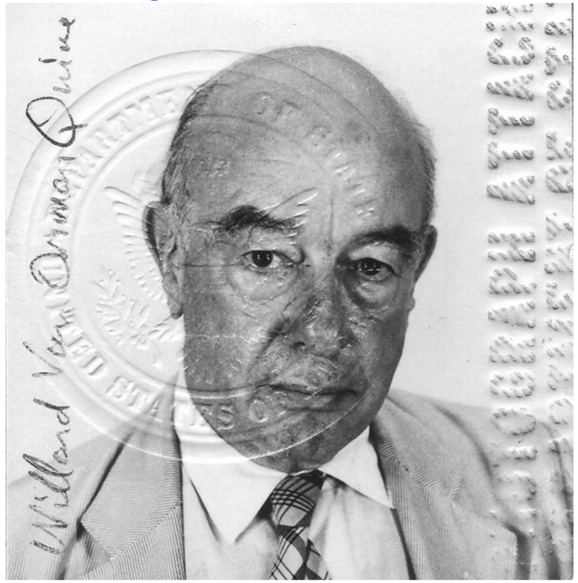
Bonus Slide: Knowing your own Source Code

HEINZ NIXDORF INSTITUTE
University of Paderborn
Algorithms and Complexity

SMN Theorem: Given program P, can construct Q that ignores its input and behaves like P on input $\langle P\rangle$.

Q knows source code of P,
but not source code of Q.

Quine Theorem: To any program P of two inputs $(\underline{x},\underline{y})$, there is a program Q on $\underline{x}$ that behaves like P on $(\underline{x},\langle Q\rangle)$.



Example: P prints $\underline{y} \;\Rightarrow\; Q$ prints $\langle Q\rangle$!

```
->++>+++>+>+>++>>+>+>+++>>+>+>++>+++>+++>+>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>+>+>++>
>>+++>>>>>+++>+>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>+++>>>>>>>>>++>+++>+++>+>>+++>>+++>+>+++>+>++>
+++>>>>+>+>+>+>++>+++>+>+>>+++>>>>>>>>>+>+>>>>+>+>++>+++>+++>+>>+++>+++>+>+++>+>++>+++
>++>>+>+>++>+++>+>>+++>>>+++>>>++>+++>+>>+++>+>>+++>+>++>+++>>+++>+>++>>+++>>>+>+++>+>++>>+[[>
>+[>]+>+[<]<-]>>[>]<+<+++[<]<<+]>>+[>]+++[++++++++++>++[-<++++++++++++++++>]<.<-<]
```

Holds for every Turing-complete programming system!!

Fun: Write a **Quine** in your favorite program. language!

Complexity Theory

HEINZ NIXDORF INSTITUTE
University of Paderborn
Algorithms and Complexity

- Problem instance I:
  Is CNF formula $(x \vee y \vee z) \wedge (\neg x \vee \neg y \vee \neg z)$ satisfiable?
- Problem P: <u>in</u>finite collection of instances.
- Algorithmic solution of P: <u>provably</u> solves each $I \in P$
  → model of computation
- Computational complexity $\mathcal{C}_R(P)$:
  minimum amount of resource R consumed by solution
  – worst-case, asymptotically for instance size $|I|=:n \to \infty$
  – resources R: e.g. running time, memory, #processors
- upper complexity bound $\mathcal{C}_R(P) \leq f(n)$:
  <u>exists</u> algorithm solving P <u>provably</u> using at most $f(n)$ R
- lower complexity bound $\mathcal{C}_R(P) \geq g(n)$:
  <u>every</u> algorithm solving P <u>provably</u> uses at least $g(n)$ R

Lower Bounds Techniques

Upper bound: design and analyze an algorithm

$\exists$algorithm A $\forall n$ $\forall$instances I of size $|I|=n$:

A solves I and uses $\leq f(n)$ of R

Lower bound:

$\forall$algorithms A solving P $\forall n$

$\exists$instance I, $|I|=n$: A uses $\geq g(n)$ of R

Proof techniques: a) Diagonalization

b) Show problem P complete for a class (eg. **NP**

– ie. reduce SAT to P in poly.time: algorithm design!)

c) Genius still sought

Morgenstern

– in restricted models: Ben-Or, Razborov, Ukkonen...

Advertisement I:
Space Complexity

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: $\mathbf{NL}=\mathbf{coNL}\subseteq\mathbf{L}^2\subset\mathbf{PSPACE}$
- and $\mathbf{DTIME}(t(n)) \subseteq \mathbf{DSPACE}(t(n)/\log t(n))$

Advertisement II: Own Research

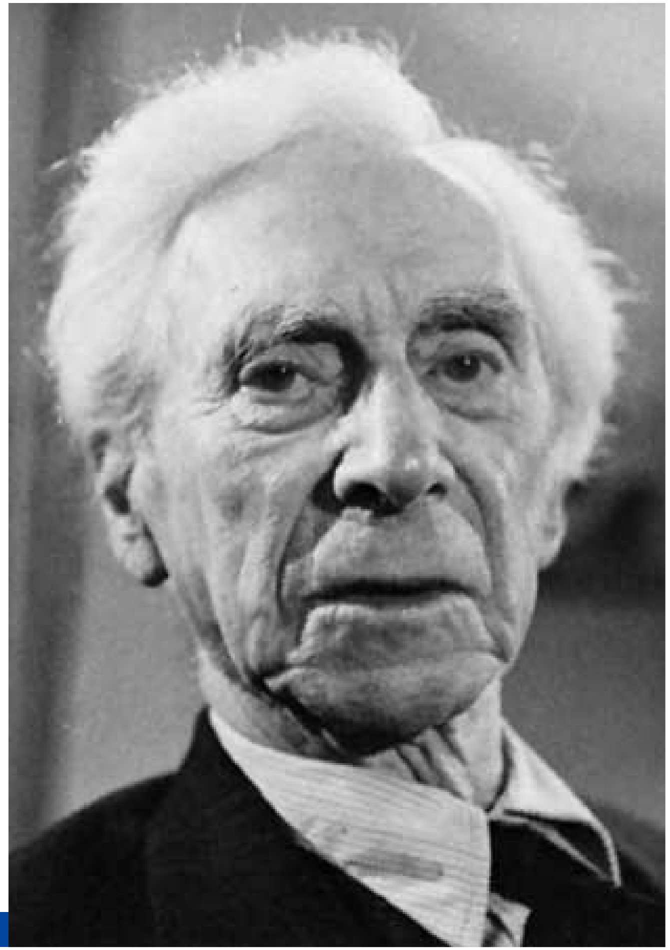
- Computability over real numbers
  - Computing $y \in \mathbb{R}$ means to output an infinite sequence of integer fractions $|q_n - y| \leq 2^{-n}$.

    $y = f(x)$ upon input of a sequence $|p_m - x| \leq 2^{-m}$.
  - Computing $x \to f(x) = y$ means to apply a finite sequence of exact arithmetical operations and comparisons.
- Physical Foundations of Computing and
- Complexity Theory of Computational Physics
- Computational Geometry in Computer Graphics

How Science Works…

Bertrand Russel (1872-1970):

„Darin besteht das Wesen der Wissenschaft:
Zuerst denkt man an etwas, das wahr sein könnte.
Dann sieht man nach, ob es der Fall ist
und im allgemeinen ist es nicht der Fall.“



Extension:

Typically one comes up with an intuitive conjecture, but fails to prove or disprove!

This has to be dealt with,
some way or another…

The End of CS493

HEINZ NIXDORF INSTITUTE
University of Paderborn
Algorithms and Complexity

Thanks a lot
for your
attention and
diligence!