5th Week

Structural Properties by Finite Automata

Synopsis.

- Deterministic/Randomized Advice
- Dissectability and Separation
- Immunity and Simplicity
- Swapping Lemmas

May 7, 2018. 23:59

Course Schedule: 16 Weeks

Subject to Change

- Week 1: Basic Computation Models
- Week 2: NP-Completeness, Probabilistic and Counting Complexity Classes
- Week 3: Space Complexity and the Linear Space Hypothesis
- Week 4: Relativizations and Hierarchies
- Week 5: Structural Properties by Finite Automata
- Week 6: Stype-2 Computability, Multi-Valued Functions, and State Complexity
- Week 7: Cryptographic Concepts for Finite Automata
- Week 8: Constraint Satisfaction Problems
- Week 9: Combinatorial Optimization Problems
- Week 10: Average-Case Complexity
- Week 11: Basics of Quantum Information
- Week 12: BQP, NQP, Quantum NP, and Quantum Finite Automata
- Week 13: Quantum State Complexity and Advice
- Week 14: Quantum Cryptographic Systems
- Week 15: Quantum Interactive Proofs
- Week 16: Final Evaluation Day (no lecture)

YouTube Videos

- This lecture series is based on numerous papers of T. Yamakami. He gave conference talks (in English) and invited talks (in English), some of which were video-recorded and uploaded to YouTube.
- Use the following keywords to find a playlist of those videos.
- YouTube search keywords:

  Tomoyuki Yamakami conference invited talk playlist

Conference talk video

Main References by T. Yamakami

✎ T. Yamakami and T. Suzuki. Resource bounded immunity and simplicity. Theor. Comput. Sci. 347(1-2), 90-129 (2005)

✎ T. Yamakami. Swapping lemmas for regular and context-free languages. Preprint, arXiv:0808.4122 (2008)

✎ K. Tadaki, T. Yamakami, and J. C. H. Lin. Theory of one-tape linear-time Turing machines. Theor. Comput. Sci. 411(1): 22-43 (2010)

✎ T. Yamakami. The roles of advice to one-tape linear-time Turing machines and finite automata. Int. J. Found. Comput. Sci. 21(6): 941-962 (2010)

✎ T. Yamakami. Immunity and pseudorandomness of context-free languages. Theor. Comput. Sci. 412(45): 6432-6450 (2011)

✎ T. Yamakami and Y. Kato. The dissecting power of regular languages. Inf. Process. Lett. 113(4): 116-122 (2013)

I. Roles of Advice for Finite Automata

1. Motivational Discussions
2. Advice to Finite Automata
3. Deterministic Advice
4. Advised Language Families
5. Power of Advice
6. Swapping Lemmas
7. Another Characterization of REG/n
8. separation Results
9. Relationships among Advised Classes

Motivational Discussion I

- Context-free languages are one of the most fundamental types of languages in formal language theory.
- How can we describe a "complicated" nature of languages?
- E.g., consider two similar languages:
  - $L_{eq} = \{ 0^n1^n \mid n \geq 0 \}$
  - Equal = $\{ w \in \{0,1\}^* \mid \#_0(w)=\#_1(w) \}$

$\#_b(w)$ = the number of b in w

- Both languages are in CFL but not in REG.
  - $L_{3eq} = \{ 0^n1^n2^n \mid n \in N \}$
  - 3Equal = $\{ w \in \{0,1,2\}^* \mid \#_0(w)=\#_1(w)=\#_2(w) \}$
- Both languages are in CFL(2) but not in CFL.

CFL(2) = $\{ B_1 \cap B_2 \mid B_1, B_2, \in CFL \}$

Motivational Discussion II

- Recall the languages from the previous slide.
  - L_{eq} = { 0^n1^n | $n \geq 0$ }
  - Equal = { $w \in \{0,1\}^*$ | $\#_0(w)=\#_1(w)$ }
  - L_{3eq} = { $0^n1^n2^n$ | $n \in N$ }
  - 3Equal = { $w \in \{0,1,2\}^*$ | $\#_0(w)=\#_1(w)=\#_2(w)$ }
- Question: How different are the above languages?
- Time-complexity is not suitable to use for automata.
- Thus, we need to look for structural differences of languages.

Model of Finite Automata (revisited)

- Firstly, let us recall a model of one-way (one-head) finite automata.

$M = (Q,\Sigma,\delta,q_0,F)$

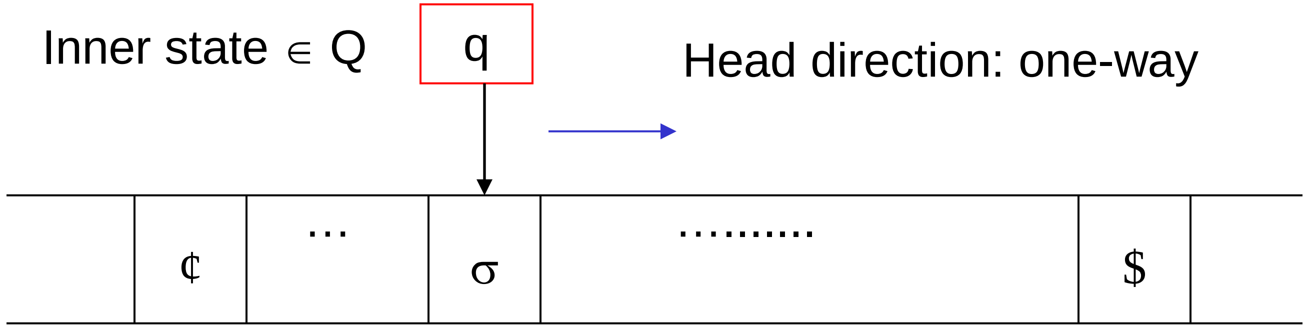
L(M) = set of strings accepted by M

Q = set of inner states
Σ = input alphabet
δ : transition function
q_0 : initial state
F = set of final (accepting) states

CPU

Inner state $\in$ Q

Head direction: one-way



End-marker An infinite read-only tape End-marker

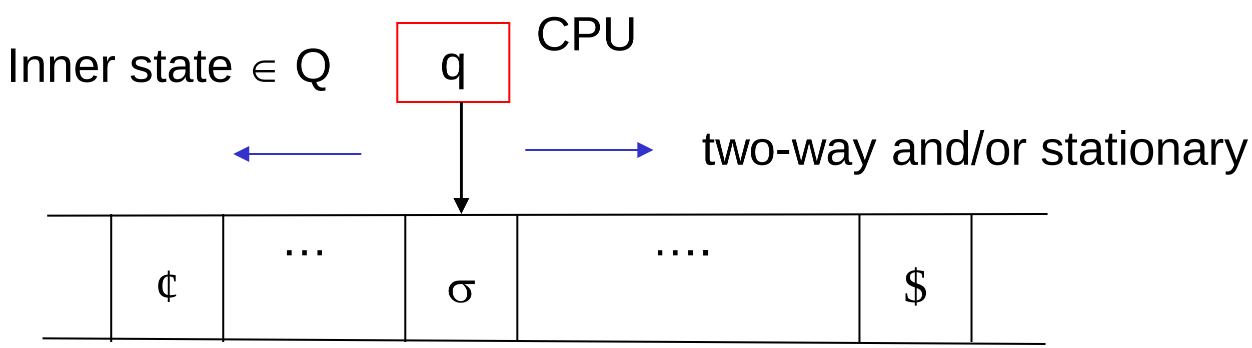
Model of 1-Tape Turing Machines (revisited)

- Secondly, let us recall a model of one-way (one-head) nondeterministic Turing machine.

$M = (Q,\Sigma,\Gamma,\delta,q_0,Q_{acc},Q_{rej})$

L(M) = set of strings accepted by M

Q,q_0 are the same
Σ = input alphabet
Γ = tape alphabet
$Q_{acc} \cup Q_{rej}$: halting states
δ : transition function



An infinite input/work tape

(¢ and $ are removable)

1-Tape Linear-Time Complexity Classes

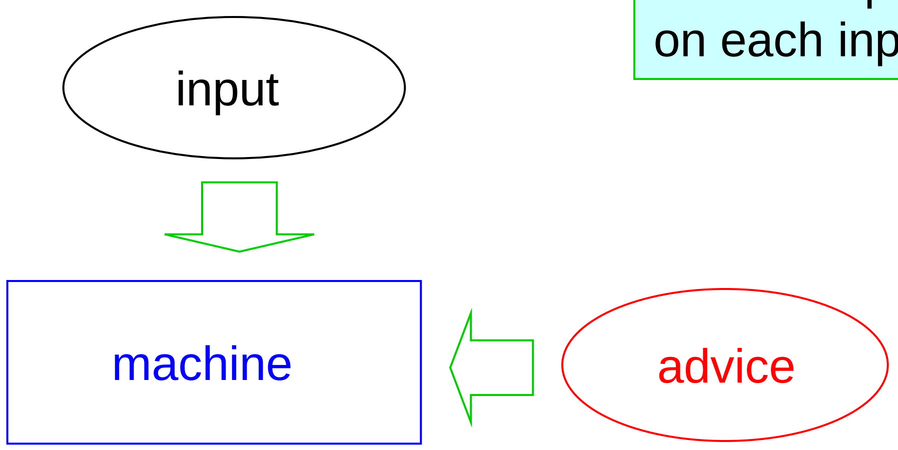
- In Week 1, we have defined the following notations.

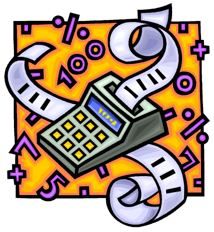
- Machine
  - 1DTM = 1-tape deterministic Turing machine
- Complexity Class
  - 1-DLIN = class of all languages that are recognized by 1DTMs in linear time
- Function Class
  - 1-FLIN = class of all functions that are computed in linear time by 1DTMs with no extra output tape

Advice of Karp and Lipton

- Advice is an external source of information.
- Advice is a way to enhance a computational power of an underlying machine.
- We use advice of the style of Karp and Lipton (1990).

Advice depends only on each input size.





Advice to Finite Automata

- In Week 3, we have already discussed the advice notion of Karp and Lipton (1990) for Turing machines.

- Damm and Holzer (1995) considered a similar advice notion, which is applied to finite automata.
- They provided advice strings next to standard input strings.

- Tadaki, Yamakami, and Lin (2004) took a slightly different way to provide advice to finite automata.
- Here, advice strings are given in parallel to input strings.

Track Notation for Advice I

- We use a track notation of [Tadaki-Yamakami-Lin (2004)].
- In these slides, we also write a track notation as [x y]^T.

$$\begin{bmatrix} x \\ w \end{bmatrix} = \begin{bmatrix} x_1 \\ w_1 \end{bmatrix}\begin{bmatrix} x_2 \\ w_2 \end{bmatrix}\cdots\begin{bmatrix} x_i \\ w_i \end{bmatrix}\cdots\begin{bmatrix} x_n \\ w_n \end{bmatrix} \quad \text{if} \quad \begin{cases} x = x_1 x_2 \cdots x_i \cdots x_n \\ w = w_1 w_2 \cdots w_i \cdots w_n \end{cases}$$

Each of them is treated as a new symbol.

x_i
w_i

new symbol

When written on an input tape:

Upper track	¢	⋯⋯	x_i	⋯⋯	$
Lower track		⋯⋯	w_i	⋯⋯	

Track Notation for Advice II

- When $|x| \neq |w|$, we pad extra #'s automatically.

- When $|x|<|w|$, the notation $[x\ w]^T$ means:

$$\begin{bmatrix} x \\ w \end{bmatrix} = \begin{bmatrix} x_1 \\ w_1 \end{bmatrix}\begin{bmatrix} x_2 \\ w_2 \end{bmatrix}\cdots\begin{bmatrix} x_i \\ w_i \end{bmatrix}\begin{bmatrix} \# \\ w_{i+1} \end{bmatrix}\cdots\begin{bmatrix} \# \\ w_n \end{bmatrix}$$

<table>
<tr><td rowspan="2">¢</td><td>x</td><td># ··· #</td><td rowspan="2">$</td></tr>
<tr><td colspan="2">w</td></tr>
</table>

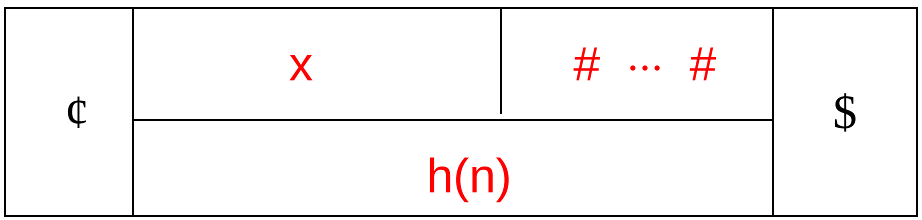
- When $|x|>|w|$, the notation $[x\ w]^T$ means:

$$\begin{bmatrix} x \\ w \end{bmatrix} = \begin{bmatrix} x_1 \\ w_1 \end{bmatrix}\begin{bmatrix} x_2 \\ w_2 \end{bmatrix}\cdots\begin{bmatrix} x_i \\ w_i \end{bmatrix}\begin{bmatrix} x_{i+1} \\ \# \end{bmatrix}\cdots\begin{bmatrix} x_n \\ \# \end{bmatrix}$$

<table>
<tr><td rowspan="2">¢</td><td colspan="2">x</td><td rowspan="2">$</td></tr>
<tr><td>w</td><td># ··· #</td></tr>
</table>

Standard (Deterministic) Advice

- Input string $x \in \Sigma^n$ over an input alphabet Σ
- Advice alphabet Γ
- Advice function $h: N \rightarrow \Gamma^*$



Advice string h(n) is given in the lower track of the tape when |x|<|h(n)|.

- NOTE: This scheme of providing advice strings is computationally equivalent to Karp-Lipton's original one for, say, polynomial time-bounded computation.

Examples of Advice

- We present a few examples of how to provide advice strings in parallel to input strings.

Σ = { 0, 1 } (input alphabet) Γ = { a, b } (advice alphabet)

Upper track	¢	0 1 0 0 1 0 0 0 1	$
Lower track		a b b a b a a b b	

Σ = { a, b, c } (input alphabet) Γ = { a, b } (advice alphabet)

Upper track	¢	c b a a c c a a c	$
Lower track		a b b a b a a # #	

Advised Language Families

- Deterministic computation with standard advice
- Let L be any language over an alphabet Σ.
- $L \in$ 1-DLIN/lin

  $\Leftrightarrow \exists M$:linear-time 1DTM $\exists \Gamma$:advice alphabet $\exists h:N \to \Gamma^*$

  1. $\forall n \in N$ [$|h(n)| = O(n)$].
  2. $\forall x \in \Sigma^n$ [$x \in L \leftrightarrow$ M accepts $[x\ h(|x|)]^T$].
- $L \in$ REG/n

  $\Leftrightarrow \exists M$:1dfa $\exists \Gamma$:advice alphabet $\exists h:N \to \Gamma^*$

  1. $\forall n \in N$ [$|h(n)| = n$].
  2. $\forall x \in \Sigma^n$ [$x \in L \leftrightarrow$ M accepts $[x\ h(|x|)]^T$].
- 1-$C_=$LIN/lin, 1-PLIN/lin, and CFL/n are similarly defined from 1-$C_=$LIN, 1-PLIN, and CFL, respectively.

Power of Advice

- Consider the context-free language:

  Center = { u1v | |u|=|v|, u,v$\in${0,1}*}.
- Fact: Center$\notin$REG.
- However, we can claim that Center$\in$REG/n.

- Let our advice function h be

$$h(n) = \begin{cases} 0^m 1 0^m & \text{if } n = 2m+1 \\ \#^{2m} & \text{if } n = 2m \end{cases}$$

Input string	x
Advice string	h(n)

Γ = { 0, 1, # }

- Let our 1dfa be s.t.

accepts x iff $[1\ 1]^T$ exists.

x	u	1	v
h(n)	0^m	1	0^m

x	u v
h(n)	$\#^m$ $\#^m$

Non-Advice Case vs. Advice Case

- For instance, we want to show:

  $\mathrm{Dup} = \{ xx \mid x \in \{0,1\}^* \} \notin \mathrm{REG/n}$.

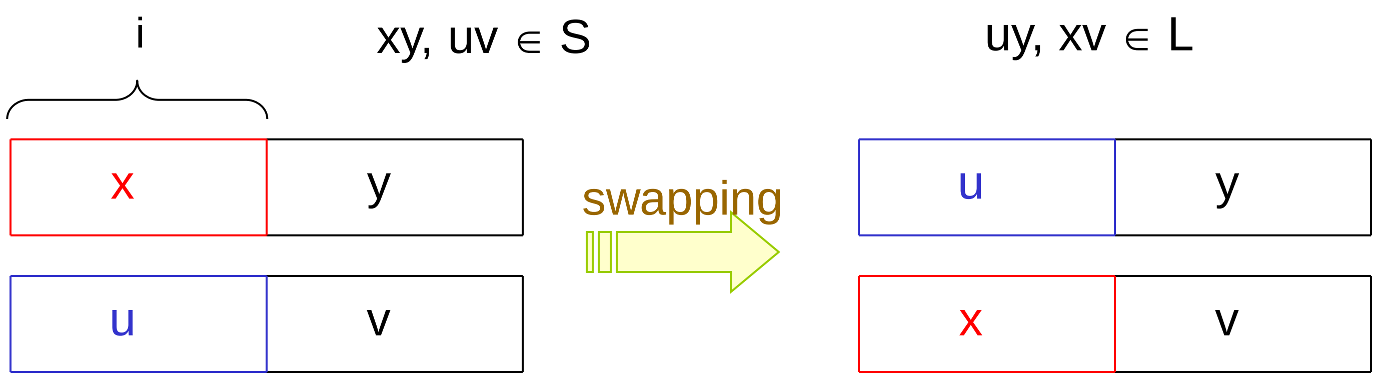
- Proof: Assume that $\mathrm{Dup} \in \mathrm{REG/n}$. That is, $\exists$ M:1dfa $\exists h:\mathbb{N}\rightarrow\Gamma^*$ s.t. $\mathrm{Dup} = \{ z \mid M \text{ accepts } [z\ h(|z|)]^T \}$.
- Let us apply the pumping lemma for REGs. Choose a long string $w = [xx\ h(|xx|)]^T$ and consider its decomposition $w = uyv$ s.t. $\forall i$ [M accepts uy^iv].
- However, this uy^iv may be no longer of the form $[z\ h(|z|)]^T$. So, we cannot get any contradiction!
- Therefore, we need another type of useful lemma for regular languages!
- That is the so-called swapping lemma for REGs.

Swapping Lemma for Regular Languages

- One of the useful properties of regular languages is a so-called swapping lemma, shown by Yamakami (2008,2010).

Swapping Lemma for REGs [Yamakami (2008,2010)]

If L is regular, then $\exists m>0$ s.t. $\forall n\in N\ \forall S\subseteq L\cap\Sigma^n\ (|S|\geq m)$ $\forall i\in[n]\ \exists xy,uv\in S\ (|x|=|u|=i)$ [$xy\neq uv$ & $uy,xv\in L$].



(*) T. Yamakami. arXiv:0808.4122 (2008) & IJFCS 21 (2010)

How to Use the Swapping Lemma for REG?

- How can we use the swapping lemma?
- (Claim) Dup $\notin$ REG/n.

❑ Proof Sketch:

- Assume Dup$\in$REG/n. That is, $\exists$ M:1dfa $\exists$h:N$\to\Gamma^*$ s.t. Dup = { z | M accepts $[z\ h(|z|)]^T$ }. Let L = { $[x\ h(|x|)]^T$ | x Dup }. Choose n'=2n and i=n.
- Let S = { $[z\ h(|z|)]^T$ | |z|=2n, M accepts $[z\ h(|z|)]^T$ } $\subseteq$ L.
- By the swapping lemma, there are two different strings xy = $[aa\ h(2n)]^T$ and uv = $[bb\ h(2n)]^T$ in S with |x|=|u|=n s.t. M accepts xv and uy.
- We then obtain xv = $[ab\ h(2n)]^T$ and uy = $[ba\ h(2n)]^T$.
- Since a$\neq$b, this is impossible! Hence, Dup$\notin$REG/n.

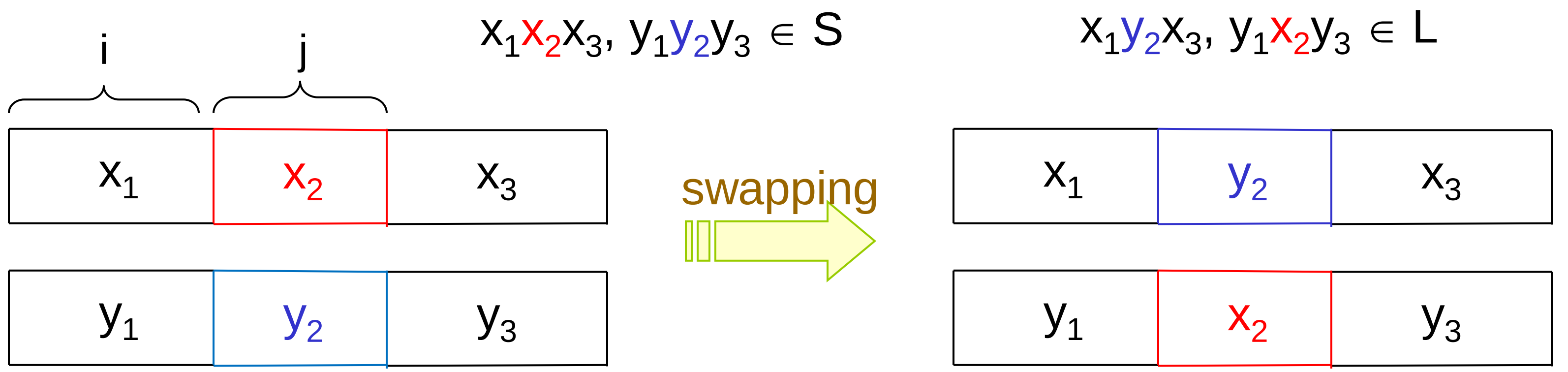
QED

Swapping Lemma for Context-Free Languages

Swapping Lemma for CFLs [Yamakami (2008,2016)]

If L is context-free, then $\exists m>0$ s.t.

$\forall n\geq 2\ \forall S\subseteq L\cap\Sigma^n\ \forall j_0,k_0 \in[2,n-1]_Z(k_0\geq 2j_0)\ \forall i\in[0,n]$
$\forall j\in[j_0,k_0](i+j\leq n)\forall u\in\Sigma^{j0}\ (\ |S_{i,u}|<|S|/m(k_0-j_0+1)(n-j_0+1)\ )$
$\exists x=x_1x_2x_3,y=y_1y_2y_3\in S\ (\ |x_1|=|y_1|=i\ )(\ |x_2|=|y_2|=j\ )(\ |x_3|=|y_3|\ )$
$[\ x_2\neq y_2 \,\&\, x_1y_2x_3,y_1x_2y_3\in L\ ].$



(*) T. Yamakami. arXiv:0808.4122 (2008) & TCS 613 (2016).

Equivalence Classes

- A (binary) relation R is a subset of a Cartesian product of two sets A and B (i.e., $R \subseteq A \times B$).
- For a set X, a relation on X is a subset of $X \times X$.
- An equivalence relation $\sim$ on X is a (binary) relation satisfying the following three conditions:
  1. (reflexivity) $x \sim x$ for any x.
  2. (symmetry) $x \sim y$ implies $y \sim x$ for any x,y.
  3. (transitivity) $x \sim y$ and $y \sim z$ imply $x \sim z$ for any x,y,z.
- The equivalence class of x is $[x] = \{ y \mid x \sim y \}$.
- $X/\sim$ is the set of all equivalence classes w.r.t. $\sim$;

  i.e., $X/\sim = \{ [x] \mid x \in X \}$.

Another Characterization of REG/n

- The characteristic function of a language S is

  $S(x) = 1$ if $x \in S$, and $S(x) = 0$ if $x \notin S$.

- Theorem: [Yamakami (2010)]

  For any language S over alphabet Σ, the following two statements are equivalent. Let $\Delta = \{(x,n) |, x \in \Sigma^*, n \in N, |x| \leq n\}$.

  1. S is in REG/n.
  2. $\exists \equiv$: equivalence relation on Δ s.t.
     a) $|\Delta/\equiv|$ is finite.
     b) $\forall n \in N\ \forall x,y \in \Sigma^*\ (|x|=|y| \leq n)$

        $(x,n) \equiv (y,n) \leftrightarrow \forall z \in \Sigma^*\ [|xz|=n \rightarrow S(xz) = S(yz)]$.

- NOTE: The swapping lemma follows from this theorem.

Separation Results I

- We will show two separation results.

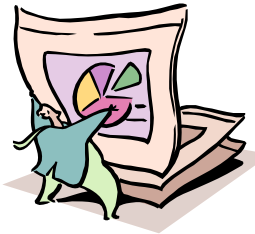
- Proposition: [Yamakami (2010)]
  $\text{1-C}_=\text{LIN} \not\subset \text{CFL/n}$.

❑ Proof Sketch:

- Let $\Sigma_6=\{a_1,a_2,\ldots,a_6,\#\}$ and consider the language
  $\text{Equal}_6 = \{ w\in\Sigma_6^* \mid \#_a(w)=\#_b(w) \text{ for } \forall a,b\in\Sigma_6 \}$.
- It is known that $\text{Equal}_6 \notin \text{CFL/n}$ by the swapping lemma [Yamakami (2008)].
- It is easy to show that $\text{Equal}_6 \in \text{1-C}_=\text{LIN}$.

QED

Separation Results II



- Theorem: [Yamakami (2010)]

  CFL $\not\subseteq$ 1-PLIN/lin.

❑ Proof Sketch:

- Let IP*={ xy | x,y$\in${0,1}*,|x|=|y|,$x^R \bullet y \equiv 1 \pmod 2$ }, where $x \bullet y$ is the (bitwise) binary inner product.
- It is known that IP*$\in$CFL.
- We exploit a certain special property of 1-PLIN/lin to show that IP*$\notin$1-PLIN/lin.

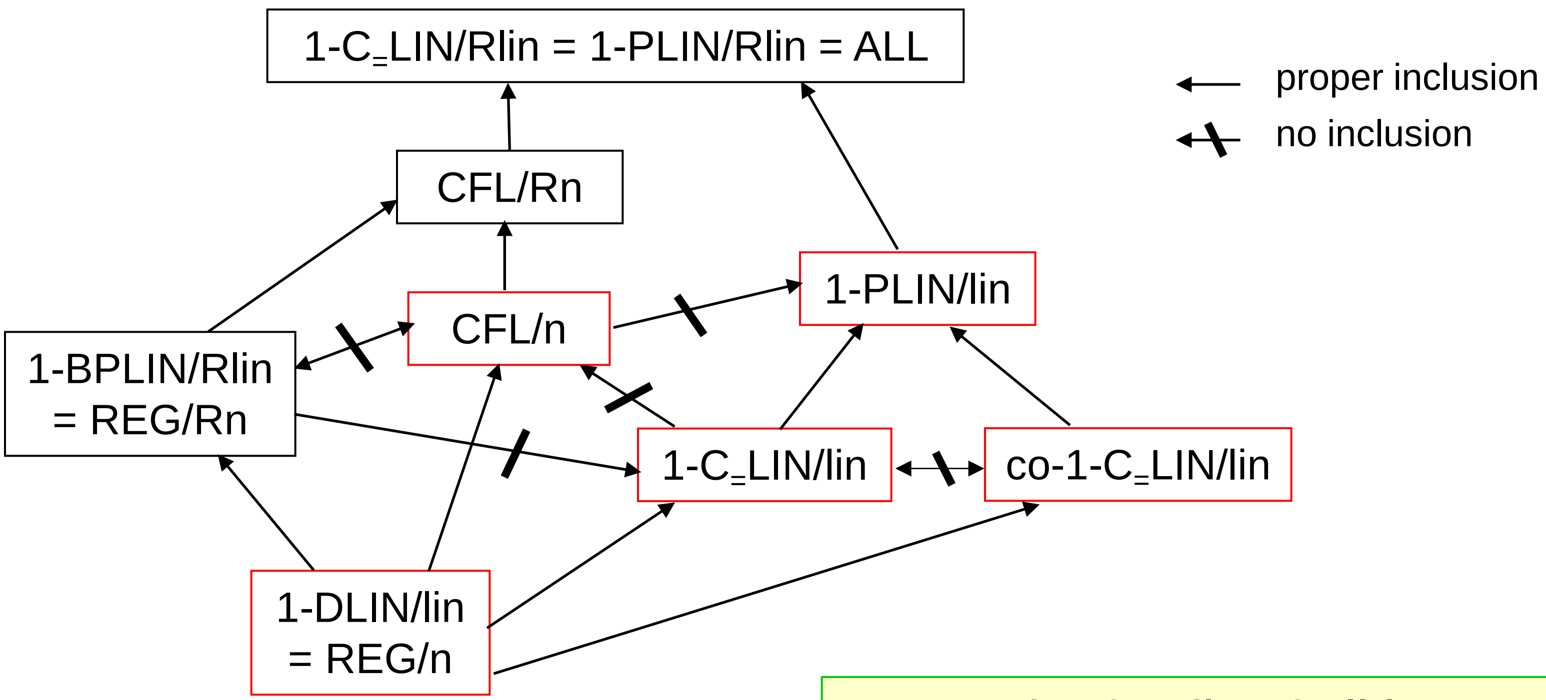
QED

- We can prove the following as well.
- Theorem: [Yamakami (2010)]

  1-$C_=$LIN/lin $\neq$ co-1-$C_=$LIN/lin $\neq$ 1-PLIN/lin.

Relationships among Advised Classes

- We summarize known class separations and collapses among advised language families.



ALL = the family of all languages

II. Randomized Advice

1. What is Randomized Advice?
2. Notation for Random variables
3. Advised Language Families
4. Power of 1-$C_{=}$LIN/Rlin and 1-PLIN/Rlin
5. Why 1-$C_{=}$LIN/Rlin = ALL?
6. Power of REG/Rn
7. Limitation of REG/Rn

What is Randomized Advice?

- In randomized advice, all advice strings are chosen at random according to a probability distribution.
- Let Γ be an advice alphabet.
- For each n, an advice probability distribution D_n over $\Gamma^{t(n)}$ generates advice strings $y \in \Gamma^{t(n)}$ with probability $D_n(y)$, where t(n) is a length function.

- Input string $x \in \Sigma^n$

D_n generates y with probability $D_n(y)$.

<table>
<tr><td rowspan="2">¢</td><td>x</td><td># ··· #</td><td rowspan="2">$</td></tr>
<tr><td colspan="2">y</td></tr>
</table>

Advice string y is given in the lower track of the tape in the case of $|x|<t(n)$.

Notation for Random Variables

- Given a probability distribution D_n over $\Gamma^{t(n)}$, we use the following succinct notation.
- The notation $[x \;\; D_n]^T$ denotes a random variable of the form $[x\ y]^T$ (where “T” indicates transpose) over all strings y in $\Gamma^{t(n)}$ according to D_n.

$$\begin{bmatrix} x \\ D_n \end{bmatrix}: \text{ random variable over } \Gamma^{t(n)} \text{ when } |x| = n$$

- In other words, we randomly pick y with prob. $D_n(y)$ and write it onto the input tape along with x to form $[x\ y]^T$.
- NOTE: we should add extra #s as before when $|y| \neq |x|$.

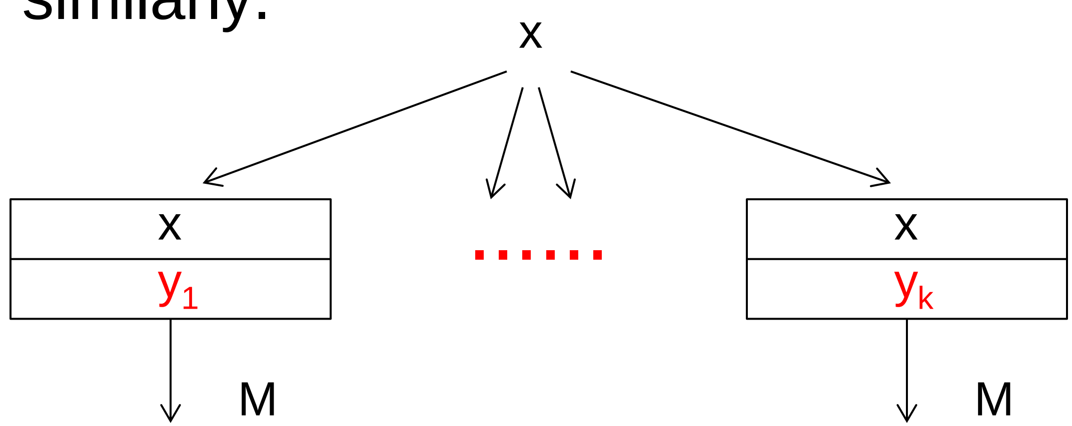
Advised Language Families I

- Let L be any language over an alphabet Σ.
- $L \in$ REG/Rn

  $\Leftrightarrow \exists M$:1dfa $\exists \varepsilon \in [0,½)$ $\exists \Gamma$ $\exists \{D_n\}_{n \in N}$:advice prob. dist.

  1. $\forall n \in N$ [D_n generates advice strings $y \in \Gamma^n$].
  2. $\forall x \in \Sigma^n$ [$x \in L \rightarrow$ M accepts $[x\ D_n]^T$ with probability $\geq 1-\varepsilon$].
  3. $\forall x \in \Sigma^n$ [$x \in L \rightarrow$ M rejects $[x\ D_n]^T$ with probability $\geq 1-\varepsilon$].
- CFL/Rn is defined similarly.

D_n generates $y_1, \ldots, y_k$



Advised Language Families II

- We also provide randomized advice to 1-tape linear-time complexity classes.
- Let L be any language over an alphabet Σ.
- $L \in$ 1-BPLIN/Rlin

  $\Leftrightarrow \exists M$:linear-time 1PTM $\exists \varepsilon \in [0,½)\ \exists \Gamma\ \exists \{D_n\}_{n \in N}$: dist.

  1. $\forall n \in N$ [D_n generates advice strings $y \in \Gamma^{O(n)}$].
  2. $\forall x \in \Sigma^n$ [$x \in L \rightarrow$ M accepts $[x\ D_n]^T$ with prob. $\geq 1-\varepsilon$].
  3. $\forall x \in \Sigma^n$ [$x \in L \rightarrow$ M rejects $[x\ D_n]^T$ with prob. $\geq 1-\varepsilon$].

- 1-$C_=$LIN/Rlin and 1-PLIN/Rlin are defined similarly by supplementing randomized advice to underlying machines associated with 1-$C_=$LIN and 1-PLIN.

Example

- Consider a language: Dup = { xx | x∈{ 0,1 }* }.
- (Claim) Dup ∉ CFL.
- (Claim) Dup ∈ REG/Rn.

❑ Proof Sketch:

- Let our randomized advice D_n be s.t.

$$D_n(w) = \begin{cases} 1/2^m & \text{if } n = 2m \text{ and } w = yy \\ 1 & \text{if } n = 2m+1 \text{ and } w = \#^n \\ 0 & \text{otherwise.} \end{cases}$$

- 1dfa works as:
  1. Compute x•y and z•y.
  2. Accept xz if x•y $\equiv_2$ z•y.

w	x	z
D_n	y	y

- We run this procedure twice independently to reduce the error probability to ¼.

Power of 1-$C_=$LIN/Rlin and 1-PLIN/Rlin

- We show another result that shows the power of randomized advice when applied to 1-$C_=$LIN and 1-PLIN.

- Proposition: [Yamakami (2010)]
  1-$C_=$LIN/Rlin = 1-PLIN/Rlin = ALL.

- In other words, the advised language family 1-$C_=$LIN/Rlin (as well as 1-PLIN/Rlin) consists of all possible languages.

- In the next slide, we will give a proof sketch.

Why 1-$C_{=}$LIN/Rlin = ALL?

❑ Proof Sketch:

- Let L be any language over Σ. For simplicity, assume $L \cap \Sigma^n \neq \Sigma^n$. Let our randomized advice D_n be

$$D_n(y) = \begin{cases} \dfrac{1}{\left|\Sigma^n - L\right|} & \text{if } y \in \Sigma^n - L, \\ 0 & \text{if } y \in L \cap \Sigma^n. \end{cases}$$

Input string	x
D_n generates	y

This means that M accepts $[x\ D_n]^T$ probabilistically.

- Let our 1PTM M work as:

$$\begin{cases} \text{if } x=y\text{, then reject } x\text{; and} \\ \text{if } x\neq y\text{, then accept/reject with equal probability } 1/2. \end{cases}$$

- It is easy to check that $x \in L \leftrightarrow \text{Prob}[\ M([x\ D_n]^T) = 1\ ] = 1/2$.
- We conclude that $L \in$ 1-$C_{=}$LIN/Rlin.

QED

Power of REG/Rn

- Yamakami (2010) showed the following class separations with regard to REG/Rn.

- Lemma: 1-BPLIN/Rlin = REG/Rn.

- Proposition: DCFL$\cap$REG/Rn $\not\subset$ REG/n.

- Proposition: REG/Rn$\cap$1-C$_=$LIN/lin $\not\subset$ CFL/n.

- Theorem: REG/Rn $\not\subset$1C$_=$LIN/lin$\cup$co1-C$_=$LIN/lin.

Limitation of REG/Rn

- REG/Rn seems quite large but there is also a clear limitation in its recognition power.

- Theorem: [Yamakami (2010)]
  CFL$\not\subset$REG/Rn.

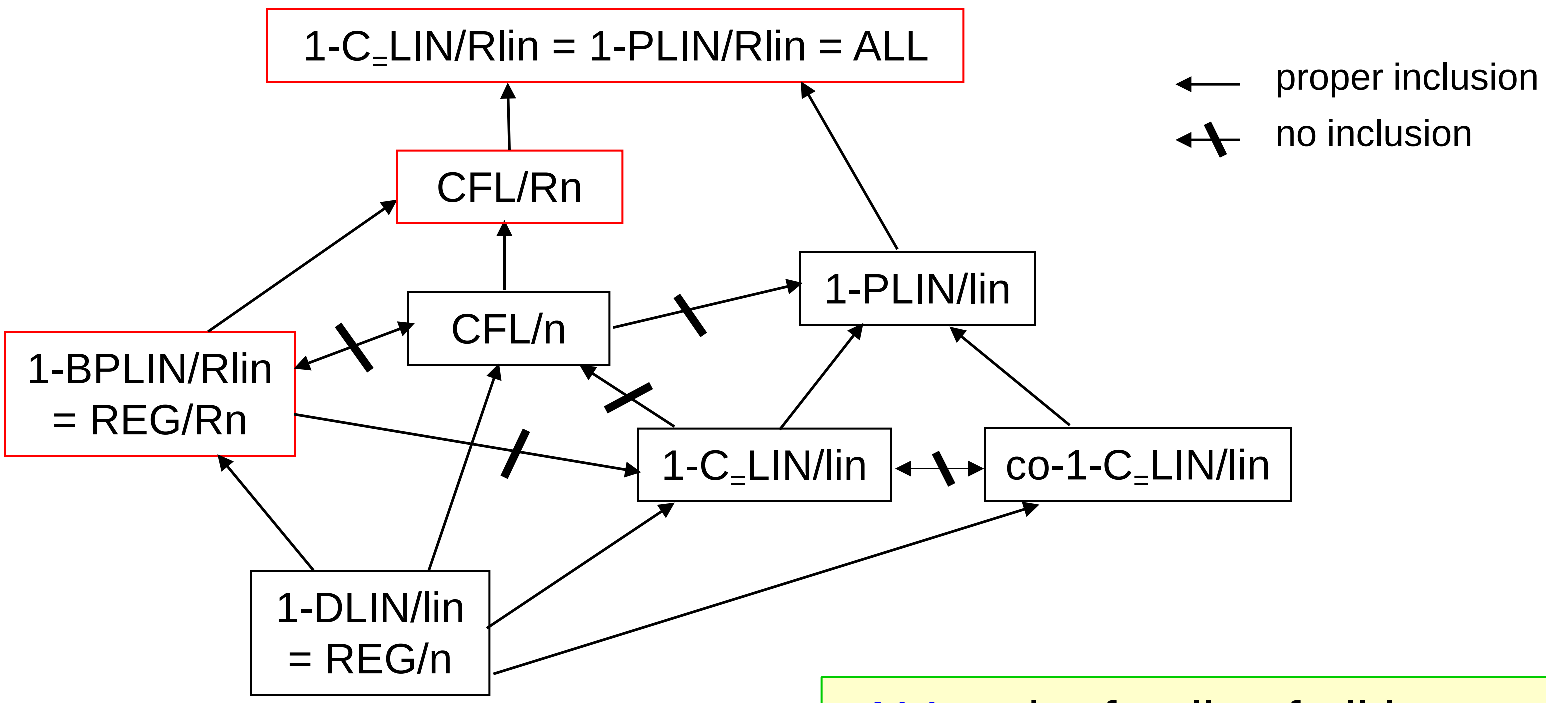
❑ Proof Idea:

- We use REG/n-pseudorandomness and average-case complexity class Aver-REG/n.
- The proof relies on the fact that, for any language L in REG/Rn, a distributional problem (A,μ) belongs to Aver-REG/n for any probability distribution μ.

QED

Relationships among Advised Classes (again)

- We summarize known class separations and collapses among advised language families.



ALL = the family of all languages

III. Dissectability

1. Structural Properties
2. “Infinite” Notation
3. CFL(k), CFL_k, and CFL_{BH}
4. A New Notion of Dissectability
5. P-Dissectability
6. Constantly Growing Languages
7. Non-REG-Dissectable Languages
8. Bounded Languages

Structural Properties

- We are interested in "structural" properties of languages.
- In the past literature, several structural properties have been discussed for regular and context-free languages.

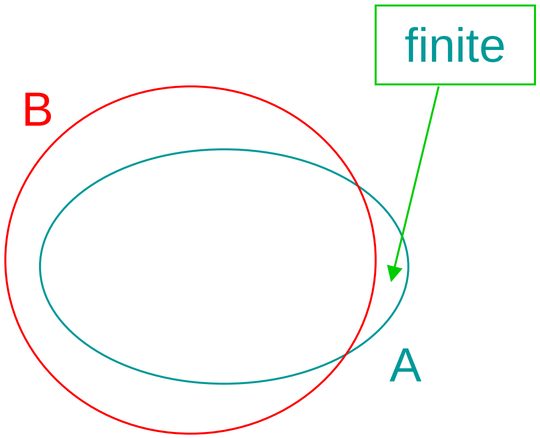
- Examples:
  i. Boolean closure properties [1960s]
  ii. Semi-linearity [Parikh (1961)]
  iii. Minimal cover [Domaratzki et al. (2002)]
  iv. Pseudorandomness [Yamakami (2011)]

  (We will discuss pseudorandomness in Week 6.)

“Infinite” Notations

- Our target is “formal languages,” which are countable sets.
- Here, we ignore “finite” portions of infinite sets.
- For this purpose, we want to simplify notations.
- A: countable set
  - $|A| < \infty \Leftrightarrow$ A is a finite set
  - $|A| = \infty \Leftrightarrow$ A is an infinite set
- A,B: infinite countable sets
  - $A \subseteq_{ae} B \Leftrightarrow |A - B| < \infty$
  - $A =_{ae} B \Leftrightarrow A \subseteq_{ae} B$ and $B \subseteq_{ae} A$

    $\Leftrightarrow |(A - B)\cup(B - A)| < \infty$



a.e. = almost everywhere

CFL(k), CFL_k, and CFL_{BH} (revisited)

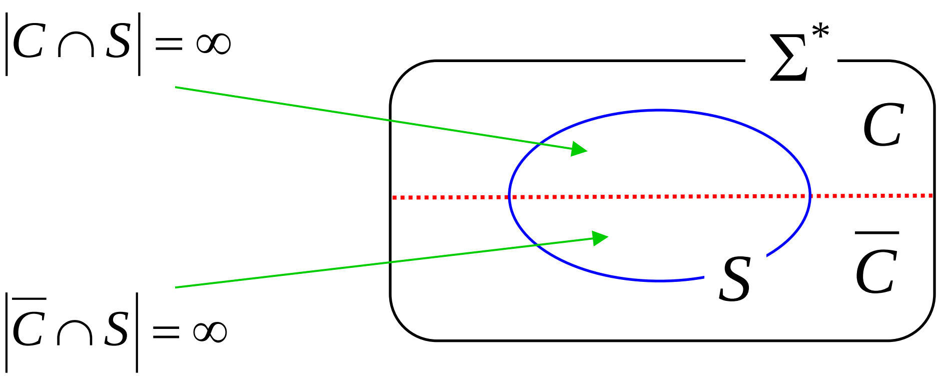
- We review several language families discussed in Week 4.
  - REG = set of all regular languages
  - CFL = set of all context-free languages
  - co-CFL = set of all complements of sets in CFL
  - CFL(k) = k-disjunctive closure, i.e.,
    - ✓ $CFL(k) = \{ L_1 \cap L_2 \cap \ldots \cap L_k \mid L_1, L_2, \ldots, L_k \in CFL \}$
  - CFL_k is defined inductively as follows:
    - ✓ $CFL_1 = CFL$
    - ✓ $CFL_{2k} = \{ A \cap B \mid A \in CFL_{2k-1}, B \in CFL \}$
    - ✓ $CFL_{2k+1} = \{ A \cup B \mid A \in CFL_{2k}, B \in CFL \}$
  - $CFL_{BH} = \cup_{k \geq 1} CFL_k$ (Boolean hierarchy over CFL)

    (In Week 4, CFL_{BH} is written as BHCFL.)

A New Notion of Dissectability

- Yamakami and Kato (2013) introduced a notion of "dissectability."
- "Dissecting" means that we can partition an infinite set into two infinite disjoint subsets.
- A language C is said to dissect an infinite language S if

$$|C \cap S| = |\overline{C} \cap S| = \infty$$



Σ = alphabet

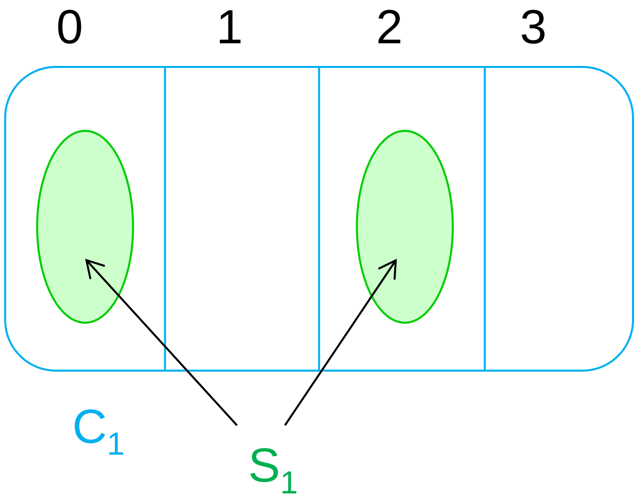
Quick Examples

- Recall that C dissects S if $\left|C \cap S\right| = \left|\overline{C} \cap S\right| = \infty$
- Let us see two simple examples.

1. Consider a non-regular language

   $S_1 = \{ a^n b^n \mid n \geq 0 \}.$

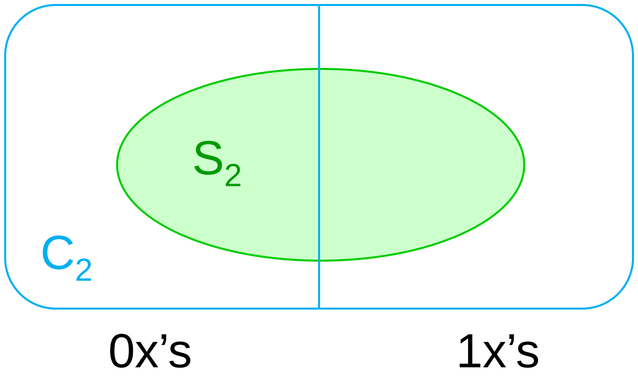
   The set $C_1 = \{ x \in \{ a,b \}^* \mid |x| \equiv 0 \pmod 4 \}$ dissects S_1.



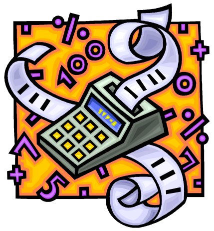
2. Consider a non-context-free language

   $S_2 = \{ ww \mid w \in \{ 0,1 \}^* \}.$

   The set $C_2 = \{ 0x \mid x \in \{ 0,1 \}^* \}$ dissects S_2.



Dissectability for Language Families



- Let $\mathbb{F}$ be an arbitrary family of languages.
- We define "$\mathbb{F}$-dissectability" as follows.

- An infinite language S is called $\mathbb{F}$-dissectable if there exists a language C in $\mathbb{F}$ that dissects S.
- A language family C is $\mathbb{F}$-dissectable if there exists an $\mathbb{F}$-dissectable language in C.

- The choice of $\mathbb{F}$ is quite important.
- Here, we are particularly interested in the case of $\mathbb{F}$ = REG (regular languages).
- In the following slide, we will explain why $\mathbb{F}$ = REG is a better choice, rather than, say, $\mathbb{F}$ = P.

P-Dissectability I

- Complexity class P may not be the best choice for $\mathbb{F}$.
- The following claim explains this statement.
- Theorem: [Yamakami-Kato (2013)]
  Every infinite recursive language is P-dissectable.

❑ Proof Sketch:

- Let L be any infinite language recognized in polynomial time by a DTM M.
- For simplicity, assume that $\Sigma = \{ 0,1 \}$.
- If $L =_{ae} \Sigma^*$, the language $C = \{ 0x \mid x \in \Sigma^* \}$ dissects L.
- Next, assume that $L \neq_{ae} \Sigma^*$.
- Let $z_0, z_1, z_2, \ldots$ be a standard lexicographic enumeration of all strings in $\Sigma^* = \{ \lambda, 0, 1, 00, 01, \ldots \}$.

P-Dissectability II

- For each string x, we determine whether $x \in C$ (or its Boolean value $C(x)$) by running the following procedure.

1. Initially, we set $A = R = \varnothing$ and $i = 0$.
2. At round i, we first recover the value $C(z_i)$ by running this entire procedure on the input z_i.
3. Next, simulate M on the input z_i within $|x|$ steps.
4. If $M(z_i) = 1$, then
   a) update A to $A \cup \{ i \}$ if $C(z_i) = 1$, and
   b) update R to $R \cup \{ i \}$ if $C(z_i) = 0$.
5. If not, then do nothing.
6. After round $|x|$, if $|A| > |R|$, then define the value $C(x) = 0$; otherwise, define $C(x) = 1$. Finish the procedure.
7. Increment i by 1 and go to Step 2.

P-Dissectability III

- The previous procedure takes only polynomial time in the length |x| of the input string x.
- By a simple diagonalization argument, we can show that

$$\left|C \cap L\right| = \left|\overline{C} \cap L\right| = \infty$$

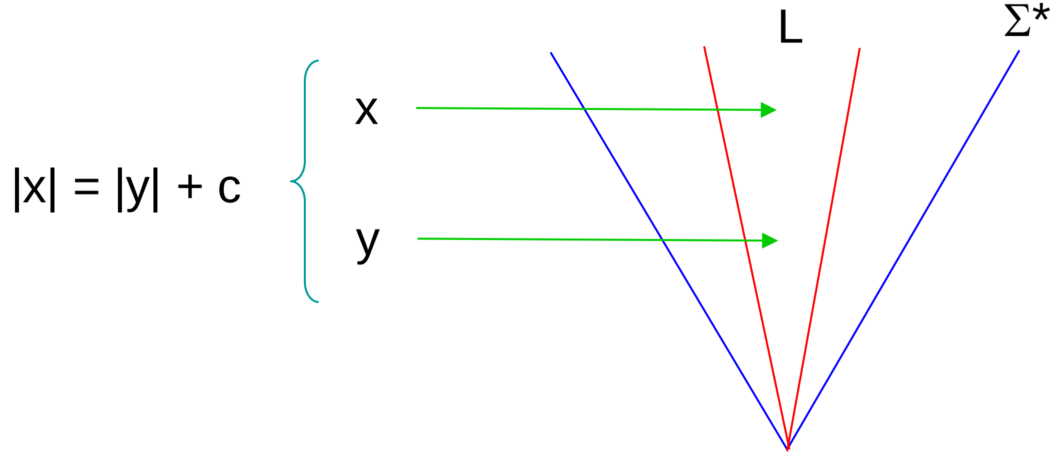
- This implies that C dissects L.
- Since C $\in$ P, L is P-dissectable.

$\boxed{\text{QED}}$

- Therefore, “P-dissectability” is not quite exciting to study.
- We then focus our attention on REG-dissectability.

Constantly Growing Languages I

- Let us consider languages composed of certain strings whose lengths are not quite far apart.
- A nonempty language L is constantly growing if there are a constant $p > 0$ and a finite subset $K \subseteq N^+$ that satisfies the following length condition:
  - for every string $x \in L$ with $|x| > p$, there exist a string $y \in L$ and a constant $c \in K$ for which $|x| = |y| + c$.



Constantly Growing Languages II

- Proposition: [Yamakami-Kato (2013)]

  Every infinite constantly-growing language is REG-dissectable.

❑ Proof Sketch:

- Let L be any infinite constantly-growing language with a constant p and a finite set K.
- Assume that $K = \{ c_1, c_2, \ldots, c_m \} \subseteq N^+$ (increasing order).
- Define $L_i = \{ x \in L \mid |x| \equiv i \pmod{(c_m+1)} \}$ for $i = 1,2,\ldots,c_m$.
- It is not difficult to prove that there are at least two distinct indices $i_1, i_2 \in [c_m]$ such that $|L_{i1}|=|L_{i2}|=\infty$.
- Consider the language $C = \{ x \mid |x| \equiv i_1 \pmod{(c_m+1)} \}$.
- This set C is regular and it clearly dissects L.

$\boxed{\text{QED}}$

Context-Free Languages

- A typical example of REG-dissectable language is context-free language.

- Theorem: [Yamakami-Kato (2013)]

  CFL is REG-dissectable.

❑ Proof Sketch:

- It is not difficult to show that every context-free language is constantly growing.
- Since any infinite constantly-growing language is REG-dissectable, the theorem immediately follows.

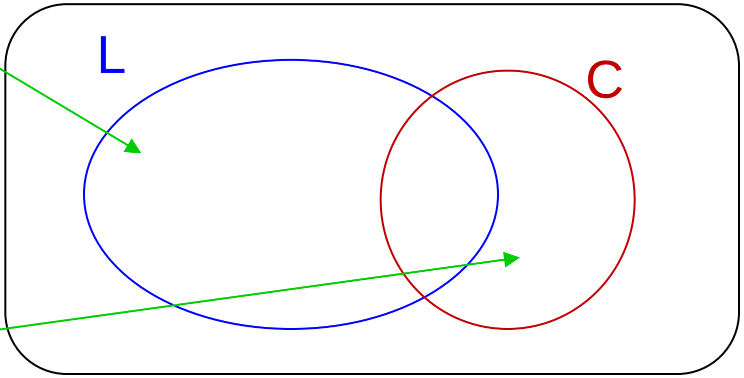
QED

Some Languages in co-CFL

- Let us consider languages in co-CFL.
- Take Fisher's language (over alphabet $\Sigma = \{ a,b \}$)

  $L = \{ (a^n b)^n \mid n \geq 0 \}$,

  which belongs to co-CFL.



- Define a regular language

  $C = \{ x \in \Sigma^* \mid \#_b(x) = \text{even} \}$.
- Since

  $L = \{ (a^n b)^n \mid n \text{ is even} \} \cup \{ (a^n b)^n \mid n \text{ is odd} \}$,

  it follows that $|C \cap L| = |\overline{C} \cap L| = \infty$
- (Open Problem) Is co-CFL REG-dissectable?

Non-REG-Dissectable Languages

- Recall the space complexity class L from Week 3.
- In fact, there are non-REG-dissectable languages in L.

- Theorem: [Yamakami-Kato (2013)]
  The complexity class L is not REG-dissectable.

❑ Proof Sketch:

- Consider the language $S = \{ 0^{n!} \mid n \geq 0 \}$ over the unary alphabet { 0 }.
- It suffices to show the following two statements.
  1. S is in L.
  2. S cannot be dissected by any regular language.

QED

Bounded Languages

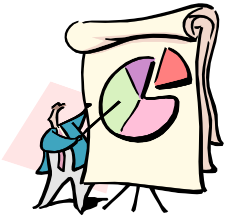
- Next, we consider special languages, called bounded languages. [Ginsburg-Spanier (1966)]

- A language L is called bounded if there is a finite set of strings $t_1,t_2,\ldots,t_k$ such that $L \subseteq t_1^*t_2^*\ldots t_k^*$.

$$x = \underbrace{t_1t_1\ldots t_1}_{i_1 \text{ times}}\;\underbrace{t_2t_2\ldots t_2}_{i_2 \text{ times}}\;\underbrace{t_3t_3\ldots t_3}_{i_3 \text{ times}}\;\cdots\;\underbrace{t_kt_k\ldots t_k}_{i_k \text{ times}}$$

- Examples:
  - $\{ a^ib^ic^j \mid i,j \geq 1 \} \quad \Leftarrow \quad t_1 = a_1,\ t_2 = b,\ t_3 = c$
  - $\{ (ab)^i(ca)^{2i}(acb)^{3i+1} \mid i \geq 1 \} \quad \Leftarrow \quad t_1 = ab,\ t_2 = ca,\ t_3 = acb$

Examples: BCFL(k)



- Recall that CFL(k) is the k-disjunctive closure of CFL.
- Here, we further consider bounded languages.

- BCFL(k) = set of all bounded languages in CFL(k)

- Theorem: [Yamakami-Kato (2013)]
  For any index $k \geq 1$, BCFL(k) is REG-(

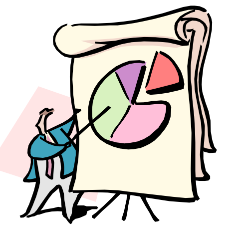
Semi-linear languages are defined by finite sets of linear equations.

❑ Proof Idea:

- Use Ginsburg's (1966) characterization of bounded context-free languages in terms of semi-linear sets.
- Since semi-linear sets are constantly-growing, we apply an argument on constantly-growing languages.

QED

Examples: $BCFL_k$



- Recall that CFL_k is the k-th level of the Boolean hierarchy over CFL.
- Moreover, we have defined $CFL_{BH} = \cup_{k\geq 1} CFL_k$.

- Here, we further consider bounded languages.

- $BCFL_k$ = set of all bounded languages in CFL_k
- $BCFL_{BH} = \cup_{k\geq 1} BCFL_k$ (Boolean hierarchy over BCFL)

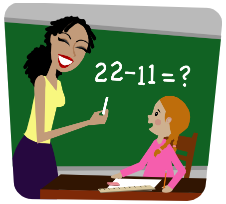
- Theorem: [Yamakami-Kato (2013)]
  $BCFL_{BH}$ is REG-dissectable.

Open Problems

- Concerning the notion of REG-dissectability, there are numerous open problems.

- The following is a short list of important open problems.

1. Is co-CFL REG-dissectable?
2. Is CFL(k) REG-dissectable?
3. Is CFL_k REG-dissectable?
4. Prove or disprove the REG-dissectability of Σ_k^{CFL}.



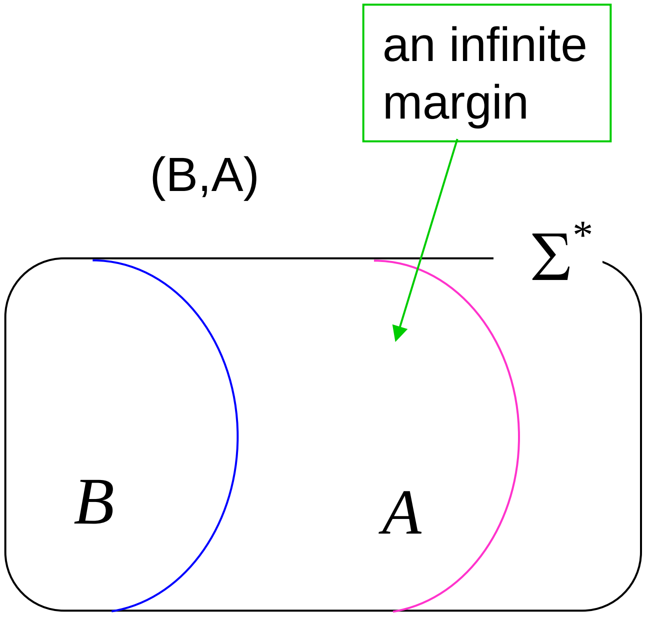
IV. Separation with Infinite Margins

1. Separation with Infinite Margins
2. Dissectability Implies i-Seperation
3. $BCFL_k$ and i-Separation

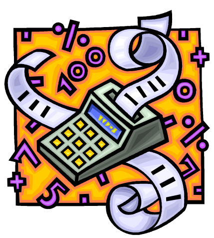
Separation with Infinite Margins I

- Let us take a quick look at an easy application of the REG-dissectability to other structural properties.

- Let A,B be any infinite languages.
- A covers B with an infinite margin (A is an i-cover of B, or A i-covers B) if $B \subseteq A$ and $A \neq_{ae} B$.
- The notation (B,A) means that A i-covers B.

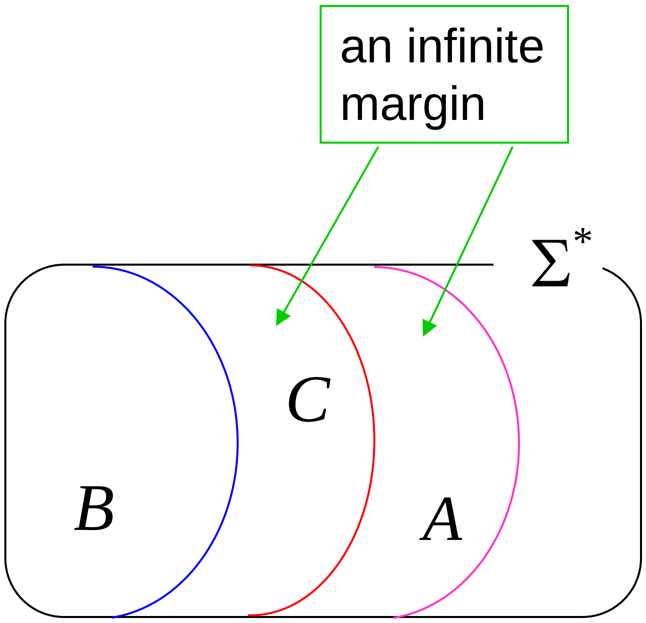


A i-covers B



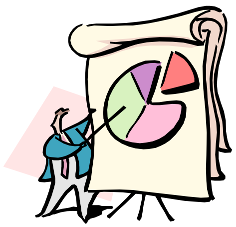
Separation with Infinite Margins II

- Let A,B,C be any infinite languages.
- C separates (B,A) with infinite margins (or C i-separates (B,A)) if $B \subseteq C \subseteq A$, $A \neq_{ae} C$, and $B \neq_{ae} C$.
- Let $\mathcal{C}$,$\mathcal{D}$ be any language families.
- Let $(\mathbb{D},\mathbb{C}) = \{ (B,A) \mid B \in \mathbb{D}, A \in \mathbb{C} \}$.
- $\mathbb{E}$ i-separates $(\mathbb{D},\mathbb{C})$ if, for every pair $(B,A) \in (\mathbb{D},\mathbb{C})$, there is a set $E \in \mathbb{E}$ that i-separates (B,A).



C i-separates (B,A)

Dissectability Implies i-separation



- Let $\mathbb{C}$,$\mathbb{D}$ be any language families.
- Theorem: [Yamakami-Kato (2013)]
  Assume that $\mathbb{C} - \mathbb{D}$ is REG-dissectable. Define $\mathbb{E} = \{ B \cup (A \cap C) \mid A \in \mathbb{C}, B \in \mathbb{D}, C \in \mathrm{REG} \}$. Then, $\mathbb{E}$ i-separates $(\mathbb{D},\mathbb{C})$.

❑ Proof Sketch:

- Let $A \in \mathbb{C}$, $B \in \mathbb{D}$, and $D = A - B$. Assume that D is infinite.
- Take a language $C \in \mathrm{REG}$ that dissects D.
- Define $E = B \cup (A \cap C)$, which belongs to $\mathbb{E}$.
- Since C dissects D, we have $|(A \cap C) - B| = |(A - C) - B| = \infty$.
- Hence, $B \subseteq E \subseteq A$ and $|A - E| = |E - B| = \infty$ hold.
- Therefore, E i-separates (B,A).

QED

BCFL_k and i-Separation

- As a consequence, we are able to prove the following theorem concerning bounded languages.

- Theorem: [Yamakami-Kato (2013)]

  BCFL_k i-separates $(\mathrm{BCFL}_k,\mathrm{BCFL}_k)$ for every $k\geq 1$.

❑ Proof Sketch:

- It suffices to prove that $\mathrm{BCFL}_k - \mathrm{BCFL}_k$ is REG-dissectable, because this helps us conclude that BCFL_k i-separates $(\mathrm{BCFL}_k,\mathrm{BCFL}_k)$ as seen before.
- The REG-dissectability of $\mathrm{BCFL}_k - \mathrm{BCFL}_k$ can be proven by induction on k.

QED

Open Problems

- We have just discussed the notion of i-separation.

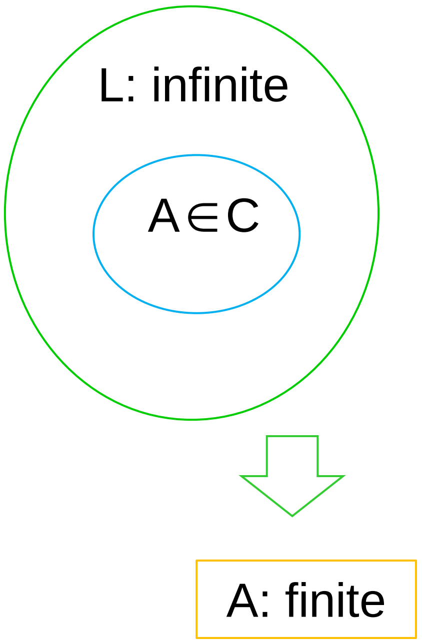
- The following is a list of important open problems.
  - Does CFL i-separate (CFL,CFL)?
  - Does CFL_k i-separate (CFL_k, CFL_k) for every $k \geq 1$?

V. Immunity and Simplicity

1. C-Immunity
2. Historical Background
3. Examples of REG- & CFL-Immune Languages
4. C-Simplicity
5. Examples of C-Simple Languages
6. REG-Bi-Immune Languages
7. Examples of REG-Bi-Immune Languages
8. Σ^p_k-Immunity and Σ^p_k-Simplicity

C-Immunity

- Flajolet and Steyaert (1974) first adapted the recursion-theoretic notion of "immunity" into complexity theory.
- Let C be any nonempty language family.
- A language L is C-immune $\Leftrightarrow$
  1. L is infinite, and
  2. no infinite subset A of L exists in C.
- A language family D is C-immune $\Leftrightarrow$
  - D contains a C-immune language.



- (Claim) C cannot be C-immune by the definition.
- (Open Question) Is NP P-immune?

Historical Background

- Flajolet and Steyaert (1974) showed:
  - $L_{eq} = \{ 0^n1^n \mid n \in N \}$ is REG-immune.
  - $L_{3eq} = \{ 0^n1^n2^n \mid n \in N \}$ is CFL-immune.
- The notion of immunity structurally differentiates the above two languages.
- In the next slide, we will give the proof of the above claim.
- But, similar languages below are not even REG-immune.
  - Equal $= \{ w \in \{0,1\}^* \mid \#_0(w)=\#_1(w) \}$
  - 3Equal $= \{ w \in \{0,1,2\}^* \mid \#_0(w)=\#_1(w)=\#_2(w) \}$
  - Because $\{(01)^n \mid n \in N \} \subseteq$ Equal $\{(012)^n \mid n \in N \} \subseteq$ 3Equal.

Proof Idea for "L_{eq}: REG-Immune"

- (Claim) [Flajolet-Steyaert (1974)]
  $L_{eq} = \{ 0^n1^n \mid n \geq 0 \}$ is REG-immune.

❑ Proof Sketch:

- We prove this claim by contradiction.
- Assume that there is an infinite subset A of L in REG.
- Take a pumping constant $m > 0$ (of the pumping lemma).
- Choose a string 0^n1^n in A with $n \geq m$ (because A is infinite).
- Let $xyz = 0^n1^n$ be a decomposition with $|y| > 0$.
- By the pumping lemma for REG, xy^kz is in A for any $k \geq 0$.
- However, clearly xy^kz does not belong to L_{eq}.
- This is a contradiction.

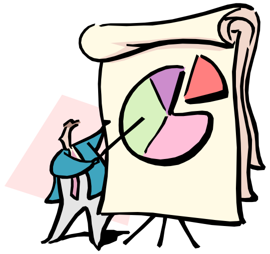
QED

Examples of REG-Immune Languages

- Proposition: [Yamakami (2013)]
  DCFL $\cap$ REG/n is REG-immune

❑ Proof Idea: Because L_{eq} is in both DCFL $\cap$ REG/n.

- Proposition: [Yamakami (2010)]
  DCFL – REG/n is REG-immune



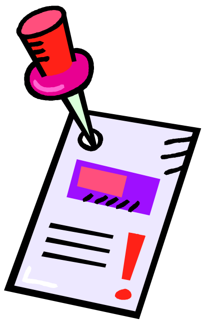
❑ Proof Idea: Because

➢ $Pal_{\#} = \{ w\#w^R \mid w \in \{0,1\}^* \}$ is REG-immune, and

➢ $Pal_{\#}$ is in DCFL – REG/n.

- In comparison, $Pal = \{ ww^R \mid w \in \{0,1\}^* \}$ is not REG-immune because $L = \{ 0^n 0^n \mid n \geq 0 \} \subseteq Pal$ and $L \in$ REG.

Examples of Immune Languages II



- Proposition: [Yamakami (2011)]
  CFL(2) $\cap$ REG/n is CFL-immune

❑ Proof Idea: Because L_{3eq} is in CFL(2) $\cap$ REG/n.

- Proposition: [Yamakami (2011)]
  L – CFL/n is CFL-immune

❑ Proof Idea: Because

  - $3Dup_{\#}$ = { w#w#w | w $\in$ {0,1}* } is CFL-immune, and
  - $3Dup_{\#}$ is in L – CFL/n.

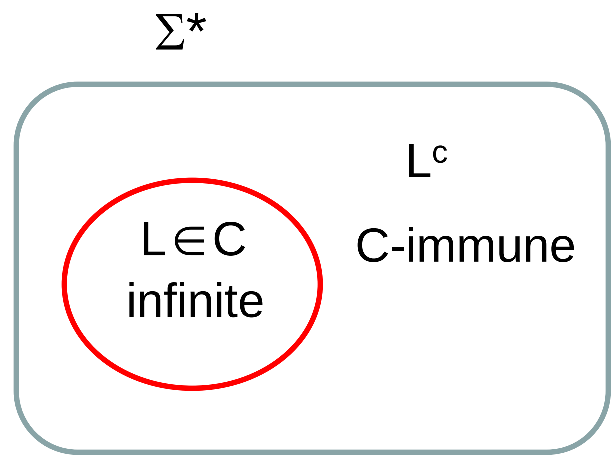
- The last result was improved by Suzuki (2016) to:
  - CFL(2) – CFL/n is CFL-immune.

(*) T. Suzuki. IAENG Int. J. Appl. Math. 46, 2016.

C-Simplicity

- There is another important notion related to immunity.
- Let C be any language family.

- A language L is C-simple $\Leftrightarrow$
  1) L is infinite,
  2) L is in C, and
  3) L^c is C-immune.

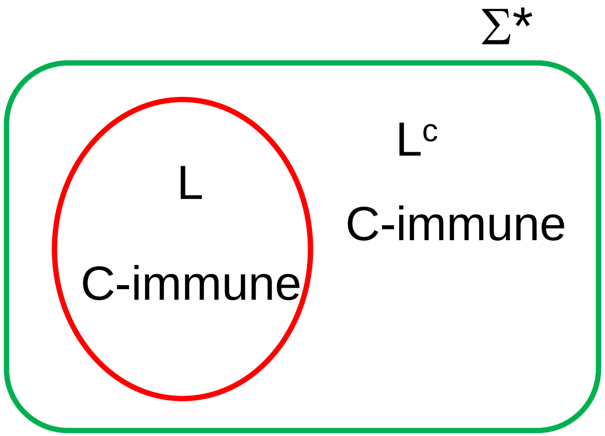


- (Claim) If a C-simple language exists, then $C \neq$ co-C.
- (Open Question) Is there any NP-simple language?

Examples of CFL-Simple Languages

- Consider the following languages ($k \geq 3$).
  - $L_{keq} = \{ a_1^n a_2^n \ldots a_k^n \mid n \in N \}$ (extensions of L_{eq})
  - $(L_{keq})^c$ is CFL-simple.
  - L_{keq} is in CFL(2) $\cap$ REG/n.
  - NOTE: Unfortunately, $(L_{keq})^c$ is not REG-immune.
- Theorem: [Yamakami (2011)]
  - There exists a CFL-simple language L.
  - Moreover, some L^c is in CFL(2) $\cap$ REG/n.
- (Open Question) Is there any REG-immune CFL-simple language?

C-Bi-Immunity



- C-bi-immunity is another extension of C-immunity.

- A language L is C-bi-immune $\Leftrightarrow$
  - L and L^c are both C-immune.
- A language family D is C-bi-immune $\Leftrightarrow$
  - D contains a C-bi-immune language.

- (Claim) EXP is P-bi-immune. [Schöning (1983)]

❑ Proof Idea:

- The desired language was constructed by diagonalization.

Examples of REG-Bi-Immune Languages

- Theorem: [Yamakami (2011)]

  L $\cap$ REG/n is REG-bi-immune.

❑ Proof Sketch:

- Consider the following two languages.
  - $L_{even} = \{w \in \{0,1\}^* \mid \exists k[2k < \log\log|w| \leq 2k+1]\} \cup \{\lambda\} \cup \{0,1\}^2$
  - $L_{odd} = \{w \in \{0,1\}^* \mid \exists k[2k+1 < \log\log|w| \leq 2k+2]\} \cup \{0,1\}$
- We can show that (1) $L_{even} \cup L_{odd} = \{0,1\}^*$, (2) $L_{even} \cap L_{odd} = \varnothing$, and (3) L_{even} and L_{odd} are both REG-immune.
- Moreover, L_{even} and L_{odd} are in L $\cap$ REG/n.

QED

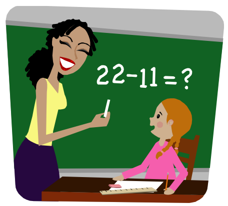
Σ^p_k-Immunity and Σ^p_k-Simplicity

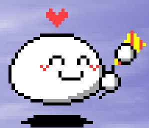
- Without detailed explanation, we describe some of the results obtained by Yamakami and Suzuki (2005).

1. Let $k \geq 1$. No Σ^p_k-simple set is h-Δ^p_k-d-complete for Σ^p_k.
2. A strongly NP^G-simple set exists relative to a Cohen-Feferman generic oracle G.
3. Let $k \geq 1$. All Σ^p_k-generic sets are honestly Σ^p_k-hyperimmune.
4. Let $k \geq 1$. No Σ^p_k-hypersimple set is P-T-complete for Σ^p_k.
5. Let $k \geq 1$. No Σ^p_k-simple set is Δ^p_k-1tt-complete for Σ^p_k if $U(\Sigma^p_k \cap \Pi^p_k) \not\subseteq SUB\Delta^{EXP}_k$.
6. If the k-immune hypothesis is true, then there exists an NP-simple set.

Open Problems

- We have just discussed the notion of i-separation.
- The following is a list of important open problems.
- Open Problems:
  - Is CFL REG-bi-immune?
  - Is CFL–REG/n REG-bi-immune?
  - Is there any REG-immune CFL-simple set?
  - Does an NP-simple language exist?





Thank you for listening

Q & A

I'm happy to take your question!

END