The Generative Power of Natural Languages

Petr Horáček, Eva Zámečníková and Ivana Burgetová

Department of Information Systems
Faculty of Information Technology
Brno University of Technology
Božetěchova 2, 612 00 Brno, CZ

BRNO UNIVERSITY OF TECHNOLOGY

FACULTY OF INFORMATION TECHNOLOGY

FRVŠ MŠMT FR97/2011/G1

Outline

- **The Generative Power of Natural Languages**

Outline

- **The Generative Power of Natural Languages**
- **Transformational Grammars**

Outline

- **The Generative Power of Natural Languages**
- **Transformational Grammars**
- **The Generative Capacity of Transformational Grammars**

Outline

- **The Generative Power of Natural Languages**
- **Transformational Grammars**
- **The Generative Capacity of Transformational Grammars**
- **Conclusion**

Topic

- **The Generative Power of Natural Languages**
- Transformational Grammars
- The Generative Capacity of Transformational Grammars
- Conclusion

- What is generative power of natural languages?
- Are natural languages recursive or not?

Outline

1. Examining of the generative power of NL.
2. Inherent generative capacity of classical transformational grammar as a formalism for language competence.

Chomsky Hierarchy

$$\mathcal{L}_3 \subset \mathcal{L}_2 \subset \mathcal{L}_1 \subset \mathcal{L}_0$$

- $\mathcal{L}_3$ ... set of regular languages,
- $\mathcal{L}_2$ ... set of context-free languages,
- $\mathcal{L}_1$ ... set of context-sensitive languages and
- $\mathcal{L}_0$ ... set of all phrase structure languages

Chomsky Hierarchy

$$\mathcal{L}_3 \subset \mathcal{L}_2 \subset \mathcal{L}_1 \subset \mathcal{L}_0$$

- $\mathcal{L}_3$ . . . set of regular languages,
- $\mathcal{L}_2$ . . . set of context-free languages,
- $\mathcal{L}_1$ . . . set of context-sensitive languages and
- $\mathcal{L}_0$ . . . set of all phrase structure languages

- NL could not be described as regular languages, because NL grammar must have **self-embedding**. (Chomsky, 1959)

Definition

A context-free grammar is self-embedding if there exists $A \in V$ such that

$$A \Rightarrow^* \alpha A \beta$$

for some $\alpha, \beta \in (V \cup X)^+$.

Theorem

A context-free language L is regular iff it possesses at least one grammar which is not self-embedding.

- Regular languages can also have self-embedding grammar.

Self-embedding in English

- $G_1 \ldots$ any grammar for X^*
- $G_2 \ldots$ any self-embedding grammar (eg. $S \rightarrow ab, S \rightarrow aSb$)
- productions of grammar G are the union of those of G_1 and $G_2 \Rightarrow$ it is self-embedding and is also a grammar for the regular set X^*

Example: Self-embedding

Self-embedding in English

- $G_1 \dots$ any grammar for X^*
- $G_2 \dots$ any self-embedding grammar (eg. $S \rightarrow ab, S \rightarrow aSb$)
- productions of grammar G are the union of those of G_1 and $G_2 \Rightarrow$ it is self-embedding and is also a grammar for the regular set X^*

Example

1. *John believes that Mary wants Bill with all his heart.*

Example: Self-embedding

Self-embedding in English

- $G_1 \ldots$ any grammar for X^*
- $G_2 \ldots$ any self-embedding grammar (eg. $S \rightarrow ab, S \rightarrow aSb$)
- productions of grammar G are the union of those of G_1 and $G_2 \Rightarrow$ it is self-embedding and is also a grammar for the regular set X^*

Example

1. *John believes that Mary wants Bill with all his heart.*
2. *John believes that Mary wants Bill to leave with all his heart.*

Self-embedding in English

- G_1 ... any grammar for X^*
- G_2 ... any self-embedding grammar (eg. $S \rightarrow ab, S \rightarrow aSb$)
- productions of grammar G are the union of those of G_1 and G_2 $\Rightarrow$ it is self-embedding and is also a grammar for the regular set X^*

Example

1. *John believes that Mary wants Bill with all his heart.*
2. *John believes that Mary wants Bill to leave with all his heart.*
3. *John believes that Mary wants Bill to tell Sam to leave with all his heart.*

Regularity of Natural Languages

Definition

We say, that two strings w_1 and w_2 are Myhill equivalent with respect to the language L, $w_1 \equiv_L w_2$, if for all strings u, v of X^* we have that

$$uw_1v \in L \Leftrightarrow uw_2 \in L.$$

Regularity of Natural Languages

Definition

We say, that two strings w_1 and w_2 are Myhill equivalent with respect to the language L, $w_1 \equiv_L w_2$, if for all strings u, v of X^* we have that

$$uw_1v \in L \Leftrightarrow uw_2 \in L.$$

Proposition

1. *If $w_1 \in L$ and $w_1 \equiv_L w_2$, then $w_2 \in L$.*
2. *If $w_1 \equiv_L w_2$ and $x \in X$, then $w_1x \equiv_L w_2x$.*

Definition

We say, that two strings w_1 and w_2 are Myhill equivalent with respect to the language L, $w_1 \equiv_L w_2$, if for all strings u, v of X^* we have that

$$uw_1v \in L \Leftrightarrow uw_2 \in L.$$

Proposition

1. *If $w_1 \in L$ and $w_1 \equiv_L w_2$, then $w_2 \in L$.*
2. *If $w_1 \equiv_L w_2$ and $x \in X$, then $w_1x \equiv_L w_2x$.*

Proof.

1. Take $u = v = \varepsilon$ in definition above.
2. For any $u, v \in X^*$:

$$\begin{aligned} u(w_1x)v \in L &\Leftrightarrow uw_1(xv) \in L \\ &\Leftrightarrow uw_2(xv) \in L \textbf{ since } w_1 \equiv_L w_2 \\ &\Leftrightarrow u(w_2x)v \in L \end{aligned}$$

Thus $w_1 \equiv_L w_2$. □

Theorem

A language L is regular if and only if the number of Myhill equivalence classes for L is finite.

Theorem

A language L is regular if and only if the number of Myhill equivalence classes for L is finite.

Proof.

Assume that L has a finite set Q of equivalence classes. We use these classes as states of a finite state machine. By previous Proposition, following definitions of $\delta : Q \times X \to Q$ and $F \subset Q$ are well defined - they do not depend on the choice of representative w from the equivalence class (w) in Q.

Theorem

A language L is regular if and only if the number of Myhill equivalence classes for L is finite.

Proof.

Assume that L has a finite set Q of equivalence classes. We use these classes as states of a finite state machine. By previous Proposition, following definitions of $\delta : Q \times X \to Q$ and $F \subset Q$ are well defined - they do not depend on the choice of representative w from the equivalence class (w) in Q.

- $\varepsilon([w], x) = [wx]$

Theorem

A language L is regular if and only if the number of Myhill equivalence classes for L is finite.

Proof.

Assume that L has a finite set Q of equivalence classes. We use these classes as states of a finite state machine. By previous Proposition, following definitions of $\delta : Q \times X \to Q$ and $F \subset Q$ are well defined - they do not depend on the choice of representative w from the equivalence class (w) in Q.

- $\varepsilon([w], x) = [wx]$
- $[w] \in F \Leftrightarrow w \in L$

Regularity of Natural Languages

Theorem

A language L is regular if and only if the number of Myhill equivalence classes for L is finite.

Proof.

Assume that L has a finite set Q of equivalence classes. We use these classes as states of a finite state machine. By previous Proposition, following definitions of $\delta : Q \times X \to Q$ and $F \subset Q$ are well defined - they do not depend on the choice of representative w from the equivalence class (w) in Q.

- $\varepsilon([w], x) = [wx]$
- $[w] \in F \Leftrightarrow w \in L$

If we now let $q_0 = [\varepsilon]$, the Myhill equivalence class of the empty string, we have that $M = (Q, q_0, \varepsilon, F)$ accepts L:

$$\varepsilon^*(q_0, w) = [w]$$

Theorem

A language L is regular if and only if the number of Myhill equivalence classes for L is finite.

Proof.

Assume that L has a finite set Q of equivalence classes. We use these classes as states of a finite state machine. By previous Proposition, following definitions of $\delta : Q \times X \to Q$ and $F \subset Q$ are well defined - they do not depend on the choice of representative w from the equivalence class (w) in Q.

- $\varepsilon([w], x) = [wx]$
- $[w] \in F \Leftrightarrow w \in L$

If we now let $q_0 = [\varepsilon]$, the Myhill equivalence class of the empty string, we have that $M = (Q, q_0, \varepsilon, F)$ accepts L:

$$\varepsilon^*(q_0, w) = [w]$$

and thus $w \in T(M)$ iff $[w] \in F$ iff $w \in L$. □

Example

The violation of the finitness property.

- Language $\{a^n b^n | n \geq 1\}$
- b^n ... different equivalence class for each choice of n.

A dependency in natural language:

Example

The violation of the finitness property.

- Language $\{a^n b^n | n \geq 1\}$
- $b^n \ldots$ different equivalence class for each choice of n.

A dependency in natural language:

1. *The dog died.*
   $[_S$NP VP$]$

Example

The violation of the finitness property.

- Language $\{a^n b^n | n \geq 1\}$
- b^n ... different equivalence class for each choice of n.

A dependency in natural language:

1. *The dog died.*
   $[_S$NP VP$]$
2. *The boy that the dog bit died.*
   $[_S[_{NP}$NP$[_S$NP VP$]]$VP$]$

Example

The violation of the finitness property.

- Language $\{a^n b^n | n \geq 1\}$
- $b^n \ldots$ different equivalence class for each choice of n.

A dependency in natural language:

1. *The dog died.*
   $[_S$NP VP$]$
2. *The boy that the dog bit died.*
   $[_S[_{NP}$NP$[_S$NP VP$]]$VP$]$
3. *The boy that the dog that the horse kicked bit died.*
   $[_S[_{NP}$NP$[_S[_{NP}$NP$[_S$NP VP$]]]$VP$]$VP$]$

Example

The violation of the finitness property.

- Language $\{a^n b^n | n \geq 1\}$
- b^n … different equivalence class for each choice of n.

A dependency in natural language:

1. *The dog died.*
   $[_S \text{NP VP}]$
2. *The boy that the dog bit died.*
   $[_S[_{NP}\text{NP}[_S \text{NP VP}]]\text{VP}]$
3. *The boy that the dog that the horse kicked bit died.*
   $[_S[_{NP}\text{NP}[_S[_{NP}\text{NP}[_S \text{NP VP}]]]\text{VP}]\text{VP}]$

- a … NP
- b … V

Example

The violation of the finitness property.

- Language $\{a^n b^n | n \geq 1\}$
- $b^n \ldots$ different equivalence class for each choice of n.

A dependency in natural language:

1. *The dog died.*
   $[_S$NP VP]
2. *The boy that the dog bit died.*
   $[_S[_{NP}$NP$[_S$NP VP]]VP]
3. *The boy that the dog that the horse kicked bit died.*
   $[_S[_{NP}$NP$[_S[_{NP}$NP$[_S$NP VP]]]VP]VP]

- $a \ldots$ NP
- $b \ldots$ V

$\Rightarrow$ English must have infinitely many Myhill equivalence classes and so it is not regular.

Topic

- The Generative Power of Natural Languages
- **Transformational Grammars**
- The Generative Capacity of Transformational Grammars
- Conclusion

Transformational Grammars

- Transformational grammars generate the class of natural languages.
- They generate not only NL, but also unnatural languages.

- Transformational grammars generate the class of natural languages.
- They generate not only NL, but also unnatural languages.

Transformational rule

- the type of rule that can generate certain construction
- altering of the structure generated by phrase structure rules by moving, adding or deleting in the string

Transformational grammar *TG*

- *TG* has three parts
  1. Phrase structure grammar G – base,
  2. a set of transformations T and
  3. a set of restrictions on these transformations R.

Transformational grammar *TG*

- *TG* has three parts
  1. Phrase structure grammar *G* – base,
  2. a set of transformations *T* and
  3. a set of restrictions on these transformations *R*.

- deep structures – the set of derivation trees generated by *G*
- restrictions of *R* specify that some transformations in *T* are obligatory

Transformational grammar *TG*

- *TG* has three parts
  1. Phrase structure grammar *G* – base,
  2. a set of transformations *T* and
  3. a set of restrictions on these transformations *R*.

- deep structures – the set of derivation trees generated by *G*
- restrictions of *R* specify that some transformations in *T* are obligatory
- surface structures – the set of trees which may be obtained from deep structures by successively applying transformations from *T* according the rules from *R*.
- *L*(*TG*) – a set of strings we may read off the surface structures.

- We will consider grammars with context-free bases.

- We will consider grammars with context-free bases.

Lemma

A transformational grammar can perform arbitrary homomorphisms, and in particular, ε-homomorphisms.

- We will consider grammars with context-free bases.

Lemma

A transformational grammar can perform arbitrary homomorphisms, and in particular, ε-homomorphisms.

Proof.

Proof demonstration:

- Substitution map $g : X^* \to 2^{Y^*}$ as a map where
  - $g(\varepsilon) = \varepsilon$
- and for each $n \geq 1$
  - $g(a_1 \dots a_n) = g(a_1)g(a_2)\dots g(a_n)$.

If $g(a)$ contains only one element Y^* for each $a \in X$, then g is called a *homomorphism*.

Example on the next page is a part of this proof demonstration. □

Example

Transformation: change *that* $\rightarrow$ *my own* and *dog* $\rightarrow$ *white cat*.

Example

Transformation: change *that* $\rightarrow$ *my own* and *dog* $\rightarrow$ *white cat*.

- $g(\textit{that}) = \textit{my own}$
- $g(\textit{dog}) = \textit{white cat}$

Example

Transformation: change *that* $\rightarrow$ *my own* and *dog* $\rightarrow$ *white cat*.

- $g(\textit{that}) = \textit{my own}$
- $g(\textit{dog}) = \textit{white cat}$

For

- *That dog likes that food.*

Example

Transformation: change *that* $\rightarrow$ *my own* and *dog* $\rightarrow$ *white cat*.

- $g(that) = my\ own$
- $g(dog) = white\ cat$

For

- *That dog likes that food.*

the resulting string will be:

Example

Transformation: change *that* $\rightarrow$ *my own* and *dog* $\rightarrow$ *white cat*.

- $g(that) = my\ own$
- $g(dog) = white\ cat$

For

- *That dog likes that food.*

the resulting string will be:

- *My own white cat likes my own food.*

Example

Transformation: change *that* $\rightarrow$ *my own* and *dog* $\rightarrow$ *white cat*.

- $g(that) = my\ own$
- $g(dog) = white\ cat$

For

- *That dog likes that food.*

the resulting string will be:

- *My own white cat likes my own food.*

When we allow ε-homomorphism, that is $g(a) = \varepsilon$, we can do arbitrary deletion by adding:

- $g(that) = \varepsilon$

Example

Transformation: change *that* $\rightarrow$ *my own* and *dog* $\rightarrow$ *white cat*.

- $g(that) = my\ own$
- $g(dog) = white\ cat$

For

- *That dog likes that food.*

the resulting string will be:

- *My own white cat likes my own food.*

When we allow ε-homomorphism, that is $g(a) = \varepsilon$, we can do arbitrary deletion by adding:

- $g(that) = \varepsilon$

Result of transformation will be:

- *Dog likes food.*

Lemma

Let G_1, G_2 be any CFGs. Then there exists a transformational grammar TG, such that TG can perform the intersection of the languages of G_1 and G_2.
That is,

$$L(TG) = (L(G_1) \cap L(G_2))$$

Proof.

Proof outline:

- TG with a context-free base
- TG has only one S production $S \rightarrow S_1 \mu S_2 \$$
  - S_1 and S_2 - start symbols for grammars G_1 and G_2, respectively.
- T_1 - transformation perfoming intersection between two CFGs.

Proof.

Proof outline:

- *TG* with a context-free base
- *TG* has only one *S* production $S \rightarrow S_1 \mu S_2 \$$
  - S_1 and S_2 - start symbols for grammars G_1 and G_2, respectively.
- T_1 - transformation perfoming intersection between two CFGs.

Transformation T_1

SD:	X	x	μ	X	y	$\$$	w	
	1	2	3	4	5	6	7	$\Rightarrow$
SC:		2	3		5	6	7 + 1	

Proof.

For generating just the intersection, transformation T_2 is needed:

$$T_2 : \mu\$ \to \varepsilon.$$

Proof.

For generating just the intersection, transformation T_2 is needed:

$$T_2 : \mu\$ \rightarrow \varepsilon.$$

Transformation T_2					
SD:	x	μ	$\$$	y	
	1	2	3	4	$\Rightarrow$
SC:	1			4	

□

Example

Example demonstrates how rules T_1 and T_2 work.

Example

Example demonstrates how rules T_1 and T_2 work.

Let G_1, G_2 be two context-free grammars

- $L(G_1) = \{a^n b^n c^m, n, m \geq 1\}$.
- $L(G_2) = \{a^n b^m c^m, n, m \geq 1\}$.

Example

Example demonstrates how rules T_1 and T_2 work.

Let G_1, G_2 be two context-free grammars

- $L(G_1) = \{a^n b^n c^m, n, m \geq 1\}$.
- $L(G_2) = \{a^n b^m c^m, n, m \geq 1\}$.

The transformational grammar generates just the intersection of these two languages, namely:

Example

Example demonstrates how rules T_1 and T_2 work.

Let G_1, G_2 be two context-free grammars

- $L(G_1) = \{a^n b^n c^m, n, m \geq 1\}$.
- $L(G_2) = \{a^n b^m c^m, n, m \geq 1\}$.

The transformational grammar generates just the intersection of these two languages, namely:

$$a^n b^n c^n, n \geq 1$$

which is not context-free.

Example

Assume the string $aabbcc\mu aabbcc\$$.

- Apply T_1 until there is no structural description that fits the rule.

Example

Assume the string $aabbcc\mu aabbcc\$$.

- Apply T_1 until there is no structural description that fits the rule.
- After all successful applications of T_1
  - string = $\mu\$aabbcc$

Example

Assume the string $aabbcc\mu aabbcc\$$.

- Apply T_1 until there is no structural description that fits the rule.
- After all successful applications of T_1
  - string = $\mu\$aabbcc$
- Apply rule T_2:
  - string = $aabbcc$; $aabbcc \in \{a^n b^n c^n; n \geq 1\}$

Example

Assume the string $aabbcc\mu aabbcc\$$.

- Apply T_1 until there is no structural description that fits the rule.
- After all successful applications of T_1
  - string = $\mu\$aabbcc$
- Apply rule T_2:
  - string = $aabbcc$; $aabbcc \in \{a^n b^n c^n; n \geq 1\}$

Now assume the string $aabbbccc\mu aabbbccc$.

Example

Assume the string $aabbcc\mu aabbcc\$$.

- Apply T_1 until there is no structural description that fits the rule.
- After all successful applications of T_1
  - string = $\mu\$aabbcc$
- Apply rule T_2:
  - string = $aabbcc$; $aabbcc \in \{a^n b^n c^n; n \geq 1\}$

Now assume the string $aabbbccc\mu aabbbccc$.

- T_1 applies since there is unequal numbers of b's.

Example

Assume the string $aabbcc\mu aabbcc\$$.

- Apply T_1 until there is no structural description that fits the rule.
- After all successful applications of T_1
  - string = $\mu\$aabbcc$
- Apply rule T_2:
  - string = $aabbcc$; $aabbcc \in \{a^n b^n c^n; n \geq 1\}$

Now assume the string $aabbbccc\mu aabbbccc$.

- T_1 applies since there is unequal numbers of b's.
- But then T_2 can not apply since the markers are not adjacent.

Example

Assume the string $aabbcc\mu aabbcc\$$.

- Apply T_1 until there is no structural description that fits the rule.
- After all successful applications of T_1
  - string = $\mu\$aabbcc$
- Apply rule T_2:
  - string = $aabbcc$; $aabbcc \in \{a^n b^n c^n; n \geq 1\}$

Now assume the string $aabbbccc\mu aabbbccc$.

- T_1 applies since there is unequal numbers of b's.
- But then T_2 can not apply since the markers are not adjacent.
- Thus, the string is not generated by the grammar.

Topic

- The Generative Power of Natural Languages
- Transformational Grammars
- **The Generative Capacity of Transformational Grammars**
- Conclusion

Theorem

There is an undecidable set S of the natural numbers N, such that S can be generated by some context-free based transformational grammar.

Generative Capacity of TG

Theorem

There is an undecidable set S of the natural numbers N, such that S can be generated by some context-free based transformational grammar.

Proof.

Proof outline:
We can construct an undecidable set $S \subseteq N$ as being homomorphic image of the intersection ot two context-free languages.

Theorem

There is an undecidable set S of the natural numbers N, such that S can be generated by some context-free based transformational grammar.

Proof.

Proof outline:
We can construct an undecidable set $S \subseteq N$ as being homomorphic image of the intersection ot two context-free languages.
That is:

$$L = \phi(L_1 \cap L_2),$$

where L_1 and L_2 are context-free languages and ϕ is a homomorphism.

□

Topic

- The Generative Power of Natural Languages
- Transformational Grammars
- The Generative Capacity of Transformational Grammars
- **Conclusion**

Question

If we consider transformational grammars as a model for constraining the form of natural languages, why should the model generate languages as powerful and unconstrained as undecidable sets?

Conclusion

Question

If we consider transformational grammars as a model for constraining the form of natural languages, why should the model generate languages as powerful and unconstrained as undecidable sets?

Answer

- We can consider that by restricting the base rules of the transformational grammar even more tightly than to the context-free, we might keep the resulting language recursive.

Question

If we consider transformational grammars as a model for constraining the form of natural languages, why should the model generate languages as powerful and unconstrained as undecidable sets?

Answer

- We can consider that by restricting the base rules of the transformational grammar even more tightly than to the context-free, we might keep the resulting language recursive.
- From the results is clear that if we want to restrict the generative power of transformational grammars, it will be necessary to constrain the form of the rules themselves rather than the base.

References

James Allen:
Natural Language Understanding,
The Benjamin/Cummings Publishing Company. Inc., 2005

Robert N. Moll, Michael A. Arbib, A. J. Kfoury:
An Introduction to Formal Language Theory,
Springer-Verlag, 1988

Thank you for your attention!

End