Syntax-Directed Translations and Quasi-Alphabetic Tree Bimorphisms

Magnus Steinby[1] and Cătălin Ionuţ Tîrnăucă[2]

[1] Department of Mathematics, University of Turku
FIN-20014 Turku, Finland
steinby@utu.fi

[2] Research Group on Mathematical Linguistics, Rovira i Virgili University
Pl. Imperial Tàrraco 1, Tarragona 43005, Spain
catalinionut.tirnauca@estudiants.urv.es

18th of July, 2007

Introduction (I)

- **Finite-state recognizers** and **transducers** have attractive properties and a well-developed theory, and are used with considerable success in natural language processing (e.g., speech and text recognition).
- They cannot perform some of the **syntax-sensitive transformations** and **reorderings** of parts of sentences frequently encountered in translations from one natural language to another.
- researchers switch attention to devices that can model trees and tree transformations:
  - **Tree transducers**;
  - **Synchronous grammars**: Syntax-Directed Translation Schema.
- In a SDTS, parse trees for the input string and the output string are implicitly produced in parallel.

Introduction (I)

- **Finite-state recognizers** and **transducers** have attractive properties and a well-developed theory, and are used with considerable success in natural language processing (e.g., speech and text recognition).
- They cannot perform some of the **syntax-sensitive transformations** and **reorderings** of parts of sentences frequently encountered in translations from one natural language to another.
- researchers switch attention to devices that can model trees and tree transformations:
  - **Tree transducers**;
  - **Synchronous grammars**: Syntax-Directed Translation Schema.
- In a SDTS, parse trees for the input string and the output string are implicitly produced in parallel.

Introduction (I)

- **Finite-state recognizers** and **transducers** have attractive properties and a well-developed theory, and are used with considerable success in natural language processing (e.g., speech and text recognition).
- They cannot perform some of the **syntax-sensitive transformations** and **reorderings** of parts of sentences frequently encountered in translations from one natural language to another.
- researchers switch attention to devices that can model trees and tree transformations:
  - **Tree transducers**;
  - **Synchronous grammars**: **S**yntax-**D**irected **T**ranslation **S**chema.
- In a SDTS, parse trees for the input string and the output string are implicitly produced in parallel.

Introduction (I)

- **Finite-state recognizers** and **transducers** have attractive properties and a well-developed theory, and are used with considerable success in natural language processing (e.g., speech and text recognition).
- They cannot perform some of the **syntax-sensitive transformations** and **reorderings** of parts of sentences frequently encountered in translations from one natural language to another.
- researchers switch attention to devices that can model trees and tree transformations:
  - **Tree transducers**;
  - **Synchronous grammars**: **S**yntax-**D**irected **T**ranslation **S**chema.
- In a SDTS, parse trees for the input string and the output string are implicitly produced in parallel.

Introduction (I)

- **Finite-state recognizers** and **transducers** have attractive properties and a well-developed theory, and are used with considerable success in natural language processing (e.g., speech and text recognition).
- They cannot perform some of the **syntax-sensitive transformations** and **reorderings** of parts of sentences frequently encountered in translations from one natural language to another.
- researchers switch attention to devices that can model trees and tree transformations:
  - **Tree transducers**;
  - **Synchronous grammars**: **S**yntax-**D**irected **T**ranslation **S**chema.
- In a SDTS, parse trees for the input string and the output string are implicitly produced in parallel.

Introduction (I)

- **Finite-state recognizers** and **transducers** have attractive properties and a well-developed theory, and are used with considerable success in natural language processing (e.g., speech and text recognition).
- They cannot perform some of the **syntax-sensitive transformations** and **reorderings** of parts of sentences frequently encountered in translations from one natural language to another.
- researchers switch attention to devices that can model trees and tree transformations:
  - **Tree transducers**;
  - **Synchronous grammars**: **S**yntax-**D**irected **T**ranslation **S**chema.
- In a SDTS, parse trees for the input string and the output string are implicitly produced in parallel.

Introduction (II)

- In natural language processing, **closure under composition** and **preservation of recognizability** are important features of translations [Knight & Graehl 2005, Knight & Hopkins & Graehl 2006].
- The tree transformations defined by many common types of tree transducers are neither closed under composition nor do they preserve the recognizability of tree languages.
- The mathematical framework of synchronous grammars appears less suitable for such operations $\Longrightarrow$ look for alternative formalisms.
- **Tree bimorphisms** provide an elegant algebraic tool for describing various classes of tree transformastions and proving their properties. By taking yields of the input and output trees, tree bimorphisms are transformed into **string-to-string translating devices**.
- How about describing synchronous-rewriting systems with the help of tree bimorphisms? [Shieber 2004]

Introduction (II)

- In natural language processing, **closure under composition** and **preservation of recognizability** are important features of translations [Knight & Graehl 2005, Knight & Hopkins & Graehl 2006].
- The tree transformations defined by many common types of tree transducers are neither closed under composition nor do they preserve the recognizability of tree languages.
- The mathematical framework of synchronous grammars appears less suitable for such operations $\Longrightarrow$ look for alternative formalisms.
- **Tree bimorphisms** provide an elegant algebraic tool for describing various classes of tree transformastions and proving their properties. By taking yields of the input and output trees, tree bimorphisms are transformed into **string-to-string translating devices**.
- How about describing synchronous-rewriting systems with the help of tree bimorphisms? [Shieber 2004]

Introduction (II)

- In natural language processing, **closure under composition** and **preservation of recognizability** are important features of translations [Knight & Graehl 2005, Knight & Hopkins & Graehl 2006].
- The tree transformations defined by many common types of tree transducers are neither closed under composition nor do they preserve the recognizability of tree languages.
- The mathematical framework of synchronous grammars appears less suitable for such operations $\Longrightarrow$ look for alternative formalisms.
- **Tree bimorphisms** provide an elegant algebraic tool for describing various classes of tree transformastions and proving their properties. By taking yields of the input and output trees, tree bimorphisms are transformed into **string-to-string translating devices**.
- How about describing synchronous-rewriting systems with the help of tree bimorphisms? [Shieber 2004]

Introduction (II)

- In natural language processing, **closure under composition** and **preservation of recognizability** are important features of translations [Knight & Graehl 2005, Knight & Hopkins & Graehl 2006].
- The tree transformations defined by many common types of tree transducers are neither closed under composition nor do they preserve the recognizability of tree languages.
- The mathematical framework of synchronous grammars appears less suitable for such operations $\Longrightarrow$ look for alternative formalisms.
- **Tree bimorphisms** provide an elegant algebraic tool for describing various classes of tree transformastions and proving their properties. By taking yields of the input and output trees, tree bimorphisms are transformed into **string-to-string translating devices**.
- How about describing synchronous-rewriting systems with the help of tree bimorphisms? [Shieber 2004]

Introduction (II)

- In natural language processing, **closure under composition** and **preservation of recognizability** are important features of translations [Knight & Graehl 2005, Knight & Hopkins & Graehl 2006].
- The tree transformations defined by many common types of tree transducers are neither closed under composition nor do they preserve the recognizability of tree languages.
- The mathematical framework of synchronous grammars appears less suitable for such operations $\Longrightarrow$ look for alternative formalisms.
- **Tree bimorphisms** provide an elegant algebraic tool for describing various classes of tree transformastions and proving their properties. By taking yields of the input and output trees, tree bimorphisms are transformed into **string-to-string translating devices**.
- How about describing synchronous-rewriting systems with the help of tree bimorphisms? [Shieber 2004]

Syntax-Directed Translation Schema (I)

What Is an SDTS?

It is a CF grammar with translation elements attached to each production. Whenever a production is used in the derivation of an input sentence, the associated translation element generates a part of the output sentence. [Aho & Ullman 1972].

Definition (Syntax-Directed Translation Schema)

A **SDTS** is a device **T**$= (N, X, Y, P, S)$, where:

- N is a finite set of **nonterminal symbols**,
- X is a finite **input alphabet**,
- Y is a finite **output alphabet**,
- $S \in N$ is the start symbol, and
- P is a finite set of **productions** of the form:

$$p = A \to u_1 A_1 u_2 \ldots u_m A_m u_{m+1}, v_1 A_{\pi(1)} v_2 \ldots v_m A_{\pi(m)} v_{m+1} \quad (*)$$

where $m \geq 0$, $A, A_1, \ldots, A_m \in N$, π is a permutation of $[m]$, and for every $i \in [m+1]$, $u_i = x_1^i \ldots x_{k_i}^i \in X^*$ $(k_i \geq 0)$ and $v_i = y_1^i \ldots y_{l_i}^i \in Y^*$ $(l_i \geq 0)$.

Syntax-Directed Translation Schema (I)

What Is an SDTS?

It is a CF grammar with translation elements attached to each production. Whenever a production is used in the derivation of an input sentence, the associated translation element generates a part of the output sentence. [Aho & Ullman 1972].

Definition (Syntax-Directed Translation Schema)

A **SDTS** is a device $\mathbf{T} = (N, X, Y, P, S)$, where:

- N is a finite set of **nonterminal symbols**,
- X is a finite **input alphabet**,
- Y is a finite **output alphabet**,
- $S \in N$ is the **start symbol**, and
- P is a finite set of **productions** of the form:

$$p = A \to u_1 A_1 u_2 \dots u_m A_m u_{m+1};\ v_1 A_{\sigma(1)} v_2 \dots v_m A_{\sigma(m)} v_{m+1}, \qquad (*)$$

where $m \geq 0$, $A, A_1, \dots, A_m \in N$, σ is a permutation of $[m]$, and for every $i \in [m+1]$, $u_i = x_i^1 \dots x_i^{k_i} \in X^*$ $(k_i \geq 0)$ and $v_i = y_i^1 \dots y_i^{l_i} \in Y^*$ $(l_i \geq 0)$.

Syntax-Directed Translation Schema (I)

What Is an SDTS?

It is a CF grammar with translation elements attached to each production. Whenever a production is used in the derivation of an input sentence, the associated translation element generates a part of the output sentence. [Aho & Ullman 1972].

Definition (Syntax-Directed Translation Schema)

A **SDTS** is a device $\mathbf{T} = (N, X, Y, P, S)$, where:

- N is a finite set of **nonterminal symbols**,
- X is a finite **input alphabet**,
- Y is a finite **output alphabet**,
- $S \in N$ is the **start symbol**, and
- P is a finite set of **productions** of the form:

$$p = A \to u_1 A_1 u_2 \ldots u_m A_m u_{m+1};\, v_1 A_{\sigma(1)} v_2 \ldots v_m A_{\sigma(m)} v_{m+1}, \qquad (*)$$

where $m \geq 0$, $A, A_1, \ldots, A_m \in N$, σ is a permutation of $[m]$, and for every $i \in [m+1]$, $u_i = x_i^1 \ldots x_i^{k_i} \in X^*$ ($k_i \geq 0$) and $v_i = y_i^1 \ldots y_i^{l_i} \in Y^*$ ($l_i \geq 0$).

Syntax-Directed Translation Schema (I)

What Is an SDTS?

It is a CF grammar with translation elements attached to each production. Whenever a production is used in the derivation of an input sentence, the associated translation element generates a part of the output sentence. [Aho & Ullman 1972].

Definition (Syntax-Directed Translation Schema)

A **SDTS** is a device **T**$= (N, X, Y, P, S)$, where:

- N is a finite set of **nonterminal symbols**,
- X is a finite **input alphabet**,
- Y is a finite **output alphabet**,
- $S \in N$ is the **start symbol**, and
- P is a finite set of **productions** of the form:

$$p = A \rightarrow u_1 A_{\mathbf{1}} u_2 \ldots u_m A_{\mathbf{m}} u_{m+1} ; v_1 A_{\sigma(\mathbf{1})} v_2 \ldots v_m A_{\sigma(\mathbf{m})} v_{m+1}, \qquad (*)$$

where $m \geq 0$, $A, A_1, \ldots, A_m \in N$, σ is a permutation of $[m]$, and for every $i \in [m+1]$, $u_i = x_i^1 \ldots x_i^{k_i} \in X^*$ ($k_i \geq 0$) and $v_i = y_i^1 \ldots y_i^{l_i} \in Y^*$ ($l_i \geq 0$).

Syntax-Directed Translation Schema (I)

What Is an SDTS?

It is a CF grammar with translation elements attached to each production. Whenever a production is used in the derivation of an input sentence, the associated translation element generates a part of the output sentence. [Aho & Ullman 1972].

Definition (Syntax-Directed Translation Schema)

A **SDTS** is a device **T**$= (N, X, Y, P, S)$, where:

- N is a finite set of **nonterminal symbols**,
- X is a finite **input alphabet**,
- Y is a finite **output alphabet**,
- $S \in N$ is the **start symbol**, and
- P is a finite set of **productions** of the form:

$$p = A \to u_1 A_{\mathbf{1}} u_2 \ldots u_m A_{\mathbf{m}} u_{m+1}; v_1 A_{\sigma(\mathbf{1})} v_2 \ldots v_m A_{\sigma(\mathbf{m})} v_{m+1}, \qquad (*)$$

where $m \geq 0$, $A, A_1, \ldots, A_m \in N$, σ is a permutation of $[m]$, and for every $i \in [m+1]$, $u_i = x_i^1 \ldots x_i^{k_i} \in X^*$ $(k_i \geq 0)$ and $v_i = y_i^1 \ldots y_i^{l_i} \in Y^*$ $(l_i \geq 0)$.

Syntax-Directed Translation Schema (I)

What Is an SDTS?

It is a CF grammar with translation elements attached to each production. Whenever a production is used in the derivation of an input sentence, the associated translation element generates a part of the output sentence. [Aho & Ullman 1972].

Definition (Syntax-Directed Translation Schema)

A **SDTS** is a device **T**$= (N, X, Y, P, S)$, where:

- N is a finite set of **nonterminal symbols**,
- X is a finite **input alphabet**,
- Y is a finite **output alphabet**,
- $S \in N$ is the **start symbol**, and
- P is a finite set of **productions** of the form:

$$p = A \rightarrow u_1 A_{\mathbf{1}} u_2 \ldots u_m A_{\mathbf{m}} u_{m+1}; v_1 A_{\sigma(\mathbf{1})} v_2 \ldots v_m A_{\sigma(\mathbf{m})} v_{m+1}, \qquad (*)$$

where $m \geq 0$, $A, A_1, \ldots, A_m \in N$, σ is a permutation of $[m]$, and for every $i \in [m+1]$, $u_i = x_i^1 \ldots x_i^{k_i} \in X^*$ $(k_i \geq 0)$ and $v_i = y_i^1 \ldots y_i^{l_i} \in Y^*$ $(l_i \geq 0)$.

Syntax-Directed Translation Schema (II)

Example

Let $T = (\{S, A, B\}, \{a, b\}, \{x, y, z\}, P, S)$, where P has the rules:

$$p_1 = S \rightarrow \mathrm{a}\,A\,\mathrm{bb}\,B\,\mathrm{a}\,B\,;\ B\,A\,\mathrm{zyx}\,B\,\mathrm{xx},$$
$$p_2 = A \rightarrow A\,A\,;\ A\,\mathrm{x}\,A,$$
$$p_3 = A \rightarrow \lambda\,;\ \lambda, \text{ and}$$
$$p_4 = B \rightarrow \mathrm{ab}\,;\ \lambda.$$

A **derivation** in T is:

$$(S, S) \overset{p_1}{\Longrightarrow}_T (aAbbBaB, BAzyxBxx) \overset{p_2}{\Longrightarrow}_T (aAAbbBaB, BAxAzyxBxx) \overset{p_3}{\Longrightarrow}{}^*_T$$
$$\overset{p_3}{\Longrightarrow}{}^*_T (abbBaB, BzyxBxx) \overset{p_4}{\Longrightarrow}{}^*_T (abbabaab, zyxxx).$$

Definition (Syntax-Directed Translation)

The **translation** defined by a SDTS T is the relation

$$\tau_T = \{(u, v) \in X^* \times Y^* \mid (S, S) \Rightarrow^*_T (u, v)\},$$

and it will be called simply **syntax-directed translation**.

Syntax-Directed Translation Schema (II)

Example

Let $T = (\{S, A, B\}, \{a, b\}, \{x, y, z\}, P, S)$, where P has the rules:

$$p_1 = \mathbf{S} \to \mathrm{a}\,\mathbf{A}\,\mathrm{bb}\,\mathbf{B}\,\mathrm{a}\,\mathbf{B}\,;\ \mathbf{B}\,\mathbf{A}\,\mathrm{zyx}\,\mathbf{B}\,\mathrm{xx},$$
$$p_2 = \mathbf{A} \to \mathbf{A}\,\mathbf{A}\,;\ \mathbf{A}\,\mathrm{x}\,\mathbf{A},$$
$$p_3 = \mathbf{A} \to \lambda\,;\ \lambda, \text{ and}$$
$$p_4 = \mathbf{B} \to \mathrm{ab}\,;\ \lambda.$$

A **derivation** in T is:

$$(S, S) \stackrel{p_1}{\Longrightarrow}_T (aAbbBaB, BAzyxBxx) \stackrel{p_2}{\Longrightarrow}_T (aAAbbBaB, BAxAzyxBxx) \stackrel{p_3}{\Longrightarrow}^*_T$$
$$\stackrel{p_3}{\Longrightarrow}^*_T (abbBaB, BzyxBxx) \stackrel{p_4}{\Longrightarrow}^*_T (abbabaab, zyxxx).$$

Definition (Syntax-Directed Translation)

The **translation** defined by a SDTS T is the relation

$$\tau_T = \{(u, v) \in X^* \times Y^* \mid (S, S) \Rightarrow^*_T (u, v)\},$$

and it will be called simply **syntax-directed translation**.

Local Tree Languages

Let Σ be a ranked alphabet, X a leaf alphabet, and $T_\Sigma(X)$ the set of Σ-terms with variables in X (labeled trees).

What Is a Fork?

The set $\mathrm{fork}(t)$ of **forks** of a ΣX-tree t is defined as:

1. $\mathrm{fork}(d) = \emptyset$ for $d \in X \cup \Sigma_0$;
2. $\mathrm{fork}(t) = \mathrm{fork}(t_1) \cup \ldots \cup \mathrm{fork}(t_m) \cup \{f(\mathrm{root}(t_1), \ldots, \mathrm{root}(t_m))\}$ for $t = f(t_1, \ldots, t_m)$ $(m > 0)$.

The (finite) set of all possible forks of ΣX-trees is denoted by $\mathrm{fork}(\Sigma, X)$.

For any $D \subseteq \Sigma \cup X$ and $E \subseteq \mathrm{fork}(\Sigma, X)$

$$L(D, E) = \{t \in T_\Sigma(X) \mid \mathrm{root}(t) \in D, \mathrm{fork}(t) \subseteq E\}.$$

A ΣX-tree language R is local (in the strict sense), if $R = L(D, E)$ for some D and E.

Local Tree Languages

Let Σ be a ranked alphabet, X a leaf alphabet, and $T_\Sigma(X)$ the set of Σ-terms with variables in X (labeled trees).

What Is a Fork?

The set $\mathrm{fork}(t)$ of **forks** of a ΣX-tree t is defined as:

1. $\mathrm{fork}(d) = \emptyset$ for $d \in X \cup \Sigma_0$;
2. $\mathrm{fork}(t) = \mathrm{fork}(t_1) \cup \ldots \cup \mathrm{fork}(t_m) \cup \{f(\mathrm{root}(t_1), \ldots, \mathrm{root}(t_m))\}$ for $t = f(t_1, \ldots, t_m)$ $(m > 0)$.

The (finite) set of all possible forks of ΣX-trees is denoted by $\mathrm{fork}(\Sigma, X)$.

Definition (Local Tree Language)

For any $D \subseteq \Sigma \cup X$ and $E \subseteq \mathrm{fork}(\Sigma, X)$,

$$L(D, E) = \{t \in T_\Sigma(X) \mid \mathrm{root}(t) \in D, \mathrm{fork}(t) \subseteq E\}.$$

A ΣX-tree language R is **local (in the strict sense)**, if $R = L(D, E)$ for some D and E. Let $\mathrm{Loc}_\Sigma(X)$ be the set of all local ΣX-tree languages, and Loc the family of local tree languages.

Local Tree Languages

Let Σ be a ranked alphabet, X a leaf alphabet, and $T_\Sigma(X)$ the set of Σ-terms with variables in X (labeled trees).

What Is a Fork?

The set $\mathrm{fork}(t)$ of **forks** of a ΣX-tree t is defined as:

1. $\mathrm{fork}(d) = \emptyset$ for $d \in X \cup \Sigma_0$;
2. $\mathrm{fork}(t) = \mathrm{fork}(t_1) \cup \ldots \cup \mathrm{fork}(t_m) \cup \{f(\mathrm{root}(t_1), \ldots, \mathrm{root}(t_m))\}$ for $t = f(t_1, \ldots, t_m)$ $(m > 0)$.

The (finite) set of all possible forks of ΣX-trees is denoted by $\mathrm{fork}(\Sigma, X)$.

Definition (Local Tree Language)

For any $D \subseteq \Sigma \cup X$ and $E \subseteq \mathrm{fork}(\Sigma, X)$,

$$L(D, E) = \{t \in T_\Sigma(X) \mid \mathrm{root}(t) \in D, \mathrm{fork}(t) \subseteq E\}.$$

A ΣX-tree language R is **local (in the strict sense)**, if $R = L(D, E)$ for some D and E. Let $\mathrm{Loc}_\Sigma(X)$ be the set of all local ΣX-tree languages and Loc the family of local tree languages.

Local Tree Languages

Let Σ be a ranked alphabet, X a leaf alphabet, and $T_\Sigma(X)$ the set of Σ-terms with variables in X (labeled trees).

What Is a Fork?

The set $\mathrm{fork}(t)$ of **forks** of a ΣX-tree t is defined as:

1. $\mathrm{fork}(d) = \emptyset$ for $d \in X \cup \Sigma_0$;
2. $\mathrm{fork}(t) = \mathrm{fork}(t_1) \cup \ldots \cup \mathrm{fork}(t_m) \cup \{f(\mathrm{root}(t_1), \ldots, \mathrm{root}(t_m))\}$ for $t = f(t_1, \ldots, t_m)$ $(m > 0)$.

The (finite) set of all possible forks of ΣX-trees is denoted by $\mathrm{fork}(\Sigma, X)$.

Definition (Local Tree Language)

For any $D \subseteq \Sigma \cup X$ and $E \subseteq \mathrm{fork}(\Sigma, X)$,

$$L(D, E) = \{t \in T_\Sigma(X) \mid \mathrm{root}(t) \in D, \mathrm{fork}(t) \subseteq E\}.$$

A ΣX-tree language R is **local (in the strict sense)**, if $R = L(D, E)$ for some D and E. Let $Loc_\Sigma(X)$ be the set of all local ΣX-tree languages, and Loc the family of local tree languages.

Local Tree Languages

Let Σ be a ranked alphabet, X a leaf alphabet, and $T_\Sigma(X)$ the set of Σ-terms with variables in X (labeled trees).

What Is a Fork?

The set $\mathrm{fork}(t)$ of **forks** of a ΣX-tree t is defined as:

1. $\mathrm{fork}(d) = \emptyset$ for $d \in X \cup \Sigma_0$;
2. $\mathrm{fork}(t) = \mathrm{fork}(t_1) \cup \ldots \cup \mathrm{fork}(t_m) \cup \{f(\mathrm{root}(t_1), \ldots, \mathrm{root}(t_m))\}$ for $t = f(t_1, \ldots, t_m)$ $(m > 0)$.

The (finite) set of all possible forks of ΣX-trees is denoted by $\mathrm{fork}(\Sigma, X)$.

Definition (Local Tree Language)

For any $D \subseteq \Sigma \cup X$ and $E \subseteq \mathrm{fork}(\Sigma, X)$,

$$L(D, E) = \{t \in T_\Sigma(X) \mid \mathrm{root}(t) \in D, \mathrm{fork}(t) \subseteq E\}.$$

A ΣX-tree language R is **local (in the strict sense)**, if $R = L(D, E)$ for some D and E. Let $\mathrm{Loc}_\Sigma(X)$ be the set of all local ΣX-tree languages, and Loc the family of local tree languages.

Local Tree Languages

Let Σ be a ranked alphabet, X a leaf alphabet, and $T_\Sigma(X)$ the set of Σ-terms with variables in X (labeled trees).

What Is a Fork?

The set $\mathrm{fork}(t)$ of **forks** of a ΣX-tree t is defined as:

1. $\mathrm{fork}(d) = \emptyset$ for $d \in X \cup \Sigma_0$;
2. $\mathrm{fork}(t) = \mathrm{fork}(t_1) \cup \ldots \cup \mathrm{fork}(t_m) \cup \{f(\mathrm{root}(t_1), \ldots, \mathrm{root}(t_m))\}$ for $t = f(t_1, \ldots, t_m)$ $(m > 0)$.

The (finite) set of all possible forks of ΣX-trees is denoted by $\mathrm{fork}(\Sigma, X)$.

Definition (Local Tree Language)

For any $D \subseteq \Sigma \cup X$ and $E \subseteq \mathrm{fork}(\Sigma, X)$,

$$L(D, E) = \{t \in T_\Sigma(X) \mid \mathrm{root}(t) \in D, \mathrm{fork}(t) \subseteq E\}.$$

A ΣX-tree language R is **local (in the strict sense)**, if $R = L(D, E)$ for some D and E. Let $\mathrm{Loc}_\Sigma(X)$ be the set of all local ΣX-tree languages, and Loc the family of local tree languages.

Quasi-Alphabetic Tree Bimorphisms (I)

Definition (Tree Homomorphism)

A **tree homomorphism** $\varphi : T_\Sigma(X) \to T_\Omega(Y)$ is determined by a mapping $\varphi_X : X \to T_\Omega(Y)$ and mappings $\varphi_m : \Sigma_m \to T_\Omega(Y \cup \Xi_m)$ $(m \geq 0,\ \Sigma_m \neq \emptyset)$:

1. $x\varphi = \varphi_X(x)$ for any $x \in X$,
2. $c\varphi = \varphi_0(c)$ for any $c \in \Sigma_0$, and
3. $t\varphi = \varphi_m(f)(\xi_1 \leftarrow t_1\varphi, \ldots, \xi_m \leftarrow t_m\varphi)$ for $t = f(t_1, \ldots, t_m)$ $(m > 0)$.

Definition (Quasi-Alphabetic Tree Homomorphism)

We call a tree homomorphism $\varphi : T_\Sigma(X) \to T_\Omega(Y)$ **quasi-alphabetic**, if

1. $\varphi_X(x) \in Y$ for every $x \in X$, and
2. for all $m \geq 0$ and $f \in \Sigma_m$, $\varphi_m(f)$ is of the form

where σ is a permutation of $[m]$, for each $i \in [m+1]$, $k_i \geq 0$ and ... and $g \in \Omega_{m'}$ for $m' = m + k_1 + \cdots + k_{m+1}$.

Let qH denote the class of all quasi-alphabetic tree homomorphisms.

Quasi-Alphabetic Tree Bimorphisms (I)

Definition (Tree Homomorphism)

A **tree homomorphism** $\varphi : T_\Sigma(X) \to T_\Omega(Y)$ is determined by a mapping $\varphi_{\mathbf{X}} : X \to T_\Omega(Y)$ and mappings $\varphi_{\mathbf{m}} : \Sigma_m \to T_\Omega(Y \cup \Xi_m)$ ($m \geq 0$, $\Sigma_m \neq \emptyset$):

1. $x\varphi = \varphi_X(x)$ for any $x \in X$,
2. $c\varphi = \varphi_0(c)$ for any $c \in \Sigma_0$, and
3. $t\varphi = \varphi_m(f)(\xi_1 \leftarrow t_1\varphi, \ldots, \xi_m \leftarrow t_m\varphi)$ for $t = f(t_1, \ldots, t_m)$ ($m > 0$).

Definition (Quasi-Alphabetic Tree Homomorphism)

We call a tree homomorphism $\varphi : T_\Sigma(X) \to T_\Omega(Y)$ **quasi-alphabetic**, if

1. $\varphi_X(x) \in Y$ for every $x \in X$, and

Quasi-Alphabetic Tree Bimorphisms (I)

Definition (Tree Homomorphism)

A **tree homomorphism** $\varphi : T_\Sigma(X) \to T_\Omega(Y)$ is determined by a mapping $\varphi_X : X \to T_\Omega(Y)$ and mappings $\varphi_m : \Sigma_m \to T_\Omega(Y \cup \Xi_m)$ ($m \geq 0$, $\Sigma_m \neq \emptyset$):

1. $x\varphi = \varphi_X(x)$ for any $x \in X$,
2. $c\varphi = \varphi_0(c)$ for any $c \in \Sigma_0$, and
3. $t\varphi = \varphi_m(f)(\xi_1 \leftarrow t_1\varphi, \ldots, \xi_m \leftarrow t_m\varphi)$ for $t = f(t_1, \ldots, t_m)$ ($m > 0$).

Definition (Quasi-Alphabetic Tree Homomorphism)

We call a tree homomorphism $\varphi : T_\Sigma(X) \to T_\Omega(Y)$ **quasi-alphabetic**, if

1. $\varphi_X(x) \in Y$ for every $x \in X$, and
2. for all $m \geq 0$ and $f \in \Sigma_m$, $\varphi_m(f)$ is of the form

$$g(y_1^1, \ldots, y_1^{k_1}, \xi_{\sigma(1)}, y_2^1, \ldots, y_2^{k_2}, \ldots, y_m^1, \ldots, y_m^{k_m}, \xi_{\sigma(m)}, y_{m+1}^1, \ldots, y_{m+1}^{k_{m+1}}),$$

where σ is a permutation of $[m]$, for each $i \in [m+1]$, $k_i \geq 0$ and $y_i^1, \ldots, y_i^{k_i} \in Y$, and $g \in \Omega_{m'}$ for $m' = m + k_1 + \ldots + k_{m+1}$.

Let qH denote the class of all quasi-alphabetic tree homomorphisms.

Quasi-Alphabetic Tree Bimorphisms (I)

Definition (Tree Homomorphism)

A **tree homomorphism** $\varphi : T_\Sigma(X) \to T_\Omega(Y)$ is determined by a mapping $\varphi_\mathbf{X} : X \to T_\Omega(Y)$ and mappings $\varphi_\mathbf{m} : \Sigma_m \to T_\Omega(Y \cup \Xi_m)$ ($m \geq 0$, $\Sigma_m \neq \emptyset$):

1. $x\varphi = \varphi_X(x)$ for any $x \in X$,
2. $c\varphi = \varphi_0(c)$ for any $c \in \Sigma_0$, and
3. $t\varphi = \varphi_m(f)(\xi_1 \leftarrow t_1\varphi, \ldots, \xi_m \leftarrow t_m\varphi)$ for $t = f(t_1, \ldots, t_m)$ ($m > 0$).

Definition (Quasi-Alphabetic Tree Homomorphism)

We call a tree homomorphism $\varphi : T_\Sigma(X) \to T_\Omega(Y)$ **quasi-alphabetic**, if

1. $\varphi_X(x) \in Y$ for every $x \in X$, and
2. for all $m \geq 0$ and $f \in \Sigma_m$, $\varphi_m(f)$ is of the form

$$g(y_1^1, \ldots, y_1^{k_1}, \xi_{\sigma(1)}, y_2^1, \ldots, y_2^{k_2}, \ldots, y_1^{k_m}, \ldots, y_m^{k_m}, \xi_{\sigma(m)}, y_{m+1}^1, \ldots, y_{m+1}^{k_{m+1}}),$$

   where σ is a permutation of $[m]$, for each $i \in [m+1]$, $k_i \geq 0$ and $y_i^1, \ldots, y_i^{k_i} \in Y$, and $g \in \Omega_{m'}$ for $m' = m + k_1 + \ldots + k_{m+1}$.

Let qH denote the class of all quasi-alphabetic tree homomorphisms.

Quasi-Alphabetic Tree Bimorphisms (I)

Definition (Tree Homomorphism)

A **tree homomorphism** $\varphi : T_\Sigma(X) \to T_\Omega(Y)$ is determined by a mapping $\varphi_{\mathbf{X}} : X \to T_\Omega(Y)$ and mappings $\varphi_{\mathbf{m}} : \Sigma_m \to T_\Omega(Y \cup \Xi_m)$ ($m \geq 0$, $\Sigma_m \neq \emptyset$):

1. $x\varphi = \varphi_X(x)$ for any $x \in X$,
2. $c\varphi = \varphi_0(c)$ for any $c \in \Sigma_0$, and
3. $t\varphi = \varphi_m(f)(\xi_1 \leftarrow t_1\varphi, \ldots, \xi_m \leftarrow t_m\varphi)$ for $t = f(t_1, \ldots, t_m)$ $(m > 0)$.

Definition (Quasi-Alphabetic Tree Homomorphism)

We call a tree homomorphism $\varphi : T_\Sigma(X) \to T_\Omega(Y)$ **quasi-alphabetic**, if

1. $\varphi_X(x) \in Y$ for every $x \in X$, and
2. for all $m \geq 0$ and $f \in \Sigma_m$, $\varphi_m(f)$ is of the form

$$g(y_1^1, \ldots, y_1^{k_1}, \xi_{\sigma(\mathbf{1})}, y_2^1, \ldots, y_2^{k_2}, \ldots, y_1^{k_m}, \ldots, y_m^{k_m}, \xi_{\sigma(\mathbf{m})}, y_{m+1}^1, \ldots, y_{m+1}^{k_{m+1}}),$$

   where σ is a permutation of $[m]$, for each $i \in [m+1]$, $k_i \geq 0$ and $y_i^1, \ldots, y_i^{k_i} \in Y$, and $g \in \Omega_{m'}$ for $m' = m + k_1 + \ldots + k_{m+1}$.

Let qH denote the class of all quasi-alphabetic tree homomorphisms.

Quasi-Alphabetic Tree Bimorphisms (I)

Definition (Tree Homomorphism)

A **tree homomorphism** $\varphi : T_\Sigma(X) \to T_\Omega(Y)$ is determined by a mapping $\varphi_{\mathbf{X}} : X \to T_\Omega(Y)$ and mappings $\varphi_{\mathbf{m}} : \Sigma_m \to T_\Omega(Y \cup \Xi_m)$ ($m \geq 0$, $\Sigma_m \neq \emptyset$):

1. $x\varphi = \varphi_X(x)$ for any $x \in X$,
2. $c\varphi = \varphi_0(c)$ for any $c \in \Sigma_0$, and
3. $t\varphi = \varphi_m(f)(\xi_1 \leftarrow t_1\varphi, \ldots, \xi_m \leftarrow t_m\varphi)$ for $t = f(t_1, \ldots, t_m)$ ($m > 0$).

Definition (Quasi-Alphabetic Tree Homomorphism)

We call a tree homomorphism $\varphi : T_\Sigma(X) \to T_\Omega(Y)$ **quasi-alphabetic**, if

1. $\varphi_X(x) \in Y$ for every $x \in X$, and
2. for all $m \geq 0$ and $f \in \Sigma_m$, $\varphi_m(f)$ is of the form

$$g(y_1^1, \ldots, y_1^{k_1}, \xi_{\sigma(\mathbf{1})}, y_2^1, \ldots, y_2^{k_2}, \ldots, y_1^{km}, \ldots, y_m^{km}, \xi_{\sigma(\mathbf{m})}, y_{m+1}^1, \ldots, y_{m+1}^{k_{m+1}}),$$

   where σ is a permutation of $[m]$, for each $i \in [m+1]$, $k_i \geq 0$ and $y_i^1, \ldots, y_i^{k_i} \in Y$, and $g \in \Omega_{m'}$ for $m' = m + k_1 + \ldots + k_{m+1}$.

Let qH denote the class of all quasi-alphabetic tree homomorphisms.

Quasi-Alphabetic Tree Bimorphisms (II)

Basic Properties

A quasi-alphabetic tree homomorphism is

- linear: no copying,
- non-deleting: no subtree information is lost,
- symbol-to-symbol: each symbol of arity greater than 0 is mapped to a tree of height 1, and the order of the subtrees can be modified, and
- each constant symbol is mapped to a tree of height 0 or 1.

Definition (Quasi-Alphabetic Tree Bimorphism)

A **tree bimorphism** is a triple $B = (\varphi, R, \psi)$, where

- $R \subseteq T_\Gamma(Z)$ is a **regular tree language**, and
- $\varphi : T_\Gamma(Z) \to T_\Sigma(X)$ and $\psi : T_\Gamma(Z) \to T_\Omega(Y)$ are tree homomorphisms.

The tree transformation defined by B is

$$\tau_B = \varphi^{-1} \circ \delta_R \circ \psi = \{(t\varphi, t\psi) \mid t \in R\} \; (\subseteq T_\Sigma(X) \times T_\Omega(Y))$$

The translation defined by B is

$$yd(\tau_B) = \{(yd(t\varphi), yd(t\psi)) \mid t \in R\} \; (\subseteq X^* \times Y^*)$$

Quasi-Alphabetic Tree Bimorphisms (II)

Basic Properties

A quasi-alphabetic tree homomorphism is

- linear: no copying,
- non-deleting: no subtree information is lost,
- symbol-to-symbol: each symbol of arity greater than 0 is mapped to a tree of height 1, and the order of the subtrees can be modified, and
- each constant symbol is mapped to a tree of height 0 or 1.

Definition (Quasi-Alphabetic Tree Bimorphism)

A **tree bimorphism** is a triple $B = (\varphi, R, \psi)$, where

- $R \subseteq T_\Gamma(Z)$ is a **regular tree language**, and

Quasi-Alphabetic Tree Bimorphisms (II)

Basic Properties

A quasi-alphabetic tree homomorphism is

- linear: no copying,
- non-deleting: no subtree information is lost,
- symbol-to-symbol: each symbol of arity greater than 0 is mapped to a tree of height 1, and the order of the subtrees can be modified, and
- each constant symbol is mapped to a tree of height 0 or 1.

Definition (Quasi-Alphabetic Tree Bimorphism)

A **tree bimorphism** is a triple $B = (\varphi, R, \psi)$, where

- $R \subseteq T_\Gamma(Z)$ is a **regular tree language**, and
- $\varphi : T_\Gamma(Z) \to T_\Sigma(X)$ and $\psi : T_\Gamma(Z) \to T_\Omega(Y)$ are tree homomorphisms.

The tree transformation defined by B is

$$\tau_B = \varphi^{-1} \circ \delta_R \circ \psi = \{(r\varphi, r\psi) \mid r \in R\} \ (\subseteq T_\Sigma(X) \times T_\Omega(Y)).$$

The translation defined by B is

$$\mathrm{yd}(\tau_B) = \{(\mathrm{yd}(r\varphi), \mathrm{yd}(r\psi)) \mid r \in R\} \ (\subseteq X^* \times Y^*).$$

Quasi-Alphabetic Tree Bimorphisms (II)

Basic Properties

A quasi-alphabetic tree homomorphism is

- linear: no copying,
- non-deleting: no subtree information is lost,
- symbol-to-symbol: each symbol of arity greater than 0 is mapped to a tree of height 1, and the order of the subtrees can be modified, and
- each constant symbol is mapped to a tree of height 0 or 1.

Definition (Quasi-Alphabetic Tree Bimorphism)

A **tree bimorphism** is a triple $B = (\varphi, R, \psi)$, where

- $R \subseteq T_\Gamma(Z)$ is a **regular tree language**, and
- $\varphi : T_\Gamma(Z) \to T_\Sigma(X)$ and $\psi : T_\Gamma(Z) \to T_\Omega(Y)$ are **tree homomorphisms**.

The tree transformation defined by B is

$$\tau_B = \varphi^{-1} \circ \delta_R \circ \psi = \{(\varphi(t), \psi(t)) \mid t \in R\} \ (\subseteq T_\Sigma(X) \times T_\Omega(Y))$$

The translation defined by B is

$$yd(\tau_B) = \{(yd(\varphi(t)), yd(\psi(t))) \mid t \in R\} \ (\subseteq X^* \times Y^*)$$

Quasi-Alphabetic Tree Bimorphisms (II)

Basic Properties

A quasi-alphabetic tree homomorphism is

- linear: no copying,
- non-deleting: no subtree information is lost,
- symbol-to-symbol: each symbol of arity greater than 0 is mapped to a tree of height 1, and the order of the subtrees can be modified, and
- each constant symbol is mapped to a tree of height 0 or 1.

Definition (Quasi-Alphabetic Tree Bimorphism)

A **tree bimorphism** is a triple $B = (\varphi, R, \psi)$, where

- $R \subseteq T_\Gamma(Z)$ is a **regular tree language**, and
- $\varphi : T_\Gamma(Z) \rightarrow T_\Sigma(X)$ and $\psi : T_\Gamma(Z) \rightarrow T_\Omega(Y)$ are **tree homomorphisms.**

The **tree transformation** defined by B is

$$\tau_{\mathbf{B}} = \varphi^{-1} \circ \delta_R \circ \psi = \{(r\varphi, r\psi) \mid r \in R\} \ \ (\subseteq T_\Sigma(X) \times T_\Omega(Y)).$$

The **translation** defined by B is

$$\mathrm{yd}(\tau_{\mathbf{B}}) = \{(\mathrm{yd}(r\varphi), \mathrm{yd}(r\psi)) \mid r \in R\} \ \ (\subseteq X^* \times Y^*).$$

Quasi-Alphabetic Tree Bimorphisms (II)

Basic Properties

A quasi-alphabetic tree homomorphism is

- linear: no copying,
- non-deleting: no subtree information is lost,
- symbol-to-symbol: each symbol of arity greater than 0 is mapped to a tree of height 1, and the order of the subtrees can be modified, and
- each constant symbol is mapped to a tree of height 0 or 1.

Definition (Quasi-Alphabetic Tree Bimorphism)

A **tree bimorphism** is a triple $B = (\varphi, R, \psi)$, where

- $R \subseteq T_\Gamma(Z)$ is a **regular tree language**, and
- $\varphi : T_\Gamma(Z) \to T_\Sigma(X)$ and $\psi : T_\Gamma(Z) \to T_\Omega(Y)$ are **tree homomorphisms**.

The **tree transformation** defined by B is

$$\tau_B = \varphi^{-1} \circ \delta_R \circ \psi = \{(r\varphi, r\psi) \mid r \in R\} \ \ (\subseteq T_\Sigma(X) \times T_\Omega(Y)).$$

The **translation** defined by B is

$$\mathrm{yd}(\tau_B) = \{(\mathrm{yd}(r\varphi), \mathrm{yd}(r\psi)) \mid r \in R\} \ \ (\subseteq X^* \times Y^*).$$

Quasi-Alphabetic Tree Bimorphisms (II)

Basic Properties

A quasi-alphabetic tree homomorphism is

- linear: no copying,
- non-deleting: no subtree information is lost,
- symbol-to-symbol: each symbol of arity greater than 0 is mapped to a tree of height 1, and the order of the subtrees can be modified, and
- each constant symbol is mapped to a tree of height 0 or 1.

Definition (Quasi-Alphabetic Tree Bimorphism)

A **tree bimorphism** is a triple $B = (\varphi, R, \psi)$, where

- $R \subseteq T_\Gamma(Z)$ is a **regular tree language**, and
- $\varphi : T_\Gamma(Z) \to T_\Sigma(X)$ and $\psi : T_\Gamma(Z) \to T_\Omega(Y)$ are **tree homomorphisms**.

The **tree transformation** defined by B is

$$\tau_{\mathbf{B}} = \varphi^{-1} \circ \delta_R \circ \psi = \{(r\varphi, r\psi) \mid r \in R\} \ \ (\subseteq T_\Sigma(X) \times T_\Omega(Y)).$$

The **translation** defined by B is

$$\mathrm{yd}(\tau_{\mathbf{B}}) = \{(\mathrm{yd}(r\varphi), \mathrm{yd}(r\psi)) \mid r \in R\} \ \ (\subseteq X^* \times Y^*).$$

Quasi-Alphabetic Tree Bimorphisms (II)

Basic Properties

A quasi-alphabetic tree homomorphism is

- linear: no copying,
- non-deleting: no subtree information is lost,
- symbol-to-symbol: each symbol of arity greater than 0 is mapped to a tree of height 1, and the order of the subtrees can be modified, and
- each constant symbol is mapped to a tree of height 0 or 1.

Definition (Quasi-Alphabetic Tree Bimorphism)

A **tree bimorphism** is a triple $B = (\varphi, R, \psi)$, where

- $R \subseteq T_\Gamma(Z)$ is a **regular tree language**, and
- $\varphi : T_\Gamma(Z) \rightarrow T_\Sigma(X)$ and $\psi : T_\Gamma(Z) \rightarrow T_\Omega(Y)$ are **tree homomorphisms**.

The **tree transformation** defined by B is

$$\tau_{\mathbf{B}} = \varphi^{-1} \circ \delta_R \circ \psi = \{(r\varphi, r\psi) \mid r \in R\} \;\; (\subseteq T_\Sigma(X) \times T_\Omega(Y)).$$

The **translation** defined by B is

$$\mathrm{yd}(\tau_{\mathbf{B}}) = \{(\mathrm{yd}(r\varphi), \mathrm{yd}(r\psi)) \mid r \in R\} \;\; (\subseteq X^* \times Y^*).$$

Quasi-Alphabetic Tree Bimorphisms (III)

Notations

For any classes H_1 and H_2 of tree homomorphisms and any class $\mathcal{R}$ of regular tree languages,

- $\mathbf{B}(H_1, \mathcal{R}, H_2)$ are all tree bimorphisms $B = (\varphi, R, \psi)$ with $\varphi \in H_1$, $R \in \mathcal{R}$ and $\psi \in H_2$;
- $\mathcal{B}(H_1, \mathcal{R}, H_2)$ is the corresponding class of tree transformations;
- $\mathbf{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ is the class of **quasi-alphabetic tree bimorphisms** in which the two tree homomorphisms are quasi-alphabetic and the tree language is local;
- $\mathcal{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ is the class of all the tree transformations defined by quasi-alphabetic tree bimorphisms.

An example

Let $B = (\varphi, R, \psi)$ be a bimorphism, where:

- $Z = \{z\}$, $\Gamma_3 = \{f\}$, $\Gamma_0 = \{c\}$, $\Gamma = \Gamma_3 \cup \Gamma_0$;
- $X = \{x, y\}$, $\Sigma_6 = \{g\}$, $\Sigma_2 = \{i\}$, $\Sigma = \Sigma_6 \cup \Sigma_2$;
- $Y = \{0, 1\}$, $\Omega_7 = \{h\}$, $\Omega_0 = \{j\}$, $\Omega = \Omega_7 \cup \Omega_0$;
- $R = T_\Gamma(Z)$;
- $\varphi : T_\Gamma(Z) \to T_\Sigma(X)$ and $\psi : T_\Gamma(Z) \to T_\Omega(Y)$ quasi-alphabetic tree homomorphisms:
  - $\varphi_Z(z) = x$, $\psi_Z(z) = 0$;
  - $\varphi_0(c) = i$, $\psi_0(c) = j$;
  - $\varphi_3(f) = g(y, \xi_2, \xi_1, x, x, \xi_3)$, $\psi_3(f) = h(\xi_3, 1, 1, \xi_2, 0, \xi_1, 1)$.

Quasi-Alphabetic Tree Bimorphisms (III)

Notations

For any classes H_1 and H_2 of tree homomorphisms and any class $\mathcal{R}$ of regular tree languages,

- $\mathbf{B}(H_1, \mathcal{R}, H_2)$ are all tree bimorphisms $B = (\varphi, R, \psi)$ with $\varphi \in H_1$, $R \in \mathcal{R}$ and $\psi \in H_2$;
- $\mathcal{B}(H_1, \mathcal{R}, H_2)$ is the corresponding class of tree transformations;
- $\mathbf{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ is the class of **quasi-alphabetic tree bimorphisms** in which the two tree homomorphisms are quasi-alphabetic and the tree language is local;
- $\mathcal{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ is the class of all the tree transformations defined by quasi-alphabetic tree bimorphisms.

An example

Let $B = (\varphi, R, \psi)$ be a bimorphism, where:

- $Z = \{z\}$, $\Gamma_3 = \{f\}$, $\Gamma_0 = \{c\}$, $\Gamma = \Gamma_3 \cup \Gamma_0$;
- $X = \{x, y\}$, $\Sigma_6 = \{g\}$, $\Sigma_2 = \{i\}$, $\Sigma = \Sigma_6 \cup \Sigma_2$;
- $Y = \{0, 1\}$, $\Omega_7 = \{h\}$, $\Omega_0 = \{j\}$, $\Omega = \Omega_7 \cup \Omega_0$;
- $R = T_\Gamma(Z)$;
- $\varphi : T_\Gamma(Z) \to T_\Sigma(X)$ and $\psi : T_\Gamma(Z) \to T_\Omega(Y)$ quasi-alphabetic tree homomorphisms:
  - $\varphi_Z(z) = x$, $\psi_Z(z) = 0$;
  - $\varphi_0(c) = i$, $\psi_0(c) = j$;
  - $\varphi_3(f) = g(y, \xi_2, \xi_1, x, x, \xi_3)$, $\psi_3(f) = h(\xi_3, 1, 1, \xi_2, 0, \xi_1, 1)$.

Quasi-Alphabetic Tree Bimorphisms (III)

Notations

For any classes H_1 and H_2 of tree homomorphisms and any class $\mathcal{R}$ of regular tree languages,

- $\mathbf{B}(H_1, \mathcal{R}, H_2)$ are all tree bimorphisms $B = (\varphi, R, \psi)$ with $\varphi \in H_1$, $R \in \mathcal{R}$ and $\psi \in H_2$;
- $\mathcal{B}(H_1, \mathcal{R}, H_2)$ is the corresponding class of tree transformations;
- $\mathbf{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ is the class of **quasi-alphabetic tree bimorphisms** in which the two tree homomorphisms are quasi-alphabetic and the tree language is local;
- $\mathcal{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ is the class of all the tree transformations defined by quasi-alphabetic tree bimorphisms.

An example

Let $B = (\varphi, R, \psi)$ be a bimorphism, where:

- $Z = \{z\}$, $\Gamma_3 = \{f\}$, $\Gamma_0 = \{c\}$, $\Gamma = \Gamma_3 \cup \Gamma_0$;
- $X = \{x, y\}$, $\Sigma_6 = \{g\}$, $\Sigma_2 = \{i\}$, $\Sigma = \Sigma_6 \cup \Sigma_2$;
- $Y = \{0, 1\}$, $\Omega_7 = \{h\}$, $\Omega_0 = \{j\}$, $\Omega = \Omega_7 \cup \Omega_0$;
- $R = T_\Gamma(Z)$;
- $\varphi : T_\Gamma(Z) \to T_\Sigma(X)$ and $\psi : T_\Gamma(Z) \to T_\Omega(Y)$ quasi-alphabetic tree homomorphisms:
  - $\varphi_Z(z) = x$, $\psi_Z(z) = 0$;
  - $\varphi_0(c) = i$, $\psi_0(c) = j$;
  - $\varphi_3(f) = g(y, \xi_2, \xi_1, x, x, \xi_3)$, $\psi_3(f) = h(\xi_3, 1, 1, \xi_2, 0, \xi_1, 1)$.

Quasi-Alphabetic Tree Bimorphisms (III)

Notations

For any classes H_1 and H_2 of tree homomorphisms and any class $\mathcal{R}$ of regular tree languages,

- $\mathbf{B}(H_1, \mathcal{R}, H_2)$ are all tree bimorphisms $B = (\varphi, R, \psi)$ with $\varphi \in H_1$, $R \in \mathcal{R}$ and $\psi \in H_2$;
- $\mathcal{B}(H_1, \mathcal{R}, H_2)$ is the corresponding class of tree transformations;
- $\mathbf{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ is the class of **quasi-alphabetic tree bimorphisms** in which the two tree homomorphisms are quasi-alphabetic and the tree language is local;
- $\mathcal{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ is the class of all the tree transformations defined by quasi-alphabetic tree bimorphisms.

An example

Let $B = (\varphi, R, \psi)$ be a bimorphism, where:

- $Z = \{z\}$, $\Gamma_3 = \{f\}$, $\Gamma_0 = \{c\}$, $\Gamma = \Gamma_3 \cup \Gamma_0$;
- $X = \{x, y\}$, $\Sigma_6 = \{g\}$, $\Sigma_2 = \{i\}$, $\Sigma = \Sigma_6 \cup \Sigma_2$;
- $Y = \{0, 1\}$, $\Omega_7 = \{h\}$, $\Omega_0 = \{j\}$, $\Omega = \Omega_7 \cup \Omega_0$;
- $R = T_\Gamma(Z)$;
- $\varphi : T_\Gamma(Z) \to T_\Sigma(X)$ and $\psi : T_\Gamma(Z) \to T_\Omega(Y)$ quasi-alphabetic tree homomorphisms:
  - $\varphi_Z(z) = x$, $\psi_Z(z) = 0$;
  - $\varphi_0(c) = i$, $\psi_0(c) = j$;
  - $\varphi_3(f) = g(y, \xi_2, \xi_1, x, x, \xi_3)$, $\psi_3(f) = h(\xi_3, 1, 1, \xi_2, 0, \xi_1, 1)$.

Quasi-Alphabetic Tree Bimorphisms (III)

Notations

For any classes H_1 and H_2 of tree homomorphisms and any class $\mathcal{R}$ of regular tree languages,

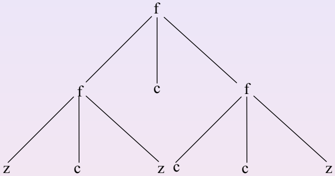
- $\mathbf{B}(H_1, \mathcal{R}, H_2)$ are all tree bimorphisms $B = (\varphi, R, \psi)$ with $\varphi \in H_1$, $R \in \mathcal{R}$ and $\psi \in H_2$;
- $\mathcal{B}(H_1, \mathcal{R}, H_2)$ is the corresponding class of tree transformations;
- $\mathbf{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ is the class of **quasi-alphabetic tree bimorphisms** in which the two tree homomorphisms are quasi-alphabetic and the tree language is local;
- $\mathcal{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ is the class of all the tree transformations defined by quasi-alphabetic tree bimorphisms.

An example

Let $B = (\varphi, R, \psi)$ be a bimorphism, where:

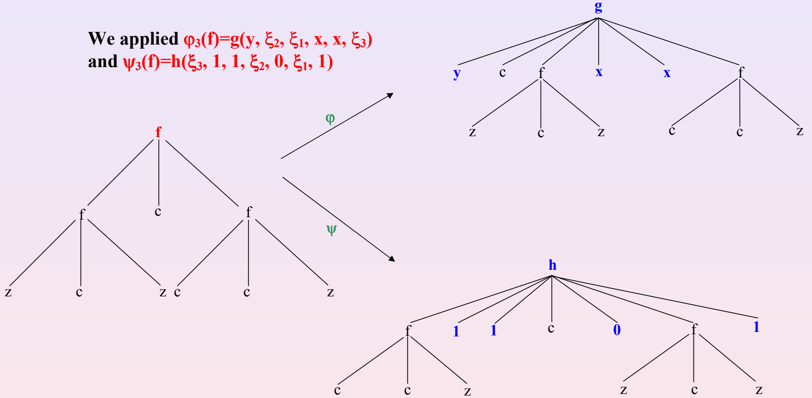
- $Z = \{z\}$, $\Gamma_3 = \{f\}$, $\Gamma_0 = \{c\}$, $\Gamma = \Gamma_3 \cup \Gamma_0$;
- $X = \{x, y\}$, $\Sigma_6 = \{g\}$, $\Sigma_2 = \{i\}$, $\Sigma = \Sigma_6 \cup \Sigma_2$;
- $Y = \{0, 1\}$, $\Omega_7 = \{h\}$, $\Omega_0 = \{j\}$, $\Omega = \Omega_7 \cup \Omega_0$;
- $R = T_\Gamma(Z)$;
- $\varphi : T_\Gamma(Z) \to T_\Sigma(X)$ and $\psi : T_\Gamma(Z) \to T_\Omega(Y)$ quasi-alphabetic tree homomorphisms:
  - $\varphi_Z(z) = x$, $\psi_Z(z) = 0$;
  - $\varphi_0(c) = i$, $\psi_0(c) = j$;
  - $\varphi_3(f) = g(y, \xi_2, \xi_1, x, x, \xi_3)$, $\psi_3(f) = h(\xi_3, 1, 1, \xi_2, 0, \xi_1, 1)$.

Quasi-Alphabetic Tree Bimorphisms (III)



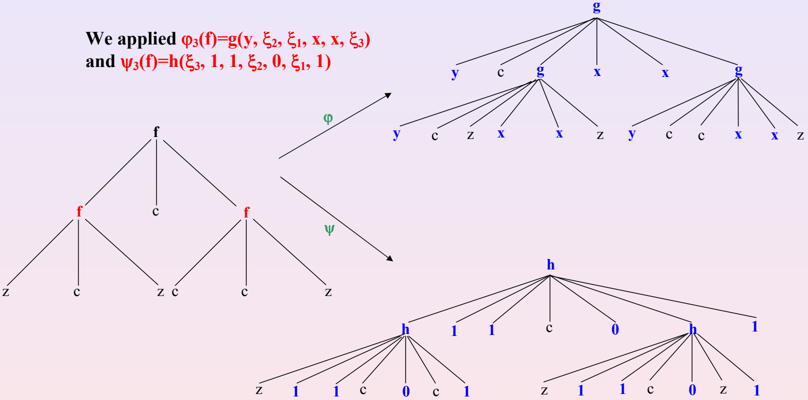
Quasi-Alphabetic Tree Bimorphisms (III)

We applied $\varphi_3(f)=g(y, \xi_2, \xi_1, x, x, \xi_3)$
and $\psi_3(f)=h(\xi_3, 1, 1, \xi_2, 0, \xi_1, 1)$



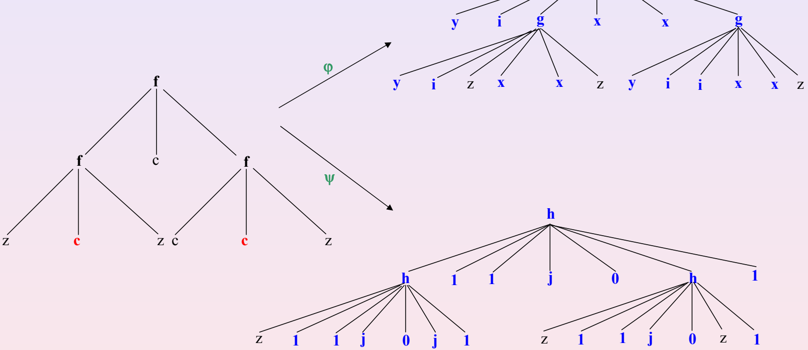
Quasi-Alphabetic Tree Bimorphisms (III)

We applied $\varphi_3(f)=g(y, \xi_2, \xi_1, x, x, \xi_3)$
and $\psi_3(f)=h(\xi_3, 1, 1, \xi_2, 0, \xi_1, 1)$



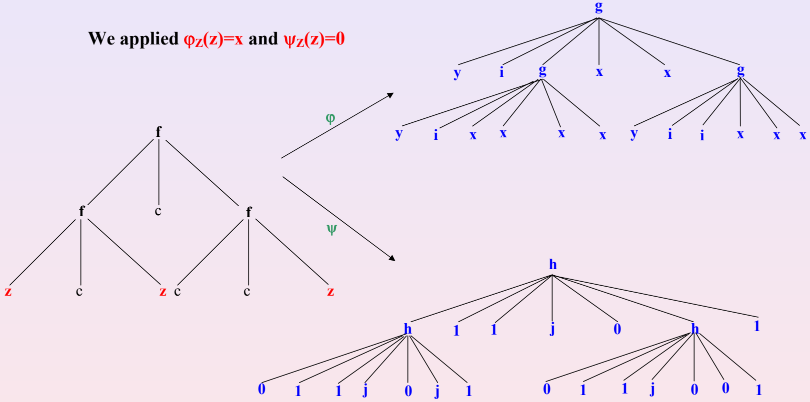
Quasi-Alphabetic Tree Bimorphisms (III)

We applied $\varphi_0(c)=i$ and $\psi_0(c)=j$



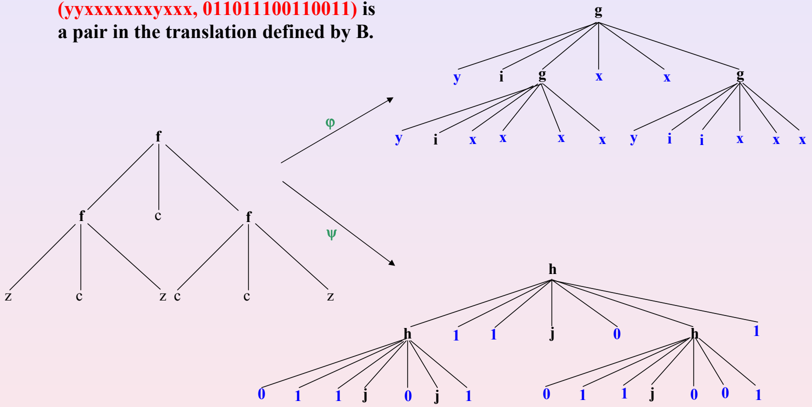
Quasi-Alphabetic Tree Bimorphisms (III)

We applied $\varphi_Z(z)=x$ and $\psi_Z(z)=0$



Quasi-Alphabetic Tree Bimorphisms (III)

(yyxxxxxxxyxxx, 011011100110011) is a pair in the translation defined by B.



The Connection between SDTSs and $\mathbf{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$

Proposition

For every SDTS T, one can define a tree bimorphism B in $\mathbf{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ such that $\tau_T = \mathrm{yd}(\tau_B)$.

Proposition

For each quasi-alphabetic tree bimorphism B, one can define a syntax-directed translation schema T such that $\mathrm{yd}(\tau_B) = \tau_T$.

Theorem

The class of syntax-directed translations is effectively equal to the class of translations defined by quasi-alphabetic tree bimorphisms.

Lemma (Preservation of Recognizability)

Quasi-alphabetic tree bimorphisms preserve recognizability, i.e., if $B \in \mathbf{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ and R' is a regular tree language, then so is $R'\tau_B$.

Using the connection expressed by the above theorem and the tree language theory, properties of syntax-directed translations can be proved (e.g., the domain and range are context-free languages).

The Connection between SDTSs and $\mathbf{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$

Proposition

For every SDTS T, one can define a tree bimorphism B in $\mathbf{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ such that $\tau_T = \mathrm{yd}(\tau_B)$.

Proposition

For each quasi-alphabetic tree bimorphism B, one can define a syntax-directed translation schema T such that $\mathrm{yd}(\tau_B) = \tau_T$.

Theorem

The class of syntax-directed translations is effectively equal to the class of translations defined by quasi-alphabetic tree bimorphisms.

Lemma (Preservation of Recognizability)

Quasi-alphabetic tree bimorphisms preserve recognizability, i.e., if $B \in \mathbf{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ and R' is a regular tree language, then so is $R'\tau_B$.

Using the connection expressed by the above theorem and the tree language theory, properties of syntax-directed translations can be proved (e.g., the domain and range are context-free languages).

The Connection between SDTSs and $\mathbf{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$

Proposition

For every SDTS T, one can define a tree bimorphism B in $\mathbf{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ such that $\tau_T = \mathrm{yd}(\tau_B)$.

Proposition

For each quasi-alphabetic tree bimorphism B, one can define a syntax-directed translation schema T such that $\mathrm{yd}(\tau_B) = \tau_T$.

Theorem

The class of syntax-directed translations is effectively equal to the class of translations defined by quasi-alphabetic tree bimorphisms.

Lemma (Preservation of Recognizability)

Quasi-alphabetic tree bimorphisms preserve recognizability, i.e., if $B \in \mathbf{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ and R' is a regular tree language, then so is $R'\tau_B$.

Using the connection expressed by the above theorem and the tree language theory, properties of syntax-directed translations can be proved (e.g., the domain and range are context-free languages).

The Connection between SDTSs and $\mathbf{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$

Proposition

For every SDTS T, one can define a tree bimorphism B in $\mathbf{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ such that $\tau_T = \mathrm{yd}(\tau_B)$.

Proposition

For each quasi-alphabetic tree bimorphism B, one can define a syntax-directed translation schema T such that $\mathrm{yd}(\tau_B) = \tau_T$.

Theorem

The class of syntax-directed translations is effectively equal to the class of translations defined by quasi-alphabetic tree bimorphisms.

Lemma (Preservation of Recognizability)

Quasi-alphabetic tree bimorphisms preserve recognizability, i.e., if $B \in \mathbf{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ and R' is a regular tree language, then so is $R'\tau_B$.

Using the connection expressed by the above theorem and the tree language theory, properties of syntax-directed translations can be proved (e.g., the domain and range are context-free languages).

The Connection between SDTSs and $\mathbf{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$

Proposition

For every SDTS T, one can define a tree bimorphism B in $\mathbf{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ such that $\tau_T = \mathrm{yd}(\tau_B)$.

Proposition

For each quasi-alphabetic tree bimorphism B, one can define a syntax-directed translation schema T such that $\mathrm{yd}(\tau_B) = \tau_T$.

Theorem

The class of syntax-directed translations is effectively equal to the class of translations defined by quasi-alphabetic tree bimorphisms.

Lemma (Preservation of Recognizability)

Quasi-alphabetic tree bimorphisms preserve recognizability, i.e., if $B \in \mathbf{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ and R' is a regular tree language, then so is $R'\tau_B$.

Using the connection expressed by the above theorem and the tree language theory, properties of syntax-directed translations can be proved (e.g., the domain and range are context-free languages).

Classes Equivalent to $\mathbf{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$

The essential feature of the bimorphisms in the class $\mathbf{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ is that the tree homomorphisms are quasi-alphabetic; we may either limit or extend the class of tree languages allowed.

Theorem

The class of all syntax-directed translations is effectively equal to the class of translations defined by the tree bimorphisms belonging to the class $\mathbf{B}(\mathrm{qH}, \mathrm{Loc} \cap \mathrm{DRec}, \mathrm{qH})$.

Theorem

The class of all syntax-directed translations is effectively equal to the class of translations defined by the tree bimorphisms belonging to the class $\mathbf{B}(\mathrm{qH}, \mathrm{Rec}, \mathrm{qH})$.

Classes Equivalent to $\mathbf{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$

The essential feature of the bimorphisms in the class $\mathbf{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ is that the tree homomorphisms are quasi-alphabetic; we may either limit or extend the class of tree languages allowed.

Theorem

The class of all syntax-directed translations is effectively equal to the class of translations defined by the tree bimorphisms belonging to the class $\mathbf{B}(\mathrm{qH}, \mathrm{Loc} \cap \mathrm{DRec}, \mathrm{qH})$.

Theorem

The class of all syntax-directed translations is effectively equal to the class of translations defined by the tree bimorphisms belonging to the class $\mathbf{B}(\mathrm{qH}, \mathrm{Rec}, \mathrm{qH})$.

Classes Equivalent to $\mathbf{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$

The essential feature of the bimorphisms in the class $\mathbf{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ is that the tree homomorphisms are quasi-alphabetic; we may either limit or extend the class of tree languages allowed.

Theorem

The class of all syntax-directed translations is effectively equal to the class of translations defined by the tree bimorphisms belonging to the class $\mathbf{B}(\mathrm{qH}, \mathrm{Loc} \cap \mathrm{DRec}, \mathrm{qH})$.

Theorem

The class of all syntax-directed translations is effectively equal to the class of translations defined by the tree bimorphisms belonging to the class $\mathbf{B}(\mathrm{qH}, \mathrm{Rec}, \mathrm{qH})$.

Closure Properties of $\mathcal{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$

Theorem

The class $\mathcal{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ is closed under inverses.

Theorem

The class $\mathcal{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ is closed under composition.

Closure Properties of $\mathcal{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$

Theorem

The class $\mathcal{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ is closed under inverses.

Theorem

The class $\mathcal{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ is closed under composition.

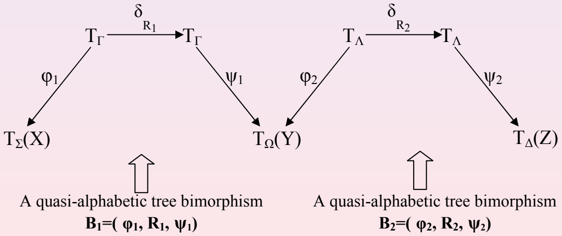
Closure Properties of $\mathcal{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$

Theorem

The class $\mathcal{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ is closed under inverses.

Theorem

The class $\mathcal{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ is closed under composition.



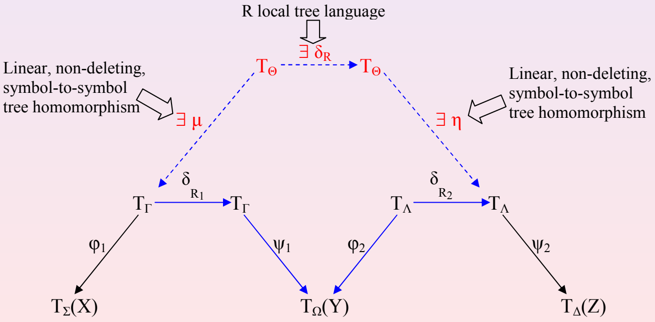
Closure Properties of $\mathcal{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$

Theorem

The class $\mathcal{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ is closed under inverses.

Theorem

The class $\mathcal{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ is closed under composition.



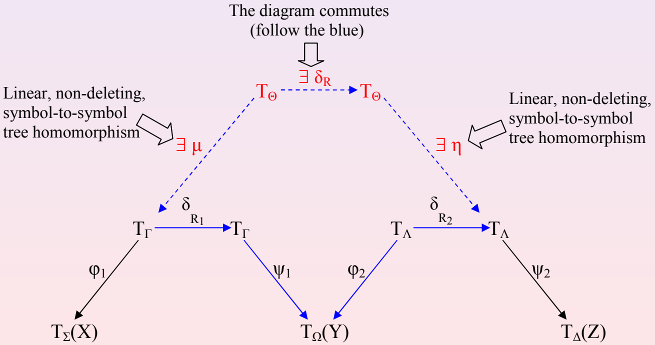
Closure Properties of $\mathcal{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$

Theorem

The class $\mathcal{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ is closed under inverses.

Theorem

The class $\mathcal{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ is closed under composition.



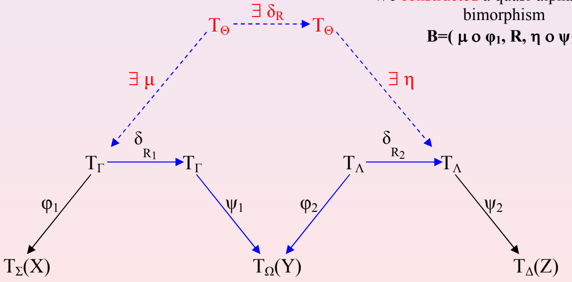
Closure Properties of $\mathcal{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$

Theorem

The class $\mathcal{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ is closed under inverses.

Theorem

The class $\mathcal{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ is closed under composition.



We constructed a quasi-alphabetic tree bimorphism

B=(μ o φ₁, R, η o ψ₁)

Conclusions

What We Did

- We introduce the new class of quasi-alphabetic tree bimorphisms.
- We show that the class of syntax-directed translations is effectively equal to the class of translations defined by quasi-alphabetic tree bimorphisms.
- We show that the class of tree transformations defined by quasi-alphabetic tree bimorphisms is closed under composition and inverses, and preserves recognizability $\Longrightarrow$ we solved (partially) an open problem mentioned by Shierber [Shieber 2004], pp.8: "...the bimorphism characterization of tree transducers has led to a series of composition closure results. Similar techniques may now be applicable to synchronous formalisms, where no composition results are known..."

Other (Recent) Work

- There is a **super class** of tree transformations closed under composition [Maletti 2007].
- Synchronous context-free grammars [Satta & Peserico 2005] perform the same translations as quasi-alphabetic tree bimorphisms [Tîrnăucă 2007].

Future Work

- to investigate other closure properties of $\mathbf{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ (e.g., intersection);
- to see if other synchronous rewriting formalisms can be modeled in terms of quasi-alphabetic tree bimorphisms (e.g., inversion grammars [Wu 1997]).

Conclusions

What We Did

- We introduce the new class of quasi-alphabetic tree bimorphisms.
- We show that the class of syntax-directed translations is effectively equal to the class of translations defined by quasi-alphabetic tree bimorphisms.
- We show that the class of tree transformations defined by quasi-alphabetic tree bimorphisms is closed under composition and inverses, and preserves recognizability $\Longrightarrow$ we solved (partially) an open problem mentioned by Shierber [Shieber 2004], pp.8: "...the bimorphism characterization of tree transducers has led to a series of composition closure results. Similar techniques may now be applicable to synchronous formalisms, where no composition results are known..."

Other (Recent) Work

- There is a **super class** of tree transformations closed under composition [Maletti 2007].

Future Work

- to investigate other closure properties of $\mathbf{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ (e.g., intersection);

Conclusions

What We Did

- We introduce the new class of quasi-alphabetic tree bimorphisms.
- We show that the class of syntax-directed translations is effectively equal to the class of translations defined by quasi-alphabetic tree bimorphisms.
- We show that the class of tree transformations defined by quasi-alphabetic tree bimorphisms is closed under composition and inverses, and preserves recognizability $\Longrightarrow$ we solved (partially) an open problem mentioned by Shierber [Shieber 2004], pp.8: "...the bimorphism characterization of tree transducers has led to a series of composition closure results. Similar techniques may now be applicable to synchronous formalisms, where no composition results are known..."

Other (Recent) Work

- There is a **super class** of tree transformations closed under composition [Maletti 2007].
- **Synchronous context-free grammars** [Satta & Peserico 2005], perform the same translations as quasi-alphabetic tree bimorphisms [Tîrnăucă 2007].

Future Work

- to investigate other closure properties of $\mathbf{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ (e.g., intersection);

Conclusions

What We Did

- We introduce the new class of quasi-alphabetic tree bimorphisms.
- We show that the class of syntax-directed translations is effectively equal to the class of translations defined by quasi-alphabetic tree bimorphisms.
- We show that the class of tree transformations defined by quasi-alphabetic tree bimorphisms is closed under composition and inverses, and preserves recognizability $\Longrightarrow$ we solved (partially) an open problem mentioned by Shierber [Shieber 2004], pp.8: "...the bimorphism characterization of tree transducers has led to a series of composition closure results. Similar techniques may now be applicable to synchronous formalisms, where no composition results are known..."

Other (Recent) Work

- There is a **super class** of tree transformations closed under composition [Maletti 2007].
- **Synchronous context-free grammars** [Satta & Peserico 2005], perform the same translations as quasi-alphabetic tree bimorphisms [Tîrnăucă 2007].

Future Work

- to investigate other closure properties of $\mathbf{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ (e.g., intersection);

Conclusions

What We Did

- We introduce the new class of quasi-alphabetic tree bimorphisms.
- We show that the class of syntax-directed translations is effectively equal to the class of translations defined by quasi-alphabetic tree bimorphisms.
- We show that the class of tree transformations defined by quasi-alphabetic tree bimorphisms is closed under composition and inverses, and preserves recognizability $\Longrightarrow$ we solved (partially) an open problem mentioned by Shierber [Shieber 2004], pp.8: "...the bimorphism characterization of tree transducers has led to a series of composition closure results. Similar techniques may now be applicable to synchronous formalisms, where no composition results are known..."

Other (Recent) Work

- There is a **super class** of tree transformations closed under composition [Maletti 2007].
- **Synchronous context-free grammars** [Satta & Peserico 2005], perform the same translations as quasi-alphabetic tree bimorphisms [Tîrnăucă 2007].

Future Work

- to investigate other closure properties of **B**$(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ (e.g., intersection);
- to see if other synchronous rewriting formalisms can be model in terms of quasi-alphabetic tree bimorphisms (e.g., inversion grammars [Wu 1997]).

Conclusions

What We Did

- We introduce the new class of quasi-alphabetic tree bimorphisms.
- We show that the class of syntax-directed translations is effectively equal to the class of translations defined by quasi-alphabetic tree bimorphisms.
- We show that the class of tree transformations defined by quasi-alphabetic tree bimorphisms is closed under composition and inverses, and preserves recognizability $\Longrightarrow$ we solved (partially) an open problem mentioned by Shierber [Shieber 2004], pp.8: "...the bimorphism characterization of tree transducers has led to a series of composition closure results. Similar techniques may now be applicable to synchronous formalisms, where no composition results are known..."

Other (Recent) Work

- There is a **super class** of tree transformations closed under composition [Maletti 2007].
- **Synchronous context-free grammars** [Satta & Peserico 2005], perform the same translations as quasi-alphabetic tree bimorphisms [Tîrnăucă 2007].

Future Work

- to investigate other closure properties of $\mathbf{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ (e.g., intersection);
- to see if other synchronous rewriting formalisms can be model in terms of quasi-alphabetic tree bimorphisms (e.g., inversion grammars [Wu 1997]).

Conclusions

What We Did

- We introduce the new class of quasi-alphabetic tree bimorphisms.
- We show that the class of syntax-directed translations is effectively equal to the class of translations defined by quasi-alphabetic tree bimorphisms.
- We show that the class of tree transformations defined by quasi-alphabetic tree bimorphisms is closed under composition and inverses, and preserves recognizability $\Longrightarrow$ we solved (partially) an open problem mentioned by Shierber [Shieber 2004], pp.8: "...the bimorphism characterization of tree transducers has led to a series of composition closure results. Similar techniques may now be applicable to synchronous formalisms, where no composition results are known..."

Other (Recent) Work

- There is a **super class** of tree transformations closed under composition [Maletti 2007].
- **Synchronous context-free grammars** [Satta & Peserico 2005], perform the same translations as quasi-alphabetic tree bimorphisms [Tîrnăucă 2007].

Future Work

- to investigate other closure properties of $\mathbf{B}(\mathrm{qH}, \mathrm{Loc}, \mathrm{qH})$ (e.g., intersection);
- to see if other synchronous rewriting formalisms can be model in terms of quasi-alphabetic tree bimorphisms (e.g., inversion grammars [Wu 1997]).

References I

A.V. Aho, J.D. Ullman:
The Theory of Parsing, Translation, and Compiling, Volume I: Parsing.
Prentice Hall Professional Technical Reference, New Jersey, 1972.

K. Knight, J. Graehl:
An overview of probabilistic tree transducers for natural language processing.
Proc. of CICLing'05. LNCS 3406: 1-24, 2005.

K. Knight, M. Hopkins, J. Graehl:
Extended top-down tree transducers.
Submitted to HLT-NAACL '06, 2006.

S. Shieber:
Synchronous grammars as tree transducers.
Proc. of TAG+ 7, 2004.

A. Maletti:
Compositions of extended top-down tree transducers
Proc. of the LATA '07, 379–390, Tarragona, 2007.

References II

- C. I. Tîrnăucă:
  Synchronous context-free grammars by means of tree bimorphisms.
  Proc. of the ForLing ’07, Budapest, 2007.
- D. Wu:
  Stochastic inversion transduction grammars and bilingual parsing of parallel corpora.
  Computational Linguistics **23** (3), 377–403, 1997.
- G. Satta, E. Peserico.
  Some computational complexity results for synchronous context-free grammars.
  Proc. of the HLT ’05/EMNLP ’05 , 803–810, Vancouver, 2005.

That's all folks!

Thank you!