Why Synchronous Tree Substitution Grammars?

Andreas Maletti

Universitat Rovira i Virgili
Tarragona, Spain

andreas.maletti@urv.cat

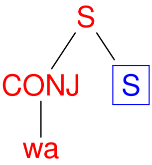
Los Angeles — June 4, 2010

Synchronous Tree Substitution Grammars

S S

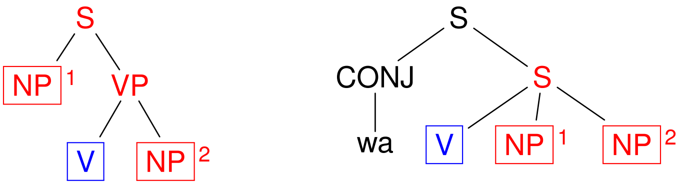
Weight: 1

Synchronous Tree Substitution Grammars



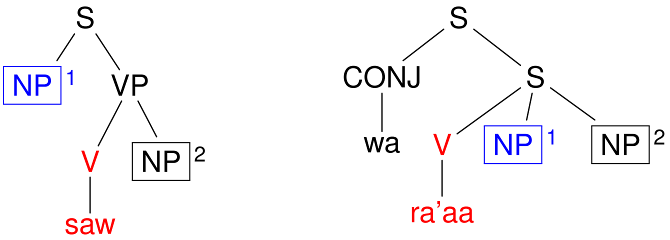
Weight: $1 \cdot 0.5$

Synchronous Tree Substitution Grammars



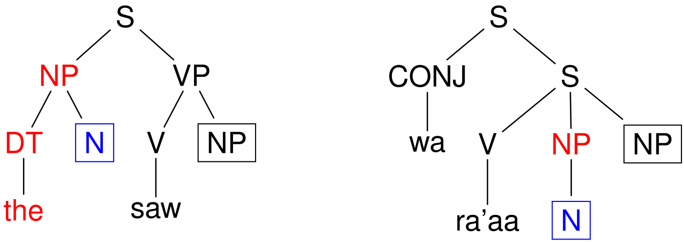
Weight: $1 \cdot 0.5 \cdot 0.25$

Synchronous Tree Substitution Grammars



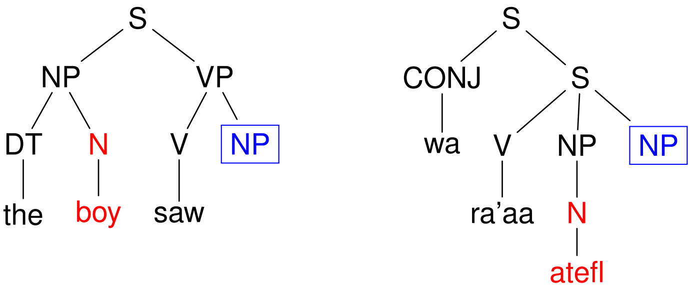
Weight: $1 \cdot 0.5 \cdot 0.25 \cdot 0.03$

Synchronous Tree Substitution Grammars



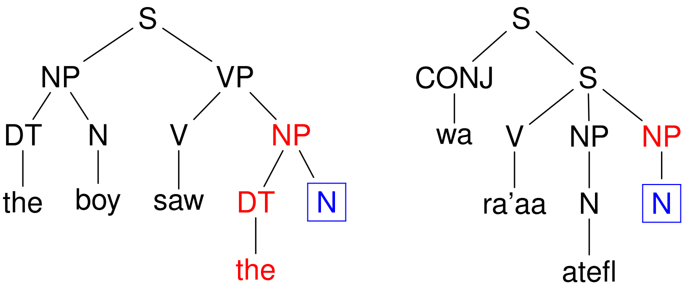
Weight: $1 \cdot 0.5 \cdot 0.25 \cdot 0.03 \cdot 0.25$

Synchronous Tree Substitution Grammars



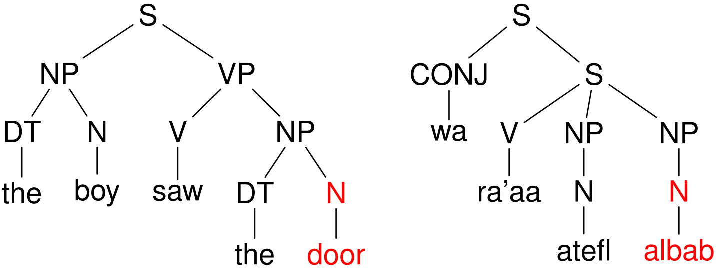
Weight: $1 \cdot 0.5 \cdot 0.25 \cdot 0.03 \cdot 0.25 \cdot 0.1$

Synchronous Tree Substitution Grammars



Weight: $1 \cdot 0.5 \cdot 0.25 \cdot 0.03 \cdot 0.25 \cdot 0.1 \cdot 0.25$

Synchronous Tree Substitution Grammars



Weight: $1 \cdot 0.5 \cdot 0.25 \cdot 0.03 \cdot 0.25 \cdot 0.1 \cdot 0.25 \cdot 0.05$

Synchronous Tree Substitution Grammars (cont'd)

Advantages

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

(Obvious) Disadvantages

- computes joint-probability (→ *generative story*)
- no state behavior (→ *local behavior*)

Implementation

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

Synchronous Tree Substitution Grammars (cont'd)

Advantages

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

(Obvious) Disadvantages

- computes joint-probability ($\rightarrow$ *generative story*)
- no state behavior ($\rightarrow$ *local behavior*)

Implementation

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

Synchronous Tree Substitution Grammars (cont'd)

Advantages

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

(Obvious) Disadvantages

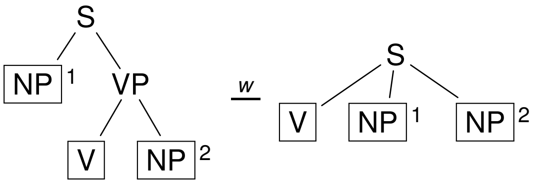
- computes joint-probability ($\rightarrow$ *generative story*)
- no state behavior ($\rightarrow$ *local behavior*)

Implementation

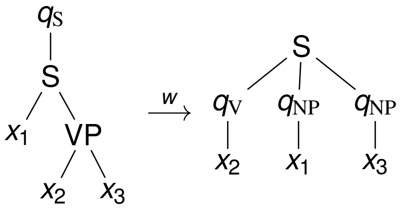
- extended top-down tree transducer in TIBURON [MAY, KNIGHT '06]

Synchronous Tree Substitution Grammars (cont'd)

Synchronous tree substitution grammar rule:



Corresponding extended top-down tree transducer rule:



Extended Top-down Tree Transducer

Advantages

- input-driven model (can easily compute conditional probability)
- state behavior

Disadvantages (also of STSG)

- not binarizable
  [AHO, ULLMAN '72; ZHANG, HUANG, GILDEA, KNIGHT '06]
- inefficient input/output restriction (BAR-HILLEL construction)
  [M., SATTA '10]
- not composable
  [ARNOLD, DAUCHET '82]

Extended Top-down Tree Transducer

Advantages

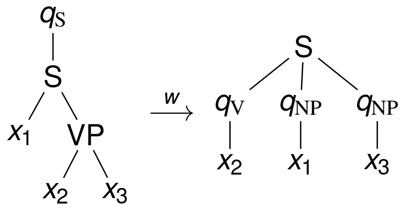
- input-driven model (can easily compute conditional probability)
- state behavior

Disadvantages (also of STSG)

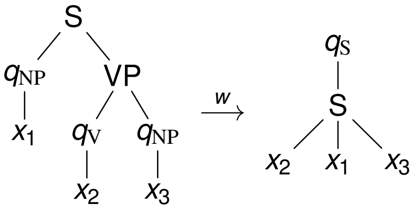
- not binarizable
  [AHO, ULLMAN '72; ZHANG, HUANG, GILDEA, KNIGHT '06]
- inefficient input/output restriction (BAR-HILLEL construction)
  [M., SATTA '10]
- not composable
  [ARNOLD, DAUCHET '82]

Extended Bottom-up Tree Transducer

Top-down tree transducer rule:



Corresponding extended bottom-up tree transducer rule:



Extended Bottom-up Tree Transducer (cont'd)

Theorem

For every STSG we can construct an equivalent extended bottom-up tree transducer in linear time.

Question

Do they have better properties?

Extended Bottom-up Tree Transducer (cont'd)

Theorem

For every STSG we can construct an equivalent extended bottom-up tree transducer in linear time.

Question

Do they have better properties?

Roadmap

Syntax

Definition

Weighted extended multi bottom-up tree transducer (XMBOT) is a system $(Q, \Sigma, \Delta, F, R)$ with

- Q *ranked* alphabet of *states*
- Σ and Δ ranked alphabets of input and output symbols
- $F \subseteq Q_1$ final states
- R finite set of rules $l \xrightarrow{w} r$ with $w \in \mathbb{R}_{\geq 0}$, linear $l \in T_\Sigma(Q(X))$, linear $r \in Q(T_\Delta(X))$ such that $\text{var}(l) = \text{var}(r)$

Syntax (cont'd)

Definition

XMBOT $(Q, \Sigma, \Delta, F, R)$ is proper if $\{l, r\} \not\subseteq Q(X)$ for every $l \xrightarrow{w} r \in R$.

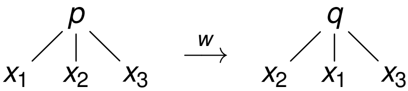
Syntax (cont'd)

Definition

XMBOT $(Q, \Sigma, \Delta, F, R)$ is proper if $\{l, r\} \not\subseteq Q(X)$ for every $l \xrightarrow{w} r \in R$.

Example

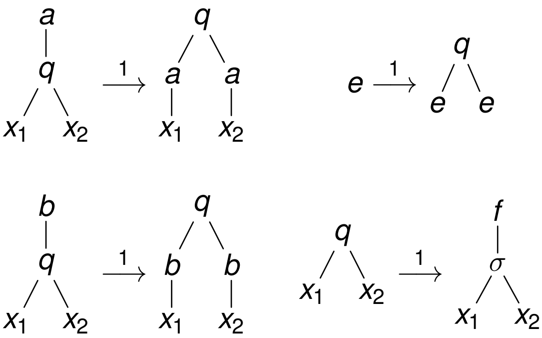
Disallowed rule for properness:



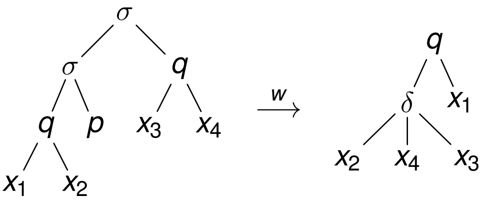
Syntax — An Example

Example

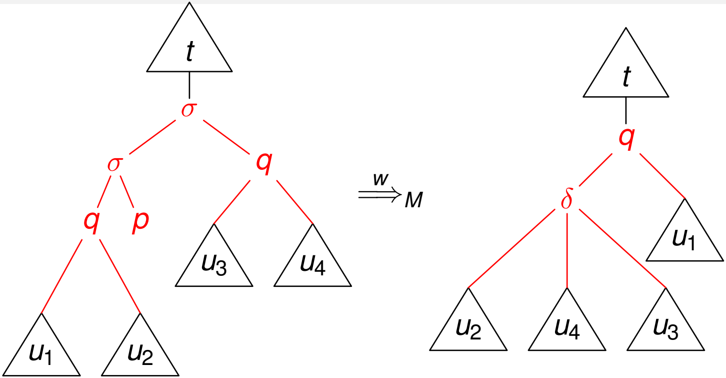
- $Q = \{f^{(1)}, q^{(2)}\}$ and $F = \{f\}$
- $\Sigma = \{a^{(1)}, b^{(1)}, e^{(0)}\}$ and $\Delta = \Sigma \cup \{\sigma^{(2)}\}$
- the following rules



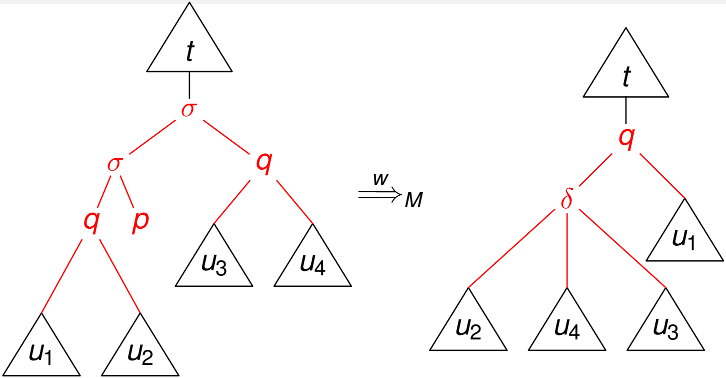
Semantics



Semantics



Semantics



Semantics

- $\mathrm{wt}(\xi_1 \overset{w_1}{\Longrightarrow}_M \cdots \overset{w_{n-1}}{\Longrightarrow}_M \xi_n) = w_1 \cdot \ldots \cdot w_{n-1}$
- $\mathrm{wt}(t, u) = \sum_{q \in F, d \colon t \Longrightarrow^*_M q(u)} \mathrm{wt}(d)$

Semantics — An Example

Used rule

$e \longrightarrow q(e, e)$

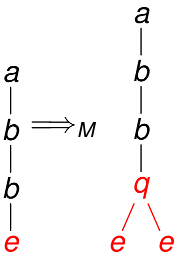
Example

Semantics — An Example

Used rule

$e \longrightarrow q(e, e)$

Example

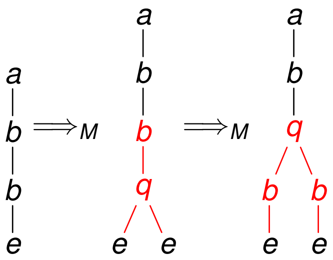


Semantics — An Example

Used rule

$b(q(x_1, x_2)) \longrightarrow q(b(x_1), b(x_2))$

Example

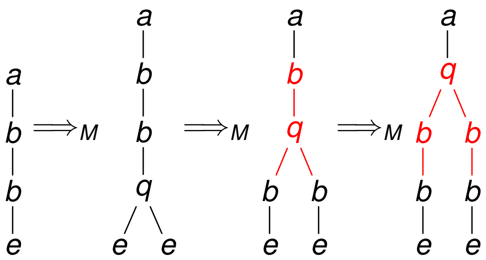


Semantics — An Example

Used rule

$$b(q(x_1, x_2)) \longrightarrow q(b(x_1), b(x_2))$$

Example

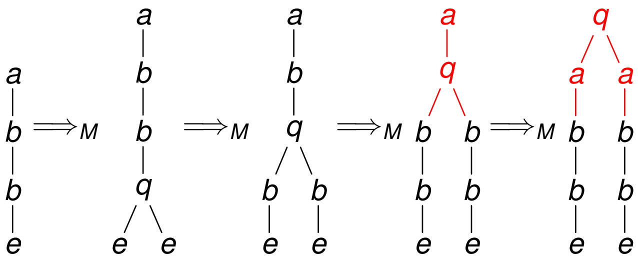


Semantics — An Example

Used rule

$$a(q(x_1, x_2)) \longrightarrow q(a(x_1), a(x_2))$$

Example

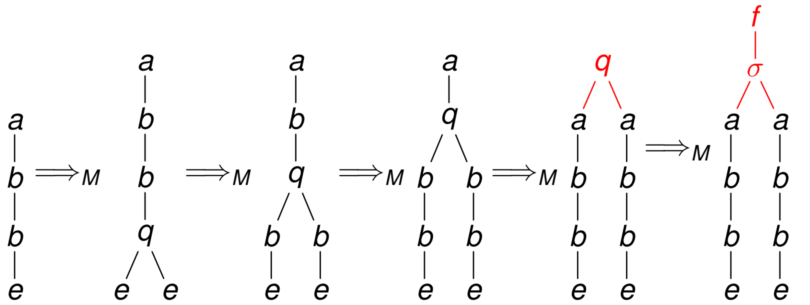


Semantics — An Example

Used rule

$q(x_1, x_2) \longrightarrow f(\sigma(x_1, x_2))$

Example



Roadmap

One-Symbol Normal Form

Definition

XMBOT $(Q, \Sigma, \Delta, F, R)$ is in one-symbol normal form
if exactly one input or output symbol occurs in each rule.

Theorem

For every proper XMBOT there exists an equivalent XMBOT in one-symbol normal form. It can be constructed in linear time.

Corollary

For every proper XMBOT the transition from joint-distribution to conditional-distribution is linear time.

One-Symbol Normal Form

Definition

XMBOT $(Q, \Sigma, \Delta, F, R)$ is in one-symbol normal form
if exactly one input or output symbol occurs in each rule.

Theorem

For every proper XMBOT there exists an equivalent XMBOT in one-symbol normal form. It can be constructed in linear time.

Corollary

For every proper XMBOT the transition from joint-distribution to conditional-distribution is linear time.

One-Symbol Normal Form

Definition

XMBOT $(Q, \Sigma, \Delta, F, R)$ is in one-symbol normal form
if exactly one input or output symbol occurs in each rule.

Theorem

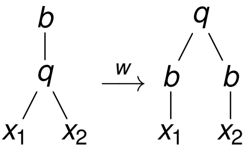
For every proper XMBOT there exists an equivalent XMBOT in one-symbol normal form. It can be constructed in linear time.

Corollary

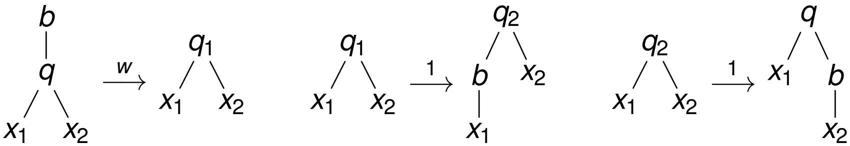
For every proper XMBOT the transition from joint-distribution to conditional-distribution is linear time.

One-Symbol Normal Form (cont'd)

Rule not in one-symbol normal form:



Replacement rules for this rule:



Binarization

Definition

An XMBOT is fully binarized if each rule contains at most 3 states. ($\leq$ 2 in each left-hand side)

Theorem

Every proper XMBOT can be fully binarized in linear time.

Proof.

First binarize the trees in the rules and then transform into one-symbol normal form. □

Binarization

Definition

An XMBOT is fully binarized if each rule contains at most 3 states.
($\leq$ 2 in each left-hand side)

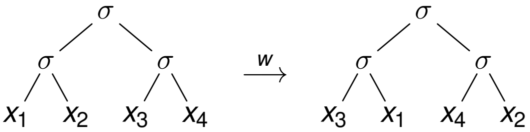
Theorem

Every proper XMBOT can be fully binarized in linear time.

Proof.

First binarize the trees in the rules and then transform into one-symbol normal form. □

Binarization (cont'd)



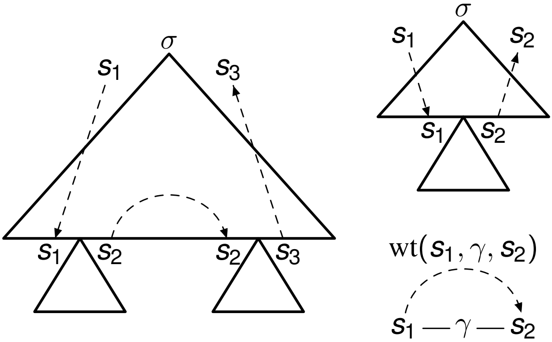
Comparison

In general, STSG cannot be binarized, but people try . . .
[Zhang, Huang, Gildea, Knight '06; DeNero, Pauls, Klein '09]

BAR-HILLEL Construction

Definition

The input product of a weighted tree transformation $\tau\colon T_\Sigma \times T_\Delta \to S$ with a power series $\varphi\colon \Sigma^* \to S$ is $\tau'(s, t) = \tau(s, t) \cdot \varphi(\mathrm{yd}(s))$.



BAR-HILLEL Construction (cont'd)

Theorem

The input product of an XMBOT M with a WSA S can be computed in time $O(|M| \cdot |S|^3)$.

Note

The output product of an XMBOT M with a WSA S can be computed in time $O(|M| \cdot |S|^{2\,\mathrm{rk}(M)+2})$.

Comparison

The input/output product of an STSG M with a WSA S can be computed in time $O(|M| \cdot |S|^{2\,\mathrm{rk}(M)+5})$.
[M., SATTA '10]

BAR-HILLEL Construction (cont'd)

Theorem

The input product of an XMBOT M with a WSA S can be computed in time $O(|M| \cdot |S|^3)$.

Note

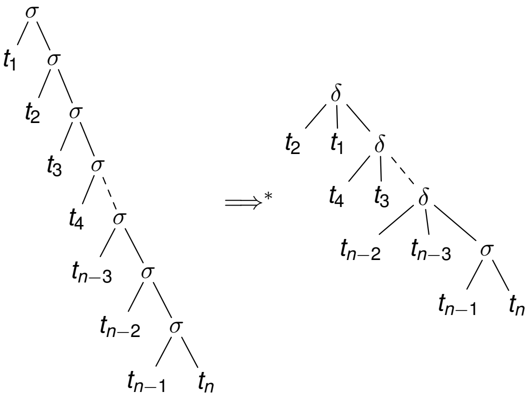
The output product of an XMBOT M with a WSA S can be computed in time $O(|M| \cdot |S|^{2\,\mathrm{rk}(M)+2})$.

Comparison

The input/output product of an STSG M with a WSA S can be computed in time $O(|M| \cdot |S|^{2\,\mathrm{rk}(M)+5})$.
[M., SATTA '10]

Roadmap

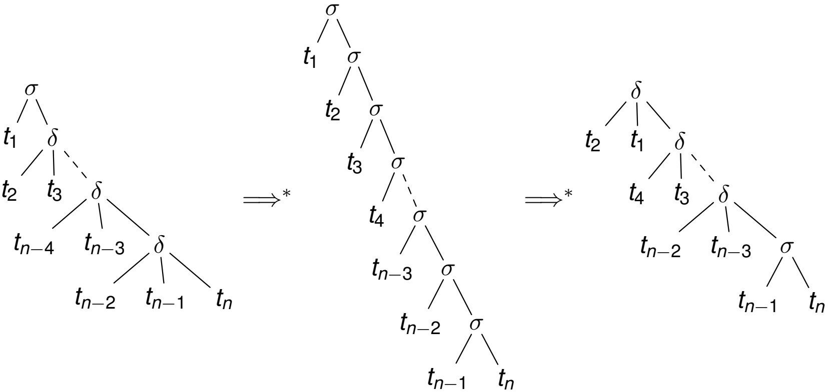
Composition of STSG



Conclusion

STSGs are not composable!

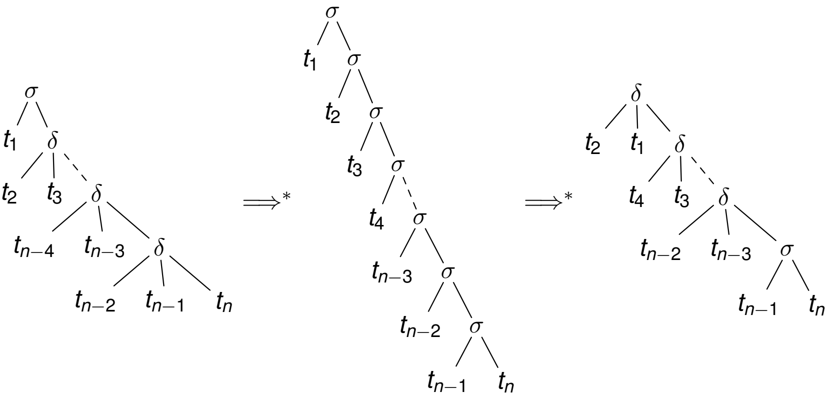
Composition of STSG



Conclusion

STSGs are not composable!

Composition of STSG



Conclusion

STSGs are not composable!

Composition Construction

Definition

for XMBOT $M = (Q, \Sigma, \Gamma, F, R)$ and $N = (Q', \Gamma, \Delta, G, P)$ construct

$$M \,;\, N = (Q(Q'), \Sigma, \Delta, F(G), R')$$

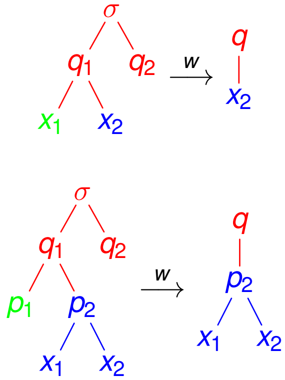
with three types of rules:

1. input-consuming rules constructed from input-consuming rules of R (with their weight)
2. epsilon rules constructed from epsilon-rules of P
3. epsilon rules constructed from an epsilon rule of R followed by an input consuming rule of P (product of the weights)

Composition construction (cont'd)

Example

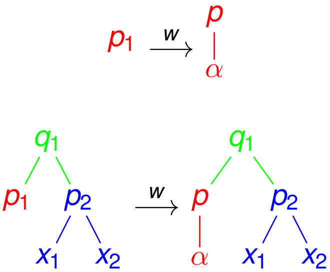
Input consuming rule of R and resulting rule:



Composition construction (cont'd)

Example

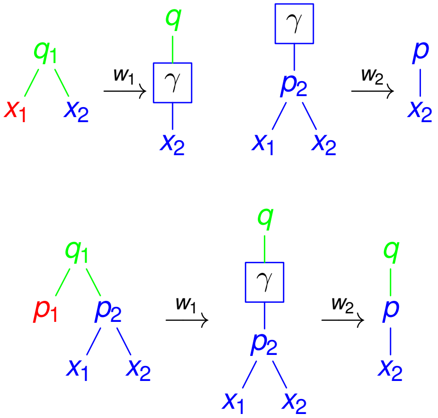
Epsilon rule of P and resulting rule:



Composition construction (cont'd)

Example

Epsilon rule of R and input consuming of P and resulting rule:



Composition construction (cont'd)

Note

The constructed XMBOT might be non-proper.

Theorem

For all proper XMBOTs M and N such that

- *M has no cyclic input epsilon rules or*
- *N has no cyclic output epsilon rules,*

then there exists a proper XMBOT that computes the composition of the transformations computed by M and N.

Composition construction (cont'd)

Note

The constructed XMBOT might be non-proper.

Theorem

For all proper XMBOTs M and N such that

- *M has no cyclic input epsilon rules or*
- *N has no cyclic output epsilon rules,*

then there exists a proper XMBOT that computes the composition of the transformations computed by M and N.

Summary

Algorithm \ Device	STSG	XMBOT
Binarization	✗	$O(\lvert M\rvert)$
Input product	$O(\lvert M\rvert \cdot \lvert S\rvert^{2\,\mathrm{rk}(M)+5})$	$O(\lvert M\rvert \cdot \lvert S\rvert^{3})$
Output product	$O(\lvert M\rvert \cdot \lvert S\rvert^{2\,\mathrm{rk}(M)+5})$	$O(\lvert M\rvert \cdot \lvert S\rvert^{2\,\mathrm{rk}(M)+2})$
Composition	✗	$O(\lvert M_1\rvert \cdot \lvert M_2\rvert^{\mathrm{rk}(M_1)+1})$

Summary

Algorithm \ Device	STSG	XMBOT
Binarization	✗	$O(\|M\|)$
Input product	$O(\|M\| \cdot \|S\|^{2\,\mathrm{rk}(M)+5})$	$O(\|M\| \cdot \|S\|^3)$
Output product	$O(\|M\| \cdot \|S\|^{2\,\mathrm{rk}(M)+5})$	$O(\|M\| \cdot \|S\|^{2\,\mathrm{rk}(M)+2})$
Composition	✗	$O(\|M_1\| \cdot \|M_2\|^{\mathrm{rk}(M_1)+1})$

Summary

Algorithm \ Device	STSG	XMBOT
Binarization	✗	$O(\lvert M\rvert)$
Input product	$O(\lvert M\rvert \cdot \lvert S\rvert^{2\,\mathrm{rk}(M)+5})$	$O(\lvert M\rvert \cdot \lvert S\rvert^{3})$
Output product	$O(\lvert M\rvert \cdot \lvert S\rvert^{2\,\mathrm{rk}(M)+5})$	$O(\lvert M\rvert \cdot \lvert S\rvert^{2\,\mathrm{rk}(M)+2})$
Composition	✗	$O(\lvert M_1\rvert \cdot \lvert M_2\rvert^{\mathrm{rk}(M_1)+1})$

Summary

Algorithm \ Device	STSG	XMBOT
Binarization	✗	$O(\lvert M\rvert)$
Input product	$O(\lvert M\rvert \cdot \lvert S\rvert^{2\,\mathrm{rk}(M)+5})$	$O(\lvert M\rvert \cdot \lvert S\rvert^{3})$
Output product	$O(\lvert M\rvert \cdot \lvert S\rvert^{2\,\mathrm{rk}(M)+5})$	$O(\lvert M\rvert \cdot \lvert S\rvert^{2\,\mathrm{rk}(M)+2})$
Composition	✗	$O(\lvert M_1\rvert \cdot \lvert M_2\rvert^{\mathrm{rk}(M_1)+1})$

Summary

Algorithm \ Device	STSG	XMBOT
Binarization	✗	$O(\lvert M\rvert)$
Input product	$O(\lvert M\rvert \cdot \lvert S\rvert^{2\,\mathrm{rk}(M)+5})$	$O(\lvert M\rvert \cdot \lvert S\rvert^{3})$
Output product	$O(\lvert M\rvert \cdot \lvert S\rvert^{2\,\mathrm{rk}(M)+5})$	$O(\lvert M\rvert \cdot \lvert S\rvert^{2\,\mathrm{rk}(M)+2})$
Composition	✗	$O(\lvert M_1\rvert \cdot \lvert M_2\rvert^{\mathrm{rk}(M_1)+1})$
Reversal	$O(\lvert M\rvert)$	✗
Pres. of REC	✓	✗

Summary

Algorithm \ Device	STSG	XMBOT
Binarization	✗	$O(\lvert M\rvert)$
Input product	$O(\lvert M\rvert \cdot \lvert S\rvert^{2\,\mathrm{rk}(M)+5})$	$O(\lvert M\rvert \cdot \lvert S\rvert^{3})$
Output product	$O(\lvert M\rvert \cdot \lvert S\rvert^{2\,\mathrm{rk}(M)+5})$	$O(\lvert M\rvert \cdot \lvert S\rvert^{2\,\mathrm{rk}(M)+2})$
Composition	✗	$O(\lvert M_1\rvert \cdot \lvert M_2\rvert^{\mathrm{rk}(M_1)+1})$
Reversal	$O(\lvert M\rvert)$	✗
Pres. of REC	✓	✗

References (1/2)

- AHO, ULLMAN: *The theory of parsing, translation, and compiling.* Prentice Hall. 1972
- ARNOLD, DAUCHET: Morphismes et bimorphismes d'arbres. *Theoret. Comput. Sci.* 20(1):33–93, 1982
- CHIANG, KNIGHT: An introduction to synchronous grammars. Tutorial at *ACL*. 2006
- DENERO, PAULS, KLEIN: Asynchronous binarization for synchronous grammars. In *ACL*, p. 141–144, 2009
- ENGELFRIET: Bottom-up and top-down tree transformations — a comparison. *Math. Systems Theory* 9(3), 1975
- ENGELFRIET, LILIN, MALETTI: Extended multi bottom-up tree transducers — composition and decomposition. *Acta Inf.* 46(8):561–590, 2009
- GÉCSEG, STEINBY: *Tree Automata*. Akadémiai Kiadó, Budapest 1984

References (2/2)

- LILIN: *Une généralisation des transducteurs d'états finis d'arbres: les S-transducteurs.* Université de Lille. 1978
- LILIN: Propriétés de clôture d'une extension de transducteurs d'arbres déterministes. In *CAAP*, LNCS 112, p. 280–289, 1981
- MALETTI, SATTA: Parsing and translation algorithms based on weighted extended tree transducers. In *ATANLP* 2010
- MAY, KNIGHT: TIBURON — a weighted tree automata toolkit. In *CIAA*, LNCS 4094, p. 102–113, 2006
- ZHANG, HUANG, GILDEA, KNIGHT: Synchronous binarization for machine translation. In *NAACL-HLT*, p. 256–263, 2006

Thank you for your attention!