Monte Carlo Computability

Rupert Hölzl
Bundeswehr University Munich

Joint work with Vasco Brattka and Rutger Kuyper

1 Motivation

Motivation

1. Ziegler; Brattka, de Brecht, and Pauly; as well as Brattka, Gherardi and Hölzl used the Weihrauch lattice to study computability with advice on *infinite* objects.

Motivation

computability	acceptance criterion	KL characterisation
nondeterministic	some path works	$\leq_W$ WKL
Las Vegas	$p > 0$; error detection	$\leq_W$ WWKL

1. Ziegler; Brattka, de Brecht, and Pauly; as well as Brattka, Gherardi and Hölzl used the Weihrauch lattice to study computability with advice on *infinite* objects.

Motivation

computability	acceptance criterion	KL characterisation
nondeterministic	some path works	$\leq_W$ WKL
Las Vegas	$p > 0$; error detection	$\leq_W$ WWKL

1. Ziegler; Brattka, de Brecht, and Pauly; as well as Brattka, Gherardi and Hölzl used the Weihrauch lattice to study computability with advice on *infinite* objects.
2. Continuing this line of work, we (somewhat vaguely) asked:

Motivation

computability	acceptance criterion	KL characterisation
nondeterministic	some path works	$\leq_W$ WKL
Las Vegas	$p > 0$; error detection	$\leq_W$ WWKL
	— ? —	

1. Ziegler; Brattka, de Brecht, and Pauly; as well as Brattka, Gherardi and Hölzl used the Weihrauch lattice to study computability with advice on *infinite* objects.
2. Continuing this line of work, we (somewhat vaguely) asked:
   - What happens without error detection (in some sense)?

Motivation

computability	acceptance criterion	KL characterisation
nondeterministic	some path works	$\leq_W$ WKL
Las Vegas	$p > 0$; error detection	$\leq_W$ WWKL
	— ? —	— ? —

1. Ziegler; Brattka, de Brecht, and Pauly; as well as Brattka, Gherardi and Hölzl used the Weihrauch lattice to study computability with advice on *infinite* objects.
2. Continuing this line of work, we (somewhat vaguely) asked:
   - What happens without error detection (in some sense)?
   - What happens with higher versions of WWKL?

Motivation

computability	acceptance criterion	KL characterisation
nondeterministic	some path works	$\leq_W$ WKL
Las Vegas	$p > 0$; error detection	$\leq_W$ WWKL
	— ? —	— ? —

1. Ziegler; Brattka, de Brecht, and Pauly; as well as Brattka, Gherardi and Hölzl used the Weihrauch lattice to study computability with advice on *infinite* objects.
2. Continuing this line of work, we (somewhat vaguely) asked:
   - What happens without error detection (in some sense)?
   - What happens with higher versions of WWKL?
3. The resulting computability notion is the topic of this talk.

Motivation

computability	acceptance criterion	KL characterisation
nondeterministic	some path works	$\leq_W$ WKL
Las Vegas	$p > 0$; error detection	$\leq_W$ WWKL
Monte Carlo	$p > 0$	$\leq_W \mathrm{WWKL}' \times \mathrm{C}'_{\mathbb{N}}$

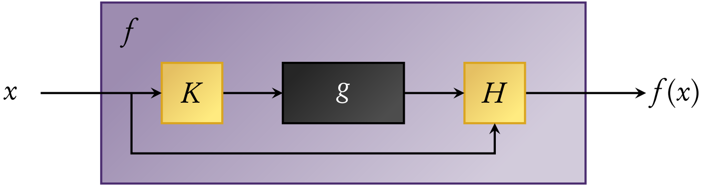
1. Ziegler; Brattka, de Brecht, and Pauly; as well as Brattka, Gherardi and Hölzl used the Weihrauch lattice to study computability with advice on *infinite* objects.
2. Continuing this line of work, we (somewhat vaguely) asked:
   - What happens without error detection (in some sense)?
   - What happens with higher versions of WWKL?
3. The resulting computability notion is the topic of this talk.

2 Preliminaries

Weihrauch degrees

1. We look at mathematical tasks where a problem instance is given to a black box, and the black box has to return a solution.
2. For one such instance there may be multiple solutions. This is modeled by *multi-valued* functions mapping to *sets* of solutions.
3. The problem and solution descriptions may be infinite objects.
4. **Example.** Given a bounded sequence of rationals, the black box has to produce an accumulation point.
5. **Reducibility.** Assume we have a black box solving a certain problem g. Can we use it to solve another problem f?

(Weak) Weihrauch reducibility $f \leq_W g$

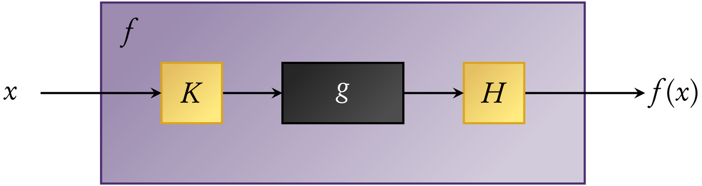


1. K and H are Turing functionals.*
2. The decoding procedure H has access to the original input.
3. That is, the computed function is

$$f : x \mapsto H(g(K(x)), x).$$

*modulo representations

Strong Weihrauch reducibility $f \leq_{sW} g$



1. K and H are Turing functionals.*
2. The decoding procedure H has *no* access to the original input.
3. That is, the computed function is

$$f : x \mapsto H(g(K(x))).$$

*modulo representations

Choice

1. **Closed Choice** C_X**:** Get closed $A \subseteq X$, need to output an $x \in A$.
2. **Positive Closed Choice** PC_X**:** C_X restricted to $\{A : \mu_X(A) > 0\}$.
3. **Weak Kőnig's Lemma:** $\mathrm{WKL} = \mathrm{C}_{2^{\mathbb{N}}}$.
4. **Weak Weak Kőnig's Lemma:** $\mathrm{WWKL} = \mathrm{PC}_{2^{\mathbb{N}}}$.
5. **Positive** G_δ **Choice** $\mathbf{\Pi}^0_2\mathrm{PC}_X$**:** Like PC_X, but for G_δ sets.

Algebraic operations

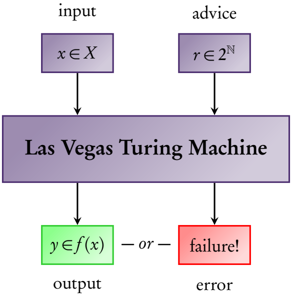
1. **Product** $A \times B$**.** Get instances of A and B, need to solve both.
2. **Compositional product** $A * B$**.** The degree of the hardest problems that can be solved by first (weakly) reducing to B, then making some computation, then (weakly) reducing to A.
3. **Jump** A'**.** The same as A, but instead of an instance of A we are only given a sequence converging to such an instance.

Nondeterministic computation

1. **Definition (Ziegler).*** f is called *nondeterministically computable* if there is a nondeterministic one-way Turing machine that on input x has at least one computation that outputs a $y \in f(x)$, and only such computations are allowed to run indefinitely.
2. **Observation (Brattka, de Brecht, Pauly).** f is nondeterministically computable if and only if $f \leq_W \mathrm{WKL}$.

*modulo representations

Las Vegas computability

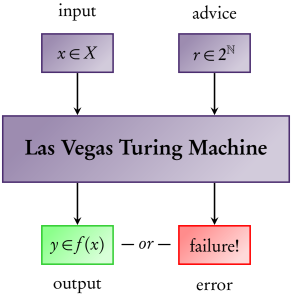


1 Definition (Brattka, Gherardi, Hölzl).* f is called *Las Vegas computable* if there exists a one-way Turing machine which on input x and advice r computes f in the following way:

- if it produces an infinite output y then $y \in f(x)$;
- otherwise it must signal failure after a finite number of steps;
- for all x's the r's not causing failure have positive measure.

*modulo representations

Las Vegas computability



1. **Definition (Brattka, Gherardi, Hölzl).*** f is called *Las Vegas computable* if there exists a one-way Turing machine which on input x and advice r computes f in the following way:
   - if it produces an infinite output y then $y \in f(x)$;
   - otherwise it must signal failure after a finite number of steps;
   - for all x's the r's not causing failure have positive measure.
2. **Observation (Brattka, Gherardi, Hölzl).**
   f is Las Vegas computable if and only if $f \leq_W$ WWKL.

*modulo representations

3 Monte Carlo Computability

Definition

1 **Definition.*** $f\colon \subseteq X \rightrightarrows Y$ is *Monte Carlo computable* if there are

- computable $F_1\colon \subseteq 2^{\mathbb{N}} \times 2^{\mathbb{N}} \to 2^{\mathbb{N}}$,
- limit-comp. $F_2\colon \subseteq 2^{\mathbb{N}} \times 2^{\mathbb{N}} \to 2^{\mathbb{N}}$ with $\mathrm{dom}(f) \times 2^{\mathbb{N}} \subseteq \mathrm{dom}(F_2)$,
- and for each $x \in \mathrm{dom}(f)$ it holds that
  - for $R_x := \{r \in 2^{\mathbb{N}} : F_2(x,r) = 0^{\mathbb{N}}\}$ we have $\mu_{2^{\mathbb{N}}}(R_x) > 0$,
  - $F_1(x,r) \in f(x)$ for all $r \in R_x$.

2 **Remark.** It's a generalisation of Las Vegas computability where F_2 need not be computable, but only limit-computable.

*modulo representations, simplified

Algebraic characterisation

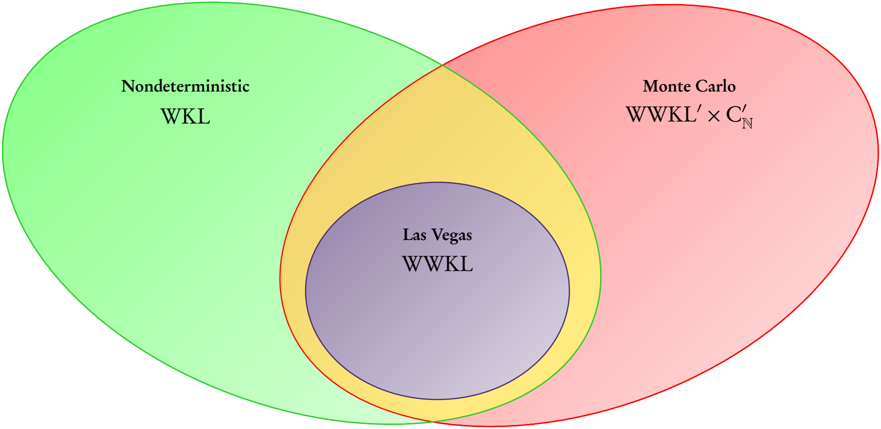
1. **Theorem.** f is Monte Carlo computable $\Leftrightarrow f \leq_W \mathbf{\Pi}^0_2\mathrm{PC}_{2^{\mathbb{N}}}$.
2. **Theorem (Brattka, Gherardi, Hölzl, Nobrega, Pauly).**

$$\mathbf{\Pi}^0_2\mathrm{PC}_{2^{\mathbb{N}}} \equiv_{sW} \mathrm{PC}'_{\mathbb{R}} \equiv_{sW} \mathrm{WWKL}' \times \mathrm{C}'_{\mathbb{N}}.$$

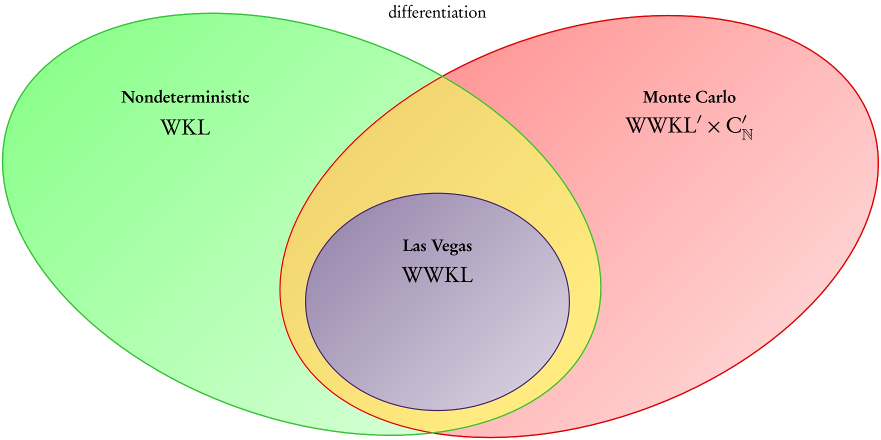
3. **Corollary.** f is Monte Carlo computable
$\Leftrightarrow \quad f \leq_W \mathrm{PC}'_{\mathbb{R}}$
$\Leftrightarrow \quad f \leq_W \mathrm{WWKL}' \times \mathrm{C}'_{\mathbb{N}}$.
4. **Theorem (Bienvenu, Kuyper).** $\mathrm{PC}'_{\mathbb{R}} * \mathrm{PC}'_{\mathbb{R}} \equiv_W \mathrm{PC}'_{\mathbb{R}}$.
5. **Corollary.** Compositions of Monte Carlo computable functions are again Monte Carlo computable.

4 Example tasks

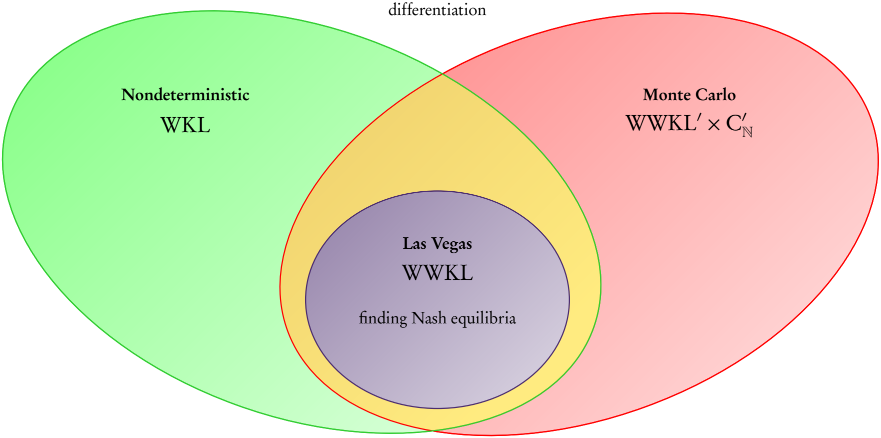
Example tasks



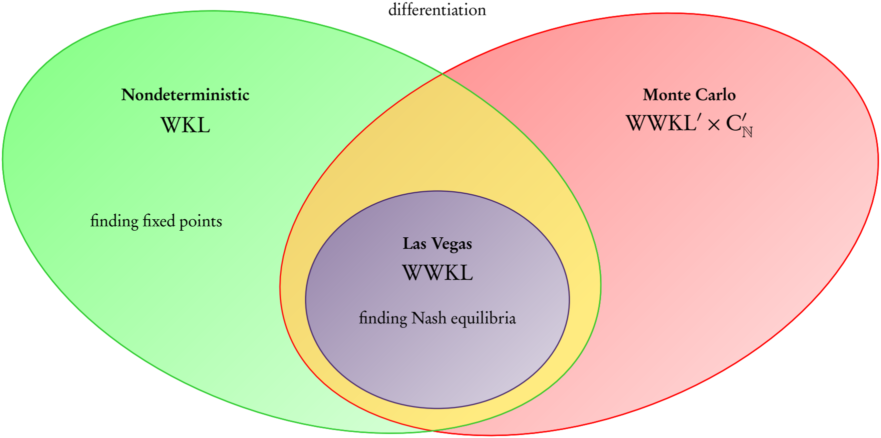
Example tasks



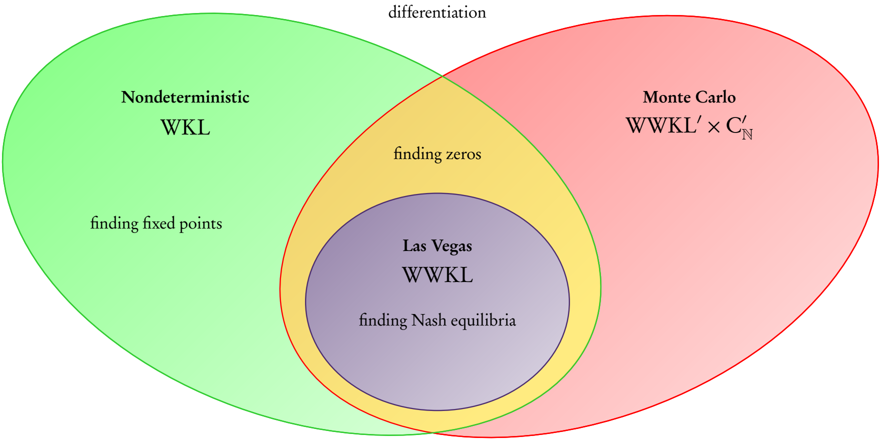
Example tasks



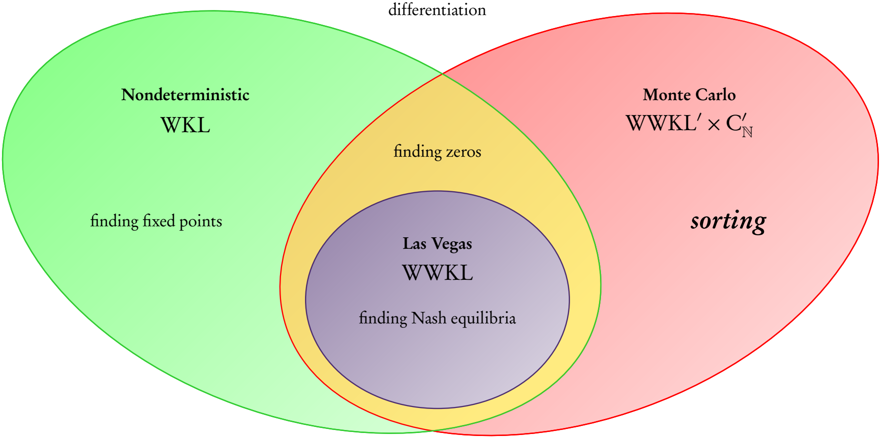
Example tasks



Example tasks

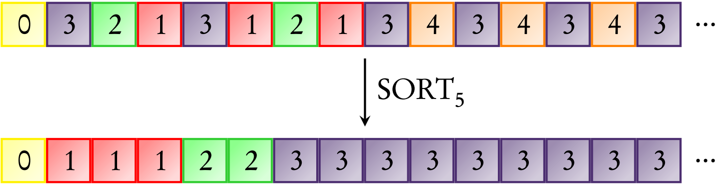


Example tasks



Sorting

Definition (Neumann, Pauly; generalised from $n = 2$).



$\mathrm{SORT}_n : \{0, 1, \ldots, n-1\}^{\mathbb{N}} \to \{0, 1, \ldots, n-1\}^{\mathbb{N}}$ is the map

$$x \mapsto 0^{k_0} 1^{k_1} 2^{k_2} \ldots (m-1)^{k_{m-1}} m^{\mathbb{N}}$$

where

- $m < n$ is the smallest number appearing infinitely often in x,
- each $i < m$ appears exactly k_i times in x.

Sorting is Monte Carlo computable

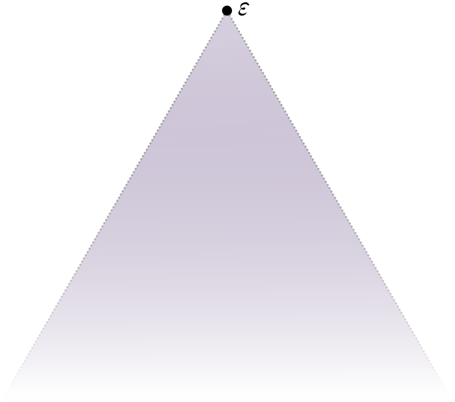
1 **Proposition.** $\mathrm{SORT}_n \leq_{\mathrm{sW}} \mathrm{WWKL}'$.

Sorting is Monte Carlo computable

1. **Proposition.** $\mathrm{SORT}_n \leq_{\mathrm{sW}} \mathrm{WWKL}'$.
2. **Proof.** Inspect the first i symbols of the input to define T_i:

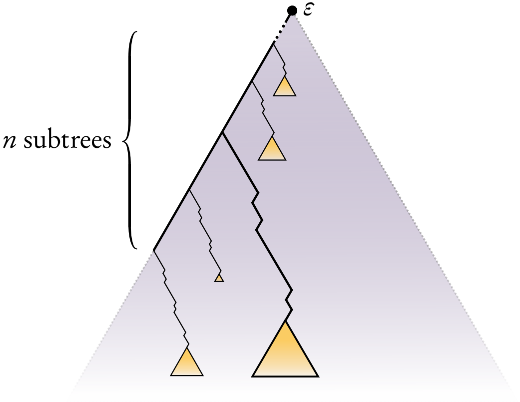
Sorting is Monte Carlo computable

1. **Proposition.** $\mathrm{SORT}_n \leq_{\mathrm{sW}} \mathrm{WWKL}'$.
2. **Proof.** Inspect the first i symbols of the input to define T_i:



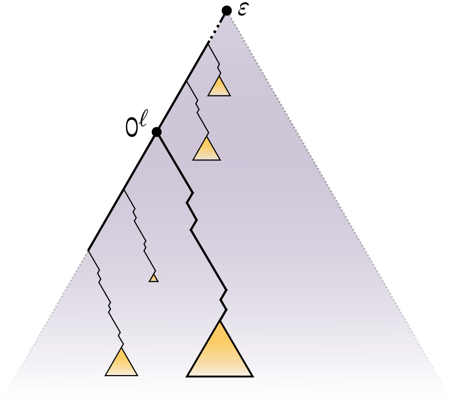
Sorting is Monte Carlo computable

1. **Proposition.** $\mathrm{SORT}_n \leq_{\mathrm{sW}} \mathrm{WWKL}'$.
2. **Proof.** Inspect the first i symbols of the input to define T_i:



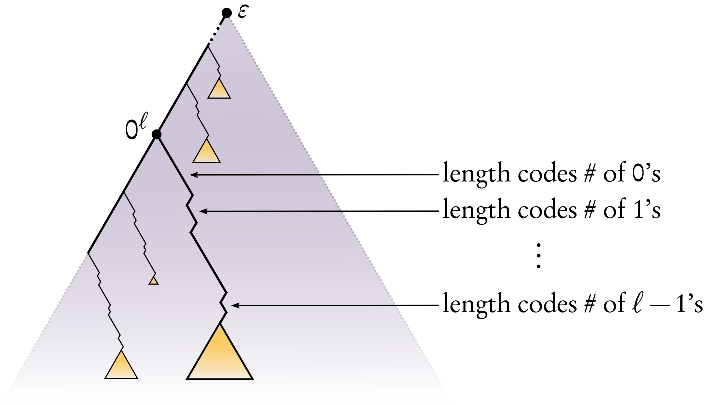
Sorting is Monte Carlo computable

1. **Proposition.** $\mathrm{SORT}_n \leq_{\mathrm{sW}} \mathrm{WWKL}'$.
2. **Proof.** Inspect the first i symbols of the input to define T_i:



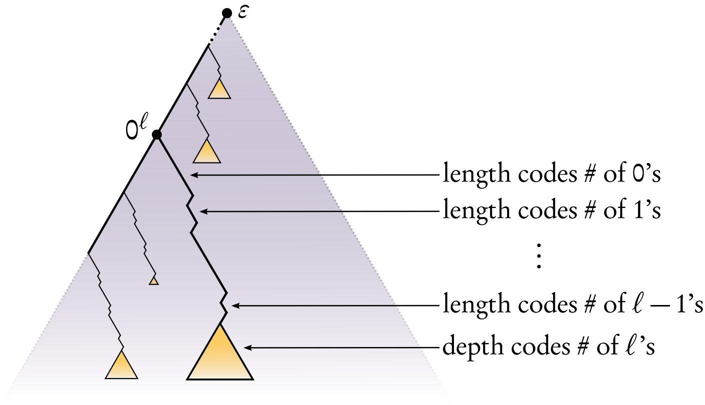
Sorting is Monte Carlo computable

1. **Proposition.** $\mathrm{SORT}_n \leq_{\mathrm{sW}} \mathrm{WWKL}'$.
2. **Proof.** Inspect the first i symbols of the input to define T_i:



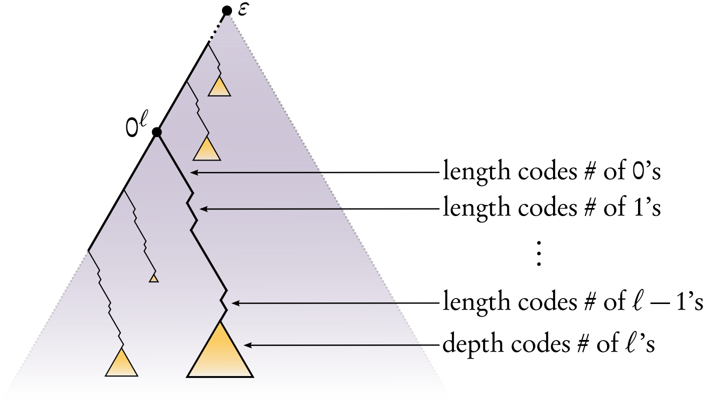
Sorting is Monte Carlo computable

1. **Proposition.** $\mathrm{SORT}_n \leq_{\mathrm{sW}} \mathrm{WWKL}'$.
2. **Proof.** Inspect the first i symbols of the input to define T_i:



Sorting is Monte Carlo computable

1. **Proposition.** $\mathrm{SORT}_n \leq_{\mathrm{sW}} \mathrm{WWKL}'$.
2. **Proof.** Inspect the first i symbols of the input to define T_i:



As $i \to \infty$, more symbols are found; so the lengths may increase.

Sorting is Monte Carlo computable

1. Let m be the smallest number infinitely often in the input.

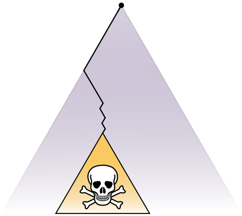
Sorting is Monte Carlo computable

1. Let m be the smallest number infinitely often in the input.
2. Then there are three possible outcomes for the ℓ-th subtree:

$\ell < m$ $\ell > m$ $\ell = m$

Sorting is Monte Carlo computable

1. Let m be the smallest number infinitely often in the input.
2. Then there are three possible outcomes for the ℓ-th subtree:



$\ell < m$ $\qquad$ $\ell > m$ $\qquad$ $\ell = m$

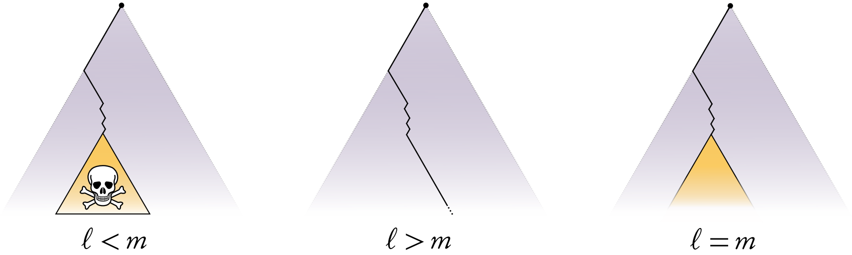
Sorting is Monte Carlo computable

1. Let m be the smallest number infinitely often in the input.
2. Then there are three possible outcomes for the ℓ-th subtree:

$\ell < m$ $\ell > m$ $\ell = m$

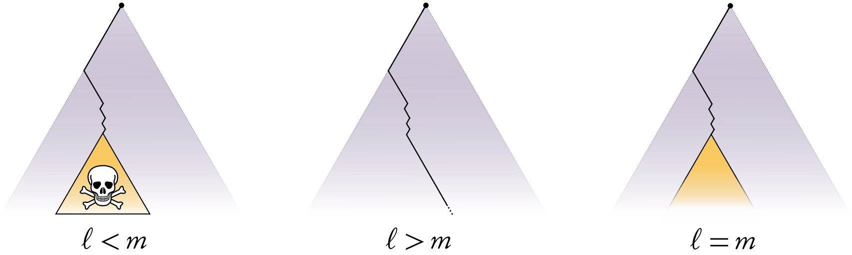
Sorting is Monte Carlo computable

1. Let m be the smallest number infinitely often in the input.
2. Then there are three possible outcomes for the ℓ-th subtree:



Sorting is Monte Carlo computable

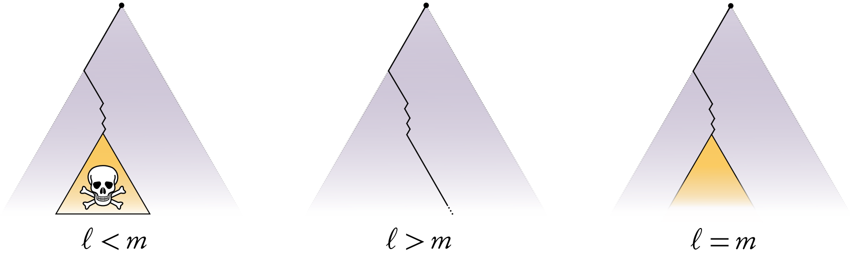
1. Let m be the smallest number infinitely often in the input.
2. Then there are three possible outcomes for the ℓ-th subtree:



3. WWKL$'$ sees $\lim_{i\to\infty} T_i$, and must return an infinite path from it.

Sorting is Monte Carlo computable

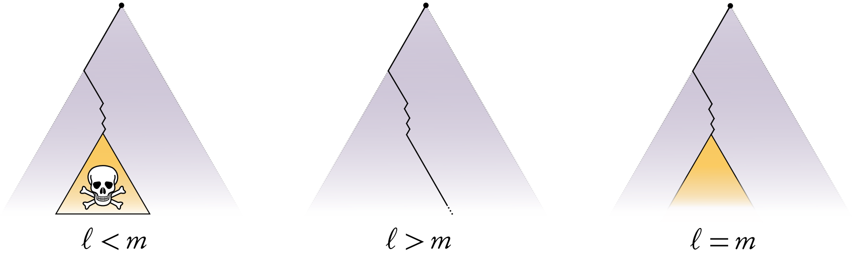
1. Let m be the smallest number infinitely often in the input.
2. Then there are three possible outcomes for the ℓ-th subtree:



3. WWKL' sees $\lim_{i \to \infty} T_i$, and must return an infinite path from it.
4. So we obtain a path from case $\ell > m$ or from case $\ell = m$.

Sorting is Monte Carlo computable

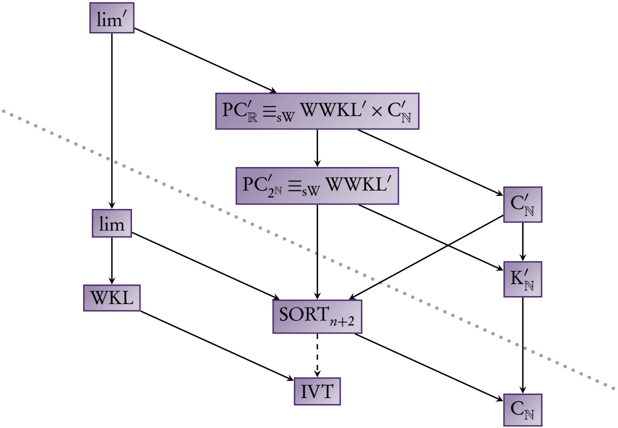
1. Let m be the smallest number infinitely often in the input.
2. Then there are three possible outcomes for the ℓ-th subtree:



3. WWKL' sees $\lim\limits_{i\to\infty} T_i$, and must return an infinite path from it.
4. So we obtain a path from case $\ell > m$ or from case $\ell = m$.
5. Then it is easy to output the sorted sequence we sought. $\square$

5 The unavoidable zoo slide

The unavoidable zoo slide



Thank you for your attention.

Proceedings of STACS 2017