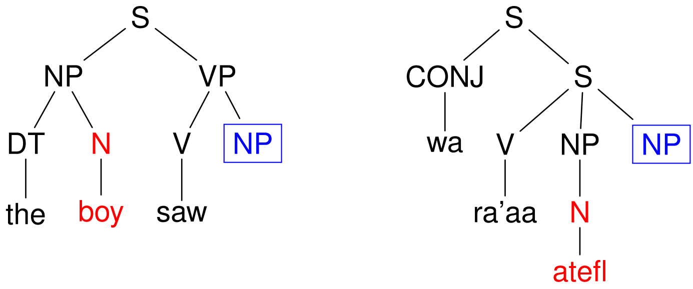
Synchronous Tree Substitution Grammar



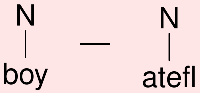
Used rule



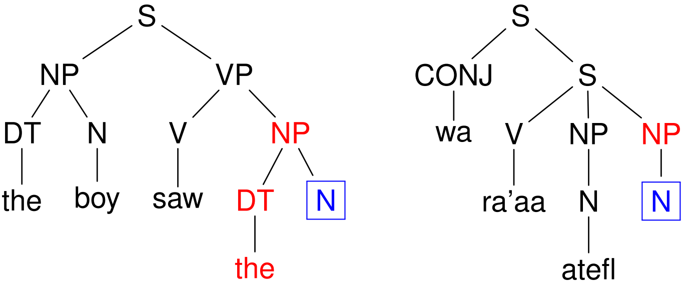
Synchronous Tree Substitution Grammar



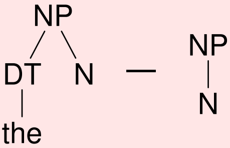
Used rule



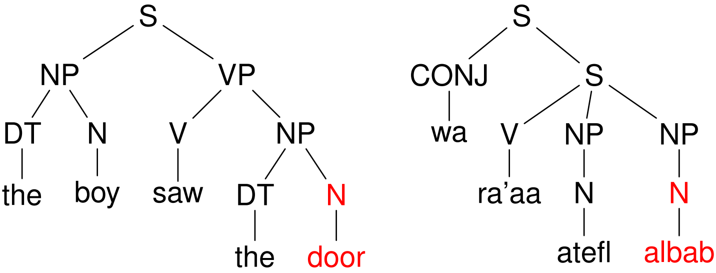
Synchronous Tree Substitution Grammar



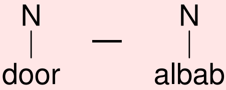
Used rule



Synchronous Tree Substitution Grammar



Used rule



Synchronous Tree Substitution Grammar (cont'd)

Advantages

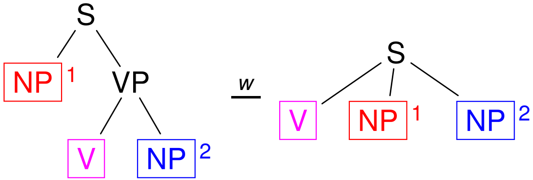
- simple and natural model
- easy to train (from linguistic resources)
- symmetric

Implementation

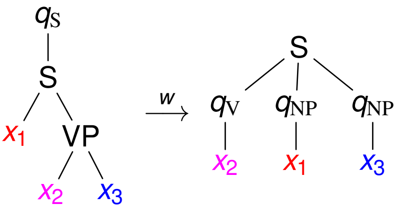
- extended top-down tree transducer in TIBURON [MAY, KNIGHT '06]

Synchronous Tree Substitution Grammar (cont'd)

Synchronous tree substitution grammar rule:

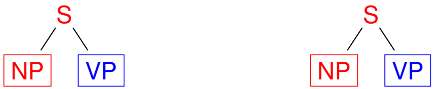


Corresponding extended top-down tree transducer rule:

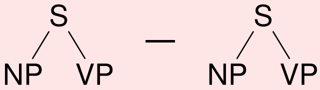


Synchronous Tree-Adjoining Grammar

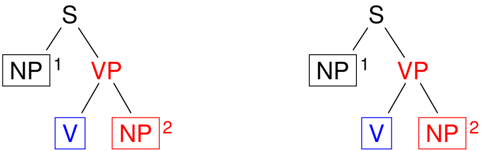
Synchronous Tree-Adjoining Grammar



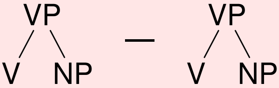
Used substitution rule



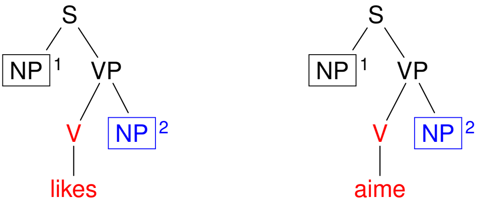
Synchronous Tree-Adjoining Grammar



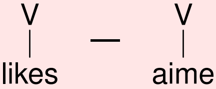
Used substitution rule



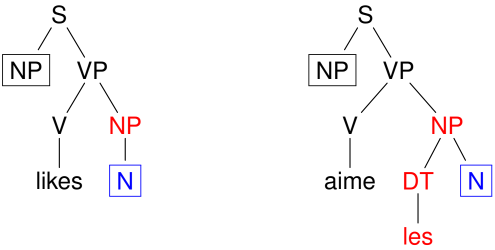
Synchronous Tree-Adjoining Grammar



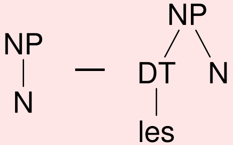
Used substitution rule



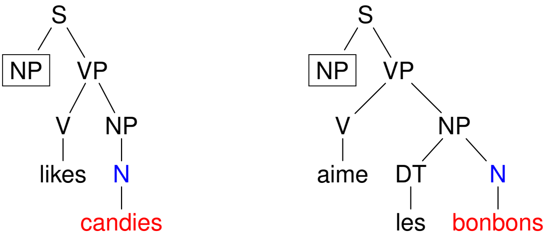
Synchronous Tree-Adjoining Grammar



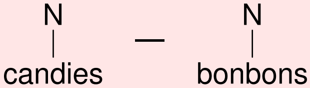
Used substitution rule



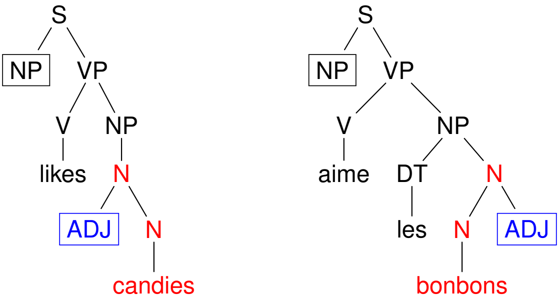
Synchronous Tree-Adjoining Grammar



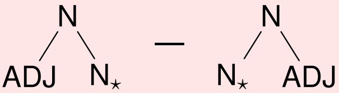
Used substitution rule



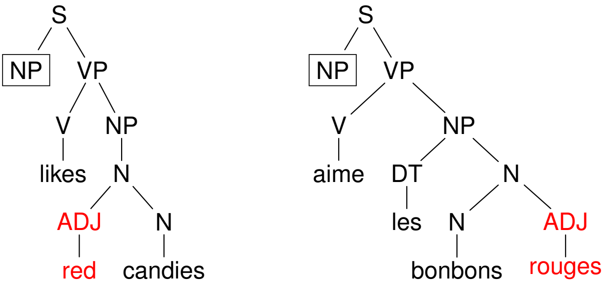
Synchronous Tree-Adjoining Grammar



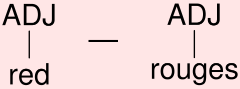
Used adjunction rule



Synchronous Tree-Adjoining Grammar



Used substitution rule



Main Question

Theorem

Every STSG is an STAG.

Question

Are they further related?

Roadmap

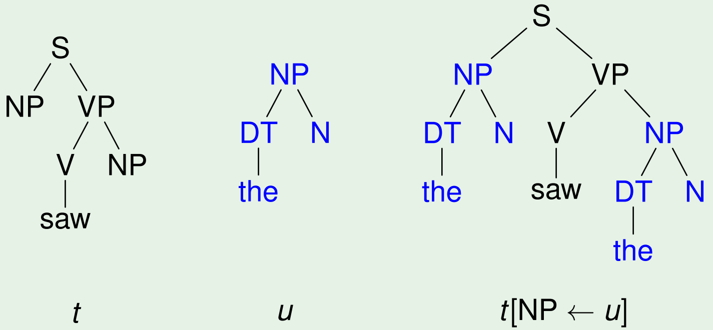
1. Motivation
2. Explicit Substitution
3. Synchronous Tree-Adjoining Grammar
4. Main Result
5. Application

First-Order Substitution

Definition

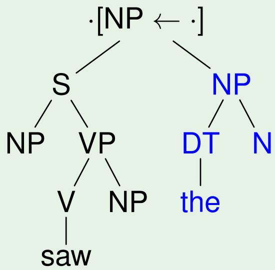
$t[v_1 \leftarrow t_1, \ldots, v_k \leftarrow t_k]$ denotes the result obtained by replacing (in parallel) all occurrences of leaves labelled v_i in t by t_i.

Example



Second-Order Substitution

Example

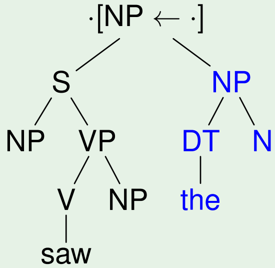


Explicit substitution

- keep an explicit representation of substitutions in tree
- any number of substitutions allowed at any level

Second-Order Substitution

Example



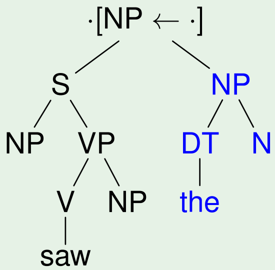
Evaluation

$$\text{eval}(\cdot[x \leftarrow \cdot](t, u)) = \text{eval}(t)[x \leftarrow \text{eval}(u)]$$

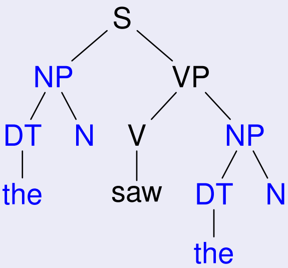
$$\text{eval}(\sigma(t_1, \ldots, t_k)) = \sigma(\text{eval}(t_1), \ldots, \text{eval}(t_k))$$

Second-Order Substitution

Example



Evaluation



Evaluation

$$\mathrm{eval}(\cdot[x \leftarrow \cdot](t,u)) = \mathrm{eval}(t)[x \leftarrow \mathrm{eval}(u)]$$

$$\mathrm{eval}(\sigma(t_1,\ldots,t_k)) = \sigma(\mathrm{eval}(t_1),\ldots,\mathrm{eval}(t_k))$$

Roadmap

1. Motivation
2. Explicit Substitution
3. Synchronous Tree-Adjoining Grammar
4. Main Result
5. Application

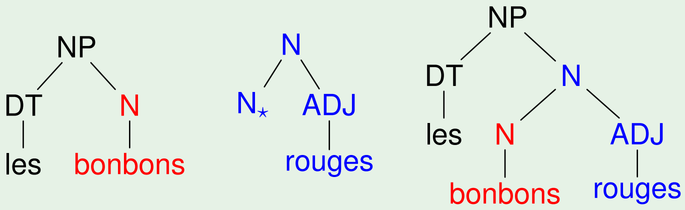
Tree-Adjoining Grammar

Intuition

A TAG has two types of rules:

- substitution rules (as in TSG)
- adjunction rules

Example (Adjunction)



derived tree

auxiliary tree

adjunction

Tree-Adjoining Grammar (cont'd)

Simplifications (see [Shieber '06])

- no substitution rules
- adjunction mandatory (if possible)
- each adjunction spot used at most once
- root nodes of auxiliary trees are never adjunction spots

Definition

A TAG is a finite set of

- derived trees (initial trees) and
- auxiliary trees (those containing a starred node)

Tree-Adjoining Grammar (cont'd)

Simplifications (see [SHIEBER '06])

- no substitution rules
- adjunction mandatory (if possible)
- each adjunction spot used at most once
- root nodes of auxiliary trees are never adjunction spots

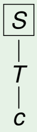
Definition

A TAG is a finite set of

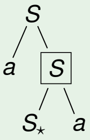
- derived trees (initial trees) and
- auxiliary trees (those containing a starred node)

Tree-Adjoining Grammar (cont'd)

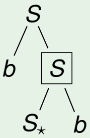
Example



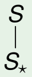
initial tree



auxiliary tree

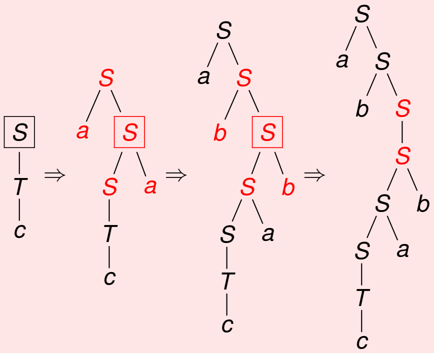


auxiliary tree



auxiliary tree

Derivation

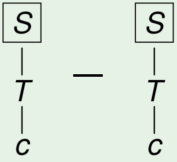


String language

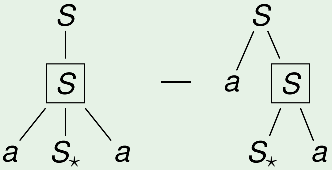
$$\{wcw \mid w \in \{a, b\}^*\}$$

Synchronous Tree-Adjoining Grammar

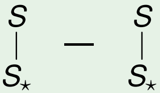
Example



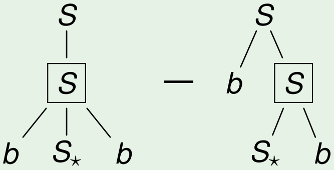
initial tree pair



auxiliary tree pair



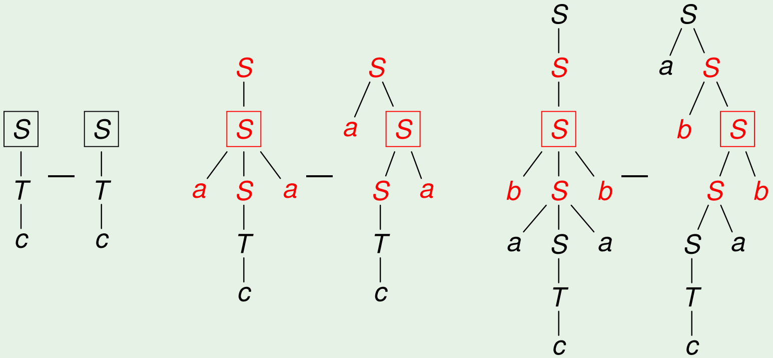
auxiliary tree pair



auxiliary tree pair

Synchronous Tree-Adjoining Grammar (cont'd)

Example



String translation

$$\{(wcw^{\mathrm{R}}, wcw) \mid w \in \{a, b\}^*\}$$

Roadmap

1. Motivation

2. Explicit Substitution

3. Synchronous Tree-Adjoining Grammar

4. Main Result

5. Application

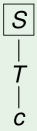
Simulation

Question

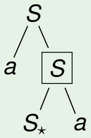
Can we simulate an STAG by some STSG?

Simulation of Adjunction

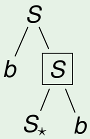
Example (TAG)



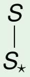
initial tree



auxiliary tree

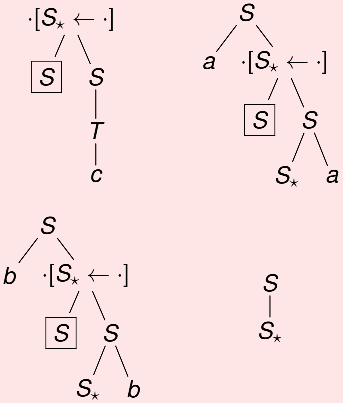


auxiliary tree



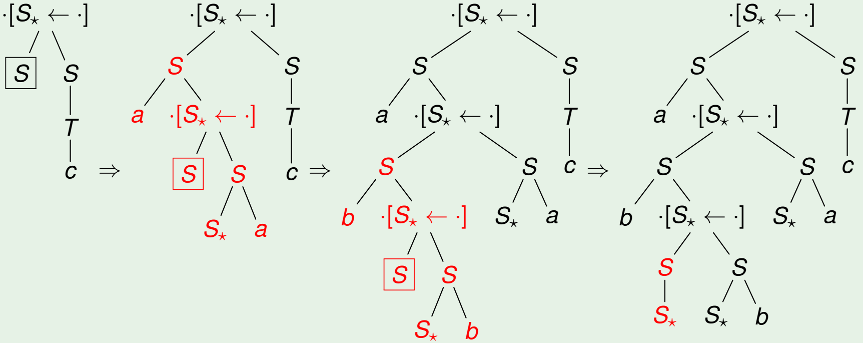
auxiliary tree

Correspondence (TSG)



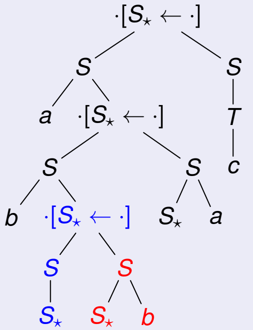
Simulation of Adjunction (cont'd)

Example



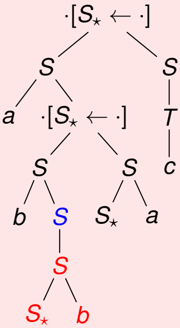
Simulation of Adjunction (cont'd)

TSG result



$\cdot[S_\star \leftarrow \cdot]$
S
S
a
$\cdot[S_\star \leftarrow \cdot]$
T
S
S
c
b
$\cdot[S_\star \leftarrow \cdot]$
$S_\star$
a
S
S
$S_\star$
$S_\star$
b

Evaluation



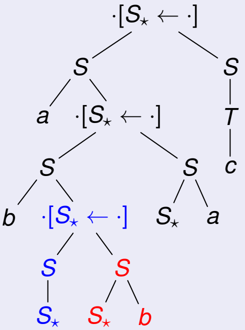
$\cdot[S_\star \leftarrow \cdot]$
S
S
a
$\cdot[S_\star \leftarrow \cdot]$
T
S
S
c
b
S
$S_\star$
a
S
$S_\star$
b

Note

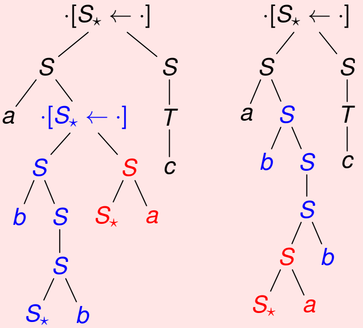
coincides with the result obtained by TAG

Simulation of Adjunction (cont'd)

TSG result



Evaluation

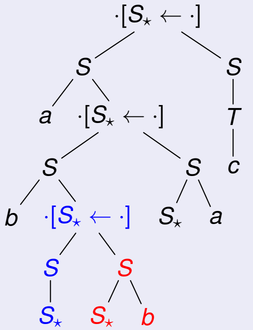


Note

coincides with the result obtained by TAG

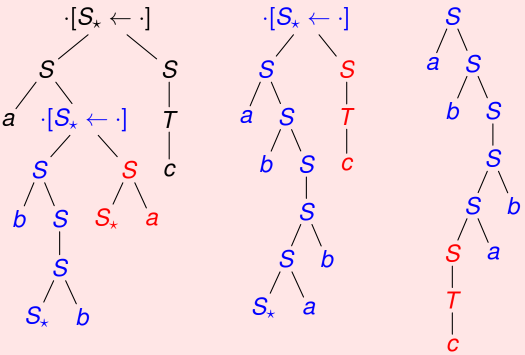
Simulation of Adjunction (cont'd)

TSG result



$\cdot[S_\star \leftarrow \cdot]$
S
S
a
$\cdot[S_\star \leftarrow \cdot]$
T
c
S
S
b
$\cdot[S_\star \leftarrow \cdot]$
$S_\star$
a
S
S
$S_\star$
$S_\star$
b

Evaluation



$\cdot[S_\star \leftarrow \cdot]$
S
S
a
$\cdot[S_\star \leftarrow \cdot]$
T
c
S
S
b
S
$S_\star$
a
S
$S_\star$
b
$\cdot[S_\star \leftarrow \cdot]$
S
S
a
S
T
b
S
c
S
S
b
$S_\star$
a
S
a
S
b
S
S
S
b
S
a
T
c

Note

coincides with the result obtained by TAG

Main Result

Theorem

For every TAG G there exists a TSG G' such that

$$L(G) = \{\mathrm{eval}(t) \mid t \in L(G')\}$$

Main Result

Theorem

For every TAG G there exists a TSG G′ such that

$$L(G) = \{\mathrm{eval}(t) \mid t \in L(G')\}$$

Theorem

For every STAG G there exists a STSG G' such that

$$T(G) = \{(\mathrm{eval}(t), \mathrm{eval}(u)) \mid (t, u) \in T(G')\}$$

Main Result

Theorem

For every STAG G there exists a STSG G' such that

$$T(G) = \{(\text{eval}(t), \text{eval}(u)) \mid (t, u) \in T(G')\}$$

Proof.

□

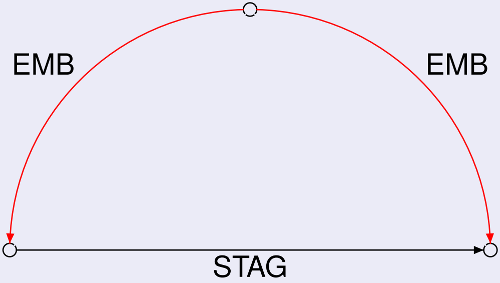
Main Result

Theorem

For every STAG G there exists a STSG G' such that

$$T(G) = \{(\text{eval}(t), \text{eval}(u)) \mid (t, u) \in T(G')\}$$

Proof.



□

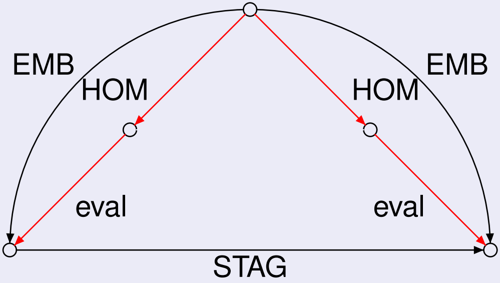
Main Result

Theorem

For every STAG G there exists a STSG G' such that

$$T(G) = \{(\mathrm{eval}(t), \mathrm{eval}(u)) \mid (t, u) \in T(G')\}$$

Proof.



□

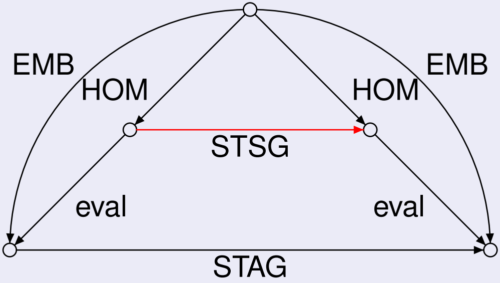
Main Result

Theorem

For every STAG G there exists a STSG G' such that

$$T(G) = \{(\text{eval}(t), \text{eval}(u)) \mid (t, u) \in T(G')\}$$

Proof.



□

Roadmap

1. Motivation
2. Explicit Substitution
3. Synchronous Tree-Adjoining Grammar
4. Main Result
5. Application

Application

Overview

- run an STAG in TIBURON (which can run STSGs)
- translate STSG algorithms to STAGs (factorization, etc.)
- integrate explicit substitution into semantics
- separate “context-free” and “context-sensitive” behavior

References

- ARNOLD, DAUCHET: Morphismes et bimorphismes d'arbres. *Theoret. Comput. Sci.* 20. 1982
- CHIANG, KNIGHT: An introduction to synchronous grammars. Tutorial at *ACL*. 2006
- ENGELFRIET, VOGLER: Macro tree transducers. *J. Comput. Syst. Sci.* 31. 1985
- MAY, KNIGHT: TIBURON — a weighted tree automata toolkit. In *CIAA*, LNCS 4094. 2006
- NEDERHOF: Weighted parsing of trees. In *IWPT*. 2009
- SHIEBER, SCHABES: Synchronous tree-adjoining grammars. *Computational Linguistics* 3. 1990
- SHIEBER: Unifying synchronous tree adjoining grammars and tree transducers via bimorphisms. In *EACL*. 2006

Thank you for your attention!