4^{th} Week

Relativizations and Hierarchies

Synopsis.

- Oracle Turing Machines and Relativization
- The Polynomial Hierarchy
- Generic Oracles
- Collapsing Recursive Oracles
- The CFL Hierarchy

April 30, 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

- ✎ L. Fortnow and T. Yamakami. Generic separation. Journal of Computer and System Sciences, Vol. 52, pp. 191-197 (1996)
- ✎ S. A. Cook, R. Impagliazzo, and T. Yamakami. A tight relationship between generic oracles and type-2 complexity Theory. Inf. Comput. 137(2): 159-170 (1997)
- ✎ T. Yamakami. Collapsing Recursive Oracles for Relativized Polynomial Hierarchies. In the Proc. of FCT 2005, Lecture Notes in Computer Science, vol. 3623, pp. 149-160 (2005)
- ✎ T. Yamakami and Y. Kato. The dissecting power of regular languages. Inf. Process. Lett. 113(4): 116-122 (2013)
- ✎ T. Yamakami. Oracle pushdown automata, nondeterministic reducibilities, and the hierarchy over the family of context-free languages. In Proc. of SOFSEM 2014, Lecture Notes in Computer Science, vol. 8327, pp. 514-525 (2014). Complete version is available at arXiv:1303.1717.

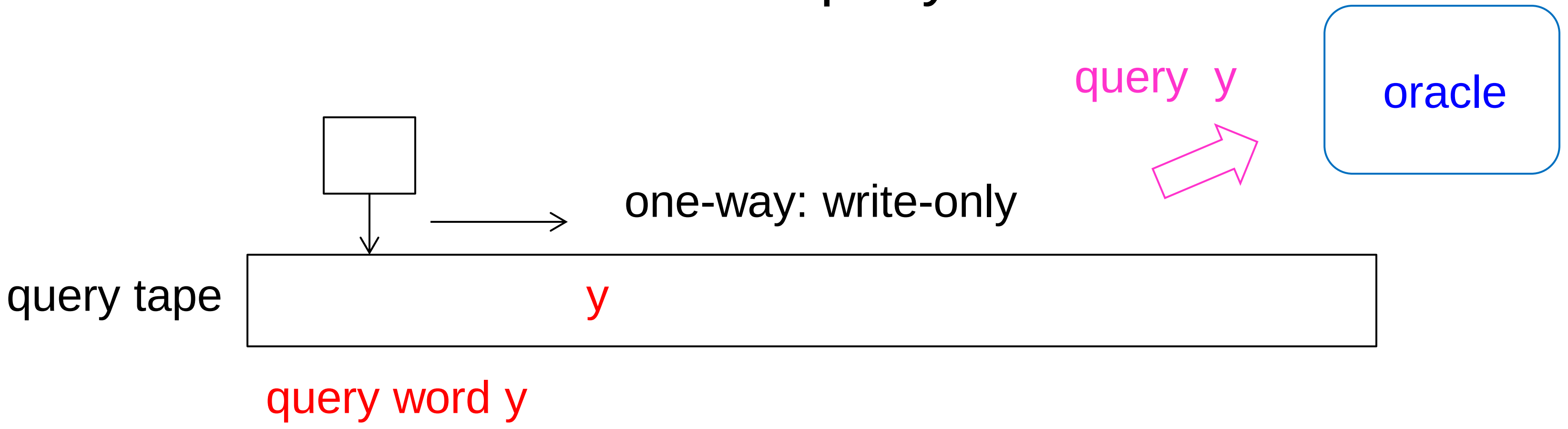
I. Oracle Turing Machines

1. Oracle Turing Machines
2. Oracle Computation
3. Turing Reductions
4. Turing Complete Problems
5. Relativizations and Relativized Worlds

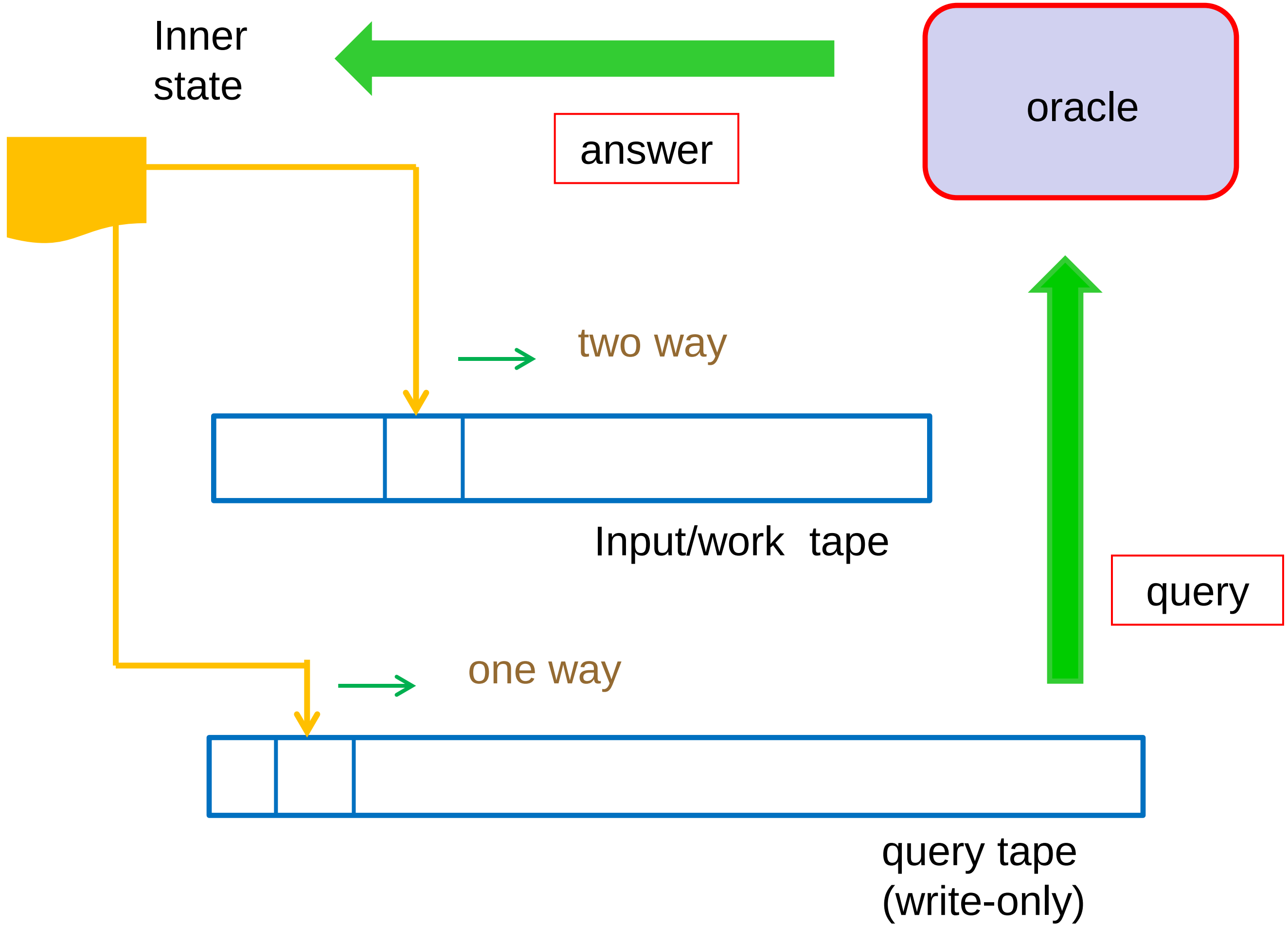
Oracle Turing Machines I (revisited)

- An oracle is an external information source, which can provide an underlying machine with necessary information via a process of query and answer.

- An oracle Turing machine (OTM) is equipped with an extra one-way write-only tape, called a query tape, by which the machine make a query to an oracle.



Oracle Turing Machines II (revisited)



Oracle Computation I

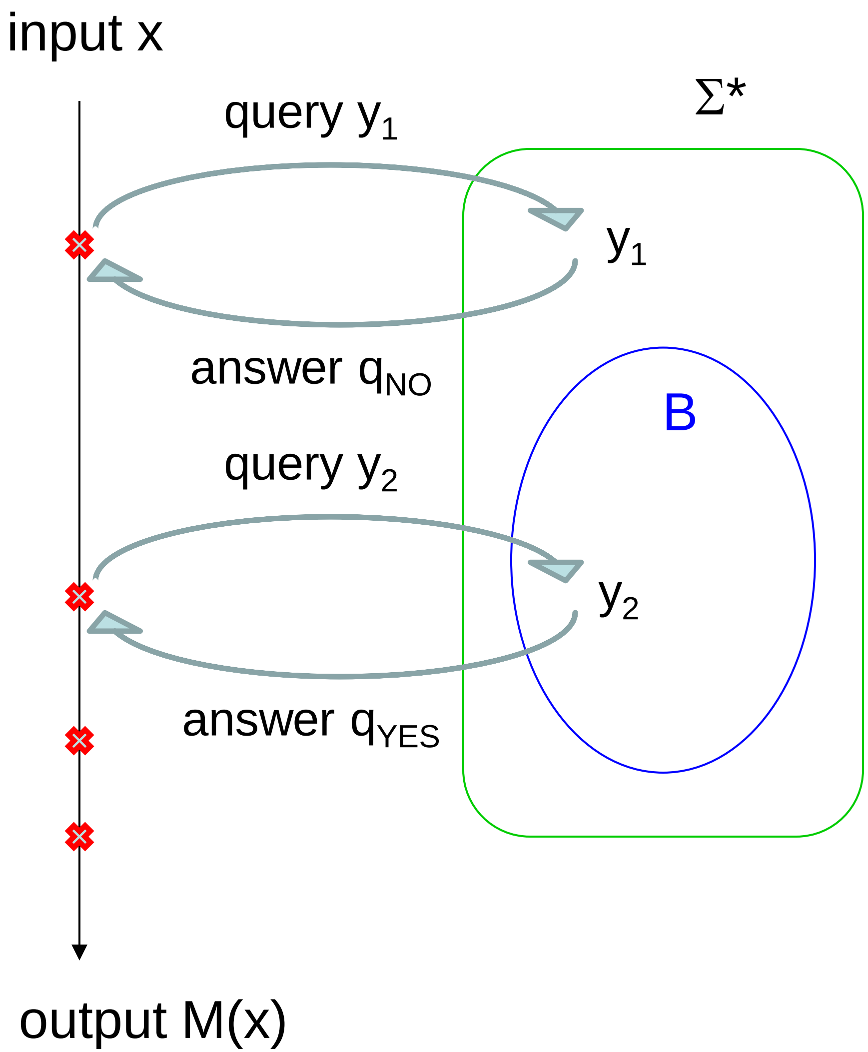
- Let M be an oracle Turing machine (OTM).
- Let x be any string in Σ^*.
- Let B be an oracle (which is now a language).

1. M starts with input x.
2. Whenever M writes a query word y on its query tape and enters a query state q_{query}, y is automatically sent to oracle B.
3. The oracle B returns its answer (YES/NO) by changing M's inner state from q_{query} to either q_{yes} or q_{no}, depending on whether $y \in B$ or $y \notin B$, respectively.
4. M resumes its computation, starting with q_{yes} or q_{no}.
5. If M halts, output M(x). Otherwise, go to Step 2.

Oracle Computation II (revisited)

- M: OTM, B: oracle

1. M starts with input x.
2. Whenever M writes a query word y on its query tape and enters a query state q_{query}, y is automatically sent to B.
3. The oracle B returns its answer (YES/NO) by changing M's inner state from q_{query}, to either q_{yes} or q_{no}.
4. M resumes its computation, starting with q_{yes} or q_{no}.
5. If M halts, output M(x). Otherwise, go to Step 2.



Languages Recognized by OTMs

- Let M be an OTM.
- Let B be an oracle (which is a language).

- We define a language recognized by M relative to B.

  $L(M,B) = \{ x \in \Sigma^* \mid M^B$ accepts x with oracle B $\}$.

- Note that L(M,B) is depending on the choice of oracle B.
- If we choose a different oracle, say, C, then L(M,C) may be different from L(M,B).

Turing Reductions

- We have already defined polynomial-time many-one reductions.
- Here, we define polynomial-time Turing reductions.
- For two languages A and B, we say that A is polynomial-time Turing reducible (p-T-reducible or $\leq_T^p$ -reducible) to B (written as $A \leq_T^p B$) if there is an OTM M such that
  - A = L(M,B); that is, for every input x, $x \in A \leftrightarrow M^B$ accepts x via making queries to the oracle B,
  - M runs in polynomial time.
- In other words,

$$A \leq_T^p B \iff \exists\, M\text{: p-time OTM } [\, A = L(M,B) \,]$$

Turing Complete Problems I

- Similar to $\leq_m^p$-educibility, $\leq_T^p$ -reducibility gives rise to a notion of completeness for a complexity class.
- To emphasize the use of Turing reductions, we often call this completeness notion by Turing completeness.

- A decision problem (or a language) L is said to be Turing complete for NP (p-T-complete for NP or $\leq_T^p$ -complete for NP) if
  1) $L \in NP$, and
  2) $A \leq_T^p L$ holds for every language $A \in NP$.

Turing Complete Problems II

- Note that all $\leq_T^p$ -complete problems are also $\leq_m^p$ -complete, because polynomial-time many-one reductions are also polynomial-time Turing reductions.
- However, the converse does not hold; namely,

$$\exists\, X,Y \text{ s.t. } X \not\leq_m^p Y \text{ and } X \leq_T^p Y.$$

- (Claim) SAT is Turing complete for NP. [Cook (1971)]
- Open Problem: are all p-T-complete problems for NP also p-m-complete for NP?

Relativizations and Relativized Worlds

- Providing an oracle to an underlying OTM M (resp., a complexity class C defined by OTMs) is called a relativization of M (resp., C).
- An informal term "relativized world" means a situation that is caused by a certain oracle.

- P^B = collection of all languages L(M,B) for polynomial-time deterministic oracle Turing machines M
- NP^B = collection of all languages L(M,B) for polynomial-time nondeterministic oracle Turing machines M

- We will see later a relativized world where P = NP and another relativized word where P $\neq$ NP.

II. Relativizations and the Hierarchy

1. Relativizations of P and NP
2. Relativized Worlds for P, NP, and co-NP
3. The Polynomial Hierarchy
4. Properties of the Polynomial Hierarchy
5. Complexity Class Θ_k^p
6. Advised Complexity Classes

Relativizations of P and NP I

- Recall that "relativization" means that we replace the original machines by oracle machines of the same types.
- Recall that P^B and NP^B are relativizations of P and NP relative to oracle B, respectively.

- Baker, Gill, and Solvay (1975) proved the following conflicting results.
  1) (Claim) There exists an oracle A s.t. $P^A = NP^A$.
  2) (Claim) There exists an oracle B s.t. $P^B \neq NP^B$.

- Next, we will explain why these claims are true.
- However, the above claims do not imply P = NP or P $\neq$ NP.

Relativizations of P and NP II

- (Claim) There exists an oracle A s.t. $P^A = NP^A$.

❑ Proof Sketch:

- Choose a PSPACE-complete set A.
- Note that $P^A = P^{PSPACE} = PSPACE$ because PSPACE is closed under polynomial-time Turing reductions.
- Similarly, $NP^A = NP^{PSPACE} = PSPACE$ by the same reasoning and $NP \subseteq PSPACE$.
- Therefore, $P^A = NP^A$ follows.

QED

Relativizations of P and NP III

- (Claim) There exists an oracle B s.t. $P^B \neq NP^B$.

❑ Proof Sketch

1. Enumerate all polynomial-time DTMs as $M_1, M_2, \ldots$
2. Let p_i be a polynomial that bounds the runtime of M_i.
3. Define an example language L^A as

   $$L^A = \{ x \mid \exists y \, [\, |y|=|x| \wedge y \in A \,] \}$$

   for any oracle A.
4. Note that $L^A \in NP^A$ for any A (by guessing y and querying it to A).
5. We want to construct the desired set B in the following way.

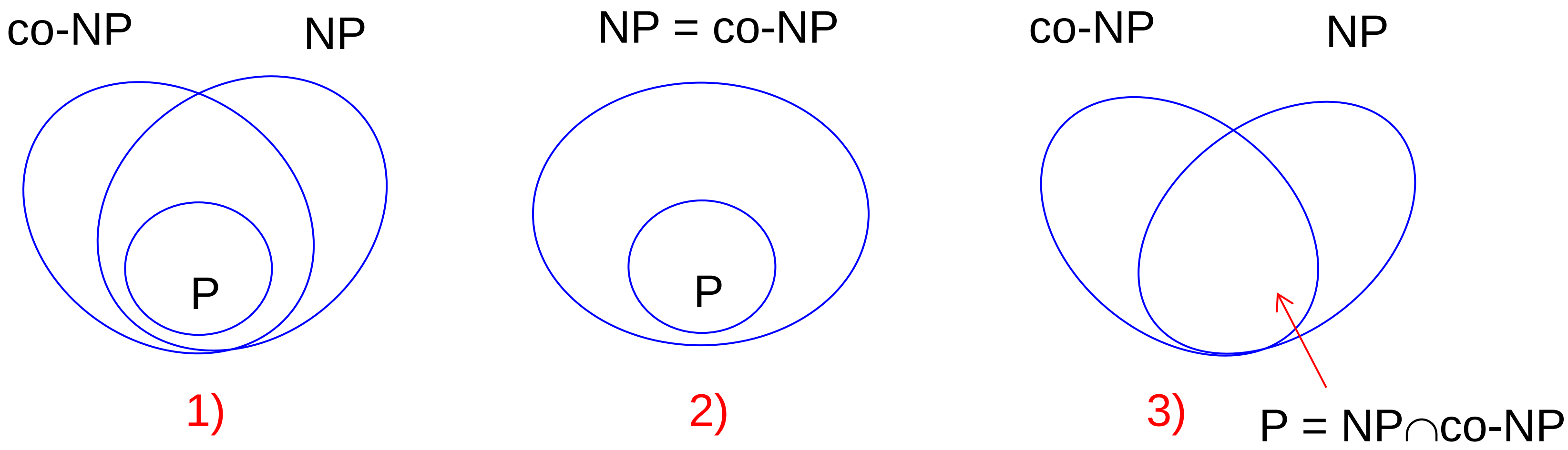
Relativizations of P and NP VI

6. Let $B_0 = \varnothing$ and let $t(1) = p_1(1)+1$.
7. For each $n \geq 1$, run M_n on input $1^{t(n)}$ with oracle B_{n-1}.
8. M_n makes only $p_n(t(n))$ queries to B_{n-1}. Let Q be the set of all queried words. Note that $|Q \cap \Sigma^{t(n)}| \leq p_n(t(n)) < 2^{t(n)}$.
9. There is a y_0 such that $y_0 \in \{0,1\}^{t(n)}$ but $y_0 \notin Q$.
10. Define $B_n = B_{n-1} \cup \{ y_0 \}$ if M_n rejects $1^{t(n)}$ with B_{n-1}, and $B_n = B_{n-1}$ otherwise.
11. Thus, $x \in L(M_i, B_n) \leftrightarrow x \notin L^{Bn}$.
12. Define $t(n+1)$ s.t. $t(n+1) > t(n)$ and $2^{t(n+1)} > p_{n+1}(t(n+1))$.
13. This means that M_i on input $1^{t(n)}$ cannot query strings in $\{ y \mid y \in \{0,1\}^{t(n+1)} \}$.
14. Finally, we set $B = \bigcup_{n \geq 1} B_n$

$\boxed{\text{QED}}$

Relativized Worlds for P, NP, and co-NP

- Three oracles A,B,C below provide quite different relativized worlds.
- (Claims) [Baker-Gill-Solvay (1975)]
  1) There is an oracle A s.t. $P^A \neq NP^A \cap coNP^A \neq NP^A$.
  2) There is an oracle B s.t. $P^B \neq NP^B = coNP^B$.
  3) There is an oracle C s.t. $P^C = NP^C \cap coNP^C \neq NP^A$.



The Polynomial Hierarchy I

- Meyer and Stockmeyer (1972,1973) introduced a notion of the polynomial-time hierarchy over NP.
- Customarily nowadays, we drop the word "-time" and call this hierarchy the polynomial hierarchy.

- The polynomial hierarchy consists of the following complexity classes: for every index k ≥ 1,

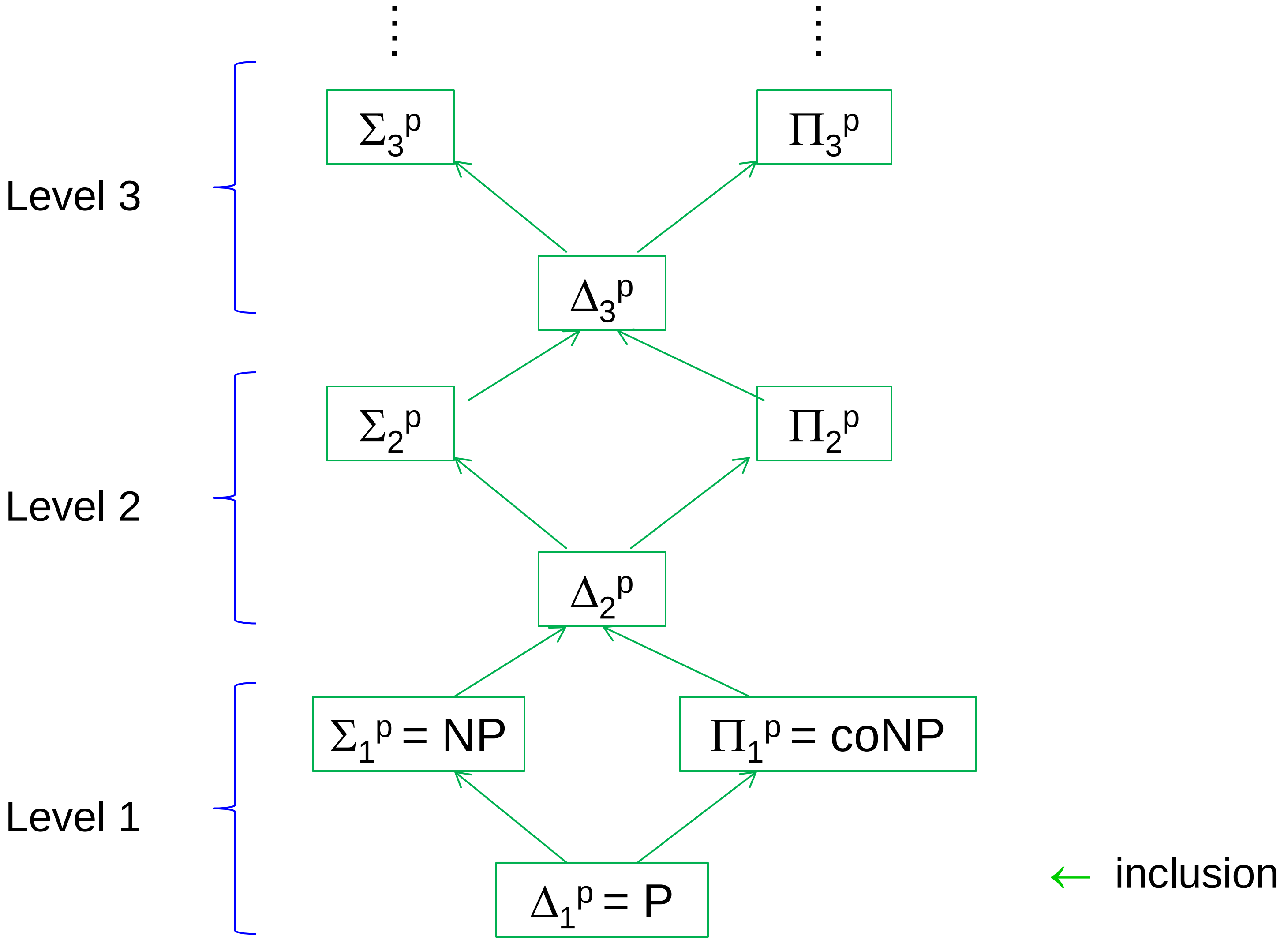
$$\Delta_1^p = P; \quad \Sigma_1^p = NP; \quad \Pi_1^p = co\text{-}NP$$

$$\Delta_{k+1}^p = P^{\Sigma_k^p}; \quad \Sigma_{k+1}^p = NP^{\Sigma_k^p}; \quad \Pi_k^p = co\text{-}\Sigma_k^p$$

relativizations

complementation

The Polynomial Hierarchy II

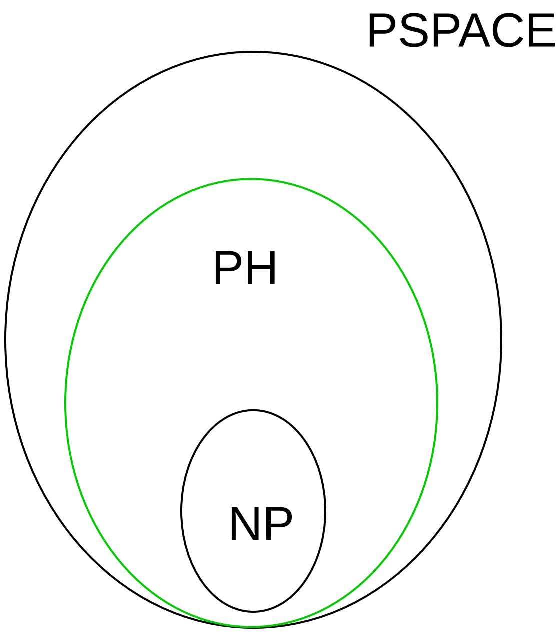


Properties of the Polynomial Hierarchy

- We define the complexity class PH as follows:

$$\mathrm{PH} = \bigcup_{k \in \mathbb{N}} \left(\Sigma_k^{\mathrm{p}} \cup \Pi_k^{\mathrm{p}} \right)$$

- The complexity class PH includes all classes in the polynomial hierarchy.
- (Claim) $\mathrm{NP} \subseteq \mathrm{PH} \subseteq \mathrm{PSPACE}$
- (Claim) If P = NP, then P = PH.
- (Claim) $\mathrm{P}^{\mathrm{PH}} = \mathrm{NP}^{\mathrm{PH}} = \mathrm{PH}$.



- Open Problems
  - Is $\Delta_k^{\mathrm{p}} = \Sigma_k^{\mathrm{p}}$ for each $k \geq 1$?
  - Is $\Delta_k^{\mathrm{p}} = \Sigma_k^{\mathrm{p}} \cap \Pi_k^{\mathrm{p}}$ for each $k \geq 1$?

Complexity Class Θ_k^p

- Wagner (1987) introduced the complexity class Θ_2^p.

- A language A is in $\Theta_2^p \Leftrightarrow$ there exist a polynomial-time deterministic OTM M, and a language $B \in NP$ such that
  1. $A = L(M,B)$, and
  2. on each input x, M makes queries $O(\log(|x|))$ times.

- We can naturally extend Θ_2^p to Θ_k^p for any $k \geq 2$ by setting $\Theta_1^p = P$ and by replacing NP ($= \Sigma_1^p$) in the above definition with Σ_{k-1}^p.

- (Claim) For any $k \geq 2$, $\Sigma_{k-1}^p \cup \Pi_{k-1}^p \subseteq \Theta_k^p \subseteq \Delta_k^p$.
- (Claim) $\Theta_2^p \subseteq PP$.

  [Beigel-Hemachandra-Wechsung (1991)]

Advised Complexity Classes

- We have seen the notion of advice in Week 3.
- Let us supply advice to the polynomial-time hierarchy.
- For any index $k \geq 1$ and for any language L,

$L \in \Delta_k^p$/poly $\Leftrightarrow$ there exists a language $A \in \Delta_k^p$ and an advice function h such that

1) $|h(n)| = n^{O(1)}$ for any $n \in N$, and
2) $L = \{ x \mid \langle x,h(|x|)\rangle \in A \}$.

- Similarly, we can define Σ_k^p/poly and Θ_k^p/poly.
- (Claim) $\Delta_k^p/\mathrm{poly} \subseteq \Theta_k^p/\mathrm{poly} \subseteq \Sigma_k^p/\mathrm{poly}$ for every $k \geq 1$.

III. Collapsing Recursive Oracles

1. Relativized Polynomial Hierarchies
2. Basic Oracle Separations and Collapses
3. Collapsing Recursive Oracles
4. Separations and Collapses
5. Sparse Sets

Relativized Polynomial Hierarchies

- We can relativize the polynomial hierarchy by taking an oracle A.
- We write $\Delta_k^p(A)$, $\Sigma_k^p(A)$, and $\Pi_k^p(A)$ as relativizations of Δ_k^p, Σ_k^p, and Π_k^p, respectively, with respect to oracle A.
- More formally, for every index $k \geq 1$, we define:

$$\Delta_1^p(A) = P^A; \quad \Sigma_1^p(A) = NP^A; \quad \Pi_1^p(A) = co\text{-}NP^A$$

$$\Delta_{k+1}^p(A) = P^{\Sigma_k^p(A)}; \quad \Sigma_{k+1}^p(A) = NP^{\Sigma_k^p(A)}; \quad \Pi_k^p(A) = co\text{-}\Sigma_k^p(A)$$

- A relativized polynomial hierarchy relative to A is

  $\{ \Delta_k^p(A), \Sigma_k^p(A), \Pi_k^p(A) \mid k \geq 1 \}$.

- We can also relativize Θ_k^p to oracle A and obtain $\Theta_k^p(A)$.

Basic Oracle Separations and Collapses

- Yao (1985) constructed a recursive oracle A s.t.
  - (Claim) $\Delta_k^p(A) \neq \Sigma_k^p(A) \neq \Pi_k^p(A)$ for all $k \geq 1$.
- Ko (1989)
  - (Claim) $\Sigma_k^p(B_k) \neq \Sigma_{k+1}^p(B_k) = \Sigma_{k+2}^p(B_k)$ for each $k \geq 1$.
- Heller (1984)
  - (Claim) $\Delta_2^p(C) \neq \Sigma_2^p(C) = \Pi_2^p(C)$.
  - (Claim) $\Sigma_1^p(D) \neq \Delta_2^p(D) = \Sigma_2^p(D)$.
- Bruschi (1992)
  - (Claim) $\Delta_k^p(E) \neq \Sigma_k^p(E) = \Pi_k^p(E)$ for all $k \geq 3$.
  - (Claim) $\Sigma_{k-1}^p(F) \neq \Delta_k^p(F) = \Sigma_k^p(F)$ for all $k \geq 3$.
- Sheu and Long (1994)
  - (Claim) $\Theta_k^p(H) \neq \Delta_k^p(H) \neq \Sigma_k^p(H)$ for all $k \geq 2$.
  - (Claim) $\Sigma_{k-1}^p(J) \neq \Theta_k^p(J) = \Sigma_k^p(J)$ for all $k \geq 2$.

Collapsing Recursive Oracles

- As we have seen earlier, Ko (1989) constructed recursive oracles (i.e., computable oracle) that force a relativized polynomial-time hierarchy to collapse to any fixed level.
- We call such oracles collapsing oracles for simplicity.
- Yamakami (2005) defined the collapsing recursive oracle polynomial (CROP) hierarchy as:

  - $\mathrm{CRO}\Sigma_k^p = \{ A \in \mathrm{REC} \mid \Sigma_k^p(A) = \Sigma_{k+1}^p(A) \}$
  - $\mathrm{CRO}\Delta_k^p = \{ A \in \mathrm{REC} \mid \Delta_k^p(A) = \Delta_{k+1}^p(A) \}$
  - $\mathrm{CRO}\Theta_k^p = \{ A \in \mathrm{REC} \mid \Theta_k^p(A) = \Theta_{k+1}^p(A) \}$
  - $\mathrm{CROPH} = \cup_{i \geq 1} \mathrm{CRO}\Delta_i^p$

REC = class of recursive languages

Basic Properties

- (Claim) For any $k \geq 1$, it follows that

$$\mathrm{CRO}\Delta_k^p \subseteq \mathrm{CRO}\Sigma_k^p \subseteq \mathrm{CRO}\Theta_k^p \subseteq \mathrm{CRO}\Pi_k^p .$$

- (Claim) All PSPACE-complete problems are in $\mathrm{CRO}\Delta_1^p$.

- A set is said to be coinfinite if its complement (with respect to a fixed universe) is infinite.

- Lemma: [Yamakami (2005)]

  Let $k \geq 1$ and assume that $\Delta_k^p \neq \Sigma_k^p$. If $A \in \mathrm{CRO}\Sigma_k^p$, then A is infinite and coinfinite.

Separations and Collapses

- Yamakami (2005) showed the following properties of the CROP hierarchy.

- Theorem: [Yamakami (2005)] Let $k \geq 1$.
  $\mathrm{CRO}\Delta_k^p \neq \mathrm{CRO}\Sigma_k^p \neq \mathrm{CRO}\Theta_k^p \neq \mathrm{CRO}\Pi_k^p$.

- Proposition: [Yamakami (2005)]
  Let $k \geq 1$. The following equivalences hold.
  - $\mathrm{NP} \subseteq \mathrm{CRO}\Delta_k^p \Leftrightarrow \Delta_k^p = \Sigma_k^p$.
  - $\mathrm{NP} \subseteq \mathrm{CRO}\Sigma_k^p \Leftrightarrow \Sigma_k^p = \Pi_k^p$.
  - $\mathrm{NP} \subseteq \mathrm{CRO}\Theta_k^p \Leftrightarrow \Theta_k^p = \Sigma_k^p$.

Sparse Sets

- A set S over alphabet Σ is called polynomially sparse (or simply sparse) if there is a polynomial p such that

$$| S \cap \Sigma^{\leq n} | \leq p(|x|)$$

for all $n \in N$, where $\Sigma^{\leq n} = \{ x \in \Sigma^* \mid |x| \leq n \}$.
- Let rSPARSE = { A | A is recursive and sparse }.

- Proposition: [Yamakami (2005)]

Let $k \geq 2$.

1. $\mathrm{rSPARSE} \cap \mathrm{CRO}\Delta_k^p \neq \emptyset \Leftrightarrow \Sigma_k^p \subseteq \Delta_k^p/\mathrm{poly}$
2. $\mathrm{rSPARSE} \cap \mathrm{CRO}\Sigma_k^p \neq \emptyset \Leftrightarrow \Pi_k^p \subseteq \Sigma_k^p/\mathrm{poly}$
3. $\mathrm{rSPARSE} \cap \mathrm{CRO}\Theta_k^p \neq \emptyset \Leftrightarrow \Sigma_k^p \subseteq \Theta_k^p/\mathrm{poly}$

IV. Generic Oracles

1. Conditions and Extensions
2. Generic Oracles
3. Complexity Class UP
4. Separations by Generic Oracles
5. Extension of UP to $U\Delta_k^p$
6. Generic Separations
7. Proof Ideas

Conditions and Extensions I

- Blum and Impagliazzo (1987) considered a class of oracles, which are called generic oracles.
- Recall that χ_A denotes the characteristic function of language A (i.e., $\forall x\ [(x \in A \to \chi_A(x)=1) \wedge [(x \notin A \to \chi_A(x)=0)]$).

- A condition σ is a partial function from $\{0,1\}^*$ to $\{0,1\}$ with a finite domain $dom(\sigma)$.
- Hence, a condition is often identified with a string.
- A condition τ extends σ (denoted by $\sigma \sqsubseteq \tau$) if $dom(\sigma) \subseteq dom(\tau)$ and $\sigma(x) = \tau(x)$ for every $x \in dom(\sigma)$.
- A set A extends σ (denoted by $\sigma \sqsubseteq A$) if $\chi_A(x) = \sigma(x)$ for every $x \in dom(\sigma)$.

Conditions and Extensions II

- Consider the following example for (σ,τ,A).

x:	λ	0	1	00	01	10	11	000	001	.....
σ:	*	*	*	1	0	*	*	1	*	.....
τ:	0	*	*	1	0	1	*	1	0	......
χ_A:	0	1	1	1	0	1	0	1	0	......

$\mathrm{dom}(\sigma) = \{\ 00, 01, 000, \ldots\ \}$

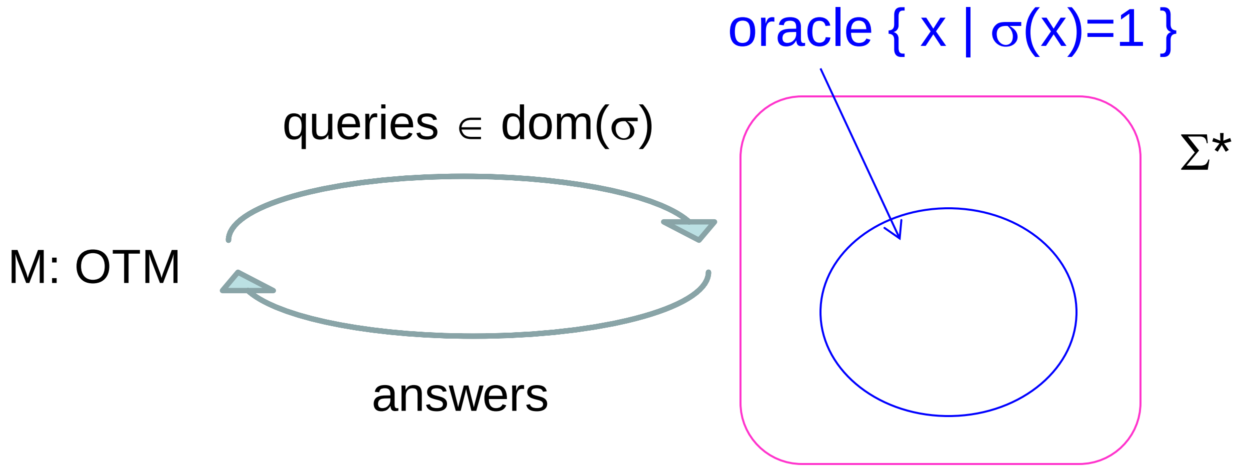
$\mathrm{dom}(\tau) = \{\ \lambda, 00, 01, 10, 000, 001, \ldots\ \}$

$A = \{\ 0, 1, 00, 10, 000, \ldots\ \}$

- (Claim) $\sigma \sqsubseteq \tau \sqsubseteq A$, because:
  1. $\mathrm{dom}(\sigma) \subseteq \mathrm{dom}(\tau)$ $\leftarrow$ blue & red circles
  2. $\sigma(x) = \tau(x)$ for all $x \in \mathrm{dom}(\sigma)$ $\leftarrow$ red circles
  3. $\tau(x) = \chi_A(x)$ for all $x \in \mathrm{dom}(\tau)$ $\leftarrow$ blue & red circles

Generic Oracles I

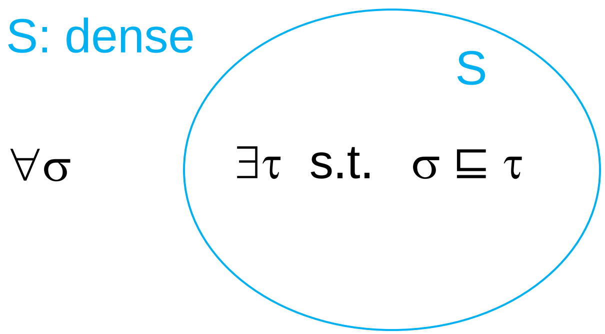
- For an oracle Turing machine M, the machine M^σ behaves like M with access to the oracle $\{ x \mid \sigma(x)=1 \}$ and all queries may be made within dom(σ).
- When some queries are made outsides of dom(σ), we treat M^σ as being undefined.



Generic Oracles II

- A set S of conditions is dense if, for every condition σ, there is another condition $\tau \in$ S such that τ extends σ.
- A set A $\subseteq \Sigma^*$ meets a set S of conditions if there is a condition $\sigma \in$ S such that A extends σ.
- A set S is arithmetical if S is exactly definable in first-order arithmetic.
- A set G is (Cohen) generic if G meets every dense arithmetic set of conditions.

meaning: easy to define in an arithmetic way



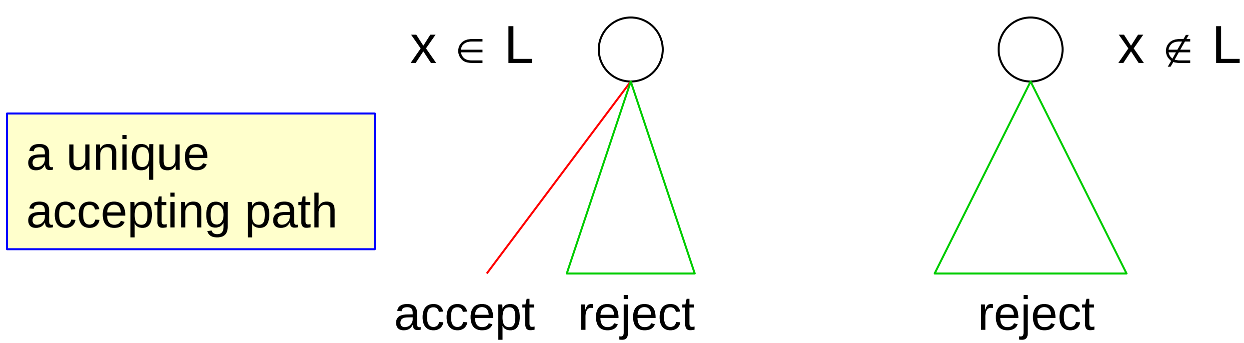
Complexity Class UP

- We introduce a complexity class called UP.
- For any language L,

  $L \in$ UP $\Leftrightarrow$ there exists an NTM M such that, for any x,

  1. $x \in L \rightarrow$ M accepts x,
  2. $x \notin L \rightarrow$ M rejects x, and
  3. M has at most one accepting path on input x.

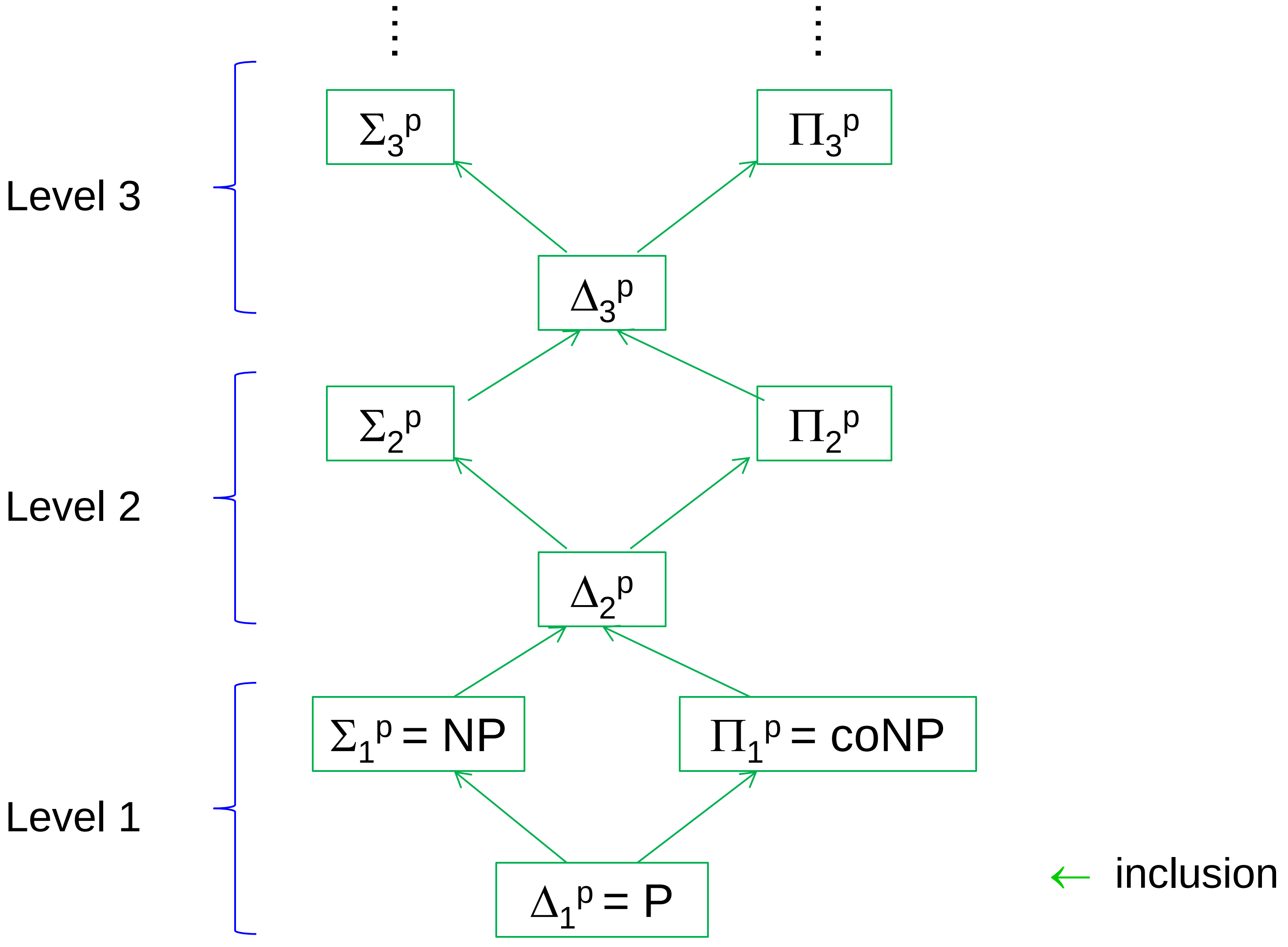
- We can relativize UP to UP^A, using oracle A.



Separations by Generic Oracles

- With generic oracles, we do not have conflicting results for most complexity classes.
- For example:
  - If a single generic oracle G satisfies $P^G = NP^G$ or $P^G \neq NP^G$, then, for all generic oracles H, we obtain $P^H = NP^H$ or $P^H \neq NP^H$, respectively
- (Claim) [Blum-Impagliazzo (1987)]
  - For any generic oracle G, $P^G \neq NP^G$.
  - If P = NP, then $P^G = UP^G = NP^G \cap \text{co-}NP^G$ for any generic oracle G.
- (*) Can we extend this result to any higher level of the polynomial hierarchy?

The Polynomial Hierarchy (revisited)



Extension of UP to $U\Delta_k^p$

- We have already defined the complexity class UP.
- Here, we extend it to fit into the k-th level of the polynomial hierarchy.

- Fortnow and Yamakami (1996) defined the following extension of UP.

$$U\Delta_1^p = P; \quad U\Delta_{k+1}^p = UP^{\Sigma_k^p} \quad \text{for any } k \geq 1.$$

- (Claim) $\Delta_k^p \subseteq U\Delta_k^p \subseteq \Sigma_k^p \cap \Pi_k^p$ for each index $k \geq 1$.

Generic Separations

- Fortnow and Yamakami (1996) proved that the polynomial hierarchy is infinite regarding generic oracles.

- Theorem: [Fortnow-Yamakami (1996)] Let $k \geq 2$.
  - $U\Delta_k^p(G) \cap \Pi_k^p(G) \neq \Delta_k^p(G)$ for any generic oracle G.

- As a result, we obtain $\Sigma_k^p(G) \cap \Pi_k^p(G) \neq \Delta_k^p(G)$.

❑ Proof Idea:

- Let A be any oracle.
- We define a function $f_n^A: \{0,1\}^n \rightarrow \{0,1\}$ as follows.

Proof Idea I

- This a new function f_n^A is defined as:

$$f_n{}^A(x) = \chi_A(x0^{n|})\chi_A(x0^{n-1}1)\chi_A(x0^{|n-2}1^2)\cdots\chi_A(x01^{n-1})$$

- Here, we define a permutation as a bijection (i.e., a one-one and onto function) on all binary strings of length n.
- Let PERM^A = $\{ 1^n \mid f_{2n}{}^A$ is a permutation $\}$
- Let $S_{2n} = 1^n(0+1)^n+(0+1)^{n-1}0^{n+1}$ (regular expression)
- Define L(A) = $\{ 1^n \in \text{PERM}^A \mid$

  $\exists y\in\{0,1\}^n\ \forall z\in\{0,1\}^n\ [\ f_{2n}{}^A(yz) \in S_{2n}\ ]\ \}$

Proof Idea II

- It suffices to prove that, for any $k \geq 2$,
  1) $L(A) \in U\Delta_k^p(A) \cap \Pi_k^p(A)$ for any oracle A, and
  2) $L(G) \notin \Delta_k^p(G)$ for any generic oracle G.

- Item 1) can be shown directly.
- For the case of $k = 2$, item 2) can be directly proven.
- When $k \geq 3$, the proof of 2) requires the notion of random restrictions and Håstad's switching lemma for circuits.

QED

Open Problems

- Concerning generic oracles, there is still a wide room to obtain interesting results.

1. Is it true that $\Delta_k^p(G) \neq U\Delta_k^p(G) \cap \text{co-}U\Delta_k^p(G)$ for any $k \geq 2$ and for any generic oracle G?
2. Prove more separations and collapses of complexity classes relative to generic oracles.

- (*) The notion of generic oracle turns out to be closely related to type-2 computability. This subject will be discussed in Week 6.

V. Many-One Reductions by 1npda's

1. m-Reduction 1npda's as Oracle 1npda's
2. Nondeterministic Pushdown Automata with Query Tapes
3. Many-One CFL-Reducibility
4. Examples
5. Characterization by Dyck Languages
6. K-Fold Application of Reductions

m-Reduction 1npda's as Oracle 1npda's

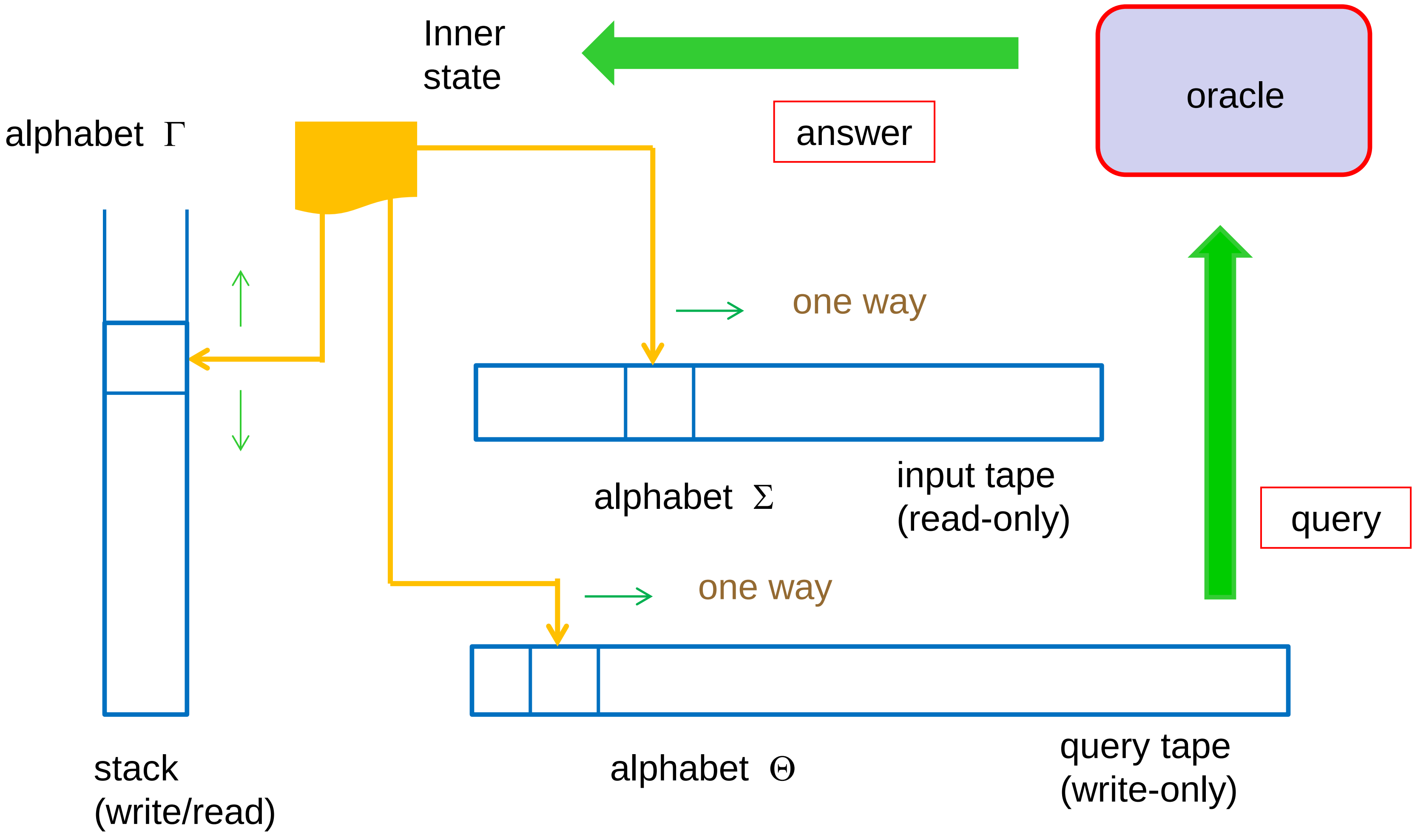
- Yamakami (2014) considered nondeterministic many-one reducibility based on 1npda's.
- An m-reduction 1npda or an oracle 1npda M
  - M = $(Q,\Sigma,\{¢,\$\},\Theta,\Gamma,\delta,q_0,Z_0,Q_{acc},Q_{rej})$ is a standard 1npda plus a write-only query tape and a special transition function δ:

    $$\delta : (Q - Q_{halt}) \times ($$

This is because all context-free languages are recognized by O(n)-time 1npda's.

- Termination condition of M:
  - All computation paths (both accepting and rejecting) should terminate (reaching halting states) within O(n) steps.
- $ACC_M(x)$ = set of accepting computation paths of M on x

One-Way Nondeterministic Pushdown Automata with Query Tapes



Many-One CFL-Reducibility

- Let L be a language over Σ and A be a language over Θ.
- L is many-one CFL-reducible to A $\Leftrightarrow$

  $\exists$ M (m-reduction 1npda) s.t. $\forall x \in \Sigma^*$

  1) along any accepting path p, M on x produces a valid string $y_p \in \Theta^*$ (called a query word) on a query tape, and
  2) $x \in L \Leftrightarrow \exists p \in ACC_M(x)$ [$y_p \in A$].
- In this case, we write $L \in CFL_m^A$ or $L \in CFL_m(A)$.
- The language A is customarily called an oracle.
- Given a language family F,
  - $CFL_m^F = CFL_m(F)$ = union of CFL_m^A for all $A \in F$.

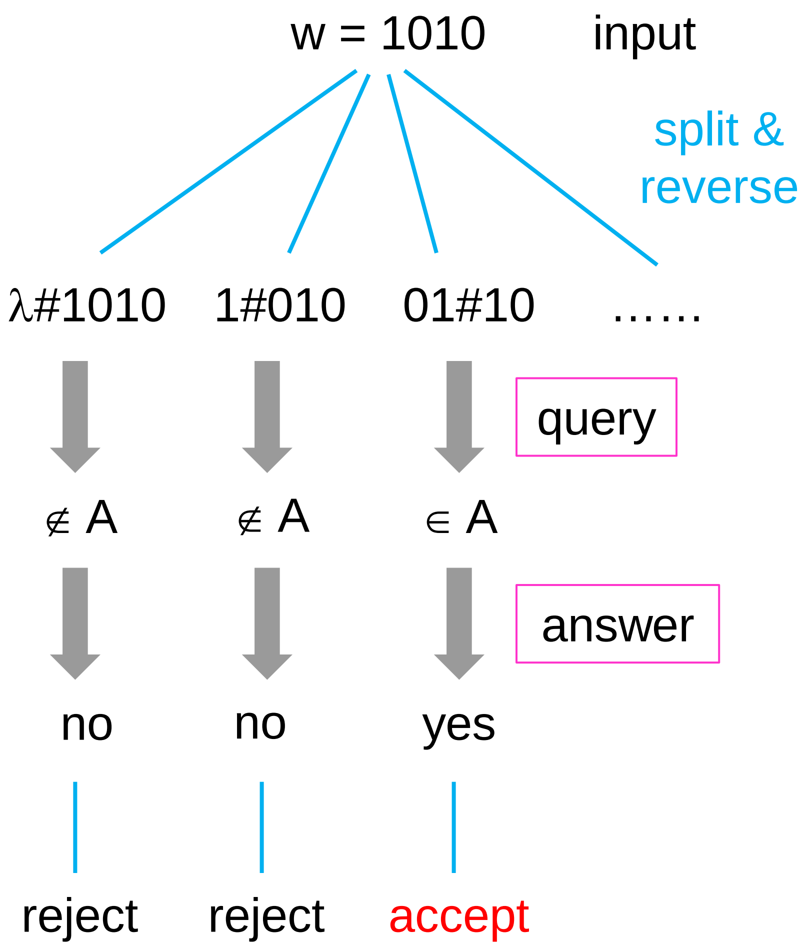
Example: Dup_2 I

- Consider the following language Dup_2.
- $Dup_2 = \{ xx \mid x \in \Sigma^* \}$ (duplication) over alphabet Σ
  - Is not context-free
  - Is many-one CFL-reducible to $A = \{ x^R\#x \mid x \in \Sigma^* \} \in CFL$.
  - Belongs to $CFL_m^A \subseteq CFL_m^{CFL}$.
- A reduction is made by the following oracle 1npda M.
  1. On input w, nondeterministically split it into xy.
  2. Using a stack, produce $x^R\#y$ on a query tape, where # is a special symbol not in Σ.
  3. Make a query to oracle.
  4. If oracle answers "yes," accept w; otherwise, reject w.
- (*) See the next slide for illustration.

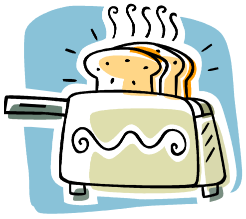
Example: Dup_2 II

- The oracle 1npda M works as follows with oracle A. (again)
  1. On input w, nondeterministically split it into xy.
  2. Using a stack, produce $x^R\#y$ on a query tape, where # is a special symbol not in Σ.
  3. Make a query to oracle.
  4. If oracle A answers “yes,” accept w; otherwise, reject w.

$A = \{ x^R\#x \mid x \in \Sigma^* \}$



Example: Dup_2 III



- The oracle npda M for Dup_2 (again)
  1. On input w, nondeterministically split it into xy.
  2. Using a stack, produce $x^R\#y$ on a query tape, where # is a special symbol.
  3. Make a query to oracle.
  4. If oracle answers "yes," accept w; otherwise, reject w.
- More formally, we define M as follows: where $\sigma,\tau\in\Sigma$

$$\delta(q_0,c,Z_0)=\{(q_0,Z_0,\lambda)\}$$
$$\delta(q_0,\$,Z_0)=\{(q_{acc},Z_0,\#)\}$$
$$\delta(q_0,\sigma,Z_0)=\{(q_1,\sigma Z_0,\lambda)\}$$
$$\delta(q_1,\sigma,\tau)=\{(q_1,\sigma\tau,\lambda),(q_2,\sigma\tau,\lambda)\}$$
$$\delta(q_2,\lambda,\tau)=\{(q_2,\lambda,\tau)\}$$
$$\delta(q_2,\lambda,Z_0)=\{(q_3,Z_0,\#)\}$$
$$\delta(q_3,\sigma,Z_0)=\{(q_3,Z_0,\sigma)\}$$
$$\delta(q_3,\$,Z_0)=\{(q_{acc},Z_0,\lambda)\}$$

$$\delta:(Q-Q_{halt})\times(\breve{\Sigma}\cup\{\lambda\})\times\Gamma\to P(Q\times\Gamma^*\times(\Theta\cup\{\lambda\}))$$

Similar Examples

- Let us consider other but similar languages.
- $Dup_3 = \{ xxx \mid x \in \Sigma^* \}$ (3 copies) over Σ
  - Is not context-free
  - Is many-one CFL-reducible to $B = \{ x^R\#x\#x\#x^R \mid x \in \Sigma^* \}$.
  - Belongs to $CFL_m^B \subseteq CFL_m^{CFL}$.
- $Match = \{ x\#w \mid \exists u,v \in \Sigma^* [w = uxv] \}$ (matching) over Σ
  - Is not context-free
  - Is many-one CFL-reducible to $C = \{ x^R\#x \mid x \in \Sigma^* \}$
  - Belongs to $CFL_m^C \subseteq CFL_m^{CFL}$.
- Consequence: $CFL \neq CFL_m^{CFL}$.
  - ➢ Compare this with $NP = NP_m^{NP}$.

Example: Sq I

- Consider a slightly more complex language.
- $Sq = \{0^n 1^{n^2} \mid n \geq 1\}$ (squared length) over alphabet {0,1}
  - Is not context-free
  - Is CFL-m-reducible to

$$D = \{\ w' = 0^i \# 1^{j_1} \# 1^{j_2} \# \cdots \# 1^{j_k} \mid \text{(i) and (ii)}\ \},\ \text{where}$$

$$\text{(i)}\ \ j_2 = j_3,\ j_4 = j_5, \cdots$$

$$\text{(ii)}\ \ i = k$$

D∈CFL

  - Belongs to $CFL_m^D \subseteq CFL_m^{CFL}$.
- The second item shown above can be proven by the following oracle 1npda M using D as an oracle.

Example: Sq II

$$D = \{\ w' = 0^i \# 1^{j_1} \# 1^{j_2} \# \cdots \# 1^{j_k} \mid \text{(i) and (ii)}\ \},$$
$$\text{where (i) } j_2 = j_3, j_4 = j_5, \cdots \quad \text{(ii) } i = k$$

- The desired oracle 1npda M works as follows.

1. On input w, check if w is of the form 0^i1^j for some i,j $\geq$ 0.
2. Simultaneously, guess $(j_1,j_2,\ldots,j_k)$ to satisfy:

$$\text{(1)}\ \ j = j_1 + j_2 + \cdots + j_k$$
$$\text{(2)}\ \ j_1 = j_2, j_3 = j_4, \cdots$$

3. Produce w′ (shown in the above box) on a query tape.
4. Make a query to oracle D.
5. If "yes," then accept w; otherwise, reject w.

Characterization by Dyck Languages

- Dyck languages over $\Sigma = \{ \sigma_1,\sigma_2,\ldots,\sigma_d \} \cup \{ \sigma'_1,\sigma'_2,\ldots,\sigma'_d \}$
  - A Dyck language L is generated by a context-free grammar with the following production set:

$$\{ S \to \lambda \mid SS \mid \sigma_i S \sigma'_i : i = 1,2,\ldots,d \}$$

  - E.g., When d = 2, L is a set of all balanced parentheses.
  - DYCK = set of all Dyck languages
- Proposition: [Yamakami (2014)]

$CFL_m^{CFL} = CFL_m^{DCFL} = CFL_m^{DYCK} = NFA_m^{DYCK} = CFL_m(NFA_m^{DYCK})$.

- This lemma suggest that DYCK may be considered as the most difficult language in CFL under many-one CFL reductions.

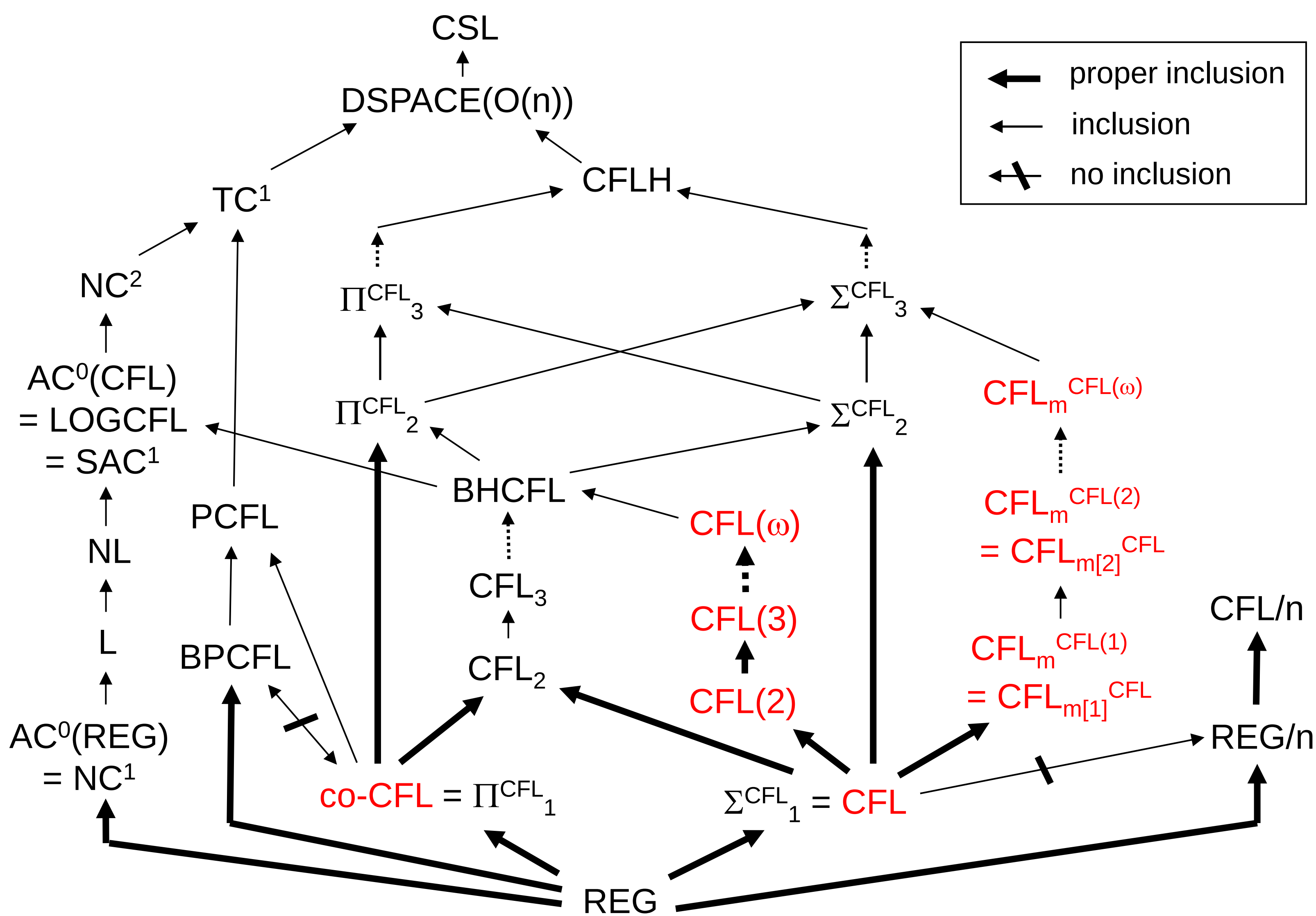
K-Fold Application of Reductions

- Many-one CFL-reducibility lacks the transitivity property.

Namely, $CFL_m^{CFL} \neq CFL$.

- K-fold application of reductions
  - $CFL_{m[1]}^A = CFL_m^A$
  - $CFL_{m[k+1]}^A = CFL_m(CFL_{m[k]}^A)$
  - $CFL_{m[k]}^F$ = union of $CFL_{m[k]}^A$ for all $A \in F$.
- K-conjunctive closure of CFL
  - $CFL(k) = \{ A \mid \exists B_1, B_2, \ldots, B_k \in CFL \; [\, A = B_1 \cap B_2 \cap \ldots \cap B_k \,] \}$
  - $CFL(\omega) = CFL(1) \cup CFL(2) \cup \ldots$
  - $\{ CFL(k) \mid k \in \mathbb{N}^+ \}$ is an infinite hierarchy. [Liu-Weiner (1973)]
- Lemma: [Yamakami (2014)]
  $CFL_{m[k]}^{CFL} = CFL_m^{CFL(k)}$ and $\bigcup_{k \geq 1} CFL_{m[k]}^{CFL} = CFL_m^{CFL(\omega)}$.

Inclusion Relations among Language Families



VI. Turing Reductions by 1npda's

1. T-Reduction 1npda's as Oracle 1npda's
2. Adaptive Queries in Nondeterministic Computation
3. Turing CFL-Reducibility
4. Example
5. The Boolean Hierarchy over CFL

T-Reduction 1npda's as Oracle 1npda's

- Similarly to m-reductions, Yamakami (2014) considered Turing reductions.
- A T-reduction 1npda or an oracle 1npda M
  - M = $(Q,\Sigma,\{¢,\$\},\Theta,\Gamma,\delta,q_0,Z_0, Q_{oracle},Q_{acc},Q_{rej})$ is a standard 1npda with a write-only query tape, a special state set Q_{oracle}, and δ such that

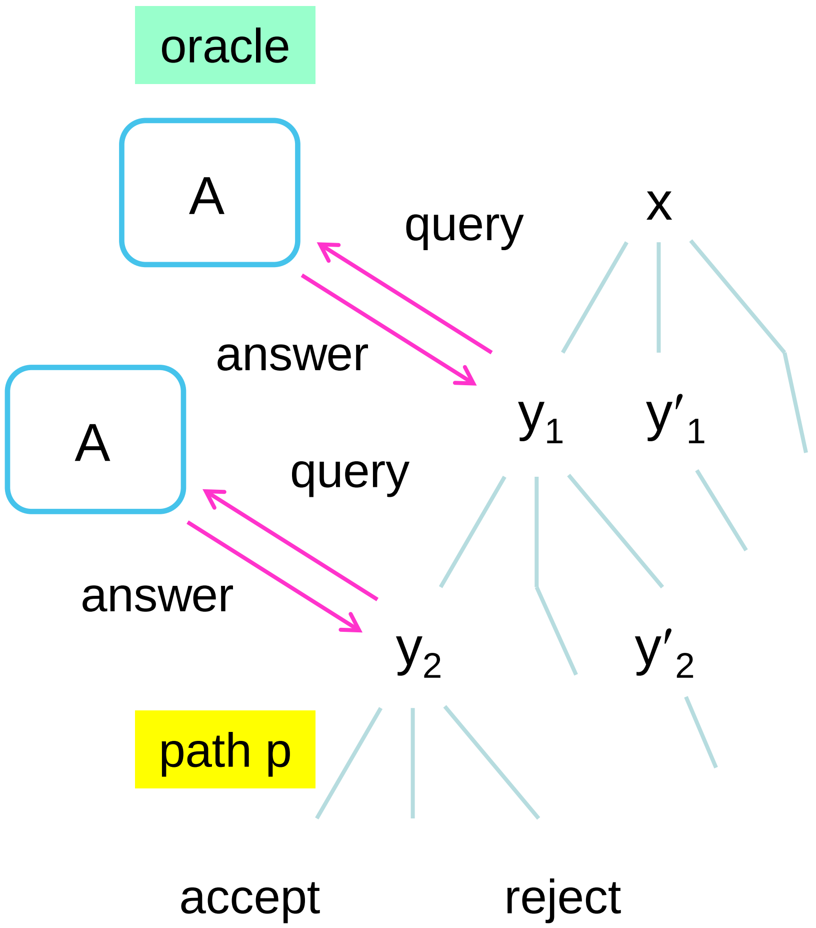
$$Q_{oracle} = \{q_{query}, q_{yes}, q_{no}\}$$

$$\delta : (Q - Q_{halt} \cup \{q_{query}\}) \times (\breve{\Sigma} \cup \{\lambda\}) \times \Gamma \rightarrow P((Q - \{q_{yes}, q_{no}\}) \times \Gamma^{*} \times (\Theta \cup \{\lambda\}))$$

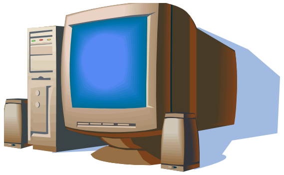
- Termination condition of M:
  - All computation paths (both accepting and rejecting paths) should terminate (reaching halting states) within O(n) time, no matter what oracle is provided.

Adaptive Queries in Nondeterministic Computation

- Let M be an oracle 1npda.
- Input x is given.
- M queries y_1 along a computation path p.
- The query tape is automatically reset.
- Depending on an oracle answer, M makes another query on y_2 along path p.
- M continues making queries.
- At the time when path p terminates, M must enter a halting state.



Turing CFL-Reducibility



- Let L be a language over Σ and A be a language over Θ.

- L is Turing CFL-reducible to A $\Leftrightarrow$

  $\exists$ M (T-reduction 1npda) s.t. $\forall x \in \Sigma^*$
  - $x \in L \Leftrightarrow$ M accepts x using oracle A
  - termination condition of M

All computation paths (both accepting and rejecting) should terminate (reaching halting states) within O(n) time, no matter what oracle is provided.

- In the above case, we write
  - $L \in CFL_T^A$ or $L \in CFL_T(A)$.
- Given a language family F, let
  - $CFL_T^F = CFL_T(F)$ = union of CFL_T^A for all $A \in F$.

The Boolean Hierarchy over CFL

- There is a nice relationship between CFL_T^A and CFL_m^A (as well as NFA_m^A).

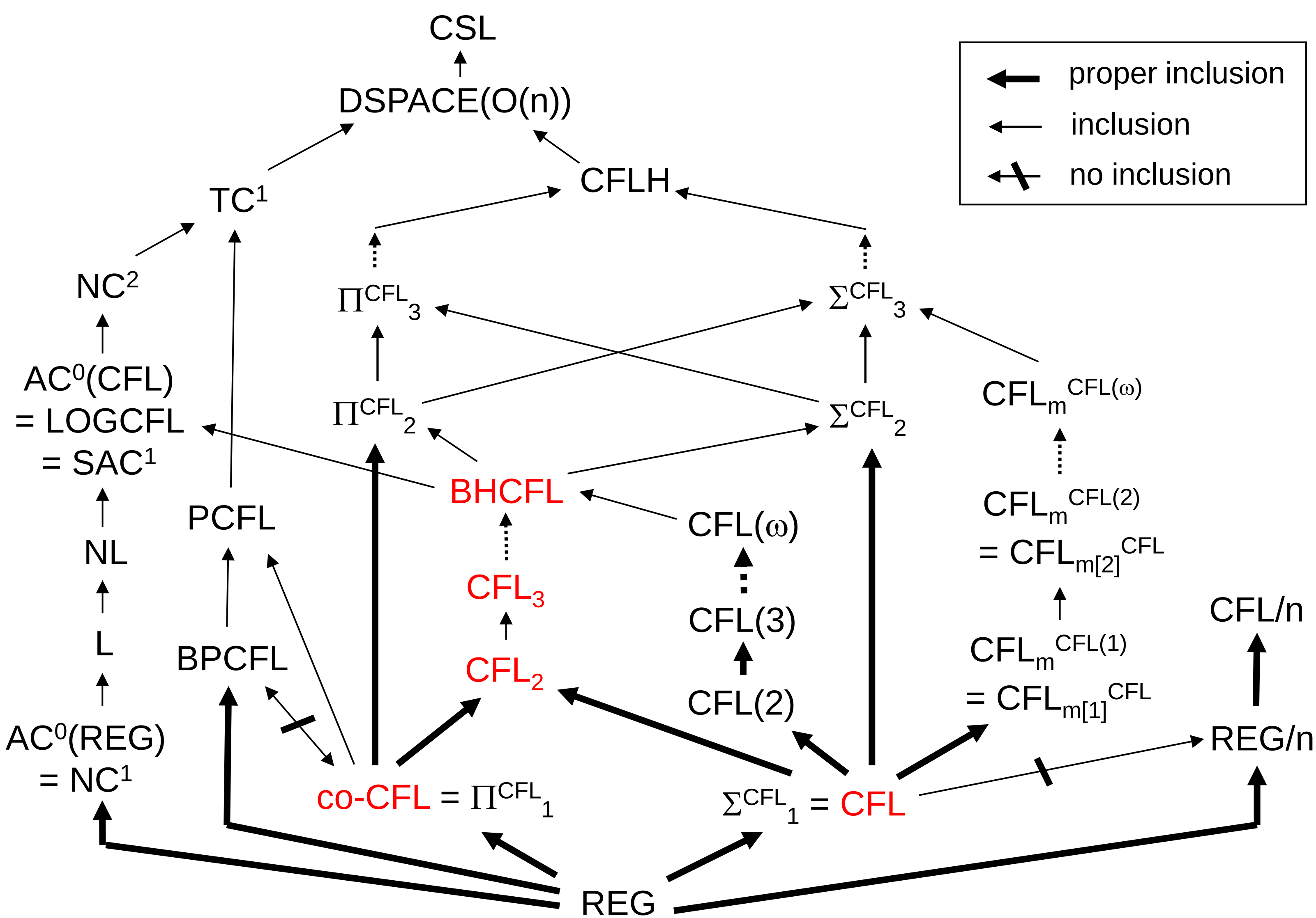
- Proposition [Yamakami (2014)]

  $\mathrm{CFL}_T^{\mathrm{CFL}} = \mathrm{CFL}_m(\mathrm{CFL}_2) = \mathrm{NFA}_m(\mathrm{CFL}_2)$.

- We define the notation of the Boolean hierarchy (over CFL).
- Boolean hierarchy over CFL [Yamakami-Kato (2013)]
  - $\mathrm{CFL}_1 = \mathrm{CFL}$
  - $\mathrm{CFL}_{2k} = \{ A \cap B \mid A \in \mathrm{CFL}_{2k-1} \wedge B \in \text{co-CFL} \}$
  - $\mathrm{CFL}_{2k+1} = \{ A \cup B \mid A \in \mathrm{CFL}_{2k} \wedge B \in$
  - $\mathrm{BHCFL} = \mathrm{CFL}_1 \cup \mathrm{CFL}_2 \cup \mathrm{CFL}_3 \cup \ldots$
  - E.g., $\mathrm{CFL}_2 = \{ A \cap B \mid A \in \mathrm{CFL}, B \in \text{co-CFL} \}$

Since co-CFL$\subseteq$$\mathrm{CFL}_2$,
we obtain CFL $\neq$ CFL_2.

Inclusion Relations among Language Families



VII. The CFL Hierarchy

1. The CFL Hierarchy (CFLH)
2. Closure Properties of CFLH
3. Examples
4. Structural Properties of CFLH
5. Relationships to the Polynomial Hierarchy

The CFL Hierarchy (CFLH) I

- Yamakami (2014) defined a hierarchy over CFL using oracle 1npda's.
- The CFL hierarchy is $\{ \Delta_k^{CFL}, \Sigma_k^{CFL}, \Pi_k^{CFL} \mid k \in \mathbb{N} \}$ whose elements are defined as follows.
  - $\Delta_1^{CFL} = \mathrm{DCFL}$
  - $\Sigma_1^{CFL} = \mathrm{CFL}$
  - $\Delta_{k+1}^{CFL} = \mathrm{DCFL}_T(\Sigma_k^{CFL})$ for any $k \geq 1$
  - $\Sigma_{k+1}^{CFL} = \mathrm{CFL}_T(\Sigma_k^{CFL})$ for any $k \geq 1$
  - $\Pi_k^{CFL} = \text{co-}\Sigma_k^{CFL}$ for any $k \geq 1$
- We further define CFLH as

$$\mathrm{CFLH} = \Sigma_1^{CFL} \cup \Sigma_2^{CFL} \cup \Sigma_3^{CFL} \cup \ldots.$$

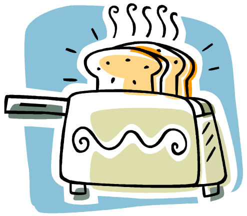
The CFL Hierarchy (CFLH) II

- The CFL hierarch satisfies the following basic properties.

- Proposition: [Yamakami (2014)] Let $k \geq 1$.
  1) $CFL_T(\Sigma_k^{CFL}) = CFL_T(\Pi_k^{CFL})$
  2) $DCFL_T(\Sigma_k^{CFL}) = DCFL_T(\Pi_k^{CFL})$
  3) $\Sigma_k^{CFL} \cup \Pi_k^{CFL} \subseteq \Delta_{k+1}^{CFL} \subseteq \Sigma_{k+1}^{CFL} \cap \Pi_{k+1}^{CFL}$
  4) $CFLH \subseteq DSPACE(O(n))$

- NOTE: item 4) comes from the termination condition of oracle 1npda's.

Closure Properties of CFLH



- Each Σ_k^{CFL} (k≥1) is closed under the following operations.
  - Length-nondecreasing substitution
  - Concatenation
  - Union
  - Reversal
  - Kleene closure
  - λ-free homomorphism
  - Inverse homomorphism

A substitution $s:\Sigma\rightarrow P(\Theta^*)$ is length nondecreasing $\Leftrightarrow$ $s(\sigma) \neq \emptyset$ and $\lambda \notin s(\sigma)$ for all $\sigma\in\Sigma$.

A homomorphism $h:\Sigma\rightarrow\Theta^*$ is λ-free $\Leftrightarrow$ $h(\sigma) \neq \lambda$ for all $\sigma\in\Sigma$.

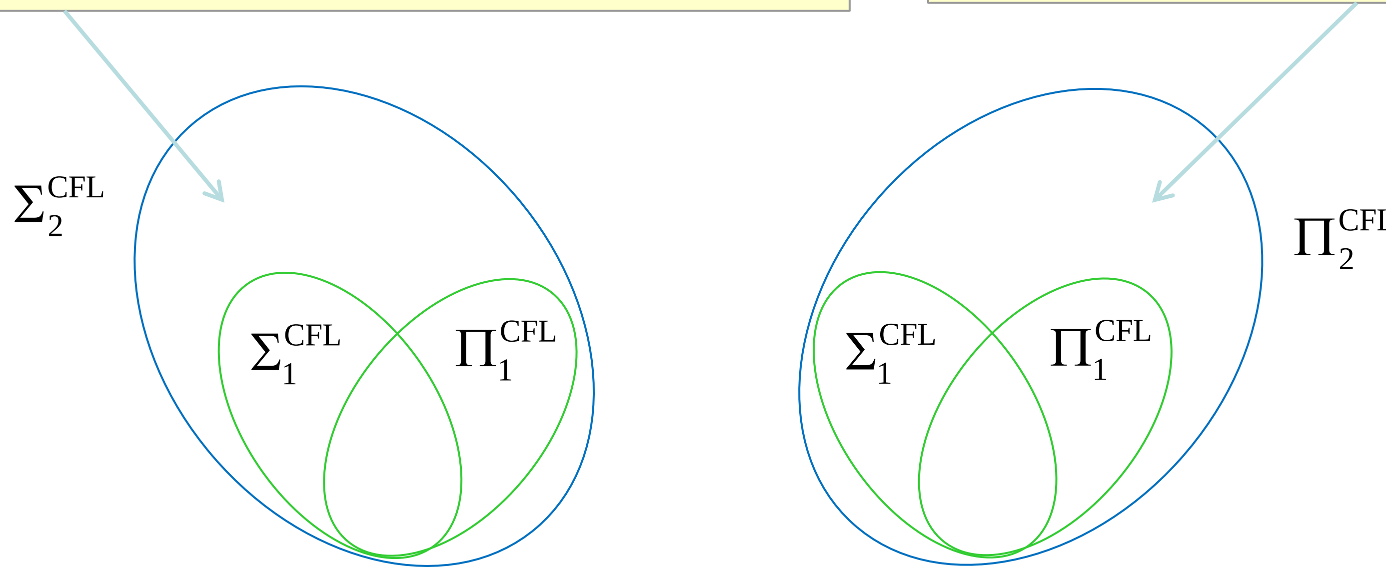
- Σ_1^{CFL} is not closed under intersection nor complementation, because Σ_1^{CFL} = CFL.

Examples of Languages

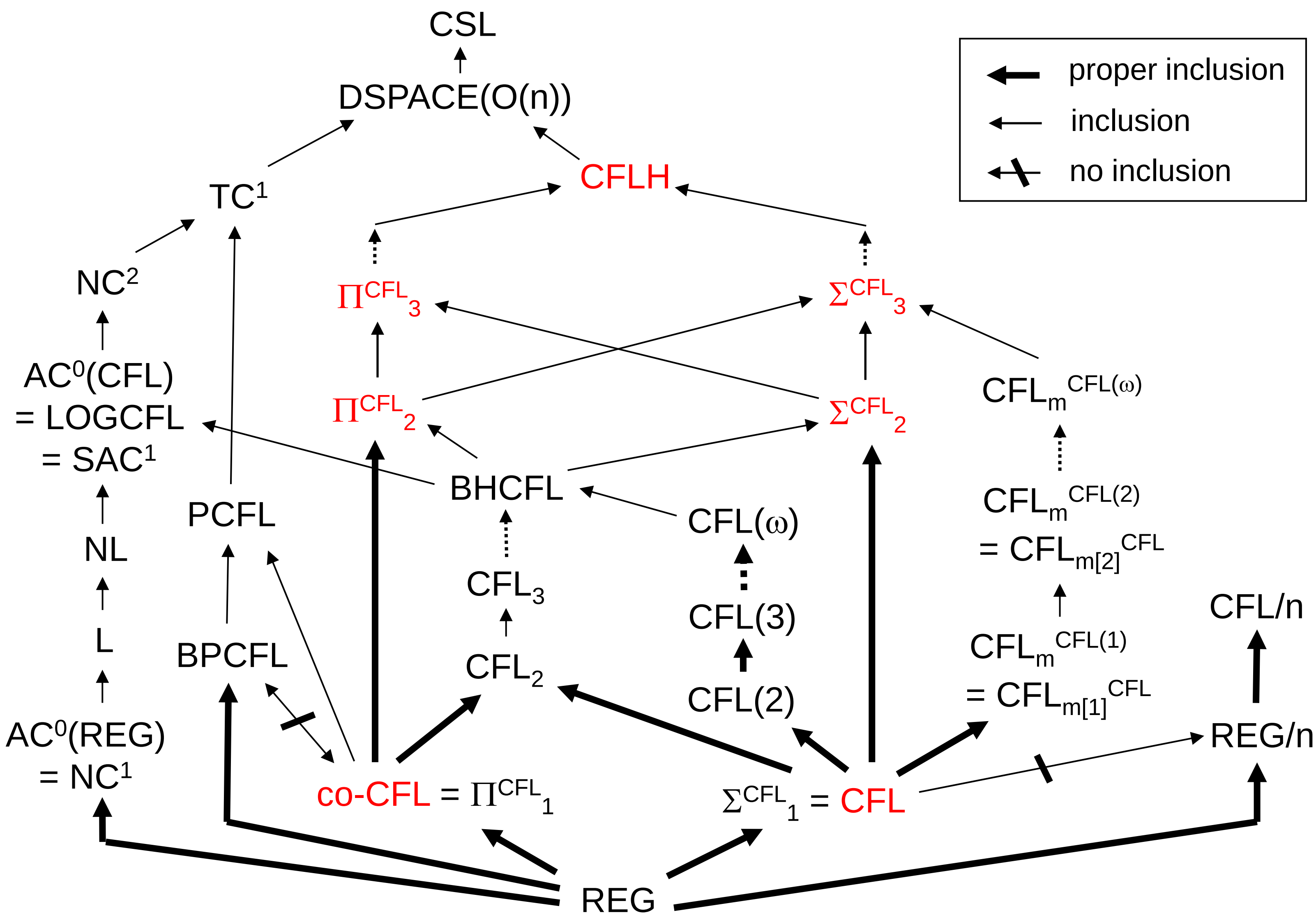
- Some languages are nicely placed in the CFL hierarchy.

- $Dup_2 = \{ xx \mid x \in \{0,1\}^* \}$ (duplication)
- $Dup_3 = \{ xxx \mid x \in \{0,1\}^* \}$ (3 copies)
- $Sq = \{ 0^n 1^k \mid k = n^2, n \in \mathbb{N} \}$ (squared length)

- $Prim = \{ 0^n \mid n \text{ is a prime} \}$ (prime length)



Inclusion Relations among Language Families



Structural Properties of CFLH

- The CFL hierarchy has the following properties.
- Theorem (upward collapse properties): [Yamakami (2014)]

  Let $k \geq 2$

  1. $\Sigma_k^{CFL} = \Sigma_{k+1}^{CFL} \Leftrightarrow \text{CFLH} = \Sigma_k^{CFL}$.
  2. $\Sigma_k^{CFL} = \Pi_k^{CFL} \Leftrightarrow \text{BH}\Sigma_k^{CFL} = \Sigma_k^{CFL}$.
  3. $\Sigma_k^{CFL} = \Pi_k^{CFL} \Leftrightarrow \Sigma_k^{CFL} = \Sigma_{k+1}^{CFL}$.

- Similarly to CFL_e and BHCFL, Yamakami (2014) defined:
- Boolean Hierarchy over Σ_k^{CFL}
  - $\Sigma_{k,e}^{CFL}$ = e-th level of the Boolean hierarchy over Σ_k^{CFL}.
  - $\text{BH}\Sigma_k^{CFL}$ = union of $\Sigma_{k,e}^{CFL}$ for all $e \in \mathbb{N}$.

Relationships to the Polynomial Hierarchy

- The CFL hierarchy has a close connection to the polynomial hierarchy.

- Theorem: [Yamakami (2014)]

  Let $k \geq 1$. If $\Sigma_{k+1}^{CFL} = \Sigma_{k+2}^{CFL}$, then $\Sigma_k^p = \Sigma_{k+1}^p$.

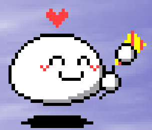
- In other words, if the polynomial hierarchy is truly an infinite hierarchy, then so is the CFL hierarchy.

❑ Proof Idea:

- It suffices to show that each Σ_{k+1}^{CFL} contains a language that is p-T-complete for Σ_k^p.

Open problems

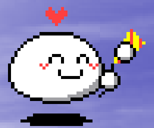
- Prove that $\Sigma_{k+1}^{CFL} \neq \Sigma_{k+2}^{CFL}$ for all $k \geq 1$.
  - Note that proving that $\Sigma_{k+1}^{CFL} = \Sigma_{k+2}^{CFL}$ is much more difficult because this implies $\Sigma_k^P = \Sigma_{k+1}^P$.
- Find interesting relativized worlds regarding Σ_k^{CFL}.
  - E.g., $BPCFL^A \not\subseteq \Sigma_2^{CFL}(A) \cap \Pi_2^{CFL}(A)$ for some A.
- Prove the REG-dissectability of Σ_{k+1}^{CFL} .
  - The dissectability will be discussed in Week 5.



Thank you for listening

Q & A

I’m happy to take your question!



END