Model Syntax-Directed Translations by Tree Transducers[1]

Cătălin Ionuţ Tîrnăucă

Research Group on Mathematical Linguistics
Rovira i Virgili University
Tarragona, Spain
catalinionut.tirnauca@estudiants.urv.cat

31st of August, 2009

[1] This work was financially supported by the ESF programme “Automata: from Mathematics to Applications (AutoMathA)” and by the Spanish Minister of Education and Research’s project MTM-2007-63422.

Syntax-Based Machine Translation

- **Syntax-based machine translation** was established to meet with the demand for systems used in practical translations between natural languages [Knight 2007]

- An ideal such system should [Knight 2007]
  1. capture syntax-sensitive transformations (i.e., tree transformations)
  2. do difficult rotations (reorder parts of sentences)
  3. preserve recognizability of tree languages
  4. be closed under inverses
  5. have composability (smaller parts easier to test, train, etc.)
  6. be efficiently trainable

Syntax-Based Machine Translation

- **Syntax-based machine translation** was established to meet with the demand for systems used in practical translations between natural languages [Knight 2007]
- An ideal such system should [Knight 2007]
  1. capture syntax-sensitive transformations (i.e., tree transformations)
  2. do difficult rotations (reorder parts of sentences)
  3. preserve recognizability of tree languages
  4. be closed under inverses
  5. have composability (smaller parts easier to test, train, etc.)
  6. be efficiently trainable

Syntax-Based Machine Translation

- **Syntax-based machine translation** was established to meet with the demand for systems used in practical translations between natural languages [Knight 2007]

- An ideal such system should [Knight 2007]
  1. capture syntax-sensitive transformations (i.e., tree transformations)
  2. do difficult rotations (reorder parts of sentences)
  3. preserve recognizability of tree languages
  4. be closed under inverses
  5. have composability (smaller parts easier to test, train, etc.)
  6. be efficiently trainable

Syntax-Based Machine Translation

- **Syntax-based machine translation** was established to meet with the demand for systems used in practical translations between natural languages [Knight 2007]

- An ideal such system should [Knight 2007]
  1. capture syntax-sensitive transformations (i.e., tree transformations)
  2. do difficult rotations (reorder parts of sentences)
  3. preserve recognizability of tree languages
  4. be closed under inverses
  5. have composability (smaller parts easier to test, train, etc.)
  6. be efficiently trainable

Syntax-Based Machine Translation

- **Syntax-based machine translation** was established to meet with the demand for systems used in practical translations between natural languages [Knight 2007]
- An ideal such system should [Knight 2007]
  1. capture syntax-sensitive transformations (i.e., tree transformations)
  2. do difficult rotations (reorder parts of sentences)
  3. preserve recognizability of tree languages
  4. be closed under inverses
  5. have composability (smaller parts easier to test, train, etc.)
  6. be efficiently trainable

Syntax-Based Machine Translation

- **Syntax-based machine translation** was established to meet with the demand for systems used in practical translations between natural languages [Knight 2007]

- An ideal such system should [Knight 2007]
  1. capture syntax-sensitive transformations (i.e., tree transformations)
  2. do difficult rotations (reorder parts of sentences)
  3. preserve recognizability of tree languages
  4. be closed under inverses
  5. have composability (smaller parts easier to test, train, etc.)
  6. be efficiently trainable

Syntax-Based Machine Translation

- **Syntax-based machine translation** was established to meet with the demand for systems used in practical translations between natural languages [Knight 2007]

- An ideal such system should [Knight 2007]
  1. capture syntax-sensitive transformations (i.e., tree transformations)
  2. do difficult rotations (reorder parts of sentences)
  3. preserve recognizability of tree languages
  4. be closed under inverses
  5. have composability (smaller parts easier to test, train, etc.)
  6. be efficiently trainable

Syntax-Based Machine Translation

- **Syntax-based machine translation** was established to meet with the demand for systems used in practical translations between natural languages [Knight 2007]
- An ideal such system should [Knight 2007]
  1. capture syntax-sensitive transformations (i.e., tree transformations)
  2. do difficult rotations (reorder parts of sentences)
  3. preserve recognizability of tree languages
  4. be closed under inverses
  5. have composability (smaller parts easier to test, train, etc.)
  6. be efficiently trainable

How to Capture Tree Transformations?

1. **Synchronous grammars**
   - naturally define all kinds of difficult rotations: e.g. Arabic-English
   - are trainable
   - very few mathematical properties are known [Shieber 2004]
2. **Tree bimorphisms**
   - algebraic mechanisms, harder to implement (no available tools)
   - no trainability results are known
   - naturally closed under inverses
   - composition and preservation of recognizability easier to establish by imposing suitable restrictions on their constituents [Arnold & Dauchet 1982, Bozapalidis 1992]
3. **Tree transducers**
   - easy to implement: many available tools, e.g. TIBURON/ISI
   - are trainable [Graehl,Knight & May 2008]
   - closure under composition and preservation of recognizability does not hold for the main types [Gécseg & Steinby 1984, Knight 2007]

How to Capture Tree Transformations?

1. **Synchronous grammars**
   - naturally define all kinds of difficult rotations: e.g. Arabic-English
   - are trainable
   - very few mathematical properties are known [Shieber 2004]
2. **Tree bimorphisms**
   - algebraic mechanisms, harder to implement (no available tools)
   - no trainability results are known
   - naturally closed under inverses
   - composition and preservation of recognizability easier to establish by imposing suitable restrictions on their constituents [Arnold & Dauchet 1982, Bozapalidis 1992]
3. **Tree transducers**
   - easy to implement: many available tools, e.g. TIBURON/ISI
   - are trainable [Graehl,Knight & May 2008]
   - closure under composition and preservation of recognizability does not hold for the main types [Gécseg & Steinby 1984, Knight 2007]

How to Capture Tree Transformations?

1. **Synchronous grammars**
   - naturally define all kinds of difficult rotations: e.g. Arabic-English
   - are trainable
   - very few mathematical properties are known [Shieber 2004]
2. **Tree bimorphisms**
   - algebraic mechanisms, harder to implement (no available tools)
   - no trainability results are known
   - naturally closed under inverses
   - composition and preservation of recognizability easier to establish by imposing suitable restrictions on their constituents [Arnold & Dauchet 1982, Bozapalidis 1992]
3. **Tree transducers**
   - easy to implement: many available tools, e.g. TIBURON/ISI
   - are trainable [Graehl,Knight & May 2008]
   - closure under composition and preservation of recognizability does not hold for the main types [Gécseg & Steinby 1984, Knight 2007]

Synchronous Grammars as Tree Bimorphisms

- Why not relate synchronous grammars and tree transducers via tree bimorphisms? [Shieber 2004]
  1. mathematical framework of synchronous grammars is improved
  2. link similar tree transformations defining formalisms
  3. the tree bimorphisms are easier to implement
- [Steinby & Tîrnăucă 2007] introduced the class of **quasi-alphabetic tree bimorphisms** which:
  1. is effectively equal to syntax-directed translation (SDT) devices (in terms of translations)
  2. is closed under composition and inverses, and preserves recognizability
  3. naturally describes the tree transformations defined by SDTs
- Moreover, we:
  1. introduced the **quasi-alphabetic tree transducer**
  2. proved that this type of transducer computes the tree transformations defined by quasi-alphabetic tree bimorphisms

Synchronous Grammars as Tree Bimorphisms

- Why not relate synchronous grammars and tree transducers via tree bimorphisms? [Shieber 2004]
  1. mathematical framework of synchronous grammars is improved
  2. link similar tree transformations defining formalisms
  3. the tree bimorphisms are easier to implement
- [Steinby & Tîrnăucă 2007] introduced the class of **quasi-alphabetic tree bimorphisms** which:
  1. is effectively equal to syntax-directed translation (SDT) devices (in terms of translations)
  2. is closed under composition and inverses, and preserves recognizability
  3. naturally describes the tree transformations defined by SDTs
- Moreover, we:
  1. introduced the **quasi-alphabetic tree transducer**
  2. proved that this type of transducer computes the tree transformations defined by quasi-alphabetic tree bimorphisms

Synchronous Grammars as Tree Bimorphisms

- Why not relate synchronous grammars and tree transducers via tree bimorphisms? [Shieber 2004]

  1. mathematical framework of synchronous grammars is improved
  2. link similar tree transformations defining formalisms
  3. the tree bimorphisms are easier to implement

- [Steinby & Tîrnăucă 2007] introduced the class of **quasi-alphabetic tree bimorphisms** which:

  1. is effectively equal to syntax-directed translation (SDT) devices (in terms of translations)
  2. is closed under composition and inverses, and preserves recognizability
  3. naturally describes the tree transformations defined by SDTs

- Moreover, we:

  1. introduced the **quasi-alphabetic tree transducer**
  2. proved that this type of transducer computes the tree transformations defined by quasi-alphabetic tree bimorphisms

Synchronous Grammars as Tree Bimorphisms

- Why not relate synchronous grammars and tree transducers via tree bimorphisms? [Shieber 2004]
  1. mathematical framework of synchronous grammars is improved
  2. link similar tree transformations defining formalisms
  3. the tree bimorphisms are easier to implement
- [Steinby & Tîrnăucă 2007] introduced the class of **quasi-alphabetic tree bimorphisms** which:
  1. is effectively equal to syntax-directed translation (SDT) devices (in terms of translations)
  2. is closed under composition and inverses, and preserves recognizability
  3. naturally describes the tree transformations defined by SDTs
- Moreover, we:
  1. introduced the **quasi-alphabetic tree transducer**
  2. proved that this type of transducer computes the tree transformations defined by quasi-alphabetic tree bimorphisms

Synchronous Grammars as Tree Bimorphisms

- Why not relate synchronous grammars and tree transducers via tree bimorphisms? [Shieber 2004]
  1. mathematical framework of synchronous grammars is improved
  2. link similar tree transformations defining formalisms
  3. the tree bimorphisms are easier to implement
- [Steinby & Tîrnăucă 2007] introduced the class of **quasi-alphabetic tree bimorphisms** which:
  1. is effectively equal to syntax-directed translation (SDT) devices (in terms of translations)
  2. is closed under composition and inverses, and preserves recognizability
  3. naturally describes the tree transformations defined by SDTs
- Moreover, we:
  1. introduced the **quasi-alphabetic tree transducer**
  2. proved that this type of transducer computes the tree transformations defined by quasi-alphabetic tree bimorphisms

Synchronous Grammars as Tree Bimorphisms

- Why not relate synchronous grammars and tree transducers via tree bimorphisms? [Shieber 2004]

  1. mathematical framework of synchronous grammars is improved
  2. link similar tree transformations defining formalisms
  3. the tree bimorphisms are easier to implement

- [Steinby & Tîrnăucă 2007] introduced the class of **quasi-alphabetic tree bimorphisms** which:

  1. is effectively equal to syntax-directed translation (SDT) devices (in terms of translations)
  2. is closed under composition and inverses, and preserves recognizability
  3. naturally describes the tree transformations defined by SDTs

- Moreover, we:

  1. introduced the **quasi-alphabetic tree transducer**
  2. proved that this type of transducer computes the tree transformations defined by quasi-alphabetic tree bimorphisms

Synchronous Grammars as Tree Bimorphisms

- Why not relate synchronous grammars and tree transducers via tree bimorphisms? [Shieber 2004]
  1. mathematical framework of synchronous grammars is improved
  2. link similar tree transformations defining formalisms
  3. the tree bimorphisms are easier to implement
- [Steinby & Tîrnăucă 2007] introduced the class of **quasi-alphabetic tree bimorphisms** which:
  1. is effectively equal to syntax-directed translation (SDT) devices (in terms of translations)
  2. is closed under composition and inverses, and preserves recognizability
  3. naturally describes the tree transformations defined by SDTs
- Moreover, we:
  1. introduced the **quasi-alphabetic tree transducer**
  2. proved that this type of transducer computes the tree transformations defined by quasi-alphabetic tree bimorphisms

Synchronous Grammars as Tree Bimorphisms

- Why not relate synchronous grammars and tree transducers via tree bimorphisms? [Shieber 2004]
  1. mathematical framework of synchronous grammars is improved
  2. link similar tree transformations defining formalisms
  3. the tree bimorphisms are easier to implement
- [Steinby & Tîrnăucă 2007] introduced the class of **quasi-alphabetic tree bimorphisms** which:
  1. is effectively equal to syntax-directed translation (SDT) devices (in terms of translations)
  2. is closed under composition and inverses, and preserves recognizability
  3. naturally describes the tree transformations defined by SDTs
- Moreover, we:
  1. introduced the **quasi-alphabetic tree transducer**
  2. proved that this type of transducer computes the tree transformations defined by quasi-alphabetic tree bimorphisms

Synchronous Grammars as Tree Bimorphisms

- Why not relate synchronous grammars and tree transducers via tree bimorphisms? [Shieber 2004]
  1. mathematical framework of synchronous grammars is improved
  2. link similar tree transformations defining formalisms
  3. the tree bimorphisms are easier to implement
- [Steinby & Tîrnăucă 2007] introduced the class of **quasi-alphabetic tree bimorphisms** which:
  1. is effectively equal to syntax-directed translation (SDT) devices (in terms of translations)
  2. is closed under composition and inverses, and preserves recognizability
  3. naturally describes the tree transformations defined by SDTs
- Moreover, we:
  1. introduced the **quasi-alphabetic tree transducer**
  2. proved that this type of transducer computes the tree transformations defined by quasi-alphabetic tree bimorphisms

Synchronous Grammars as Tree Bimorphisms

- Why not relate synchronous grammars and tree transducers via tree bimorphisms? [Shieber 2004]
  1. mathematical framework of synchronous grammars is improved
  2. link similar tree transformations defining formalisms
  3. the tree bimorphisms are easier to implement
- [Steinby & Tîrnăucă 2007] introduced the class of **quasi-alphabetic tree bimorphisms** which:
  1. is effectively equal to syntax-directed translation (SDT) devices (in terms of translations)
  2. is closed under composition and inverses, and preserves recognizability
  3. naturally describes the tree transformations defined by SDTs
- Moreover, we:
  1. introduced the **quasi-alphabetic tree transducer**
  2. proved that this type of transducer computes the tree transformations defined by quasi-alphabetic tree bimorphisms

Tree Languages - Basic Facts

- Σ ranked alphabet, X leaf alphabet (variables), Ξ formal variables
- $\Xi_m = \{\xi_1, \xi_2, \ldots, \xi_m\}$ (keeps track of the subtrees)
- $T_\Sigma(X)$ = set of all trees labeled by symbols in Σ and variables X
- **tree languages** = subsets of $T_\Sigma(X)$
- $\mathrm{yd}(t)$ = the **yield** of the tree t, i.e., the leaf symbols in t, read from left to right
- $t[t_1, \ldots, t_n]$ = replace every occurrence of ξ_i in t by t_i, $\forall i \in [n]$
- **tree homomorphism** = are determined by
  - one mapping (φ_X) to transform leaf variables into output trees and
  - a family of mappings (φ_m) to transform the input symbols into output trees with formal variables as leaves
- **tree recognizer** = recognizes regular tree languages and uses states (some initial ones to accept) to process the symbols in a top-down fashion:

$$q(f(\xi_1, \xi_2, \xi_3)) \to f(q_1(\xi_1), q_2(\xi_2), q_3(\xi_3))$$

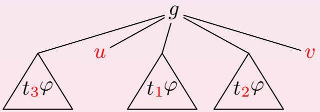
Quasi-Alphabetic Tree Homomorphisms

Formal Definition

A tree homomorphism $\varphi\colon T_\Sigma(X) \to T_\Delta(Y)$ is **quasi-alphabetic** if

1. is linear, i.e., no copying is allowed
2. is complete, i.e., no subtree information is lost
3. each variable in X is mapped into a variable in Y
4. maps each input symbol to an output symbol possibly with some output leaf symbols as direct subtrees and formal variables have to occur as direct subtrees of the root output symbol, possibly in other order

$\varphi(f(t_1,t_2,t_3)) =$



with $u, v \in Y$

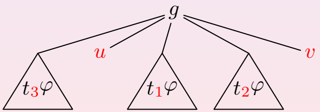
Quasi-Alphabetic Tree Homomorphisms

Formal Definition

A tree homomorphism $\varphi\colon T_\Sigma(X) \to T_\Delta(Y)$ is **quasi-alphabetic** if

1. is linear, i.e., no copying is allowed
2. is complete, i.e., no subtree information is lost
3. each variable in X is mapped into a variable in Y
4. maps each input symbol to an output symbol possibly with some output leaf symbols as direct subtrees and formal variables have to occur as direct subtrees of the root output symbol, possibly in other order

$\varphi(f(t_1, t_2, t_3)) =$



with $u, v \in Y$

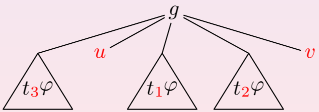
Quasi-Alphabetic Tree Homomorphisms

Formal Definition

A tree homomorphism $\varphi\colon T_\Sigma(X) \to T_\Delta(Y)$ is **quasi-alphabetic** if

1. is linear, i.e., no copying is allowed
2. is complete, i.e., no subtree information is lost
3. each variable in X is mapped into a variable in Y
4. maps each input symbol to an output symbol possibly with some output leaf symbols as direct subtrees and formal variables have to occur as direct subtrees of the root output symbol, possibly in other order

$\varphi(f(t_1, t_2, t_3)) =$



with $u, v \in Y$

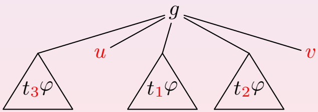
Quasi-Alphabetic Tree Homomorphisms

Formal Definition

A tree homomorphism $\varphi\colon T_\Sigma(X) \to T_\Delta(Y)$ is **quasi-alphabetic** if

1. is linear, i.e., no copying is allowed
2. is complete, i.e., no subtree information is lost
3. each variable in X is mapped into a variable in Y
4. maps each input symbol to an output symbol possibly with some output leaf symbols as direct subtrees and formal variables have to occur as direct subtrees of the root output symbol, possibly in other order

$\varphi(f(t_1, t_2, t_3)) =$



with $u, v \in Y$

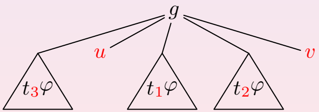
Quasi-Alphabetic Tree Homomorphisms

Formal Definition

A tree homomorphism $\varphi\colon T_\Sigma(X) \to T_\Delta(Y)$ is **quasi-alphabetic** if

1. is linear, i.e., no copying is allowed
2. is complete, i.e., no subtree information is lost
3. each variable in X is mapped into a variable in Y
4. maps each input symbol to an output symbol possibly with some output leaf symbols as direct subtrees and formal variables have to occur as direct subtrees of the root output symbol, possibly in other order

$\varphi(f(t_1, t_2, t_3)) =$



with $u, v \in Y$

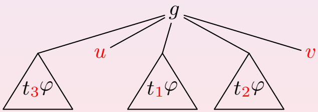
Quasi-Alphabetic Tree Homomorphisms

Formal Definition

A tree homomorphism $\varphi\colon T_\Sigma(X) \to T_\Delta(Y)$ is **quasi-alphabetic** if

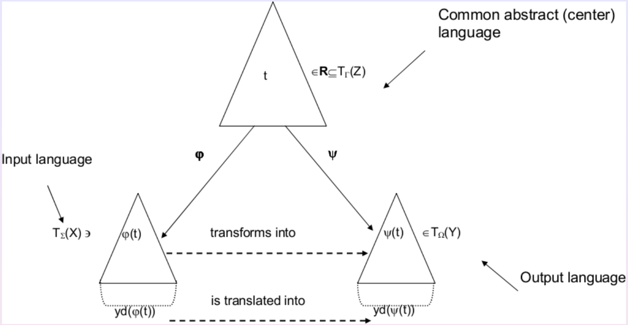
1. is linear, i.e., no copying is allowed
2. is complete, i.e., no subtree information is lost
3. each variable in X is mapped into a variable in Y
4. maps each input symbol to an output symbol possibly with some output leaf symbols as direct subtrees and formal variables have to occur as direct subtrees of the root output symbol, possibly in other order

$\varphi(f(t_1, t_2, t_3)) =$



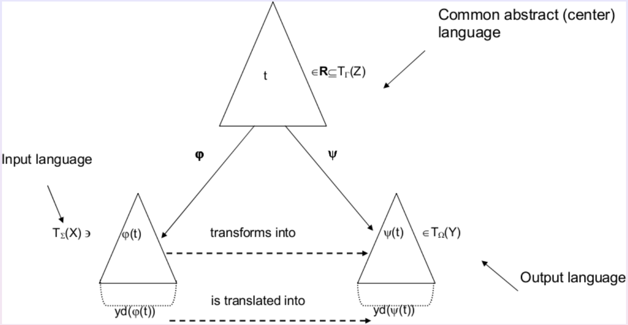
with $u, v \in Y$

Quasi-Alphabetic Tree Bimorphism



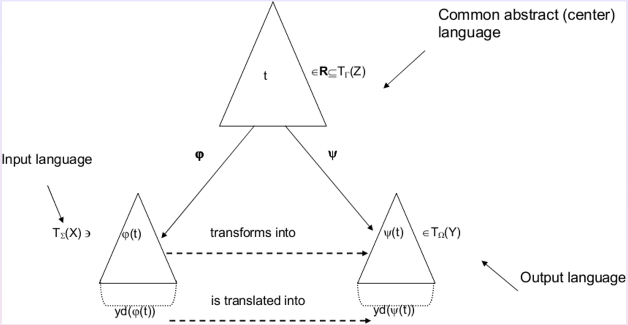
- $B = (\varphi, L, \psi)$ is a **quasi-alphabetic tree bimorphism** if
  - the input tree homomorphism φ is quasi-alphabetic
  - the center language L is regular
  - the output tree homomorphism ψ is quasi-alphabetic
- **tree transformation** defined by B: $\tau_B = \{(\varphi(t), \psi(t)) \mid t \in L\}$
- **translation** defined by B: $\mathrm{yd}(\tau_B) = \{(\mathrm{yd}(s), \mathrm{yd}(t)) \mid (s, t) \in \tau_B\}$
- **quasi-alphabetic tree relations**=class of tree transformations defined by such bimorphisms

Quasi-Alphabetic Tree Bimorphism



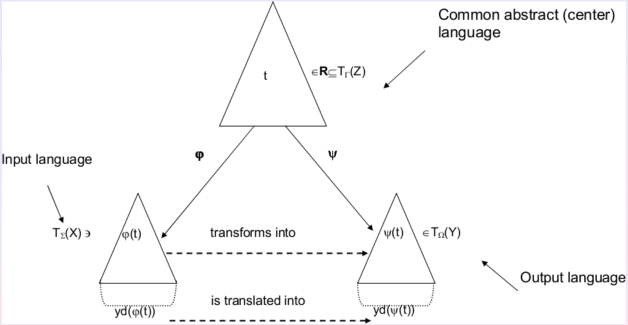
- $B = (\varphi, L, \psi)$ is a **quasi-alphabetic tree bimorphism** if
  - the input tree homomorphism φ is quasi-alphabetic
  - the center language L is regular
  - the output tree homomorphism ψ is quasi-alphabetic
- **tree transformation** defined by B: $\tau_B = \{(\varphi(t), \psi(t)) \mid t \in L\}$
- **translation** defined by B: $\mathrm{yd}(\tau_B) = \{(\mathrm{yd}(s), \mathrm{yd}(t)) \mid (s, t) \in \tau_B\}$
- **quasi-alphabetic tree relations**=class of tree transformations defined by such bimorphisms

Quasi-Alphabetic Tree Bimorphism



- $B = (\varphi, L, \psi)$ is a **quasi-alphabetic tree bimorphism** if
  - the input tree homomorphism φ is quasi-alphabetic
  - the center language L is regular
  - the output tree homomorphism ψ is quasi-alphabetic
- **tree transformation** defined by B: $\tau_B = \{(\varphi(t), \psi(t)) \mid t \in L\}$
- **translation** defined by B: $\text{yd}(\tau_B) = \{(\text{yd}(s), \text{yd}(t)) \mid (s, t) \in \tau_B\}$
- **quasi-alphabetic tree relations**=class of tree transformations defined by such bimorphisms

Quasi-Alphabetic Tree Bimorphism



- $B = (\varphi, L, \psi)$ is a **quasi-alphabetic tree bimorphism** if
  - the input tree homomorphism φ is quasi-alphabetic
  - the center language L is regular
  - the output tree homomorphism ψ is quasi-alphabetic
- **tree transformation** defined by B: $\tau_B = \{(\varphi(t), \psi(t)) \mid t \in L\}$
- **translation** defined by B: $\mathrm{yd}(\tau_B) = \{(\mathrm{yd}(s), \mathrm{yd}(t)) \mid (s, t) \in \tau_B\}$
- **quasi-alphabetic tree relations**=class of tree transformations defined by such bimorphisms

Quasi-Alphabetic Tree Transducer

Two Necessary Notations

$\Sigma(X) = \{f(x_1, \ldots, x_k) \mid f \in \Sigma_k, x_1, \ldots, x_k \in X\}$
$C^n_\Sigma(X) = \{t \in T_\Sigma(X \cup \Xi_n) \mid \forall i \in [n]$, each ξ_i appears once in $t\}$

Definition

A system $M = (Q, \Sigma, X, \Omega, Y, I, R)$ is a **quasi-alphabetic tree transducer** if

- $Q = Q_1$ is a ranked alphabet of **states** with $Q \cap (X \cup \Sigma \cup Y \cup \Omega) = \emptyset$
- X and Σ are the **input alphabets**, and Y and Ω are the **output alphabets**
- $I \subseteq Q$ is a set of **initial states**
- R is a finite set of **rules** each one of the form

  (Q1) $q(x) \to y$, $x \in X$ and $y \in Y$, or

$q(s) \to t$, with (for some $m \geqslant 0$)
$s \in \Sigma(X \cup \Xi_m) \cap C^m_\Sigma(X)$ and
$t \in \Omega(Y \cup Q(\Xi_m)) \cap C^m_{\Omega \cup Q}(Y)$

(Q2) $$q(f(\xi_1, x, \xi_2)) \to g(y, q(\xi_2), y, q(\xi_1))$$

Tree Transformation Computed by M

$$\tau_M = \{(s, t) \in T_\Sigma(X) \times T_\Omega(Y) \mid \exists q \in I \colon q(s) \Rightarrow^*_M t\}$$

Quasi-Alphabetic Tree Transducer

Two Necessary Notations

$\Sigma(X) = \{f(x_1, \ldots, x_k) \mid f \in \Sigma_k, x_1, \ldots, x_k \in X\}$
$C^n_\Sigma(X) = \{t \in T_\Sigma(X \cup \Xi_n) \mid \forall i \in [n]$, each ξ_i appears once in $t\}$

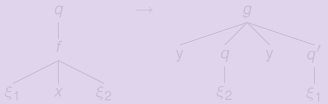
Definition

A system $M = (Q, \Sigma, X, \Omega, Y, I, R)$ is a **quasi-alphabetic tree transducer** if

- $Q = Q_1$ is a ranked alphabet of **states** with $Q \cap (X \cup \Sigma \cup Y \cup \Omega) = \emptyset$
- X and Σ are the **input alphabets**, and Y and Ω are the **output alphabets**
- $I \subseteq Q$ is a set of **initial states**
- R is a finite set of **rules** each one of the form
  (Q1) $q(x) \to y$, $x \in X$ and $y \in Y$, or

$q(s) \to t$, with (for some $m \geqslant 0$)
$s \in \Sigma(X \cup \Xi_m) \cap C^m_\Sigma(X)$ and
$t \in \Omega(Y \cup Q(\Xi_m)) \cap C^m_{\Omega \cup Q}(Y)$

(Q2)



Tree Transformation Computed by M

$$\tau_M = \{(s, t) \in T_\Sigma(X) \times T_\Omega(Y) \mid \exists q \in I\colon q(s) \Rightarrow^*_M t\}$$

Quasi-Alphabetic Tree Transducer

Two Necessary Notations

$\Sigma(X) = \{f(x_1, \ldots, x_k) \mid f \in \Sigma_k, x_1, \ldots, x_k \in X\}$
$C^n_\Sigma(X) = \{t \in T_\Sigma(X \cup \Xi_n) \mid \forall i \in [n], \text{each } \xi_i \text{ appears once in } t\}$

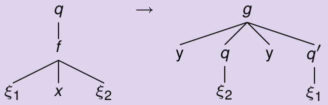
Definition

A system $M = (Q, \Sigma, X, \Omega, Y, I, R)$ is a **quasi-alphabetic tree transducer** if

- $Q = Q_1$ is a ranked alphabet of **states** with $Q \cap (X \cup \Sigma \cup Y \cup \Omega) = \emptyset$
- X and Σ are the **input alphabets**, and Y and Ω are the **output alphabets**
- $I \subseteq Q$ is a set of **initial states**
- R is a finite set of **rules** each one of the form
  (Q1) $q(x) \to y$, $x \in X$ and $y \in Y$, or

$q(s) \to t$, with (for some $m \geqslant 0$)
$s \in \Sigma(X \cup \Xi_m) \cap C^m_\Sigma(X)$ and
$t \in \Omega(Y \cup Q(\Xi_m)) \cap C^m_{\Omega \cup Q}(Y)$

(Q2)



Tree Transformation Computed by M

$$\tau_M = \{(s, t) \in T_\Sigma(X) \times T_\Omega(Y) \mid \exists q \in I \colon q(s) \Rightarrow^*_M t\}$$

Quasi-Alphabetic Tree Transducer

Two Necessary Notations

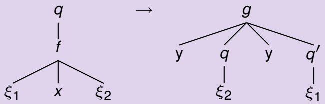
$\Sigma(X) = \{f(x_1, \ldots, x_k) \mid f \in \Sigma_k, x_1, \ldots, x_k \in X\}$
$C^n_\Sigma(X) = \{t \in T_\Sigma(X \cup \Xi_n) \mid \forall i \in [n]$, each ξ_i appears once in $t\}$

Definition

A system $M = (Q, \Sigma, X, \Omega, Y, I, R)$ is a **quasi-alphabetic tree transducer** if

- $Q = Q_1$ is a ranked alphabet of **states** with $Q \cap (X \cup \Sigma \cup Y \cup \Omega) = \emptyset$
- X and Σ are the **input alphabets**, and Y and Ω are the **output alphabets**
- $I \subseteq Q$ is a set of **initial states**
- R is a finite set of **rules** each one of the form
  (Q1) $q(x) \to y$, $x \in X$ and $y \in Y$, or

$q(s) \to t$, with (for some $m \geqslant 0$)
$s \in \Sigma(X \cup \Xi_m) \cap C^m_\Sigma(X)$ and
$t \in \Omega(Y \cup Q(\Xi_m)) \cap C^m_{\Omega \cup Q}(Y)$

(Q2) 

Tree Transformation Computed by M

$$\tau_M = \{(s, t) \in T_\Sigma(X) \times T_\Omega(Y) \mid \exists q \in I \colon q(s) \Rightarrow^*_M t\}$$

Main Result - Sketch of the Proof

Theorem

The set of all quasi-alphabetic tree relations is effectively equal to the class of all tree transformations computed by quasi-alphabetic tree transducers.

Sketch of the Proof (Tree Bimorphism to Tree Transducer)

1. Let $B = (\varphi, L, \psi)$ be a quasi-alphabetic tree bimorphism ($L \subseteq T_\Gamma(Z)$)
2. Assume w.l.o.g. that formal variables appear in φ in order ($\xi_1, \xi_2, \ldots$ etc.)
3. L is recognizable, so $\exists A = (Q, R, I)$ tree recognizer such that $L = T(A)$
4. Construct a set R' from R:

   Add $q(\varphi_Z(z)) \to \psi_Z(z)$ to R', for every $q(z) \to z$ in R

5. Take $M = (Q, \Sigma, X, \Omega, Y, I, R')$ quasi-alphabetic tree transducer
6. Then $\tau_M = \tau_B$

Example

Let $f \in \Gamma_3$, $\varphi_3(f) = g(\xi_1, x, \xi_2, \xi_3, x)$ and $\psi_3(f) = h(y, \xi_1, \xi_3, \xi_2)$. Then:

Main Result - Sketch of the Proof

Theorem

The set of all quasi-alphabetic tree relations is effectively equal to the class of all tree transformations computed by quasi-alphabetic tree transducers.

Sketch of the Proof (Tree Bimorphism to Tree Transducer)

1. Let $B = (\varphi, L, \psi)$ be a quasi-alphabetic tree bimorphism ($L \subseteq T_\Gamma(Z)$)
2. Assume w.l.o.g. that formal variables appear in φ in order ($\xi_1, \xi_2, \ldots$, etc.)
3. L is recognizable, so $\exists \mathbf{A} = (Q, R, I)$ tree recognizer such that $L = T(\mathbf{A})$
4. Construct a set R' from R:
   - Add $q(\varphi_Z(z)) \rightarrow \psi_Z(z)$ to R', for every $q(z) \rightarrow z$ in R
5. Take $M = (Q, \Sigma, X, \Omega, Y, I, R')$ quasi-alphabetic tree transducer
6. Then $\tau_M = \tau_B$

Example

Let $f \in \Gamma_3$, $\varphi_3(f) = g(\xi_1, x, \xi_2, \xi_3, x)$ and $\psi_3(f) = h(y, \xi_1, \xi_3, \xi_2)$. Then:

Main Result - Sketch of the Proof

Theorem

The set of all quasi-alphabetic tree relations is effectively equal to the class of all tree transformations computed by quasi-alphabetic tree transducers.

Sketch of the Proof (Tree Bimorphism to Tree Transducer)

1. Let $B = (\varphi, L, \psi)$ be a quasi-alphabetic tree bimorphism ($L \subseteq T_\Gamma(Z)$)
2. Assume w.l.o.g. that formal variables appear in φ in order ($\xi_1, \xi_2, \ldots$, etc.)
3. L is recognizable, so $\exists \mathbf{A} = (Q, R, I)$ tree recognizer such that $L = T(\mathbf{A})$
4. Construct a set R' from R:
   - Add $q(\varphi_Z(z)) \to \psi_Z(z)$ to R', for every $q(z) \to z$ in R
5. Take $M = (Q, \Sigma, X, \Omega, Y, I, R')$ quasi-alphabetic tree transducer
6. Then $\tau_M = \tau_B$

Example

Let $f \in \Gamma_3$, $\varphi_3(f) = g(\xi_1, x, \xi_2, \xi_3, x)$ and $\psi_3(f) = h(y, \xi_1, \xi_3, \xi_2)$. Then:

Main Result - Sketch of the Proof

Theorem

The set of all quasi-alphabetic tree relations is effectively equal to the class of all tree transformations computed by quasi-alphabetic tree transducers.

Sketch of the Proof (Tree Bimorphism to Tree Transducer)

1. Let $B = (\varphi, L, \psi)$ be a quasi-alphabetic tree bimorphism ($L \subseteq T_\Gamma(Z)$)
2. Assume w.l.o.g. that formal variables appear in φ in order ($\xi_1, \xi_2, \ldots,$ etc.)
3. L is recognizable, so $\exists \mathbf{A} = (Q, R, I)$ tree recognizer such that $L = T(\mathbf{A})$
4. Construct a set R' from R:
   - Add $q(\varphi_Z(z)) \longrightarrow \psi_Z(z)$ to R', for every $q(z) \longrightarrow z$ in R
5. Take $M = (Q, \Sigma, X, \Omega, Y, I, R')$ quasi-alphabetic tree transducer
6. Then $\tau_M = \tau_B$

Example

Let $f \in \Gamma_3$, $\varphi_3(f) = g(\xi_1, x, \xi_2, \xi_3, x)$ and $\psi_3(f) = h(y, \xi_1, \xi_3, \xi_2)$. Then:

Main Result - Sketch of the Proof

Theorem

The set of all quasi-alphabetic tree relations is effectively equal to the class of all tree transformations computed by quasi-alphabetic tree transducers.

Sketch of the Proof (Tree Bimorphism to Tree Transducer)

1. Let $B = (\varphi, L, \psi)$ be a quasi-alphabetic tree bimorphism ($L \subseteq T_\Gamma(Z)$)
2. Assume w.l.o.g. that formal variables appear in φ in order ($\xi_1, \xi_2, \ldots,$ etc.)
3. L is recognizable, so $\exists \mathbf{A} = (Q, R, I)$ tree recognizer such that $L = T(\mathbf{A})$
4. Construct a set R' from R:
   - Add $q(\varphi_Z(z)) \to \psi_Z(z)$ to R', for every $q(z) \to z$ in R
   - Add $q(\varphi_m(f)) \to \psi_m(f)[q_1(\xi_1), \ldots, q_m(\xi_m)]$ to R', for every $q(f(\xi_1, \ldots, \xi_m)) \to f(q_1(\xi_1), \ldots, q_m(\xi_m))$
5. Take $M = (Q, \Sigma, X, \Omega, Y, I, R')$ quasi-alphabetic tree transducer
6. Then $\tau_M = \tau_B$

Example

Let $f \in \Gamma_3$, $\varphi_3(f) = g(\xi_1, x, \xi_2, \xi_3, x)$ and $\psi_3(f) = h(y, \xi_1, \xi_3, \xi_2)$. Then:

Main Result - Sketch of the Proof

Theorem

The set of all quasi-alphabetic tree relations is effectively equal to the class of all tree transformations computed by quasi-alphabetic tree transducers.

Sketch of the Proof (Tree Bimorphism to Tree Transducer)

1. Let $B = (\varphi, L, \psi)$ be a quasi-alphabetic tree bimorphism ($L \subseteq T_\Gamma(Z)$)
2. Assume w.l.o.g. that formal variables appear in φ in order ($\xi_1, \xi_2, \ldots$, etc.)
3. L is recognizable, so $\exists \mathbf{A} = (Q, R, I)$ tree recognizer such that $L = T(\mathbf{A})$
4. Construct a set R' from R:
   - Add $q(\varphi_Z(z)) \longrightarrow \psi_Z(z)$ to R', for every $q(z) \longrightarrow z$ in R
   - Add $q(\varphi_m(f)) \longrightarrow \psi_m(f)[q_1(\xi_1), \ldots, q_m(\xi_m)]$ to R, for every $q(f(\xi_1, \ldots, \xi_m)) \longrightarrow f(q_1(\xi_1), \ldots, q_m(\xi_m))$
5. Take $M = (Q, \Sigma, X, \Omega, Y, I, R')$ quasi-alphabetic tree transducer
6. Then $\tau_M = \tau_B$

Example

Let $f \in \Gamma_3$, $\varphi_3(f) = g(\xi_1, x, \xi_2, \xi_3, x)$ and $\psi_3(f) = h(y, \xi_1, \xi_3, \xi_2)$. Then:

Main Result - Sketch of the Proof

Theorem

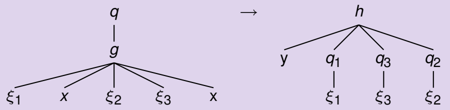
The set of all quasi-alphabetic tree relations is effectively equal to the class of all tree transformations computed by quasi-alphabetic tree transducers.

Sketch of the Proof (Tree Bimorphism to Tree Transducer)

1. Let $B = (\varphi, L, \psi)$ be a quasi-alphabetic tree bimorphism ($L \subseteq T_\Gamma(Z)$)
2. Assume w.l.o.g. that formal variables appear in φ in order ($\xi_1, \xi_2, \ldots$, etc.)
3. L is recognizable, so $\exists \mathbf{A} = (Q, R, I)$ tree recognizer such that $L = T(\mathbf{A})$
4. Construct a set R' from R:
   - Add $q(\varphi_Z(z)) \to \psi_Z(z)$ to R', for every $q(z) \to z$ in R
   - Add $q(\varphi_m(f)) \to \psi_m(f)[q_1(\xi_1), \ldots, q_m(\xi_m)]$ to R, for every $q(f(\xi_1, \ldots, \xi_m)) \to f(q_1(\xi_1), \ldots, q_m(\xi_m))$
5. Take $M = (Q, \Sigma, X, \Omega, Y, I, R')$ quasi-alphabetic tree transducer
6. Then $\tau_M = \tau_B$

Example

Let $f \in \Gamma_3$, $\varphi_3(f) = g(\xi_1, x, \xi_2, \xi_3, x)$ and $\psi_3(f) = h(y, \xi_1, \xi_3, \xi_2)$. Then:



Main Result - Sketch of the Proof

Theorem

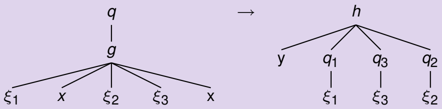
The set of all quasi-alphabetic tree relations is effectively equal to the class of all tree transformations computed by quasi-alphabetic tree transducers.

Sketch of the Proof (Tree Bimorphism to Tree Transducer)

1. Let $B = (\varphi, L, \psi)$ be a quasi-alphabetic tree bimorphism ($L \subseteq T_\Gamma(Z)$)
2. Assume w.l.o.g. that formal variables appear in φ in order ($\xi_1, \xi_2, \ldots$, etc.)
3. L is recognizable, so $\exists \mathbf{A} = (Q, R, I)$ tree recognizer such that $L = T(\mathbf{A})$
4. Construct a set R' from R:
   - Add $q(\varphi_Z(z)) \to \psi_Z(z)$ to R', for every $q(z) \to z$ in R
   - Add $q(\varphi_m(f)) \to \psi_m(f)[q_1(\xi_1), \ldots, q_m(\xi_m)]$ to R, for every $q(f(\xi_1, \ldots, \xi_m)) \to f(q_1(\xi_1), \ldots, q_m(\xi_m))$
5. Take $M = (Q, \Sigma, X, \Omega, Y, I, R')$ quasi-alphabetic tree transducer
6. Then $\tau_M = \tau_B$

Example

Let $f \in \Gamma_3$, $\varphi_3(f) = g(\xi_1, x, \xi_2, \xi_3, x)$ and $\psi_3(f) = h(y, \xi_1, \xi_3, \xi_2)$. Then:



Main Result - Sketch of the Proof

Theorem

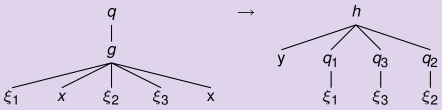
The set of all quasi-alphabetic tree relations is effectively equal to the class of all tree transformations computed by quasi-alphabetic tree transducers.

Sketch of the Proof (Tree Bimorphism to Tree Transducer)

1. Let $B = (\varphi, L, \psi)$ be a quasi-alphabetic tree bimorphism ($L \subseteq T_\Gamma(Z)$)
2. Assume w.l.o.g. that formal variables appear in φ in order ($\xi_1, \xi_2, \ldots,$ etc.)
3. L is recognizable, so $\exists \mathbf{A} = (Q, R, I)$ tree recognizer such that $L = T(\mathbf{A})$
4. Construct a set R' from R:
   - Add $q(\varphi_Z(z)) \to \psi_Z(z)$ to R', for every $q(z) \to z$ in R
   - Add $q(\varphi_m(f)) \to \psi_m(f)[q_1(\xi_1), \ldots, q_m(\xi_m)]$ to R, for every $q(f(\xi_1, \ldots, \xi_m)) \to f(q_1(\xi_1), \ldots, q_m(\xi_m))$
5. Take $M = (Q, \Sigma, X, \Omega, Y, I, R')$ quasi-alphabetic tree transducer
6. Then $\tau_M = \tau_B$

Example

Let $f \in \Gamma_3$, $\varphi_3(f) = g(\xi_1, x, \xi_2, \xi_3, x)$ and $\psi_3(f) = h(y, \xi_1, \xi_3, \xi_2)$. Then:



Some Conclusions

To construct the quasi-alphabetic tree bimorphism from the quasi-alphabetic tree transducer was more technical and, therefore, omitted. Details in the paper. The idea: the rules of the tree transducer are coded into the center language of the tree bimorphisms

Quasi-alphabetic tree transducer is a restrictive type of extended top-down tree transducer [Knight 2007] (arbitrary left-hand sides of the rules)

Synchronous tree substitution grammars, linear complete tree bimorphisms and extended top-down tree transducers were connected [Shieber 2004, Knight 2007]

No effective tree transducer was built for the most powerful synchronous grammars: synchronous tree-adjoining grammars or multitext grammars!!!

Some Conclusions

To construct the quasi-alphabetic tree bimorphism from the quasi-alphabetic tree transducer was more technical and, therefore, omitted. Details in the paper. The idea: the rules of the tree transducer are coded into the center language of the tree bimorphisms

Quasi-alphabetic tree transducer is a restrictive type of extended top-down tree transducer [Knight 2007] (arbitrary left-hand sides of the rules)

Synchronous tree substitution grammars, linear complete tree bimorphisms and extended top-down tree transducers were connected [Shieber 2004, Knight 2007]

No effective tree transducer was built for the most powerful synchronous grammars: synchronous tree-adjoining grammars or multitext grammars!!!

Some Conclusions

To construct the quasi-alphabetic tree bimorphism from the quasi-alphabetic tree transducer was more technical and, therefore, omitted. Details in the paper. The idea: the rules of the tree transducer are coded into the center language of the tree bimorphisms

Quasi-alphabetic tree transducer is a restrictive type of extended top-down tree transducer [Knight 2007] (arbitrary left-hand sides of the rules)

Synchronous tree substitution grammars, linear complete tree bimorphisms and extended top-down tree transducers were connected [Shieber 2004, Knight 2007]

No effective tree transducer was built for the most powerful synchronous grammars: synchronous tree-adjoining grammars or multitext grammars!!!

Some Conclusions

To construct the quasi-alphabetic tree bimorphism from the quasi-alphabetic tree transducer was more technical and, therefore, omitted. Details in the paper. The idea: the rules of the tree transducer are coded into the center language of the tree bimorphisms

Quasi-alphabetic tree transducer is a restrictive type of extended top-down tree transducer [Knight 2007] (arbitrary left-hand sides of the rules)

Synchronous tree substitution grammars, linear complete tree bimorphisms and extended top-down tree transducers were connected [Shieber 2004, Knight 2007]

No effective tree transducer was built for the most powerful synchronous grammars: synchronous tree-adjoining grammars or multitext grammars!!!

References I

André Arnold and Max Dauchet.
Morphismes et bimorphismes d'arbres.
Theor. Comput. Sci., 20(1):33–93, 1982.

Symeon Bozapalidis.
Alphabetic tree relations.
Theor. Comput. Sci., 99(2):177–211, 1992.

Ferenc Gécseg and Magnus Steinby.
Tree Automata.
Akadémiai Kiadó, Budapest, 1984.

Jonathan Graehl, Kevin Knight and Jonathan May.
Training tree transducers.
Comput. Ling., 34(3): 391–427, 2008.

Kevin Knight.
Capturing practical natural language transformations.
Machine Translation, 21(2):121–133, 2007.

Stuart M. Shieber.
Synchronous grammars as tree transducers.
In *Proc. TAG+7*, pages 88–95, 2004.

Magnus Steinby and Cătălin Ionuţ Tîrnăucă.
Defining syntax-directed translations by tree bimorphisms.
Theor. Comput. Sci., 410(37): 3495-3503, 2009.

That's all folks!

Thank you!