Describing Syntax with Star-Free Regular Expressions

Anssi Yli-Jyrä

Department of General Linguistics

University of Helsinki

Anssi.Yli-Jyra@Helsinki.Fi

EACL'03, Budapest

April 15, 2003

1 – The main contribution

1. Regular languages are **not** captured in first-order (FO) logic (Büchi -60)
2. Finite-State Intersection Grammar (FSIG) (Koskenniemi -90)
   allows **full** regular expressions

1 – The main contribution

1. Regular languages are **not** captured in first-order (FO) logic (Büchi -60)
2. Finite-State Intersection Grammar (FSIG) (Koskenniemi -90)
   allows **full** regular expressions

FSIGs are **not** captured in FO logic

1 – The main contribution

1. Regular languages are **not** captured in first-order (FO) logic (Büchi -60)
2. Finite-State Intersection Grammar (FSIG) (Koskenniemi -90)
   allows **full** regular expressions

FSIGs are **not** captured in FO logic

1. **ENGFSIG (Voutilainen -97) admits a star-free (regular) expression**

1 – The main contribution

1. Regular languages are **not** captured in first-order (FO) logic (Büchi -60)
2. Finite-State Intersection Grammar (FSIG) (Koskenniemi -90)
   allows **full** regular expressions

FSIGs are **not** captured in FO logic

1. **ENGFSIG (Voutilainen -97) admits a star-free (regular) expression**
2. Star-free expressions are captured **in FO logic** (McNaughton&Papert -71)
3. FO logic can express **local properties only** (Hanf -65, Gaifman -82)

1 – The main contribution

1. Regular languages are **not** captured in first-order (FO) logic (Büchi -60)
2. Finite-State Intersection Grammar (FSIG) (Koskenniemi -90) allows **full** regular expressions

---

FSIGs are **not** captured in FO logic

1. **ENGFSIG (Voutilainen -97) admits a star-free (regular) expression**
2. Star-free expressions are captured **in FO logic** (McNaughton&Papert -71)
3. FO logic can express **local properties only** (Hanf -65, Gaifman -82)

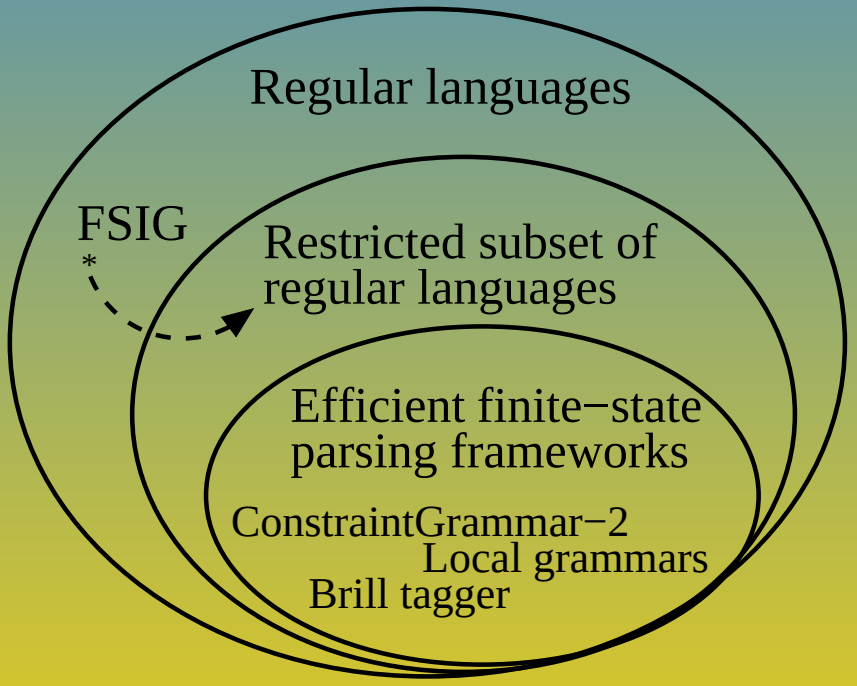
ENGFSIG describes local properties only

2 – Why locality restriction is so interesting?

Finite-state frameworks $\not\rightarrow$ **efficient parsing in practice.**

Pursue efficient parsing frameworks: make restrictions to the expressive power e.g. by requiring that the rules are **local**.

Restricted expressive power $\rightarrow$ **infer grammars more easily**.



A restriction in the expressive power of the FSIG framework is a step **towards** practical and fast parsing.

3 – The FSIG rule formalism

Structural descriptions that are encoded as strings:

bracketing	surface form	base form	morphosyntactic tags			bracketing
@<	*it*	it	PRON	NOM	SG3 SUBJ	@
	works	work	V	PRES	SG3 MV	@>

3 – The FSIG rule formalism

Structural descriptions that are encoded as strings:

bracketing	surface form	base form	morphosyntactic tags		bracketing
@<	*it*	it	PRON NOM	SG3 SUBJ	@
	works	work	V PRES	SG3 MV	@>

Checked with **context-restriction rules** (← TWOL, Koskenniemi -83). E.g.

```
V PRES SG3 ⇒  SG3 SUBJ @ ··· __ ,
                  __  AUX @ ···  SG3 SUBJ;
```

(A distributional constraint for 3^{rd} person singular finite verbs in English.)

4 – Finite-State Intersection Grammar

Wild-card symbol “dots” $\boxed{\cdots}$ stands for clause-internal tag sequences which may contain limited @<,@>-bracketing.

4 – Finite-State Intersection Grammar

Wild-card symbol “dots” $\boxed{\cdots}$ stands for clause-internal tag sequences which may contain limited @<,@>-bracketing.

Grammar components:

4 – Finite-State Intersection Grammar

Wild-card symbol "dots" $\boxed{\cdots}$ stands for clause-internal tag sequences which may contain limited @<,@>-bracketing.

Grammar components:

1. Domain D of structural description strings

4 – Finite-State Intersection Grammar

Wild-card symbol "dots" $\boxed{\cdots}$ stands for clause-internal tag sequences which may contain limited @<,@>-bracketing.

Grammar components:

1. Domain D of structural description strings
2. A finite limit d for nested brackets (center-embeddings)

4 – Finite-State Intersection Grammar

Wild-card symbol "dots" $\boxed{\cdots}$ stands for clause-internal tag sequences which may contain limited @<,@>-bracketing.

Grammar components:

1. Domain D of structural description strings
2. A finite limit d for nested brackets (center-embeddings)
3. Context restriction rules $c_1^d, c_2^d, \ldots c_n^d$

A **Finite-State Intersection Grammar (FSIG)** (Koskenniemi -90)

5 – The shift of perspective

FROM: FSIG as a parsing framework:

1. Given the set T of all morphological analyses for an input sentence, create the set S of all potential structural descriptions by inserting brackets and syntactic tags freely into the strings of T.

5 – The shift of perspective

FROM: FSIG as a parsing framework:

1. Given the set T of all morphological analyses for an input sentence, create the set S of all potential structural descriptions by inserting brackets and syntactic tags freely into the strings of T.
2. Compute intersection $S \cap D \cap c_1^d \cap c_2^d \cap \cdots \cap c_n^d$
   $\rightarrow$ **state-space explosion problem** (Tapanainen -97)

5 – The shift of perspective

FROM: FSIG as a parsing framework:

1. Given the set T of all morphological analyses for an input sentence, create the set S of all potential structural descriptions by inserting brackets and syntactic tags freely into the strings of T.
2. Compute intersection $S \cap D \cap c_1^d \cap c_2^d \cap \cdots \cap c_n^d$
   $\rightarrow$ **state-space explosion problem** (Tapanainen -97)

TO: an FSIG $G = D \cap c_1^d \cap c_2^d \cap \cdots \cap c_n^d$ as a language model:

5 – The shift of perspective

FROM: FSIG as a parsing framework:

1. Given the set T of all morphological analyses for an input sentence, create the set S of all potential structural descriptions by inserting brackets and syntactic tags freely into the strings of T.
2. Compute intersection $S \cap D \cap c_1^d \cap c_2^d \cap \cdots \cap c_n^d$
   $\rightarrow$ **state-space explosion problem** (Tapanainen -97)

TO: an FSIG $G = D \cap c_1^d \cap c_2^d \cap \cdots \cap c_n^d$ as a language model:

1. **model restrictions** for $L(G)$?

5 – The shift of perspective

FROM: FSIG as a parsing framework:

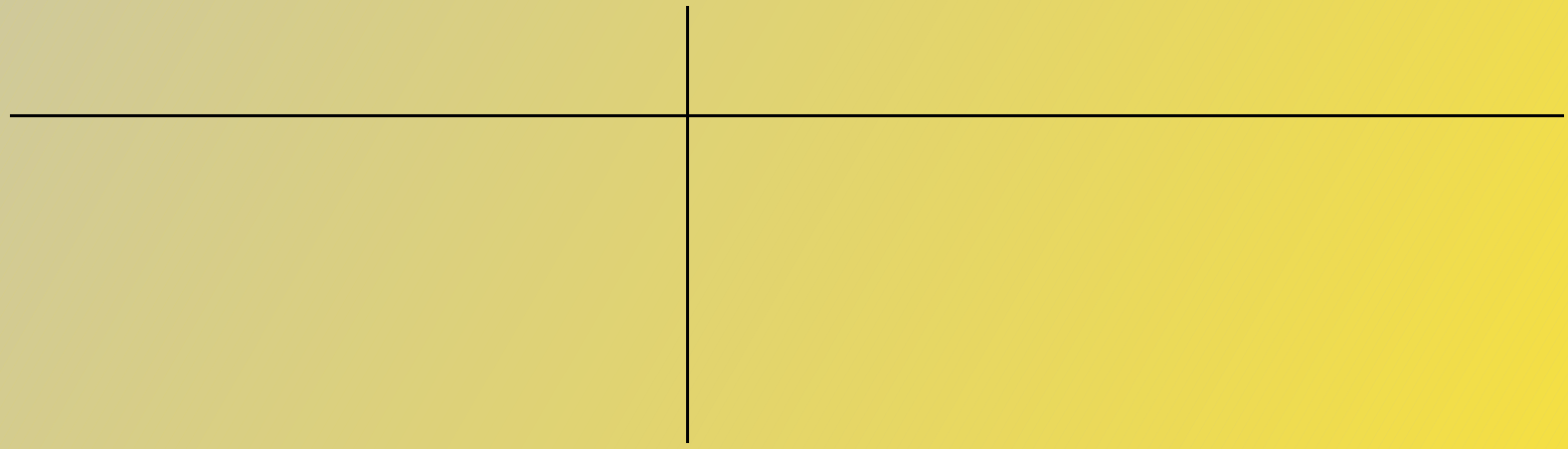
1. Given the set T of all morphological analyses for an input sentence, create the set S of all potential structural descriptions by inserting brackets and syntactic tags freely into the strings of T.
2. Compute intersection $S \cap D \cap c_1^d \cap c_2^d \cap \cdots \cap c_n^d$
   $\rightarrow$ **state-space explosion problem** (Tapanainen -97)

TO: an FSIG $G = D \cap c_1^d \cap c_2^d \cap \cdots \cap c_n^d$ as a language model:

1. **model restrictions** for $L(G)$?
2. reducing parsing to a set of **local tests**?

6 – A restricted family of languages: Star-free regular languages

Basic regular (and star-free regular) expressions: $a \in \Sigma$ and Σ^* (the universal language)



6 – A restricted family of languages: Star-free regular languages

Basic regular (and star-free regular) expressions: $a \in \Sigma$ and Σ^* (the universal language)

Regular expressions	
concatenation	$\alpha\beta$
union	$\alpha \cup \beta$
Kleene's star	α^*

6 – A restricted family of languages: Star-free regular languages

Basic regular (and star-free regular) expressions: $a \in \Sigma$ and Σ^* (the universal language)

Regular expressions		**Star-free regular expressions**	
concatenation	$\alpha\beta$	**concatenation**	$\alpha\beta$
union	$\alpha \cup \beta$	**union**	$\alpha \cup \beta$
Kleene's star	α^*	**complementation**	$\sim\alpha$

A regular language is **star-free** if it has a star-free regular expression.

7 – Elimination of stars in English FSIG

ENGFSIG (Univ. Helsinki, 1990-1997), 2600 rules (Voutilainen -94,-97).

The star-freeness of ENGFSIG by **induction**:

FSIG regular expression	Star-free equivalent
ϵ	

7 – Elimination of stars in English FSIG

ENGFSIG (Univ. Helsinki, 1990-1997), 2600 rules (Voutilainen -94,-97).

The star-freeness of ENGFSIG by **induction**:

FSIG regular expression	Star-free equivalent
ϵ	$\Sigma^* - \Sigma\Sigma^*$

7 – Elimination of stars in English FSIG

ENGFSIG (Univ. Helsinki, 1990-1997), 2600 rules (Voutilainen -94,-97).

The star-freeness of ENGFSIG by **induction**:

FSIG regular expression	Star-free equivalent
ϵ	$\Sigma^* - \Sigma\Sigma^*$
$\{\mathtt{A},\mathtt{B}\}^*$	$\Sigma^* - \Sigma^*[\Sigma - \{\mathtt{A},\mathtt{B}\}]\Sigma^*$

7 – Elimination of stars in English FSIG

ENGFSIG (Univ. Helsinki, 1990-1997), 2600 rules (Voutilainen -94,-97).

The star-freeness of ENGFSIG by **induction**:

FSIG regular expression	Star-free equivalent
ϵ	$\Sigma^* - \Sigma\Sigma^*$
$\{\mathtt{A},\mathtt{B}\}^*$	$\Sigma^* - \Sigma^*[\Sigma - \{\mathtt{A},\mathtt{B}\}]\Sigma^*$
$\mathtt{A} \Rightarrow \_\ \mathtt{B}$	$\sim[\ \Sigma^*\ \mathtt{A}\ \sim[\mathtt{B}\ \Sigma^*]]$

7 – Elimination of stars in English FSIG

ENGFSIG (Univ. Helsinki, 1990-1997), 2600 rules (Voutilainen -94,-97).

The star-freeness of ENGFSIG by **induction**:

FSIG regular expression	Star-free equivalent
ϵ	$\Sigma^* - \Sigma\Sigma^*$
$\{\mathtt{A},\mathtt{B}\}^*$	$\Sigma^* - \Sigma^*[\Sigma - \{\mathtt{A},\mathtt{B}\}]\Sigma^*$
$\mathtt{A} \Rightarrow \_\ \mathtt{B}$	$\sim[\ \Sigma^*\ \mathtt{A}\ \sim[\mathtt{B}\ \Sigma^*]]$
$\mathtt{B} \Rightarrow \mathtt{A}\ \_$	**similarly**

7 – Elimination of stars in English FSIG

ENGFSIG (Univ. Helsinki, 1990-1997), 2600 rules (Voutilainen -94,-97).

The star-freeness of ENGFSIG by **induction**:

FSIG regular expression	Star-free equivalent
ϵ	$\Sigma^* - \Sigma\Sigma^*$
$\{\mathtt{A},\mathtt{B}\}^*$	$\Sigma^* - \Sigma^*[\Sigma - \{\mathtt{A},\mathtt{B}\}]\Sigma^*$
$\mathtt{A} \Rightarrow \_\ \mathtt{B}$	$\sim[\ \Sigma^*\ \mathtt{A}\ \sim[\mathtt{B}\ \Sigma^*]]$
$\mathtt{B} \Rightarrow \mathtt{A}\ \_$	**similarly**
$[\mathtt{A}\ \mathtt{B}]^*$	$\{\mathtt{A},\mathtt{B}\}^* \cap [\mathtt{A} \Rightarrow \_\ \mathtt{B}] \cap [\mathtt{B} \Rightarrow \mathtt{A}\ \_]$

7 – Elimination of stars in English FSIG

ENGFSIG (Univ. Helsinki, 1990-1997), 2600 rules (Voutilainen -94,-97).

The star-freeness of ENGFSIG by **induction**:

FSIG regular expression	Star-free equivalent
ϵ	$\Sigma^* - \Sigma\Sigma^*$
$\{\mathtt{A},\mathtt{B}\}^*$	$\Sigma^* - \Sigma^*[\Sigma - \{\mathtt{A},\mathtt{B}\}]\Sigma^*$
$\mathtt{A} \Rightarrow \_\ \mathtt{B}$	$\sim[\ \Sigma^*\ \mathtt{A}\ \sim[\mathtt{B}\ \Sigma^*]]$
$\mathtt{B} \Rightarrow \mathtt{A}\ \_$	**similarly**
$[\mathtt{A}\ \mathtt{B}]^*$	$\{\mathtt{A},\mathtt{B}\}^* \cap [\mathtt{A} \Rightarrow \_\ \mathtt{B}] \cap [\mathtt{B} \Rightarrow \mathtt{A}\ \_]$
etc.	

8 – Implications on the descriptive complexity

Characterizations for regular languages in Finite-Model Theory:

8 – Implications on the descriptive complexity

Characterizations for regular languages in Finite-Model Theory:

- **Regular languages (can) assign sets of string positions to variables**
  $\rightarrow$ monadic 2nd-order quantification (Büchi -60)
  e.g. “Even-Parity”-language: $(0^*(10^*10^*)^*)$

8 – Implications on the descriptive complexity

Characterizations for regular languages in Finite-Model Theory:

- **Regular languages (can) assign sets of string positions to variables**
  → monadic 2nd-order quantification (Büchi -60)
  e.g. "Even-Parity"-language: $(0^*(10^*10^*)^*)$

- **Star-free languages assign string positions to variables**
  → 1st-order quantification (McNaughton and Papert -71)
  → three bounded variables are enough (Kamp -68)
  → express *local properties* (Hanf -65, Gaifman -82)

8 – Implications on the descriptive complexity

Characterizations for regular languages in Finite-Model Theory:

- **Regular languages (can) assign sets of string positions to variables**
  → monadic 2nd-order quantification (Büchi -60)
  e.g. "Even-Parity"-language: $(0^*(10^*10^*)^*)$

- **Star-free languages assign string positions to variables**
  → 1st-order quantification (McNaughton and Papert -71)
  → three bounded variables are enough (Kamp -68)
  → express *local properties* (Hanf -65, Gaifman -82)

(*Local properties* in the sense of Hanf and Gaifman)

9 – Conclusion

ENGFSIG is equivalent to a STAR-FREE regexp

9 – Conclusion

ENGFSIG is equivalent to a STAR-FREE regexp

1. ENGFSIG is FO-definable → cannot express non-local properties

9 – Conclusion

ENGFSIG is equivalent to a STAR-FREE regexp

1. ENGFSIG is FO-definable → cannot express non-local properties
2. compound context restrictions (← TWOL) compiled without transducers

9 – Conclusion

ENGFSIG is equivalent to a STAR-FREE regexp

1. ENGFSIG is FO-definable → cannot express non-local properties
2. compound context restrictions (← TWOL) compiled without transducers

Possible extensions:

- locality → ambiguity packing? → Can we avoid state-space explosion?

9 – Conclusion

ENGFSIG is equivalent to a STAR-FREE regexp

1. ENGFSIG is FO-definable → cannot express non-local properties
2. compound context restrictions (← TWOL) compiled without transducers

Possible extensions:

- locality → ambiguity packing? → Can we avoid state-space explosion?
- “NLs form a proper (star-free) subset of regular languages” (Kornai -85)

9 – Conclusion

ENGFSIG is equivalent to a STAR-FREE regexp

1. ENGFSIG is FO-definable → cannot express non-local properties
2. compound context restrictions (← TWOL) compiled without transducers

Possible extensions:

- locality → ambiguity packing? → Can we avoid state-space explosion?
- "NLs form a proper (star-free) subset of regular languages" (Kornai -85)
- varieties of star-free languages → further model restrictions?