Finite State Machines

Roger Levy

Massachusetts Institute of Technology
Department of Brain & Cognitive Sciences

Regular languages (standpoint of formal language theory)

Regular languages (standpoint of formal language theory)

- A **language** in Σ^* is a set of strings: $L \subseteq \Sigma^*$

Regular languages (standpoint of formal language theory)

- A **language** in Σ^* is a set of strings: $L \subseteq \Sigma^*$
- The set of strings matched by a regex, or accepted by an FSA, is the language defined by the regex or FSA

Regular languages (standpoint of formal language theory)

- A **language** in Σ^* is a set of strings: $L \subseteq \Sigma^*$
- The set of strings matched by a regex, or accepted by an FSA, is the language defined by the regex or FSA
- Any language defined by a regex or FSA is a **regular language**

Regular languages (standpoint of formal language theory)

- A **language** in Σ^* is a set of strings: $L \subseteq \Sigma^*$
- The set of strings matched by a regex, or accepted by an FSA, is the language defined by the regex or FSA
- Any language defined by a regex or FSA is a **regular language**
- Inductive characterization of the set of **regular languages** for Σ:

Regular languages (standpoint of formal language theory)

- A **language** in Σ^* is a set of strings: $L \subseteq \Sigma^*$
- The set of strings matched by a regex, or accepted by an FSA, is the language defined by the regex or FSA
- Any language defined by a regex or FSA is a **regular language**
- Inductive characterization of the set of **regular languages** for Σ:
  - $\emptyset$ is a regular language;

Regular languages (standpoint of formal language theory)

- A **language** in Σ^* is a set of strings: $L \subseteq \Sigma^*$
- The set of strings matched by a regex, or accepted by an FSA, is the language defined by the regex or FSA
- Any language defined by a regex or FSA is a **regular language**
- Inductive characterization of the set of **regular languages** for Σ:
  - $\emptyset$ is a regular language;
  - For all $a \in \Sigma \cup \epsilon$, $\{a\}$ is a regular language;

Regular languages (standpoint of formal language theory)

- A **language** in Σ^* is a set of strings: $L \subseteq \Sigma^*$
- The set of strings matched by a regex, or accepted by an FSA, is the language defined by the regex or FSA
- Any language defined by a regex or FSA is a **regular language**
- Inductive characterization of the set of **regular languages** for Σ:
  - $\emptyset$ is a regular language;
  - For all $a \in \Sigma \cup \epsilon$, $\{a\}$ is a regular language;
  - If L is a regular language, then so are:

Regular languages (standpoint of formal language theory)

- A **language** in Σ^* is a set of strings: $L \subseteq \Sigma^*$
- The set of strings matched by a regex, or accepted by an FSA, is the language defined by the regex or FSA
- Any language defined by a regex or FSA is a **regular language**
- Inductive characterization of the set of **regular languages** for Σ:
  - $\emptyset$ is a regular language;
  - For all $a \in \Sigma \cup \epsilon$, $\{a\}$ is a regular language;
  - If L is a regular language, then so are:
    - its **complement** $\overline{L} = \{w | w \notin L\}$;

Regular languages (standpoint of formal language theory)

- A **language** in Σ^* is a set of strings: $L \subseteq \Sigma^*$
- The set of strings matched by a regex, or accepted by an FSA, is the language defined by the regex or FSA
- Any language defined by a regex or FSA is a **regular language**
- Inductive characterization of the set of **regular languages** for Σ:
  - $\emptyset$ is a regular language;
  - For all $a \in \Sigma \cup \epsilon$, $\{a\}$ is a regular language;
  - If L is a regular language, then so are:
    - its **complement** $\overline{L} = \{w | w \notin L\}$;
    - Its **Kleene closure** $L^* = \{\forall n \in 0, 1, \cdots : w_1 w_2 \ldots w_n | \forall i \in 1 \ldots N : w_i \in L\}$;

Regular languages (standpoint of formal language theory)

- A **language** in Σ^* is a set of strings: $L \subseteq \Sigma^*$
- The set of strings matched by a regex, or accepted by an FSA, is the language defined by the regex or FSA
- Any language defined by a regex or FSA is a **regular language**
- Inductive characterization of the set of **regular languages** for Σ:
  - $\emptyset$ is a regular language;
  - For all $a \in \Sigma \cup \epsilon$, $\{a\}$ is a regular language;
  - If L is a regular language, then so are:
    - its **complement** $\overline{L} = \{w | w \notin L\}$;
    - Its **Kleene closure** $L^* = \{\forall n \in 0, 1, \cdots : w_1 w_2 \ldots w_n | \forall i \in 1 \ldots N : w_i \in L\}$;
  - If L_1 and L_2 are regular languages, then so are:

Regular languages (standpoint of formal language theory)

- A **language** in Σ^* is a set of strings: $L \subseteq \Sigma^*$
- The set of strings matched by a regex, or accepted by an FSA, is the language defined by the regex or FSA
- Any language defined by a regex or FSA is a **regular language**
- Inductive characterization of the set of **regular languages** for Σ:
  - $\emptyset$ is a regular language;
  - For all $a \in \Sigma \cup \epsilon$, $\{a\}$ is a regular language;
  - If L is a regular language, then so are:
    - its **complement** $\overline{L} = \{w | w \notin L\}$;
    - Its **Kleene closure** $L^* = \{\forall n \in 0, 1, \cdots : w_1 w_2 \ldots w_n | \forall i \in 1 \ldots N : w_i \in L\}$;
  - If L_1 and L_2 are regular languages, then so are:
    - their **concatenation** $L_1 \circ L_2 = \{w_1 w_2 | w_1 \in L_1 \wedge w_2 \in L_2\}$

Regular languages (standpoint of formal language theory)

- A **language** in Σ^* is a set of strings: $L \subseteq \Sigma^*$
- The set of strings matched by a regex, or accepted by an FSA, is the language defined by the regex or FSA
- Any language defined by a regex or FSA is a **regular language**
- Inductive characterization of the set of **regular languages** for Σ:
  - $\emptyset$ is a regular language;
  - For all $a \in \Sigma \cup \epsilon$, $\{a\}$ is a regular language;
  - If L is a regular language, then so are:
    - its **complement** $\overline{L} = \{w | w \notin L\}$;
    - Its **Kleene closure** $L^* = \{\forall n \in 0, 1, \cdots : w_1 w_2 \ldots w_n | \forall i \in 1 \ldots N : w_i \in L\}$;
  - If L_1 and L_2 are regular languages, then so are:
    - their **concatenation** $L_1 \circ L_2 = \{w_1 w_2 | w_1 \in L_1 \wedge w_2 \in L_2\}$
    - their **union** $L_1 \cup L_2$

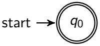
Regular languages (standpoint of formal language theory)

- A **language** in Σ^* is a set of strings: $L \subseteq \Sigma^*$
- The set of strings matched by a regex, or accepted by an FSA, is the language defined by the regex or FSA
- Any language defined by a regex or FSA is a **regular language**
- Inductive characterization of the set of **regular languages** for Σ:
  - $\emptyset$ is a regular language;
  - For all $a \in \Sigma \cup \epsilon$, $\{a\}$ is a regular language;
  - If L is a regular language, then so are:
    - its **complement** $\overline{L} = \{w|w \notin L\}$;
    - Its **Kleene closure** $L^* = \{\forall n \in 0, 1, \cdots : w_1 w_2 \ldots w_n | \forall i \in 1 \ldots N : w_i \in L\}$;
  - If L_1 and L_2 are regular languages, then so are:
    - their **concatenation** $L_1 \circ L_2 = \{w_1 w_2 | w_1 \in L_1 \wedge w_2 \in L_2\}$
    - their **union** $L_1 \cup L_2$
- **Recommendation:** compare this inductive definition with syntax & semantics of regular expressions, as they're closely related

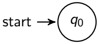
Constructing an FSA from a regex r (slightly informal)

- Base cases:

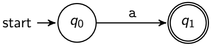
$r = \epsilon$



$r = \emptyset$

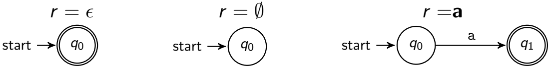


$r =$**a**



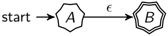
Constructing an FSA from a regex r (slightly informal)

- Base cases:

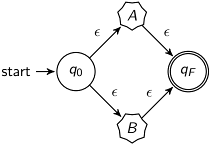


- Additional operators we need for inductive construction:

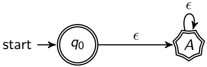
Concatenation



Disjunction

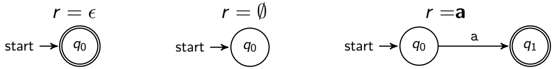


Kleene closure



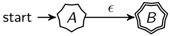
Constructing an FSA from a regex r (slightly informal)

- Base cases:

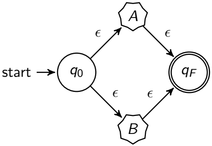


- Additional operators we need for inductive construction:

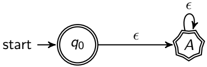
Concatenation



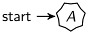
Disjunction



Kleene closure

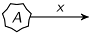


- Semantics regarding the resulting automaton A':

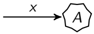


The start state of A is the start state of A'

Every final state of A is final in A'



A' has an x-labeled transition from every final state in A to wherever the arrow points to



A' has an x-labeled transition from wherever the arrow originates to the start state of A

Closure properties of regular languages

- From the past two slides, we saw that the regular languages are **closed** under:
  - Concatenation
  - Kleene closure
  - Union
  - Complementation

Closure properties of regular languages

- From the past two slides, we saw that the regular languages are **closed** under:
  - Concatenation
  - Kleene closure
  - Union
  - Complementation
- Notably, this also implies that the regular languages are also closed under **intersection** (Recommended exercise: why?)

Closure properties of regular languages

- From the past two slides, we saw that the regular languages are **closed** under:
  - Concatenation
  - Kleene closure
  - Union
  - Complementation
- Notably, this also implies that the regular languages are also closed under **intersection** (Recommended exercise: why?)
- Closure under intersection plays an important role in what's coming up!

Finite state transducers, briefly

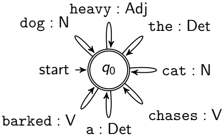
- A **finite-state transducer** consists of:
  - A finite set of N **states** $Q = \{q_0, q_1, \ldots, q_{N-1}\}$, with q_0 the **start state**
  - A finite **input alphabet** Σ of symbols
  - A finite **output alphabet** Γ of symbols
  - A set of **final states** $F \subseteq Q$
  - A **transition relation** Δ: a finite set of **transitions** of the form $q \overset{x:y}{\rightsquigarrow} q'$, with $q, q' \in Q$, $x \in \Sigma \cup \{\epsilon\}$, $y \in \Gamma \cup \{\epsilon\}$. Informally, "if you're in state q and have input x available next, you can output y and move to q'".

Finite state transducers, briefly

- A **finite-state transducer** consists of:
  - A finite set of N **states** $Q = \{q_0, q_1, \ldots, q_{N-1}\}$, with q_0 the **start state**
  - A finite **input alphabet** Σ of symbols
  - A finite **output alphabet** Γ of symbols
  - A set of **final states** $F \subseteq Q$
  - A **transition relation** Δ: a finite set of **transitions** of the form $q \overset{x:y}{\rightsquigarrow} q'$, with $q, q' \in Q$, $x \in \Sigma \cup \{\epsilon\}$, $y \in \Gamma \cup \{\epsilon\}$. Informally, "if you're in state q and have input x available next, you can output y and move to q'".
- Two example finite-state transducers:

Finite state transducers, briefly

- A **finite-state transducer** consists of:
  - A finite set of N **states** $Q = \{q_0, q_1, \ldots, q_{N-1}\}$, with q_0 the **start state**
  - A finite **input alphabet** Σ of symbols
  - A finite **output alphabet** Γ of symbols
  - A set of **final states** $F \subseteq Q$
  - A **transition relation** Δ: a finite set of **transitions** of the form $q \overset{x:y}{\rightsquigarrow} q'$, with $q, q' \in Q$, $x \in \Sigma \cup \{\epsilon\}$, $y \in \Gamma \cup \{\epsilon\}$. Informally, "if you're in state q and have input x available next, you can output y and move to q'".
- Two example finite-state transducers:

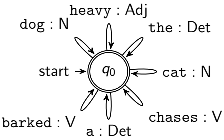


Maps words to parts of speech:

Input	`the dog chases the cat`
Output	Det N V Det N

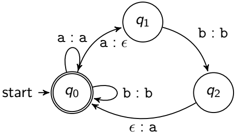
Finite state transducers, briefly

- A **finite-state transducer** consists of:
  - A finite set of N **states** $Q = \{q_0, q_1, \dots, q_{N-1}\}$, with q_0 the **start state**
  - A finite **input alphabet** Σ of symbols
  - A finite **output alphabet** Γ of symbols
  - A set of **final states** $F \subseteq Q$
  - A **transition relation** Δ: a finite set of **transitions** of the form $q \overset{x:y}{\rightsquigarrow} q'$, with $q, q' \in Q$, $x \in \Sigma \cup \{\epsilon\}$, $y \in \Gamma \cup \{\epsilon\}$. Informally, "if you're in state q and have input x available next, you can output y and move to q'".
- Two example finite-state transducers:



Maps words to parts of speech:

Input	the dog chases the cat
Output	Det N V Det N



Optional exchange of ab

Input	Possible outputs
aabbab	aabbab, ababab
	aabbba, ababba

Structural ambiguity in English syntax

- Consider the following generalizations about English:

Structural ambiguity in English syntax

- Consider the following generalizations about English:
  - A sentence can consist of the sequence `They are NP`.

Structural ambiguity in English syntax

- Consider the following generalizations about English:
  - A sentence can consist of the sequence `They are NP`.
  - A sentence can consist of the sequence `They are Vgnd NP`, where `Vgnd` is a GERUND VERB (*jumping*, *sparkling*, *sleeping*, ...).

Structural ambiguity in English syntax

- Consider the following generalizations about English:
  - A sentence can consist of the sequence `They are NP`.
  - A sentence can consist of the sequence `They are Vgnd NP`, where `Vgnd` is a GERUND VERB (*jumping*, *sparkling*, *sleeping*, ...).
  - An **NP** can consist of a noun preceded by zero or more adjectives (e.g., `table`, `precious metals`, `big green buildings`).

Structural ambiguity in English syntax

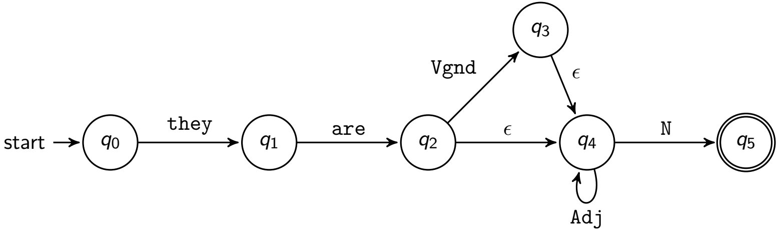
- Consider the following generalizations about English:
  - A sentence can consist of the sequence `They are NP`.
  - A sentence can consist of the sequence `They are Vgnd NP`, where `Vgnd` is a GERUND VERB (*jumping*, *sparkling*, *sleeping*, . . .).
  - An **NP** can consist of a noun preceded by zero or more adjectives (e.g., `table`, `precious metals`, `big green buildings`).
  - A gerund verb can function as an adjective inside an NP (e.g., *sleeping children*, *terrific shooting performance*), so that any English word that can serve in the part of speech `Vgnd` can also serve in the part of speech `Adj`.

Structural ambiguity in English syntax

- Consider the following generalizations about English:
  - A sentence can consist of the sequence `They are NP`.
  - A sentence can consist of the sequence `They are Vgnd NP`, where `Vgnd` is a GERUND VERB (*jumping*, *sparkling*, *sleeping*, . . .).
  - An **NP** can consist of a noun preceded by zero or more adjectives (e.g., `table`, `precious metals`, `big green buildings`).
  - A gerund verb can function as an adjective inside an NP (e.g., *sleeping children*, *terrific shooting performance*), so that any English word that can serve in the part of speech `Vgnd` can also serve in the part of speech `Adj`.
- **Suggested exercise:** before going on, try to express the following generalizations in an FSA for this fragment of English syntax, over the alphabet $\Sigma = \{\texttt{they},\texttt{are},\texttt{Vgnd},\texttt{Adj},\texttt{N}\}$

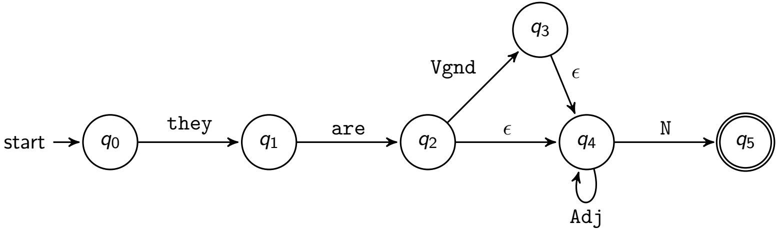
Structural ambiguity in English syntax II

- An FSA that expresses these generalization:



Structural ambiguity in English syntax II

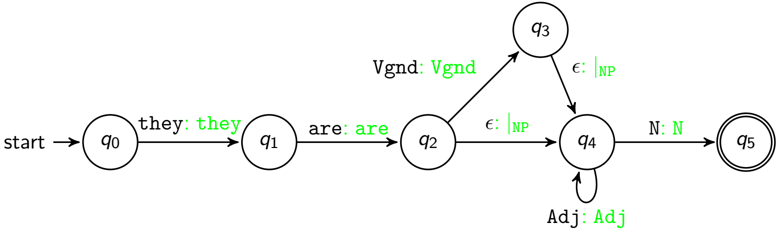
- An FSA that expresses these generalization:



- Implicitly captures a bit of syntax: everything "after" q_4 is part of the post-verbal NP, everything "before" q_4 is outside of it

Structural ambiguity in English syntax II

- An FSA that expresses these generalization:

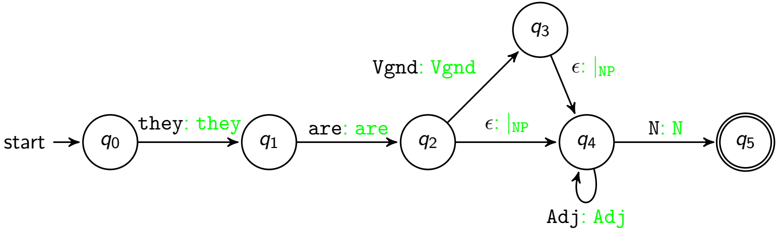


- Implicitly captures a bit of syntax: everything "after" q_4 is part of the post-verbal NP, everything "before" q_4 is outside of it
- We could make this explicit by converting the automaton into a transducer that annotates in the phrase boundary

Input	Output
`they are Vgnd N`	`they are Vgnd` $\vert_{\texttt{NP}}$ `N`
`they are Adj N`	`they are` $\vert_{\texttt{NP}}$ `Adj N`

Structural ambiguity in English syntax II

- An FSA that expresses these generalization:



- Implicitly captures a bit of syntax: everything "after" q_4 is part of the post-verbal NP, everything "before" q_4 is outside of it
- We could make this explicit by converting the automaton into a transducer that annotates in the phrase boundary

Input	Output
`they are Vgnd N`	`they are Vgnd` $|_{NP}$ `N`
`they are Adj N`	`they are` $|_{NP}$ `Adj N`

- The multiple paths through the automaton offer the possibility for different **structural descriptions** of strings

Weak vs. strong generative capacity

- **Weak generative capacity**: what languages (string sets) can be defined by a grammatical formalism?

```
They are Vgnd N
They are Vgnd Adj N
They are Adj N
⋮
```

Weak vs. strong generative capacity

- **Weak generative capacity**: what languages (string sets) can be defined by a grammatical formalism?

```
They are Vgnd N
They are Vgnd Adj N
They are Adj N
⋮
```

- **Strong generative capacity**: what sets of **structural descriptions** can be defined by a grammatical formalism?

  They are Vgnd $[_{NP}$ N $]_{NP}$
  They are Vgnd $[_{NP}$ Adj N $]_{NP}$
  They are Vgnd $[_{NP}$ Adj N $]_{NP}$
  ⋮

Weak vs. strong generative capacity

- **Weak generative capacity**: what languages (string sets) can be defined by a grammatical formalism?

```
They are Vgnd N
They are Vgnd Adj N
They are Adj N
⋮
```

- **Strong generative capacity**: what sets of **structural descriptions** can be defined by a grammatical formalism?

They are Vgnd $[_{NP}$ N $]_{NP}$
They are Vgnd $[_{NP}$ Adj N $]_{NP}$
They are Vgnd $[_{NP}$ Adj N $]_{NP}$
⋮

Phenomenon	Finite-state machine (FSM) **weak**?	FSM **strong**?
Gerund/adjective ambiguity	✓	✓

Multiple prepositional phrases in English

- Consider the following set of generalizations:
  - An NP can consist of a determiner and a noun, optionally followed by one or more **prepositional phrase**s (PPs).
  - A PP consists of a preposition followed by an NP.

Multiple prepositional phrases in English

- Consider the following set of generalizations:
  - An NP can consist of a determiner and a noun, optionally followed by one or more **prepositional phrases** (PPs).
  - A PP consists of a preposition followed by an NP.
- Example NPs that these generalizations license:

```
a joke
a joke about the woman
a joke about the woman with an umbrella
a joke about the woman with an umbrella on the street
⋮
```

Multiple prepositional phrases in English

- Consider the following set of generalizations:
  - An NP can consist of a determiner and a noun, optionally followed by one or more **prepositional phrases** (PPs).
  - A PP consists of a preposition followed by an NP.
- Example NPs that these generalizations license:

```
a joke
a joke about the woman
a joke about the woman with an umbrella
a joke about the woman with an umbrella on the street
⋮
```

- **Recommended exercise:** try writing an FSA for this before going on!

Multiple prepositional phrases in English

- Consider the following set of generalizations:
  - An NP can consist of a determiner and a noun, optionally followed by one or more **prepositional phrase**s (PPs).
  - A PP consists of a preposition followed by an NP.
- Example NPs that these generalizations license:

```
a joke
a joke about the woman
a joke about the woman with an umbrella
a joke about the woman with an umbrella on the street
⋮
```

- **Recommended exercise:** try writing an FSA for this before going on!
- **Observe:** as PPs accumulate, the meanings multiply!

```
a joke about the woman with an umbrella on the street
```

Multiple prepositional phrases in English

- Consider the following set of generalizations:
  - An NP can consist of a determiner and a noun, optionally followed by one or more **prepositional phrases** (PPs).
  - A PP consists of a preposition followed by an NP.
- Example NPs that these generalizations license:

```
a joke
a joke about the woman
a joke about the woman with an umbrella
a joke about the woman with an umbrella on the street
⋮
```

- **Recommended exercise:** try writing an FSA for this before going on!
- **Observe:** as PPs accumulate, the meanings multiply!

`a joke about the woman with an umbrella on the street`

Multiple prepositional phrases in English

- Consider the following set of generalizations:
  - An NP can consist of a determiner and a noun, optionally followed by one or more **prepositional phrase**s (PPs).
  - A PP consists of a preposition followed by an NP.
- Example NPs that these generalizations license:

```
a joke
a joke about the woman
a joke about the woman with an umbrella
a joke about the woman with an umbrella on the street
⋮
```

- **Recommended exercise:** try writing an FSA for this before going on!
- **Observe:** as PPs accumulate, the meanings multiply!

```
a joke about the woman with an umbrella on the street
```

Multiple prepositional phrases in English

- Consider the following set of generalizations:
  - An NP can consist of a determiner and a noun, optionally followed by one or more **prepositional phrases** (PPs).
  - A PP consists of a preposition followed by an NP.
- Example NPs that these generalizations license:

```
a joke
a joke about the woman
a joke about the woman with an umbrella
a joke about the woman with an umbrella on the street
⋮
```

- **Recommended exercise:** try writing an FSA for this before going on!
- **Observe:** as PPs accumulate, the meanings multiply!

```
a joke about the woman with an umbrella on the street
```

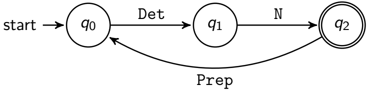
- \# meanings grows as the Catalan numbers, $C_k = \binom{2k}{k} - \binom{2k}{k-1}$ (Church & Patil, 1982)

Multiple prepositional phrases in English II

- An NP can consist of a determiner and a noun, optionally followed by one or more PREPOSITIONAL PHRASEs.
- A **prepositional phrase** (PP) consists of a preposition followed by an NP.

```
a joke
a joke about the woman
a joke about the woman with an umbrella
a joke about the woman with an umbrella on the street
```

- **Example FSA one might try:**

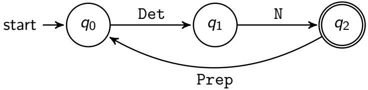


Multiple prepositional phrases in English II

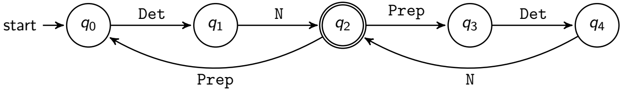
- An NP can consist of a determiner and a noun, optionally followed by one or more PREPOSITIONAL PHRASEs.
- A **prepositional phrase** (PP) consists of a preposition followed by an NP.

```
a joke
a joke about the woman
a joke about the woman with an umbrella
a joke about the woman with an umbrella on the street
```

- **Example FSA one might try:**



- **Problem:** there's only one path, so no mechanism to account for structural ambiguity. We could try adding states. . .

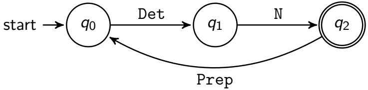


Multiple prepositional phrases in English II

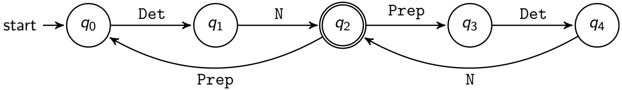
- An NP can consist of a determiner and a noun, optionally followed by one or more PREPOSITIONAL PHRASEs.
- A **prepositional phrase** (PP) consists of a preposition followed by an NP.

```
a joke
a joke about the woman
a joke about the woman with an umbrella
a joke about the woman with an umbrella on the street
```

- **Example FSA one might try:**



- **Problem:** there's only one path, so no mechanism to account for structural ambiguity. We could try adding states...



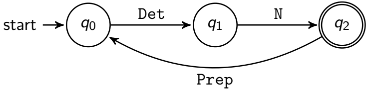
- ...but we would have to add states for every additional level of PP stacking.

Multiple prepositional phrases in English II

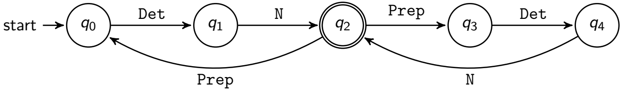
- An NP can consist of a determiner and a noun, optionally followed by one or more PREPOSITIONAL PHRASEs.
- A **prepositional phrase** (PP) consists of a preposition followed by an NP.

```
a joke
a joke about the woman
a joke about the woman with an umbrella
a joke about the woman with an umbrella on the street
```

- **Example FSA one might try:**



- **Problem:** there's only one path, so no mechanism to account for structural ambiguity. We could try adding states...



- ...but we would have to add states for every additional level of PP stacking.
- Since PP stacking is unbounded, a finite-state machine won't be able to generate enough structural descriptions for an unbounded number of PPs.

Weak vs. strong generative capacity

Phenomenon	FSM **weak**?	FSM **strong**?
Gerund/adjective ambiguity	✓	✓
NPs with stacked PP postmodifiers	✓	✗

Case study 3: object-extracted relative clauses

- Consider this sentence of English:

  `the rock that the squirrel likes can be found in the garden`

Case study 3: object-extracted relative clauses

- Consider this sentence of English:

  `the rock that the squirrel likes can be found in the garden`

- Intuitively, it involves combining these two sentences “the right way”:

  `the squirrel likes the rock`

  `the rock can be found in the garden`

Case study 3: object-extracted relative clauses

- Consider this sentence of English:

  `the rock that the squirrel likes can be found in the garden`

- Intuitively, it involves combining these two sentences "the right way":
  `the squirrel likes the rock`
  `the rock can be found in the garden`

- It involves the **syntactic construction** of **relativization**,

`the rock` `can be found in the garden`

Case study 3: object-extracted relative clauses

- Consider this sentence of English:

  `the rock that the squirrel likes can be found in the garden`

- Intuitively, it involves combining these two sentences "the right way":
  `the squirrel likes the rock`
  `the rock can be found in the garden`

- It involves the **syntactic construction** of **relativization**,

```
                    the squirrel likes the rock
 the   rock                                       can be found in the garden
```

Case study 3: object-extracted relative clauses

- Consider this sentence of English:

  `the rock that the squirrel likes can be found in the garden`

- Intuitively, it involves combining these two sentences "the right way":

  `the squirrel likes the rock`

  `the rock can be found in the garden`

- It involves the **syntactic construction** of **relativization**, **extracting** the object ~~the rock~~;

	rock	that	the	squirrel	likes	~~the rock~~							
the	rock						can	be	found	in	the	garden	

Case study 3: object-extracted relative clauses

- Consider this sentence of English:

  `the rock that the squirrel likes can be found in the garden`

- Intuitively, it involves combining these two sentences "the right way":

  `the squirrel likes the rock`

  `the rock can be found in the garden`

- It involves the **syntactic construction** of **relativization**, **extracting** the object ~~the rock~~; the resulting **relative clause** is used as a postmodifier:

	rock	that the squirrel likes ~~the rock~~	
the	rock		can be found in the garden
the	rock	that the squirrel likes	can be found in the garden

Multiple center-embedding with relative clauses

	N	that NP V ~~NP~~	
the	N		⟨rest_of_clause⟩
the	N	that NP V	⟨rest_of_clause⟩

- Subject-modifying object-extracted relative clauses can be nested:

  the rock can be found in the garden

  the rock that the squirrel likes can be found in the garden

  the rock that the squirrel that the dog chases likes can be found in the garden

  the rock that the squirrel that the dog that the woman owns chases likes can be found in the garden

  ⋮

Multiple center-embedding with relative clauses

	N	that NP V ~~NP~~	
the	N		⟨rest_of_clause⟩
the	N	that NP V	⟨rest_of_clause⟩

- Subject-modifying object-extracted relative clauses can be nested:

  `the rock can be found in the garden`

  `the rock that the squirrel likes can be found in the garden`

  `the rock that the squirrel that the dog chases likes can be found in the garden`

  `the rock that the squirrel that the dog that the woman owns chases likes can be found in the garden`

  ⋮

- The resulting sentences start to get very hard to understand, but it is theoretically productive to assume that they are implied by the relativization construction and thus part of the language

Multiple center-embedding with relative clauses

	`N`	`that NP V ~~NP~~`	
`the`	`N`		`⟨rest_of_clause⟩`
`the`	`N`	`that NP V`	`⟨rest_of_clause⟩`

- Subject-modifying object-extracted relative clauses can be nested:

  `the rock can be found in the garden`

  `the rock that the squirrel likes can be found in the garden`

  `the rock that the squirrel that the dog chases likes can be found in the garden`

  `the rock that the squirrel that the dog that the woman owns chases likes can be found in the garden`

  ⋮

- The resulting sentences start to get very hard to understand, but it is theoretically productive to assume that they are implied by the relativization construction and thus part of the language
- That is, they may tax human linguistic **performance**, but they should be part of a theory of human grammatical **competence**

Case study 3: multiple center embedding

	N	that NP V ~~NP~~	
the	N		⟨rest_of_clause⟩
the	N	that NP V	⟨rest_of_clause⟩

the rock can be found in the garden

the rock that the N V can be found in the garden

the rock that the N that the N V V can be found in the garden

the rock that the N that the N that the N V V V can be found in the garden

⋮

Case study 3: multiple center embedding

$$\frac{\begin{array}{llll} & \texttt{N} & \texttt{that NP V } \cancel{\texttt{NP}} & \\ \texttt{the} & \texttt{N} & & \langle \texttt{rest\_of\_clause} \rangle \end{array}}{\begin{array}{llll} \texttt{the} & \texttt{N} & \texttt{that NP V} & \langle \texttt{rest\_of\_clause} \rangle \end{array}}$$

`the rock can be found in the garden`

`the rock that the N V can be found in the garden`

`the rock that the N that the N V V can be found in the garden`

`the rock that the N that the N that the N V V V can be found in the garden`

⋮

- **Recommended exercise:** before going on, think of how you would try to capture this pattern (at least in part) with a finite-state model.

Case study 3: multiple center embedding

	N	that NP V ~~NP~~	
the	N		⟨rest_of_clause⟩
the	N	that NP V	⟨rest_of_clause⟩

the rock can be found in the garden

the rock that the N V can be found in the garden

the rock that the N that the N V V can be found in the garden

the rock that the N that the N that the N V V V can be found in the garden

⋮

- **Recommended exercise:** before going on, think of how you would try to capture this pattern (at least in part) with a finite-state model.
- It turns out that this pattern *cannot* be modeled with finite-state machines

Case study 3: multiple center embedding

```
        N   that NP V NP
the     N                   ⟨rest_of_clause⟩
------------------------------------------
the     N   that NP V       ⟨rest_of_clause⟩
```

`the rock can be found in the garden`

`the rock that the N V can be found in the garden`

`the rock that the N that the N V V can be found in the garden`

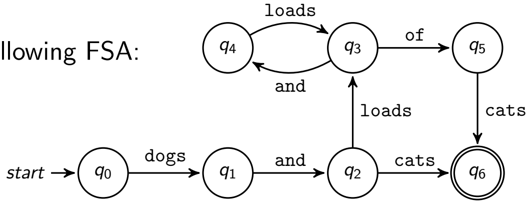
`the rock that the N that the N that the N V V V can be found in the garden`

⋮

- **Recommended exercise:** before going on, think of how you would try to capture this pattern (at least in part) with a finite-state model.
- It turns out that this pattern *cannot* be modeled with finite-state machines
- Showing this rigorously requires additional technical apparatus that we'll cover next

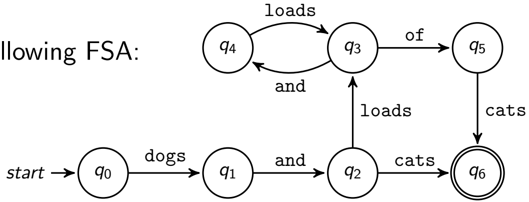
Pumping strings

- Consider the following FSA:



Pumping strings

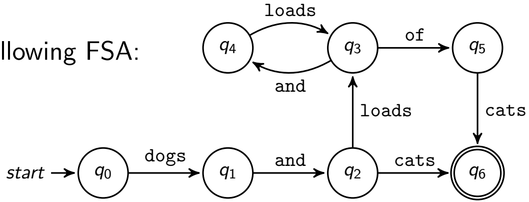
- Consider the following FSA:



- Corresponding regex: `dogs and (cats|loads (and loads)* of cats)`

Pumping strings

- Consider the following FSA:

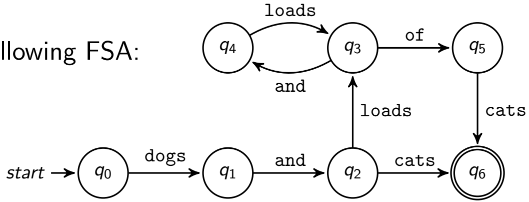


- Corresponding regex: `dogs and (cats|loads (and loads)* of cats)`
- The following sets of strings are accepted:

`dogs and cats`	$(= s_1)$
`dogs and loads of cats`	$(= s_2)$
`dogs and loads and loads of cats`	$(= s_3)$
`dogs and loads and loads and loads of cats`	$(= s_4)$
$\vdots$	

Pumping strings

- Consider the following FSA:



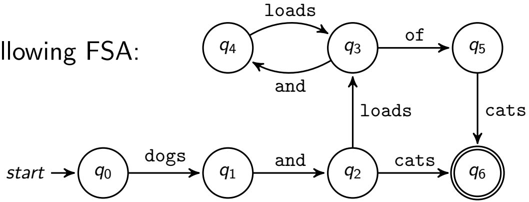
- Corresponding regex: `dogs and (cats|loads (and loads)* of cats)`
- The following sets of strings are accepted:

`dogs and cats`	$(=s_1)$
`dogs and loads of cats`	$(=s_2)$
`dogs and loads and loads of cats`	$(=s_3)$
`dogs and loads and loads and loads of cats`	$(=s_4)$
$\vdots$	

- For s_3, we could repeat the substring `and loads` as many times as we want!

Pumping strings

- Consider the following FSA:



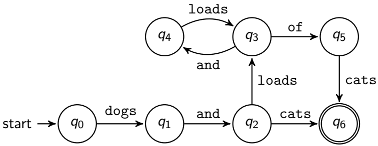
- Corresponding regex: `dogs and (cats|loads (and loads)* of cats)`
- The following sets of strings are accepted:

`dogs and cats`	$(= s_1)$
`dogs and loads of cats`	$(= s_2)$
`dogs and loads and loads of cats`	$(= s_3)$
`dogs and loads and loads and loads of cats`	$(= s_4)$
$\vdots$	

- For s_3, we could repeat the substring `and loads` as many times as we want!
- This is called **pumping** s_3 with the substring `and loads`.

The Pumping Lemma for regular languages

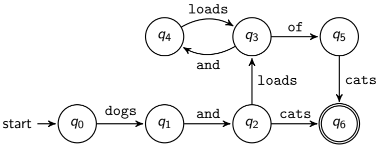
- Informally: if L is regular, then every string that is “long enough” contains some non-empty “intermediate” section that can be arbitrarily pumped without leaving L.



dogs and (cats|loads (and loads)* of cats)

The Pumping Lemma for regular languages

- Informally: if L is regular, then every string that is "long enough" contains some non-empty "intermediate" section that can be arbitrarily pumped without leaving L.



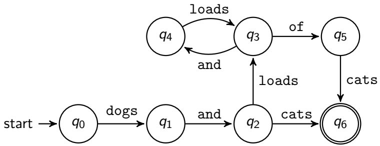
dogs and (cats|loads (and loads)* of cats)

- Formally: if $L \subseteq \Sigma^*$ is regular, then there is some integer k such that for every string $s \in L$ such that $|s| > k$ can be written as $s = xyz$ for $x, y, z \in \Sigma^*$, with:
  - $|y| \geq 1$ (y is non-empty);
  - $|xy| \leq k$;
  - for all $i \geq 0$, $xy^iz \in L$ (y can be pumped in xyz).

Example of the pumping lemma's application

if $L \subseteq \Sigma^*$ is regular, then there is some integer k such that for every string $s \in L$ such that $|s| > k$ can be written as $s = xyz$ for $x, y, z \in \Sigma^*$, with:

- $|y| \geq 1$ (y is non-empty);
- $|xy| \leq k$;
- for all $i \geq 0$, $xy^iz \in L$ (y can be pumped in xyz).



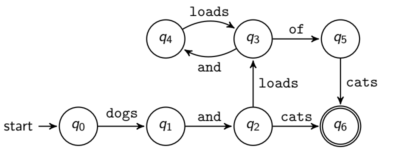
dogs and (cats|loads (and loads)* of cats)

- We can show the Pumping Lemma is satisfied by setting (e.g.) $k = 6$.

Example of the pumping lemma's application

if $L \subseteq \Sigma^*$ is regular, then there is some integer k such that for every string $s \in L$ such that $|s| > k$ can be written as $s = xyz$ for $x, y, z \in \Sigma^*$, with:

- $|y| \geq 1$ (y is non-empty);
- $|xy| \leq k$;
- for all $i \geq 0$, $xy^iz \in L$ (y can be pumped in xyz).



dogs and (cats|loads (and loads)* of cats)

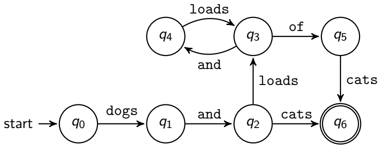
- We can show the Pumping Lemma is satisfied by setting (e.g.) $k = 6$.
- We now need to analyze the infinite set of strings of length > 6, e.g.:

$$\overbrace{\texttt{dogs and loads}}^{x}\ \overbrace{\texttt{and loads}}^{y}\ \overbrace{\texttt{of cats}}^{z}$$

Example of the pumping lemma's application

if $L \subseteq \Sigma^*$ is regular, then there is some integer k such that for every string $s \in L$ such that $|s| > k$ can be written as $s = xyz$ for $x, y, z \in \Sigma^*$, with:

- $|y| \geq 1$ (y is non-empty);
- $|xy| \leq k$;
- for all $i \geq 0$, $xy^iz \in L$ (y can be pumped in xyz).



dogs and (cats|loads (and loads)* of cats)

- We can show the Pumping Lemma is satisfied by setting (e.g.) $k = 6$.
- We now need to analyze the infinite set of strings of length > 6, e.g.:

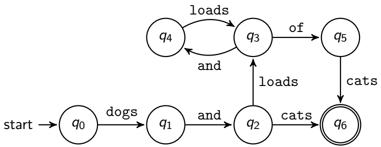
$$\overbrace{\texttt{dogs and loads}}^{x}\ \overbrace{\texttt{and loads}}^{y}\ \overbrace{\texttt{of cats}}^{z}$$

- y is non-empty;

Example of the pumping lemma's application

if $L \subseteq \Sigma^*$ is regular, then there is some integer k such that for every string $s \in L$ such that $|s| > k$ can be written as $s = xyz$ for $x, y, z \in \Sigma^*$, with:

- $|y| \geq 1$ (y is non-empty);
- $|xy| \leq k$;
- for all $i \geq 0$, $xy^iz \in L$ (y can be pumped in xyz).



dogs and (cats|loads (and loads)* of cats)

- We can show the Pumping Lemma is satisfied by setting (e.g.) $k = 6$.
- We now need to analyze the infinite set of strings of length > 6, e.g.:

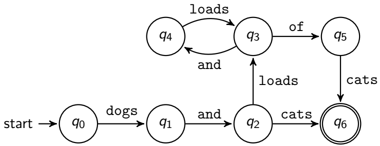
$$\overbrace{\texttt{dogs and loads}}^{x}\ \overbrace{\texttt{and loads}}^{y}\ \overbrace{\texttt{of cats}}^{z}$$

- y is non-empty;
- $|xy| \leq k$;

Example of the pumping lemma's application

if $L \subseteq \Sigma^*$ is regular, then there is some integer k such that for every string $s \in L$ such that $|s| > k$ can be written as $s = xyz$ for $x, y, z \in \Sigma^*$, with:

- $|y| \geq 1$ (y is non-empty);
- $|xy| \leq k$;
- for all $i \geq 0$, $xy^iz \in L$ (y can be pumped in xyz).



dogs and (cats|loads (and loads)* of cats)

- We can show the Pumping Lemma is satisfied by setting (e.g.) $k = 6$.
- We now need to analyze the infinite set of strings of length > 6, e.g.:

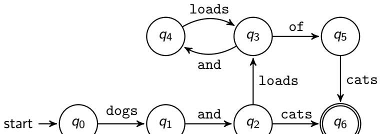
$$\overbrace{\texttt{dogs and loads}}^{x}\ \overbrace{\texttt{and loads}}^{y}\ \overbrace{\texttt{of cats}}^{z}$$

- y is non-empty;
- $|xy| \leq k$;
- xy^iz is in the language for all $i \geq 0$.

Example of the pumping lemma's application

if $L \subseteq \Sigma^*$ is regular, then there is some integer k such that for every string $s \in L$ such that $|s| > k$ can be written as $s = xyz$ for $x, y, z \in \Sigma^*$, with:

- $|y| \geq 1$ (y is non-empty);
- $|xy| \leq k$;
- for all $i \geq 0$, $xy^iz \in L$ (y can be pumped in xyz).



dogs and (cats|loads (and loads)* of cats)

- We can show the Pumping Lemma is satisfied by setting (e.g.) $k = 6$.
- We now need to analyze the infinite set of strings of length > 6, e.g.:

$$\overbrace{\texttt{dogs and loads}}^{x}\ \overbrace{\texttt{and loads}}^{y}\ \overbrace{\texttt{of cats}}^{z}$$

- y is non-empty;
- $|xy| \leq k$;
- xy^iz is in the language for all $i \geq 0$.
- We could do a similar decomposition for every other string in the language of length > 6. Thus, the pumping lemma is satisfied, and the language is regular!

The argument that English is not regular

- We define an idealization of the natural language English and call it ENGLISH.

The argument that English is not regular

- We define an idealization of the natural language English and call it $\textsc{English}$.
- We use a regular expression to define a regular language $L^\ddagger$ and call its intersection with English $L^\dagger = \textsc{English} \cap L^\ddagger$.

The argument that English is not regular

- We define an idealization of the natural language English and call it $\textsc{English}$.
- We use a regular expression to define a regular language $L^\ddagger$ and call its intersection with English $L^\dagger = \textsc{English} \cap L^\ddagger$.
- If $\textsc{English}$ is regular, then $L^\dagger$ must be regular, since regular languages are **closed under intersection**.

The argument that English is not regular

- We define an idealization of the natural language English and call it $\textsc{English}$.
- We use a regular expression to define a regular language $L^{\ddagger}$ and call its intersection with English $L^{\dagger} = \textsc{English} \cap L^{\ddagger}$.
- If $\textsc{English}$ is regular, then $L^{\dagger}$ must be regular, since regular languages are **closed under intersection**.
- We use the pumping lemma for regular languages to show that $L^{\dagger}$ is *not* regular.

The argument that English is not regular

- We define an idealization of the natural language English and call it $\textsc{English}$.
- We use a regular expression to define a regular language $L^{\ddagger}$ and call its intersection with English $L^{\dagger} = \textsc{English} \cap L^{\ddagger}$.
- If $\textsc{English}$ is regular, then $L^{\dagger}$ must be regular, since regular languages are **closed under intersection**.
- We use the pumping lemma for regular languages to show that $L^{\dagger}$ is *not* regular.
- Therefore, $\textsc{English}$ is not regular.

Evaluating multiple center-embedding with the Pumping Lemma

```
the rock can be found in the garden
the rock that the N V can be found in the garden
the rock that the N that the N V V can be found in the garden
the rock that the N that the N that the N V V V can be found in the
garden
⋮
```

- We will call this infinite set $L^{\dagger}$ and summarize it as:

  $$\texttt{the rock (that the N)}^{i}\ \texttt{V}^{i}\ \texttt{can be found in the garden}$$

  for $i \geq 0$, with the multiple appearances of ^{i} indicating that `(that the N)` and `V` must appear in place the same number of times as each other.

Evaluating multiple center-embedding with the Pumping Lemma

```
the rock can be found in the garden
the rock that the N V can be found in the garden
the rock that the N that the N V V can be found in the garden
the rock that the N that the N that the N V V V can be found in the
garden
⋮
```

- We will call this infinite set $L^{\dagger}$ and summarize it as:

  `the rock (that the N)`^{i} `V`^{i} `can be found in the garden`

  for $i \geq 0$, with the multiple appearances of ^{i} indicating that `(that the N)` and `V` must appear in place the same number of times as each other.
- Note that strings of the following form are not OK English when $i \neq j$:

  `the rock (that the N)`^{i} `V`^{j} `can be found in the garden`

  e.g., **the rock that the squirrel can be found in the garden* is not in English.

Evaluating multiple center-embedding with the Pumping Lemma

```
the rock can be found in the garden
the rock that the N V can be found in the garden
the rock that the N that the N V V can be found in the garden
the rock that the N that the N that the N V V V can be found in the the
garden
⋮
```

- We will call this infinite set $L^\dagger$ and summarize it as:

  $$\texttt{the rock (that the N)}^i\ \texttt{V}^i\ \texttt{can be found in the garden}$$

  for $i \geq 0$, with the multiple appearances of ^{i} indicating that `(that the N)` and `V` must appear in place the same number of times as each other.
- Note that strings of the following form are not OK English when $i \neq j$:

  $$\texttt{the rock (that the N)}^i\ \texttt{V}^j\ \texttt{can be found in the garden}$$

  e.g., **the rock that the squirrel can be found in the garden* is not in English.
- We assume that $L^\dagger$ is part (formally, a subset) of ENGLISH.

Intersecting $\textsc{English}$ with a regular language

$L^{\dagger} = \texttt{the rock (that the N)}^{i}\ \texttt{V}^{i}\ \texttt{can be found in the garden}$

- Call the regular language to the regex below $L^{\ddagger}$:

  `the rock (that the N)* V* can be found in the garden`

Intersecting ENGLISH with a regular language

$$L^{\dagger} = \texttt{the rock (that the N)}^{i}\ \texttt{V}^{i}\ \texttt{can be found in the garden}$$

- Call the regular language to the regex below $L^{\ddagger}$:

  `the rock (that the N)* V* can be found in the garden`

- Then

$$\textsc{English} \cap L^{\ddagger} = L^{\dagger}$$

Intersecting $\textsc{English}$ with a regular language

$L^{\dagger} = \texttt{the rock (that the N)}^{i}\ \texttt{V}^{i}\ \texttt{can be found in the garden}$

- Call the regular language to the regex below $L^{\ddagger}$:

  `the rock (that the N)* V* can be found in the garden`

- Then

$$\textsc{English} \cap L^{\ddagger} = L^{\dagger}$$

- So, if $\textsc{English}$ is regular, then $L^{\dagger}$ is regular!

Intersecting ENGLISH with a regular language

$L^{\dagger} =$ `the rock (that the N)`^{i} `V`^{i} `can be found in the garden`

- Call the regular language to the regex below $L^{\ddagger}$:

  `the rock (that the N)* V* can be found in the garden`

- Then

$$\textsc{English} \cap L^{\ddagger} = L^{\dagger}$$

- So, if ENGLISH is regular, then $L^{\dagger}$ is regular!
- We will use the contrapositive: if $L^{\dagger}$ is *not* regular, then ENGLISH is not regular either.

Intersecting ENGLISH with a regular language

$L^{\dagger} = $ `the rock (that the N)`^{i} `V`^{i} `can be found in the garden`

- Call the regular language to the regex below $L^{\ddagger}$:

  `the rock (that the N)* V* can be found in the garden`

- Then

$$\textsc{English} \cap L^{\ddagger} = L^{\dagger}$$

- So, if ENGLISH is regular, then $L^{\dagger}$ is regular!
- We will use the contrapositive: if $L^{\dagger}$ is *not* regular, then ENGLISH is not regular either.
- We will be able to use the Pumping Lemma for regular languages to show that $L^{\dagger}$ is *not* regular

Proving by contradiction that $L^\dagger$ is not regular

if $L \subseteq \Sigma^*$ is regular, then there is some integer k such that for every string $s \in L$ such that $|s| > k$ can be written as $s = xyz$ for $x, y, z \in \Sigma^*$, with:

- $|y| \geq 1$ (y is non-empty);
- $|xy| \leq k$;
- for all $i \geq 0$, $xy^iz \in L$ (y can be pumped in xyz).

- If $L^\dagger$ were regular, then there must be some k per the above.

Proving by contradiction that $L^\dagger$ is not regular

if $L \subseteq \Sigma^*$ is regular, then there is some integer k such that for every string $s \in L$ such that $|s| > k$ can be written as $s = xyz$ for $x, y, z \in \Sigma^*$, with:

- $|y| \geq 1$ (y is non-empty);
- $|xy| \leq k$;
- for all $i \geq 0$, $xy^iz \in L$ (y can be pumped in xyz).

- If $L^\dagger$ were regular, then there must be some k per the above.
- Suppose there is some such k. Then consider the following string:

  the rock (that the N)^k V^k can be found in the garden

Proving by contradiction that $L^\dagger$ is not regular

if $L \subseteq \Sigma^*$ is regular, then there is some integer k such that for every string $s \in L$ such that $|s| > k$ can be written as $s = xyz$ for $x, y, z \in \Sigma^*$, with:

- $|y| \geq 1$ (y is non-empty);
- $|xy| \leq k$;
- for all $i \geq 0$, $xy^iz \in L$ (y can be pumped in xyz).

- If $L^\dagger$ were regular, then there must be some k per the above.
- Suppose there is some such k. Then consider the following string:

  $\texttt{the rock (that the N)}^k\ \texttt{V}^k\ \texttt{can be found in the garden}$

- We should be able to find an appropriate xyz decomposition.

Proving by contradiction that $L^\dagger$ is not regular

if $L \subseteq \Sigma^*$ is regular, then there is some integer k such that for every string $s \in L$ such that $|s| > k$ can be written as $s = xyz$ for $x, y, z \in \Sigma^*$, with:

- $|y| \geq 1$ (y is non-empty);
- $|xy| \leq k$;
- for all $i \geq 0$, $xy^iz \in L$ (y can be pumped in xyz).

- If $L^\dagger$ were regular, then there must be some k per the above.
- Suppose there is some such k. Then consider the following string:

  $\texttt{the rock (that the N)}^k\ \texttt{V}^k\ \texttt{can be found in the garden}$

- We should be able to find an appropriate xyz decomposition.
- Since $|xy| \leq k$, y cannot contain any $\texttt{V}$s. We can distinguish two cases:

Proving by contradiction that $L^\dagger$ is not regular

if $L \subseteq \Sigma^*$ is regular, then there is some integer k such that for every string $s \in L$ such that $|s| > k$ can be written as $s = xyz$ for $x, y, z \in \Sigma^*$, with:

- $|y| \geq 1$ (y is non-empty);
- $|xy| \leq k$;
- for all $i \geq 0$, $xy^iz \in L$ (y can be pumped in xyz).

- If $L^\dagger$ were regular, then there must be some k per the above.
- Suppose there is some such k. Then consider the following string:

  $\texttt{the rock (that the N)}^k\ \texttt{V}^k\ \texttt{can be found in the garden}$

- We should be able to find an appropriate xyz decomposition.
- Since $|xy| \leq k$, y cannot contain any `V`s. We can distinguish two cases:
  - y could contain one or more `N`s. But then pumping y would yield a string that doesn't have the same number of `N`s and `V`s, which wouldn't be in $L^\dagger$!

Proving by contradiction that $L^\dagger$ is not regular

if $L \subseteq \Sigma^*$ is regular, then there is some integer k such that for every string $s \in L$ such that $|s| > k$ can be written as $s = xyz$ for $x, y, z \in \Sigma^*$, with:

- $|y| \geq 1$ (y is non-empty);
- $|xy| \leq k$;
- for all $i \geq 0$, $xy^iz \in L$ (y can be pumped in xyz).

- If $L^\dagger$ were regular, then there must be some k per the above.
- Suppose there is some such k. Then consider the following string:

  $\texttt{the rock (that the N)}^k\ \texttt{V}^k\ \texttt{can be found in the garden}$

- We should be able to find an appropriate xyz decomposition.
- Since $|xy| \leq k$, y cannot contain any `V`s. We can distinguish two cases:
  - y could contain one or more `N`s. But then pumping y would yield a string that doesn't have the same number of `N`s and `V`s, which wouldn't be in $L^\dagger$!
  - y might be the string `the`, `that`, or `that the`. But then pumping y will also yield a string outside of $L^\dagger$!

Proving by contradiction that $L^\dagger$ is not regular

if $L \subseteq \Sigma^*$ is regular, then there is some integer k such that for every string $s \in L$ such that $|s| > k$ can be written as $s = xyz$ for $x, y, z \in \Sigma^*$, with:

- $|y| \geq 1$ (y is non-empty);
- $|xy| \leq k$;
- for all $i \geq 0$, $xy^iz \in L$ (y can be pumped in xyz).

- If $L^\dagger$ were regular, then there must be some k per the above.
- Suppose there is some such k. Then consider the following string:

  `the rock (that the N)`^{k} `V`^{k} `can be found in the garden`

- We should be able to find an appropriate xyz decomposition.
- Since $|xy| \leq k$, y cannot contain any `V`s. We can distinguish two cases:
  - y could contain one or more `N`s. But then pumping y would yield a string that doesn't have the same number of `N`s and `V`s, which wouldn't be in $L^\dagger$!
  - y might be the string `the`, `that`, or `that the`. But then pumping y will also yield a string outside of $L^\dagger$!
- Either way, a contradiction: for ENGLISH to be regular, $L^\dagger$ had to be regular. But $L^\dagger$ can't be regular according to the Pumping Lemma!

Proving by contradiction that $L^\dagger$ is not regular

if $L \subseteq \Sigma^*$ is regular, then there is some integer k such that for every string $s \in L$ such that $|s| > k$ can be written as $s = xyz$ for $x, y, z \in \Sigma^*$, with:

- $|y| \geq 1$ (y is non-empty);
- $|xy| \leq k$;
- for all $i \geq 0$, $xy^iz \in L$ (y can be pumped in xyz).

- If $L^\dagger$ were regular, then there must be some k per the above.
- Suppose there is some such k. Then consider the following string:

  $\texttt{the rock (that the N)}^k\ \texttt{V}^k\ \texttt{can be found in the garden}$

- We should be able to find an appropriate xyz decomposition.
- Since $|xy| \leq k$, y cannot contain any `V`s. We can distinguish two cases:
  - y could contain one or more `N`s. But then pumping y would yield a string that doesn't have the same number of `N`s and `V`s, which wouldn't be in $L^\dagger$!
  - y might be the string `the`, `that`, or `that the`. But then pumping y will also yield a string outside of $L^\dagger$!
- Either way, a contradiction: for English to be regular, $L^\dagger$ had to be regular. But $L^\dagger$ can't be regular according to the Pumping Lemma!
- Thus English is **not regular**

Summary

Phenomenon	FSM **weak**?	FSM **strong**?
Gerund/adjective ambiguity	✓	✓
NPs with stacked PP postmodifiers	✓	✗
Multiply nested object relative clauses	✗	✗

Summary

Phenomenon	FSM **weak**?	FSM **strong**?
Gerund/adjective ambiguity	✓	✓
NPs with stacked PP postmodifiers	✓	✗
Multiply nested object relative clauses	✗	✗

- Both **weak** and **strong** generative capacity of grammatical formalisms are of interest

Summary

Phenomenon	FSM **weak**?	FSM **strong**?
Gerund/adjective ambiguity	✓	✓
NPs with stacked PP postmodifiers	✓	✗
Multiply nested object relative clauses	✗	✗

- Both **weak** and **strong** generative capacity of grammatical formalisms are of interest
- Finite-state models can capture some features of English syntactic structure, but have neither the strong nor the weak generative capacity for other features

Summary

Phenomenon	FSM **weak**?	FSM **strong**?
Gerund/adjective ambiguity	✓	✓
NPs with stacked PP postmodifiers	✓	✗
Multiply nested object relative clauses	✗	✗

- Both **weak** and **strong** generative capacity of grammatical formalisms are of interest
- Finite-state models can capture some features of English syntactic structure, but have neither the strong nor the weak generative capacity for other features
- **Looking ahead:** these classic results motivate more expressive grammatical formalisms that have been central to the cognitive science of language for decades