Learnability of Quantifiers

Computational Representations of Quantifiers

Shane Steinert-Threlkeld & Jakub Szymanik
University of Washington, Linguistics
ILLC
University of Amsterdam

Outline

Recap

Yesterday:

- Basic examples of generalized quantifiers
- Constraints on GQs: ISOM, EXT, CONS, MON
- Definability

Today:

- We will start by exploring consequences of the proposed universals
- We will give *computational representations* of GQs by looking at *machines* accepting corresponding *formal languages*
- Definability results of the kind we saw yesterday will correspond to the Chomsky Hierarchy

Recap

Yesterday:

- Basic examples of generalized quantifiers
- Constraints on GQs: ISOM, EXT, CONS, MON
- Definability

Today:

- We will start by exploring consequences of the proposed universals
- We will give *computational representations* of GQs by looking at *machines* accepting corresponding *formal languages*
- Definability results of the kind we saw yesterday will correspond to the Chomsky Hierarchy

Outline

1. Recap
2. Universals and representations
3. Quantifiers as Computational Problems
   - Regular Languages and Finite Automata
   - Context-Free Languages and Quantifiers

Recall...

ISOM If $(M, A, B) \cong (M', A', B')$, then $Q_{\mathbf{M}}(A, B) \Leftrightarrow Q_{\mathbf{M'}}(A', B')$

EXT If $M \subseteq M'$, then $Q_{\mathbf{M}}(A, B) \Leftrightarrow Q_{\mathbf{M'}}(A, B)$

CONS $Q_{\mathbf{M}}(A, B) \Leftrightarrow Q_{\mathbf{M}}(A, A \cap B)$

Consequences of Constraints

Theorem

A quantifier Q satisfies **CONS** *and* **EXT** *if and only if for every M, M' and $A, B \subseteq M$, $A', B' \subseteq M'$, if $|A - B| = |A' - B'|$ and $|A \cap B| = |A' \cap B'|$, then $Q_M AB \Leftrightarrow Q_{M'} A'B'$.*

GQs as Binary Relations on $\mathbb{N}$

In other words, quantifiers that satisfy **CONS** and **EXT** can be summarized succinctly as binary relations on natural numbers. Given Q we define:

$$Q^c xy \quad \Leftrightarrow \quad \exists\,(M, A, B)\; Q_M AB \text{ and } |A - B| = x, |A \cap B| = y.$$

Standard generalized quantifiers can thus be seen as particular simple cases.

$$\textit{every}^c\; xy \Leftrightarrow x = 0$$

$$\textit{some}^c\; xy \Leftrightarrow y > 0$$

$$\textit{at least three}^c\; xy \Leftrightarrow y \geq 3$$

$$\textit{most}^c\; xy \Leftrightarrow y > x$$

$$\textit{an even number of}^c\; xy \Leftrightarrow y = 2n \text{ for some } n \in \mathbb{N}$$

GQs as Binary Relations on $\mathbb{N}$

In other words, quantifiers that satisfy **CONS** and **EXT** can be summarized succinctly as binary relations on natural numbers. Given Q we define:

$$Q^c xy \quad \Leftrightarrow \quad \exists\,(M, A, B)\; Q_M AB \text{ and } |A - B| = x, |A \cap B| = y.$$

Standard generalized quantifiers can thus be seen as particular simple cases.

$$\begin{aligned} \textit{every}^c\; xy &\Leftrightarrow x = 0 \\ \textit{some}^c\; xy &\Leftrightarrow y > 0 \\ \textit{at least three}^c\; xy &\Leftrightarrow y \geq 3 \\ \textit{most}^c\; xy &\Leftrightarrow y > x \\ \textit{an even number of}^c\; xy &\Leftrightarrow y = 2n \text{ for some } n \in \mathbb{N} \end{aligned}$$

Number triangle representation

Let Q be a CE-quantifier. Then define relation: $Q(k, m)$ iff there are M and $A, B \subseteq M$ such that $\mathrm{card}(A - B) = k$, $\mathrm{card}(A \cap B) = m$, and $\mathrm{Q}_M(A, B)$.

(0,0)

(1,0) (0,1)

(2,0) (1,1) (0,2)

(3,0) (2,1) (1,2) (0,3)

(4,0) (3,1) (2,2) (1,3) (0,4)

... ...

Example 1: which?

```
                -
            -       +
        -       +       +
    -       +       +       +
-       +       +       +       +
...                                 ...
```

Example 2: which?

$$
\begin{array}{ccccccccc}
 & & & & \text{—} & & & & \\
 & & & - & & - & & & \\
 & & - & & - & & + & & \\
 & - & & - & & + & & + & \\
- & & - & & + & & + & & +
\end{array}
$$

$$\ldots$$

Example 3: which?

$$
\begin{array}{ccccccccc}
 & & & & - & & & & \\
 & & & + & & + & & & \\
 & & + & & + & & - & & \\
 & + & & + & & - & & - & \\
+ & & + & & - & & - & & -
\end{array}
$$

$$\ldots$$

Example 4: which?

$$
\begin{array}{ccccccccc}
 & & & & - & & & & \\
 & & & + & & + & & & \\
 & & + & & + & & + & & \\
 & + & & + & & + & & - & \\
+ & & + & & + & & - & & - \\
 & & & & \ldots & & & &
\end{array}
$$

Example 5: which?

$$
\begin{array}{ccccccccc}
 & & & & \text{—} & & & & \\
 & & & + & & - & & & \\
 & & + & & - & & + & & \\
 & + & & - & & + & & - & \\
 + & & - & & + & & - & & + \\
 & & & & \cdots & & & &
\end{array}
$$

Example 6: which?

$$
\begin{array}{ccccccccc}
 & & & & - & & & & \\
 & & & + & & - & & & \\
 & & + & & - & & - & & \\
 & + & & - & & - & & + & \\
+ & & - & & - & & + & & - \\
 & & & & \cdots & & & &
\end{array}
$$

Example 7: which?

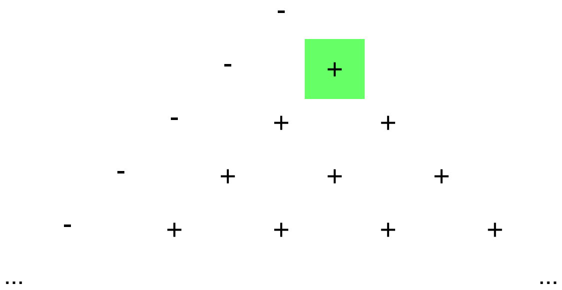
$$
\begin{array}{ccccccccccccc}
 & & & & & & — & & & & & & \\
 & & & & & − & & + & & & & & \\
 & & & & − & & − & & + & & & & \\
 & & & − & & − & & + & & + & & & \\
 & & − & & − & & − & & + & & + & & \\
 & − & & − & & − & & + & & + & & + & \\
− & & − & & − & & − & & + & & + & & +
\end{array}
$$

$\cdots$

Example 8: which?

$$
\begin{array}{c}
-\\
- \quad +\\
- \quad + \quad +\\
- \quad - \quad + \quad +\\
- \quad - \quad + \quad + \quad +\\
- \quad - \quad + \quad + \quad + \quad +\\
- \quad - \quad - \quad + \quad + \quad + \quad +\\
\dots
\end{array}
$$

↑MON



↑MON

-

- +

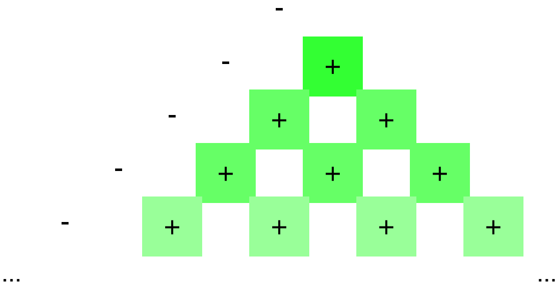
- + +

- + + +

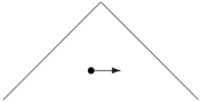
- + + + +

... ...

↑MON



MON↑



Outline

Languages/Problems - basic definitions

- *Alphabet* is any non-empty finite set of symbols, e.g., $A = \{a, b\}$ and $B = \{0, 1\}$.
- A *word (string)* is a finite sequence of symbols from a given alphabet, e.g., "1110001110".
- The *empty word*, ε, is a sequence without symbols.
- The *length of a word* is the number of symbols in it.
- For a in the alphabet and w a word, $\#_a(w)$ denotes the number of as in w.
- *The set of all words over alphabet* Γ is denoted by Γ^*, e.g., $\{0, 1\}^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$.
- Any set of words, a subset of Γ^*, will be called a *problems/language*.
- A great reference: Hopcroft and Ullman 1979.

Languages/Problems - basic definitions

- *Alphabet* is any non-empty finite set of symbols, e.g., $A = \{a, b\}$ and $B = \{0, 1\}$.
- A *word (string)* is a finite sequence of symbols from a given alphabet, e.g., "1110001110".
- The *empty word*, ε, is a sequence without symbols.
- The *length of a word* is the number of symbols in it.
- For a in the alphabet and w a word, $\#_a(w)$ denotes the number of as in w.
- *The set of all words over alphabet* Γ is denoted by Γ^*, e.g., $\{0, 1\}^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$.
- Any set of words, a subset of Γ^*, will be called a *problems/language*.
- A great reference: Hopcroft and Ullman 1979.

Languages/Problems - basic definitions

- *Alphabet* is any non-empty finite set of symbols, e.g., $A = \{a, b\}$ and $B = \{0, 1\}$.
- A *word (string)* is a finite sequence of symbols from a given alphabet, e.g., “1110001110”.
- The *empty word*, ε, is a sequence without symbols.
- The *length of a word* is the number of symbols in it.
- For a in the alphabet and w a word, $\#_a(w)$ denotes the number of as in w.
- *The set of all words over alphabet* Γ is denoted by Γ^*, e.g., $\{0, 1\}^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$.
- Any set of words, a subset of Γ^*, will be called a *problems/language*.
- A great reference: Hopcroft and Ullman 1979.

Languages/Problems - basic definitions

- *Alphabet* is any non-empty finite set of symbols, e.g., $A = \{a, b\}$ and $B = \{0, 1\}$.
- A *word (string)* is a finite sequence of symbols from a given alphabet, e.g., "1110001110".
- The *empty word*, ε, is a sequence without symbols.
- The *length of a word* is the number of symbols in it.
- For a in the alphabet and w a word, $\#_a(w)$ denotes the number of as in w.
- *The set of all words over alphabet* Γ is denoted by Γ^*, e.g., $\{0, 1\}^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$.
- Any set of words, a subset of Γ^*, will be called a *problems/language*.
- A great reference: Hopcroft and Ullman 1979.

Languages/Problems - basic definitions

- *Alphabet* is any non-empty finite set of symbols, e.g., $A = \{a, b\}$ and $B = \{0, 1\}$.
- A *word (string)* is a finite sequence of symbols from a given alphabet, e.g., "1110001110".
- The *empty word*, ε, is a sequence without symbols.
- The *length of a word* is the number of symbols in it.
- For a in the alphabet and w a word, $\#_a(w)$ denotes the number of as in w.
- *The set of all words over alphabet* Γ is denoted by Γ^*, e.g., $\{0, 1\}^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$.
- Any set of words, a subset of Γ^*, will be called a *problems/language*.
- A great reference: Hopcroft and Ullman 1979.

Languages/Problems - basic definitions

- *Alphabet* is any non-empty finite set of symbols, e.g., $A = \{a, b\}$ and $B = \{0, 1\}$.
- A *word (string)* is a finite sequence of symbols from a given alphabet, e.g., “1110001110”.
- The *empty word*, ε, is a sequence without symbols.
- The *length of a word* is the number of symbols in it.
- For a in the alphabet and w a word, $\#_a(w)$ denotes the number of as in w.
- *The set of all words over alphabet* Γ is denoted by Γ^*, e.g., $\{0, 1\}^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$.
- Any set of words, a subset of Γ^*, will be called a *problems/language*.
- A great reference: Hopcroft and Ullman 1979.

Languages/Problems - basic definitions

- *Alphabet* is any non-empty finite set of symbols, e.g., $A = \{a, b\}$ and $B = \{0, 1\}$.
- A *word (string)* is a finite sequence of symbols from a given alphabet, e.g., "1110001110".
- The *empty word*, ε, is a sequence without symbols.
- The *length of a word* is the number of symbols in it.
- For a in the alphabet and w a word, $\#_a(w)$ denotes the number of as in w.
- *The set of all words over alphabet* Γ is denoted by Γ^*, e.g., $\{0, 1\}^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$.
- Any set of words, a subset of Γ^*, will be called a *problems/language*.
- A great reference: Hopcroft and Ullman 1979.

The Main Idea

We will associate each GQ (of type $\langle 1 \rangle$ or CE of type $\langle 1, 1 \rangle$) with a formal language in $\{0, 1\}^*$. First, every finite model will get mapped to a string s by

- Starting with a model $\mathcal{M} = \langle M, A, B \rangle$
- Enumerating A as $\vec{a}$
- Writing a 0 for each element of $A \setminus B$ and a 1 for each element of $A \cap B$

From the earlier results, we have that

$$\mathcal{M} \in Q \quad \Leftrightarrow \quad \langle \#_0(s), \#_1(s) \rangle \in Q^c,$$

More Formally Defined

Definition

Let $\mathcal{M} = \langle M, A, B \rangle$ be a model, $\vec{a}$ an enumeration of A, and $n = |A|$. We define $\tau\left(\vec{a}, B\right) \in \{0,1\}^n$ by

$$\left(\tau\left(\vec{a}, B\right)\right)_i = \begin{cases} 0 & a_i \in A \setminus B \\ 1 & a_i \in A \cap B \end{cases}$$

Thus, τ defines the string corresponding to a particular finite model.

Definition

For a type $\langle 1, 1 \rangle$ quantifier Q, define *the language of Q* as

$$\mathcal{L}_Q = \left\{ s \in \{0,1\}^* \mid \langle \#_0(s), \#_1(s) \rangle \in Q^c \right\}$$

More Formally Defined

Definition

Let $\mathcal{M} = \langle M, A, B \rangle$ be a model, $\vec{a}$ an enumeration of A, and $n = |A|$. We define $\tau\left(\vec{a}, B\right) \in \{0,1\}^n$ by

$$\left(\tau\left(\vec{a}, B\right)\right)_i = \begin{cases} 0 & a_i \in A \setminus B \\ 1 & a_i \in A \cap B \end{cases}$$

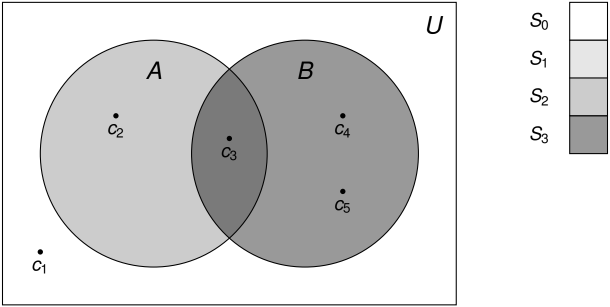
Thus, τ defines the string corresponding to a particular finite model.

Definition

For a type $\langle 1,1 \rangle$ quantifier Q, define *the language of Q* as

$$\mathcal{L}_Q = \left\{ s \in \{0,1\}^* \mid \langle \#_0(s), \#_1(s) \rangle \in Q^c \right\}$$

Example



Examples of Quantifier Languages

- $\mathcal{L}_{\text{every}} = \{w \mid \#_0(w) = 0\}$
- $\mathcal{L}_{\text{some}} = \{w \mid \#_1(w) > 0\}$
- $\mathcal{L}_{\text{most}} = \{w \mid \#_1(w) > \#_0(w)\}$

Outline

1. Recap

2. Universals and representations

3. Quantifiers as Computational Problems
   - Regular Languages and Finite Automata
   - Context-Free Languages and Quantifiers

Finite automata

Definition

A *non-deterministic finite automaton* (FA) is a tuple (A, Q, q_s, F, δ), where:

- A is an input alphabet;
- Q is a finite set of states;
- $q_s \in Q$ is an initial state;
- $F \subseteq Q$ is a set of accepting states;
- $\delta : Q \times A \longrightarrow \mathcal{P}(Q)$ is a transition function.

Regular languages

Definition

The language accepted (recognized) by some FA H, L(H), is the set of all words over the alphabet A which are accepted by H.

Definition

We say that a *language* $L \subseteq A^*$ *is regular* if and only if there exists some FA H such that $L = L(H)$.

Examples of GQ Automata

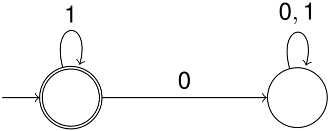


Figure: A finite state automaton for *every*.

Figure: A finite state automaton for *at least two*.

Examples of GQ Automata

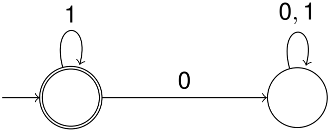


Figure: A finite state automaton for *every*.

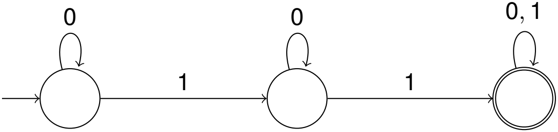


Figure: A finite state automaton for *at least two*.

First-Order Definability

Not all quantifiers whose languages are accepted by DFAs are FO definable:

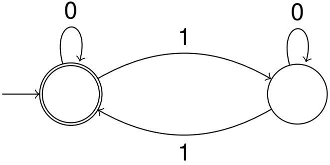


Figure: A cyclic finite state automaton for *an even number of*.

Characterizing First-Order Definable GQs

Theorem (van Benthem 1986, p.156-157)

A quantifier Q is first-order definable iff $\mathcal{L}_Q$ can be recognized by a permutation-invariant acyclic finite state automaton.

Recall from last time:

Lemma (Ehrenfeucht-Fraïssé)

Q is FO-definable iff $\exists n$ s.t.

$$\mathcal{M} \equiv_n \mathcal{M}' \Rightarrow \mathcal{M} \in Q \text{ iff } \mathcal{M}' \in Q$$

where $\equiv_n$ means Dup has n-round winning strategy in E-F game.

In our case:

$$A \sim_n B \Leftrightarrow |A| = |B| = l < n \text{ or } |A|, |B| \geq n$$

and $\mathcal{M} \equiv_n \mathcal{M}'$ iff $A \cap B, A \setminus B, B \setminus A, M \setminus (B \cup A)$ all bear $\sim_n$ to their primed counterparts.

Characterizing First-Order Definable GQs

Theorem (van Benthem 1986, p.156-157)

A quantifier Q is first-order definable iff $\mathcal{L}_Q$ can be recognized by a permutation-invariant acyclic finite state automaton.

Recall from last time:

Lemma (Ehrenfeucht-Fraïssé)

Q is FO-definable iff $\exists n$ s.t.

$$\mathcal{M} \equiv_n \mathcal{M}' \Rightarrow \mathcal{M} \in Q \text{ iff } \mathcal{M}' \in Q$$

where $\equiv_n$ means Dup has n-round winning strategy in E-F game.

In our case:

$$A \sim_n B \Leftrightarrow |A| = |B| = l < n \text{ or } |A|, |B| \geq n$$

and $\mathcal{M} \equiv_n \mathcal{M}'$ iff $A \cap B, A \setminus B, B \setminus A, M \setminus (B \cup A)$ all bear $\sim_n$ to their primed counterparts.

Interpret E-F "threshold" in "Tree of Numbers"

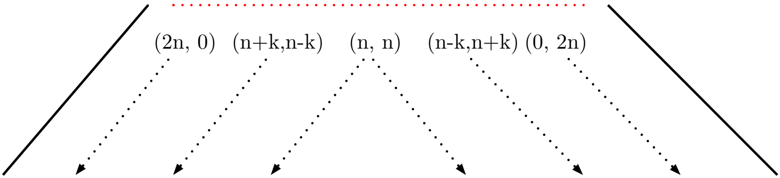


Figure: Fraïssé Threshold at level $2n$

Homework: transform the above directly into an acyclic finite automaton.

Characterizing GQs with Regular Languages

The type $\langle 1 \rangle$ *divisibility quantifier* D_n is defined:

$$\langle M, A \rangle \in D_n \quad \text{iff} \quad |A| \text{ is divisible by } n.$$

Theorem (Mostowski 1991)

Finite state automata accept exactly the class of quantifiers of type $\langle 1, \ldots, 1 \rangle$ *definable in first-order logic augmented with* D_n *for all* n.

Beyond Regular Languages

Lemma (Pumping Lemma)

Let L be a regular language. Then $\exists p \geq 1$ s.t. every $w \in L$ with $len(w) \geq p$, there are x, y, z s.t. $w = xyz$ and

(1) $len(y) \geq 1$

(2) $len(xy) \leq p$

(3) $xy^i z \in L$ *for all* $i \geq 0$

Corollary

L_{most} *is not regular.*

Proof.

Suppose otherwise, and let p be the 'pumping length'. Consider the word $w = 0^p 1^{p+1} \in L_{\text{most}}$. By assumption, $w = xyz$ with $len(xy) \leq p$ and $len(y) \geq 1$. Thus, xy contains only 0s. Then $w = xy^2 z$ would have to be in L_{most}, but it is not. □

Beyond Regular Languages

Lemma (Pumping Lemma)

Let L be a regular language. Then $\exists p \geq 1$ s.t. every $w \in L$ with $len(w) \geq p$, there are x, y, z s.t. $w = xyz$ and

(1) $len(y) \geq 1$

(2) $len(xy) \leq p$

(3) $xy^i z \in L$ for all $i \geq 0$

Corollary

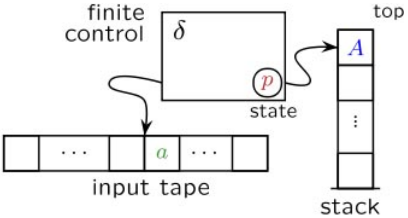
L_{most} is not regular.

Proof.

Suppose otherwise, and let p be the 'pumping length'. Consider the word $w = 0^p 1^{p+1} \in L_{\text{most}}$. By assumption, $w = xyz$ with $len(xy) \leq p$ and $len(y) \geq 1$. Thus, xy contains only 0s. Then $w = xy^2 z$ would have to be in L_{most}, but it is not. $\square$

Outline

Pushdown Automata



Essentially: augment a DFA with a last-in/first-out stack

Push down automata

Definition

A non-deterministic push-down automaton (PDA) is a tuple $(A, \Gamma, \#, Q, q_s, F, \delta)$, where:

- A is an input alphabet;
- Γ is a stack alphabet;
- $\# \not\in \Gamma$ is a stack initial symbol, empty stack consists only of it;
- Q is a finite set of states;
- $q_s \in Q$ is an initial state;
- $F \subseteq Q$ is a set of accepting states;
- $\delta : Q \times (A \cup \{\varepsilon\}) \times \Gamma \longrightarrow \mathcal{P}(Q \times \Gamma^*)$ is a transition function.

Context-free languages

Definition

We say that a language $L \subseteq A^*$ is context-free if and only if there is a PDA H such that $L = L(H)$.

The context-free languages are a strict superset of the regular languages: There is a PDA for $L_{ab} = \{a^n b^n : n \geq 1\}$, though L_{ab} is not regular (exercise!).

PDA for Proportional Quantifiers

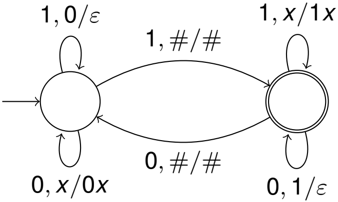


Figure: A PDA computing L_{most}.

Characterizing the PDA Quantifiers

Definition

A quantifier Q is *first-order additively definable* if there is a formula φ in the first-order language with equality and an addition symbol $\bar{+}$ such that

$$Q^c ab \quad \Leftrightarrow \quad \langle \mathbb{N}, +, a, b \rangle \models \varphi(a, b)$$

Theorem (van Benthem 1986, p.163-165)

$\mathcal{L}_Q$ is computable by a pushdown automaton if and only if Q is first-order additively definable.

Aside: Deterministic Context-Free Languages

Kanazawa 2013 has characterized the monadic quantifiers accepted by *deterministic pushdown automata*. (Unlike the finite-state case, nondeterministic PDAs are strictly more powerful than the deterministic counterparts.)

The characterization essentially blocks what we might call ‘multiply proportional quantifiers’, such as:

- Between two-fifths and seven-eights of all teenagers watch 10 hours of TV a week.

So: the DPDA quantifiers are not closed under Boolean operations. It turns out that the PDA quantifiers are the closure of the DPDA ones under Boolean operations.

Aside: Deterministic Context-Free Languages

Kanazawa 2013 has characterized the monadic quantifiers accepted by *deterministic pushdown automata*. (Unlike the finite-state case, nondeterministic PDAs are strictly more powerful than the deterministic counterparts.)

The characterization essentially blocks what we might call 'multiply proportional quantifiers', such as:

- Between two-fifths and seven-eights of all teenagers watch 10 hours of TV a week.

So: the DPDA quantifiers are not closed under Boolean operations. It turns out that the PDA quantifiers are the closure of the DPDA ones under Boolean operations.

Beyond context-free languages

The language

$$L_{abc} = \{a^k b^k c^k : k \geq 1\}$$

is not context-free. More generally, neither is:

$$L_3 = \{w \in \{a, b, c\}^* \mid \#_a(w) = \#_b(w) = \#_c(w)\}$$

Question: does this language correspond to a GQ realized in natural language? Clark n.d. and van Benthem 1987 suggest the type $\langle 1, 1, 1, 1 \rangle$ *equal number of*, as in:

- An equal number of undergraduates, graduate students, and faculty members live at Stanford.

Beyond context-free languages

The language

$$L_{abc} = \{a^k b^k c^k : k \geq 1\}$$

is not context-free. More generally, neither is:

$$L_3 = \{w \in \{a, b, c\}^* \mid \#_a(w) = \#_b(w) = \#_c(w)\}$$

Question: does this language correspond to a GQ realized in natural language? Clark n.d. and van Benthem 1987 suggest the type $\langle 1, 1, 1, 1 \rangle$ *equal number of*, as in:

- An equal number of undergraduates, graduate students, and faculty members live at Stanford.

Beyond context-free languages

The language

$$L_{abc} = \{a^k b^k c^k : k \geq 1\}$$

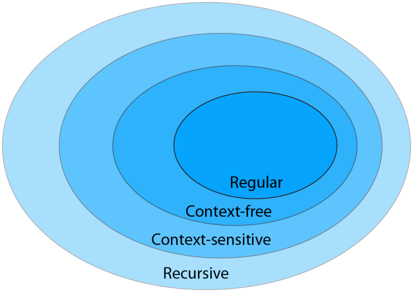
is not context-free. More generally, neither is:

$$L_3 = \{w \in \{a, b, c\}^* \mid \#_a(w) = \#_b(w) = \#_c(w)\}$$

Question: does this language correspond to a GQ realized in natural language? Clark n.d. and van Benthem 1987 suggest the type $\langle 1, 1, 1, 1 \rangle$ *equal number of*, as in:

- An equal number of undergraduates, graduate students, and faculty members live at Stanford.

Chomsky's Hierarchy



How psychologically real are those distinctions? See Szymanik 2016.