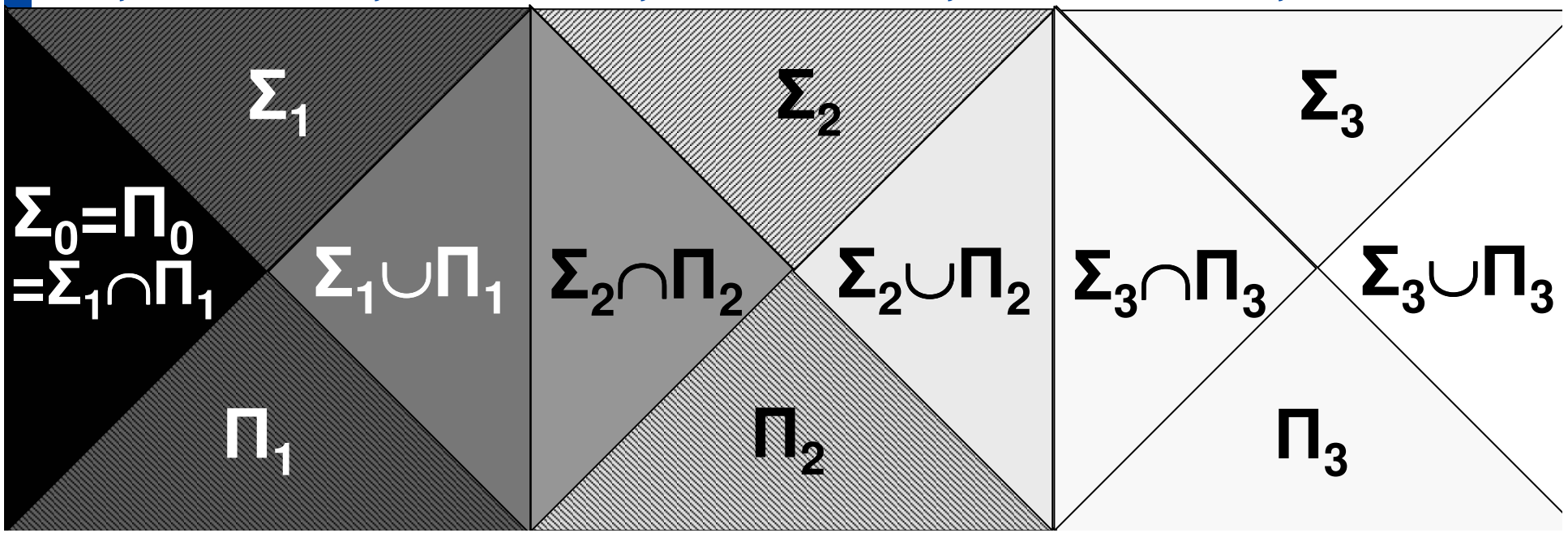
Go Figure (I)

HEINZ NIXDORF INSTITUTE
University of Paderborn
Algorithms and Complexity

$\Sigma_d = \{ \text{“}\exists\forall\exists \ldots R\text{“} \mid R \text{ decidable} \}$
($d \in \mathbb{N}$ alternating quantifiers)
$\Pi_d = \{ \text{“}\forall\exists\forall \ldots R\text{“} \mid R \text{ decidable} \}$

$\emptyset,\ H=\emptyset\text{‘},\ H^H=\emptyset\text{‘‘},\ H^{H^H}=\emptyset\text{‘‘‘},\ \emptyset\text{‘‘‘‘}=\emptyset^{(4)},\ \emptyset^{(5)} \ldots$



Go Figure (II)

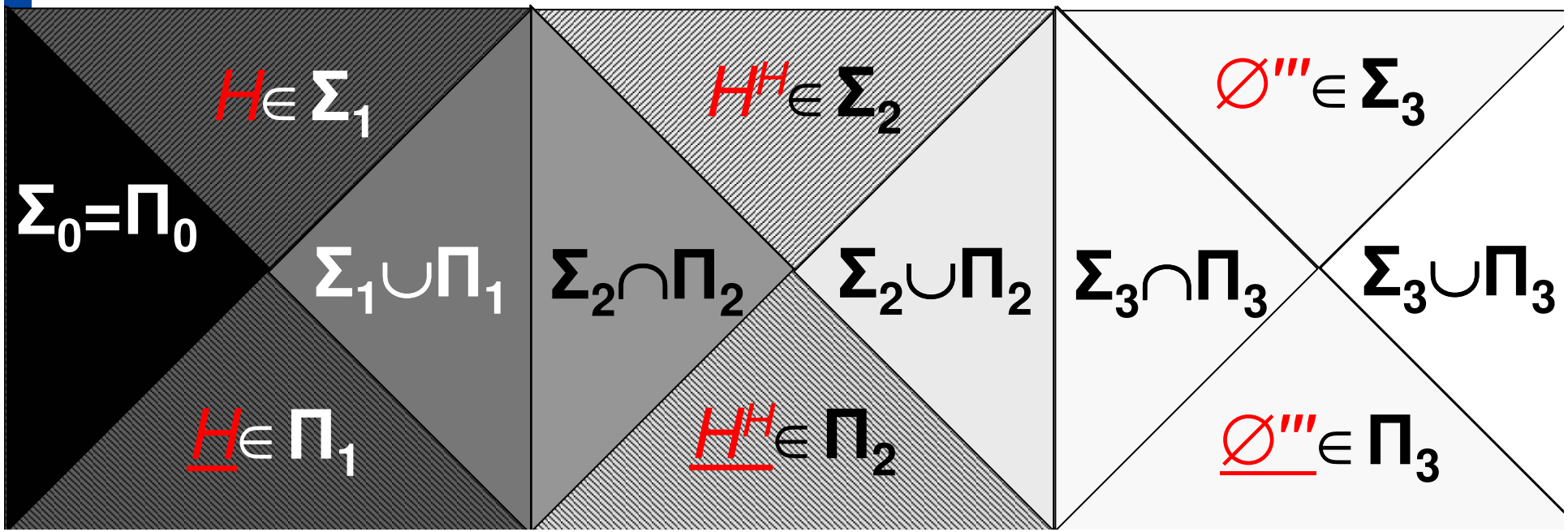
HEINZ NIXDORF INSTITUTE
University of Paderborn
Algorithms and Complexity

Theorem: For $L \subseteq \{0,1\}^*$ it holds:

a) L semi-decidable relative to $H \Leftrightarrow L \in \Sigma_2$

b) L decidable relative to $H \Leftrightarrow L \in \Sigma_2 \cap \Pi_2$

c) L semi-decidable relative to $\emptyset^{(d)} \Leftrightarrow L \in \Sigma_{d+1}$



Purpose of Complexity Theory

HEINZ NIXDORF INSTITUTE
University of Paderborn
Algorithms and Complexity

Why study complexity classes

like **P, NP, PSPACE, EXP** ?

Why classes of uncomputability?

CLASSIFICATION:

Putting some order into the variety of computational problems!

- e.g. Halting Problem: undecidable, $\in \Sigma_1 \setminus \Pi_1$;
- Totality: even more undecidable, $\in \Pi_2 \setminus \Sigma_2$
- Hamilton Cycle: **NP**-complete

But are these classes really different???

Time Hierarchy Theorem

HEINZ NIXDORF INSTITUTE
University of Paderborn
Algorithms and Complexity

Definition: For function $t:\mathbb{N}\to\mathbb{N}$, $\mathbf{DTIME}(t(n)) :=$
$\{ L \subseteq \{0,1\}^* : L \text{ decidable in time } O(t(n)) \}$.

Examples: a) $\mathbf{P} = \bigcup_k \mathbf{DTIME}(n^k)$
b) $\mathbf{EXP} = \bigcup_k \mathbf{DTIME}(2^{n^k})$

$\mathbf{NP} \ni \{ \underline{x} \mid \exists \underline{y} \in \{0,1\}^{p(|\underline{x}|)} : \langle \underline{x},\underline{y} \rangle \in R \}$,
$R \subseteq \{0,1\}^*$ decidable in polynomial time.
Note $\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{EXP}$.

Theorem: $\mathbf{P} \subset \mathbf{EXP}$.

Proof of the Time Hierarchy Theorem

HEINZ NIXDORF INSTITUTE
University of Paderborn
Algorithms and Complexity

Theorem: There exists a problem $L \in \mathbf{EXP} \setminus \mathbf{P}$.

For each $t(n) \geq n$, $\mathbf{DTIME}(t^4(n)) \setminus \mathbf{DTIME}(t(n)) \neq \emptyset$

Reminder (Exercise): Extended universal progr. decides $\{ \langle P, \underline{x}, T \rangle : P \text{ on } \underline{x} \text{ terminates in} \leq T \text{ steps} \}$ in time $O(T^2 + T \cdot (|\underline{P}| + |\underline{x}|)^2)$.

Proof of the Theorem: Consider the language D_t: decidable in time $O(n^2 \cdot t^2(n) + n^3 \cdot t(n)) \leq O(t^4(n))$. ✓

$D_t := \{ \langle P \rangle \#0^k : \text{program } P \text{ does not accept } \langle P \rangle \#0^k \text{ within } t(n) \cdot n \text{ steps}, n := |\langle P \rangle| + k \}$

$D = \{ \langle P \rangle \mid \text{program } P \text{ does not accept input } \langle P \rangle \}$

Proof of the Time Hierarchy Theorem

HEINZ NIXDORF INSTITUTE
University of Paderborn
Algorithms and Complexity

For each $t(n) \geq n$, $\mathbf{DTIME}(t^4(n)) \setminus \mathbf{DTIME}(t(n)) \neq \emptyset$.

Proof: Language D_t is decidable in time $O(t^4(n))$.

Suppose decidable in time $\leq c \cdot t(n)$ by program P.

Choose k such that $c \cdot t(n) \leq n \cdot t(n)$, $n := |\langle P \rangle| + k$.

Case $|\langle P \rangle| \# 0^k \in D_t$:

$\Rightarrow$ P accepts $|\langle P \rangle| \# 0^k$ in time $\leq c \cdot t(n) \leq n \cdot t(n)$ ↯

Case $|\langle P \rangle| \# 0^k \notin D_t$:

$\Rightarrow$ P rejects $|\langle P \rangle| \# 0^k$ in time $\leq n \cdot t(n)$ ↯

$D_t := \{ \langle P \rangle \# 0^k$: program P does not accept $\langle P \rangle \# 0^k$ within $t(n) \cdot n$ steps, $n := |\langle P \rangle| + k \}$

Reconsideration

HEINZ NIXDORF INSTITUTE
University of Paderborn
Algorithms and Complexity

For each $t(n) \geq n$, $\mathbf{DTIME}(t^4(n)) \supset \mathbf{DTIME}(t(n))$

even $\mathbf{DTIME}(t(n) \cdot \log t(n)) \supset \mathbf{DTIME}(t(n))$

There is a problem solvable in superpolynomial but not in polynomial time.

Which problem?

SAT ?

$\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{EXP}$

at least one inclusion strict:

which one(s) ? $1,000,000

Advertisement:
Space-Bounded Computation

HEINZ NIXDORF INSTITUTE
University of Paderborn
Algorithms and Complexity

So-far focus on *fast* algorithms, i.e. with low running time

Captured in complexity classes like **P**, **NP**, **EXP**

- and open questions: **P** vs. **NP** vs. **coNP** vs. **EXP**

Often at the expense of memory: Dynamic Programming.

How about *frugal* algorithms: low memory consumption?

Captured in complexity classes like **L**, **NL**, **PSPACE**

- with interesting connections to *parallel* complexity
- and surprising results: $\mathbf{NL=coNL\subseteq L^2\subset PSPACE}$
- and $\mathbf{DTIME}(t(n)) \subseteq \mathbf{DSPACE}(t(n)/\log t(n))$