A New Total Injection Betting Strategy

Stijn Vermeeren

School of Mathematics, University of Leeds, Leeds, LS2 9JT, United Kingdom,
mmsv@leeds.ac.uk

Abstract. I give a new method to construct a partial computably random sequence that is not total injection random.

Amongst infinite sequences of 0's and 1's there are some that are very regular, like

$$0101010101010101\ldots$$

whereas others seem quite random, such as a sequence beginning with

$$0110110001011001\ldots$$

might. Yet from a probabilistic point of view, both sequences are equally likely as outcomes of a random experiment, such as a repeated coin toss. Algorithmic randomness can be used to solve this paradox. In algorithmic randomness, computability theory is used to define what is to be considered a regularity in a sequence, and which sequences can be said to be truly random. The exact definition, however, can be done in many different ways that all make sense, without there being one definition that is obviously superior. Hence, the study of the relative strength of the many definitions of randomness has attracted a lot of attention over the years. In this article, I investigate the relative strength of total injection randomness and partial computable randomness. These notions were proven to be incomparable in 2009 by Bienvenu, Hölzl, Kräling and Merkle [1] by studying the initial segment complexity of random sequences of the different types. I will give a new construction of a partial computable random sequence Z that is not total injection random. My construction is more direct, as the total injection betting strategy that succeeds on Z directly approximates the supermartingale used to construct Z.

1 Notation

In this article we will consider infinite sequences of 0's and 1's, which we can treat as elements of 2^ω (i.e. functions $\mathbb{N} = \{0, 1, 2, \cdots\} \to \{0, 1\}$). Such sequences will be represented by capital letters, usually Z.

We will also use finite strings of 0's and 1's, which we can treat as elements of 2^n (i.e. functions $n = \{0, 1, \cdots, n-1\} \to \{0, 1\}$) for some $n \in \mathbb{N}$. We will use lowercase Greek letters (usually σ or τ) to represent strings. If $\sigma \in 2^n$, then $|\sigma| = n$ is called the length of the string. The unique *empty* string of length 0 is

written $\emptyset$, and $\sigma\tau$ is the string of length $|\sigma| + |\tau|$ obtained by concatenating σ and τ. The set of strings of length less than or equal to n is written $2^{\leq n}$. $2^{<\omega}$ is the set of all finite strings of any length.

If f is a function, then $f\restriction_A$ is the restriction of f to the domain A. In particular if $Z \in 2^\omega$, then $Z\restriction_n \in 2^n$ is the initial segment of Z of length n.

We fix a computable pairing function $\langle \cdot, \cdot \rangle$, that is a computable bijection

$$\langle \cdot, \cdot \rangle : \mathbb{N} \times \mathbb{N} \to \mathbb{N}.$$

So $\langle n_0, n_1 \rangle$ is a natural number that encodes the ordered pair (n_0, n_1). We extend this to k-tuples for any k, by defining inductively

$$\langle n_0, n_1, \ldots, n_{k-1} \rangle = \langle n_0, \langle n_1, \ldots, n_{k-1} \rangle \rangle.$$

We assume that the reader is familiar with the basics of computability theory, in particular with how to effectively enumerate all computable functions. We use square brackets $[s]$ to indicate that computations are only done up to that stage s. Familiarity with algorithmic randomness is not strictly required, but will be helpful to understand the proofs and to see the results in context. For background reading on algorithmic randomness, see [7] or [2].

2 Computable Randomness

Computable randomness is one of the most fundamental definitions of randomness. It is appealing because it has a very natural definition, motivated by the fact that we expect the digits of a random sequence to be *unpredictable*. This is captured by requiring that one cannot make an unbounded profit by *betting* on the value of successive digits of the random sequence. More precisely, a *betting strategy* starts with a certain amount of money (capital), and can then place bets on the digits of an infinite sequence. Placing a bet on a digit means assigning a part of the current capital to the outcome 0 and another part to the outcome 1. The money placed on the correct outcome is doubled, but the money placed on the incorrect outcome is lost. A betting strategy starts by placing a bet on the first digit, which is evaluated, and losses are substracted or gains added to the capital. With the new capital, the betting strategy can then place a bet on the second digit, which is allowed to depend on the value of the first digit, which was already evaluated. And so on. A betting strategy *succeeds* on an infinite sequence Z if the betting strategy makes unbounded profits when betting on the digits of Z. A sequence Z is *computably random* if no *computable* betting strategy succeeds on Z.

The best way to treat a betting strategy as a mathematical object, is by using martingales. For any betting strategy, we can consider a *capital function* B that maps each string $\sigma \in 2^{<\omega}$ to the amount of money the betting strategy has after betting on σ. Thus we have a function

$$B : 2^{<\omega} \to \mathbb{R}_{\geq 0}$$

that satisfies

$$B(\sigma) = \frac{B(\sigma 0) + B(\sigma 1)}{2} \tag{1}$$

for all strings σ. Any such function is called a **martingale**. Equation (1) is a fairness condition: the expected value of the new capital must be equal to the initial capital. Any betting strategy, as above, gives rise to a martingale, and every martingale corresponds to a betting strategy. A martingale B succeeds on a sequence Z if

$$\limsup_{n\to\infty} B(Z{\restriction}_n) = \infty. \tag{2}$$

A sequence Z is **computably random** (**CR**) if *no* computable martingale (that is: a martingale which is computable as a function $2^{<\omega} \to \mathbb{R}_{\geq 0}$) succeeds on Z.

Note 1. – We can loosen the requirement (1) to an inequality:

$$B(\sigma) \geq \frac{B(\sigma 0) + B(\sigma 1)}{2}. \tag{3}$$

Any function $B : 2^{<\omega} \to \mathbb{R}_{\geq 0}$ satisfying just this inequality is called a **supermartingale**. Supermartingales do not succeed on more sequences than ordinary martingales, because the only thing they can do more in terms of betting strategies, is to *throw away some money* at each bet.

- *Infinite profits* could be formalized using lim instead of lim sup, without changing the notion of computable randomness. This is justified by a lemma called the *savings trick*. (See e.g. [2], 6.3.8 or [7], 7.1.14)
- We can assume that any (super)martingale has only rational values, where we represent a rational number as a pair of natural numbers (numerator and denominator). Indeed we can effectively approximate each computable (super)martingale $B : 2^{<\omega} \to \mathbb{R}_{\geq 0}$ with a computable (super)martingale $D : 2^{<\omega} \to \mathbb{Q}_{\geq 0}$ such that whenever B succeeds on Z, then also D succeeds on Z. The important advantage of this is that with rational values, we can decide equality. ([9,10]; see also [2], 7.1.2 or [7])

The notion of computable randomness can be strengthened in two directions: by allowing partial computable martingales and by allowing nonmonotonic martingales.

A partial martingale is a partial function $B : 2^{<\omega} \to \mathbb{R}_{\geq 0}$ that still satisfies the *fairness condition*

$$B(\sigma) = \frac{B(\sigma 0) + B(\sigma 1)}{2} \tag{4}$$

when all values involved are defined. Some values of B may be undefined (the betting strategy might be forever undecided on certain bets). But once either value on the right hand side in (4) is defined, both other values in the equation must be defined (we can only bet on the next digit after σ, if we know how much money we have at σ, and once we know how much money we want to make with some outcome, we also know how much money we still have left for the other

outcome). Success for a partial martingale B on a sequence Z is defined exactly like for total martingales:

$$\limsup_{n\to\infty} B(Z{\restriction}_n) = \infty, \tag{5}$$

where now we obviously require that B is defined on all initial segments of Z. A sequence Z is **partial computably random** (**PCR**) if *no* partial computable martingale succeeds on Z.

The second variation on computable randomness is to allow betting on the digits of a sequence in a non-monotonic order. That is, we might be able to bet on the second digit first, and depending on that outcome, we can decide how to bet on the first digit. It is not important to bet on all digits of a sequence; we can ignore certain digits anyway by betting evenly on them, such that our capital stays the same no matter what the digit's value is. The order in which to bet on the digits can be fixed in advance, e.g. by a computable permutation or injection. A (partial) computable permutation/injection betting strategy is then a pair $\langle f, B\rangle$ of a computable permutation/injection f and a (partial) computable martingale B, and $\langle f, B\rangle$ succeeds on Z if

$$\limsup_{n\to\infty} B((Z \circ f){\restriction}_n) = \infty. \tag{6}$$

This gives rise to the notions of **partial/total permutation randomness** (**PPR**/**TPR**) and **partial/total injection randomness** (**PIR**/**TIR**). (Total permutation randomness will in fact turn out to be equivalent with computable randomness.) We can take the nonmonotonicity even further by not fixing the order in advance, and instead allowing the next digit to be on to depend on the outcomes of the previous bets. This gives rise to the notion of **Kolmogorov-Loveland randomness** (**KLR**), of which the partial and total variants are equivalent.

All of the Notes 1 are also valid for all partial and nonmonotonic notions of computable randomness.

We will look at the relative strength of all of these randomness notions in the final section of this article. First we concentrate on constructing a sequence that is PCR but not TIR.

3 Diagonalizing against Partial Computable Betting Strategies

We want to construct a sequence Z that is PCR. That is: no partial computable betting strategy may succeed on Z. Therefore we need to diagonalize against all partial computable martingales with some fixed starting capital, say 1. This is not difficult: if $\{B_0, B_1, \cdots\}$ is an enumeration of all such martingales, then we can add them together into one new total supermartingale L. Values that are undefined can be taken equal to 0 in this addition. In particular, we let

$$V_k(\sigma) = \begin{cases} B_k(\sigma) & \text{if } B_k(\sigma)\downarrow \\ 0 & \text{otherwise,} \end{cases}$$

and

$$L = \sum_{k \in \mathbb{N}} 2^{-k} V_k. \tag{7}$$

Then L is easily seen to be a well-defined supermartingale. Also, if any B_k succeeds on Z, then also V_k and L succeed on Z. So if we choose Z such that L fails on Z, then Z is PCR. This can be done by taking Z to be the left-most non-ascending path on L considered as a tree, i.e. if $Z\restriction_n$ is defined, then we take

$$Z(n) = \begin{cases} 0 \text{ if } L(Z\restriction_n 0) \leq L(Z\restriction_n) \\ 1 \text{ otherwise.} \end{cases}$$

In the second case, as L is a supermartingale, we have $L(Z\restriction_n 1) \leq L(Z\restriction_n)$. Hence $\limsup_{n\to\infty} B(Z\restriction_n) \leq L(\emptyset)$ and L fails on Z, as required.

Now we want Z to be PCR, but at the same time we want there to be a total injection betting strategy that succeeds on Z. This means that we need to make the construction of Z more effective, so that a betting strategy can take advantage of that. First of all we want the enumeration $\{B_0, B_1, \cdots\}$ to be effective. This can be achieved by effectively enumerating all Turing machine programs $\{\phi_0, \phi_1, \phi_2, \cdots\}$, putting $B_k(\emptyset) = 1$, and putting $B_k(\sigma 0) = \phi_k(\sigma 0)$ and $B_k(\sigma 1) = \phi_k(\sigma 1)$ at the first stage when $B_k(\sigma)$ is already defined, $\phi_k(\sigma 0)\downarrow$, $\phi_k(\sigma 1)\downarrow$ and $B_k(\sigma) = \frac{\phi_k(\sigma 0)+\phi_k(\sigma 1)}{2}$. Then $\{B_0, B_1, \cdots\}$ is an effective enumeration of all partial computable martingales with starting capital 1, in the sense that $B_k(\sigma)$ can be effectively computed from k and σ.

Next, we want to be able to compute the values of L more easily, by avoiding infinite sums in (7). This can be done by fixing an increasing sequence of natural numbers (n_k), and by applying betting strategy B_k only from position n_k onwards. Formally, we let

$$V_k(\sigma) = \begin{cases} 1 & \text{if } |\sigma| \leq n_k \\ \frac{B_k(\sigma)}{B_k(\sigma\restriction_{n_k})} & \text{if } |\sigma| > n_k,\ B_k(\sigma)\downarrow \text{ and } B_k(\sigma\restriction_{n_k}) > 0 \\ 0 & \text{otherwise,} \end{cases}$$

and

$$L = \sum_{k \in \mathbb{N}} 2^{-k} V_k$$

like before. Now, to compute any particular value of L, we only need to know the value of B_k for finitely many k. But still, if any B_k succeeds on Z, then also V_k and L succeed on Z. So we can choose Z like before to make Z PCR. The sequence we obtain for certain values of (n_k) will be called $Z^{(n_k)}$.

The techniques for constructing PCR sequences outlined above are well-known. In particular, look at [7] for more explanations and applications.

4 A Total Injection Betting Strategy That Succeeds

4.1 Approximations for L and Z

We claim that, for a suitable choice of (n_k), the PCR sequence $Z^{(n_k)}$ is not TIR. To prove this, we construct a total injection betting strategy that succeeds on

$Z^{(n_k)}$. The sequence (n_k) will not be computable, so the betting strategy will have to guess the values of (n_k). Hence we introduce the following notation:

$$V_{k,n}(\sigma) = \begin{cases} 1 & \text{if } |\sigma| \leq n \\ \frac{B_k(\sigma)}{B_k(\sigma\restriction_n)} & \text{if } |\sigma| > n,\ B_k(\sigma)\downarrow \text{ and } B_k(\sigma\restriction_n) > 0 \\ 0 & \text{otherwise,} \end{cases}$$

for any $k, n \in \mathbb{N}$, and

$$L_i^{\langle n_0,\ldots,n_{i-1}\rangle} = \sum_{k=0}^{i-1} 2^{-k} V_{k,n_k}.$$

Also, we have to make all computations in our strategy finite, so we need to approximate as follows:

$$V_{k,n}[s](\sigma) = \begin{cases} 1 & \text{if } |\sigma| \leq n \\ \frac{B_k(\sigma)}{B_k(\sigma\restriction_n)} & \text{if } |\sigma| > n,\ B_k[s](\sigma)\downarrow, \text{ and } B_k(\sigma\restriction_n) > 0 \\ 0 & \text{otherwise,} \end{cases}$$

and

$$L_i^{\langle n_0,\ldots,n_{i-1}\rangle}[s] = \sum_{k=0}^{i-1} 2^{-k} V_{k,n_k}[s].$$

So $L_i^{\langle n_0,\ldots,n_{i-1}\rangle}[s]$ sums only the first i martingales, computed up to stage s, and it uses given values for the sequence (n_k).

We let $Z_i^{\langle n_0,\ldots,n_{i-1}\rangle}$ be the left-most non-ascending path of $L_i^{\langle n_0,\ldots,n_{i-1}\rangle}$, and $Z_i^{\langle n_0,\ldots,n_{i-1}\rangle}[s]$ the left-most non-ascending path of $L_i^{\langle n_0,\ldots,n_{i-1}\rangle}[s]$. Our betting strategy will use these $Z_i^{\langle n_0,\ldots,n_{i-1}\rangle}[s]$ (which are computable) as guesses for the actual Z.

4.2 The Injection

As PCR implies TPR, we need to make essential use of the fact that we are allowed to bet on the positions of $Z^{(n_k)}$ in an order given by a computable *injection*. Equivalently, the order may be given by a computable enumeration of an infinite subset of $\mathbb{N}$. We achieve this by uniformly assigning a computation to each $k \in \mathbb{N}$, and by betting on k at the stage that the computation corresponding to k terminates, if ever. In particular, we will bet on $k = \langle i, \langle n_0, \ldots, n_{i-1}\rangle, l, m\rangle$ at the first stage s that

$$\left|\left\{ j \in \{0,\ldots,i-1\} \quad : \quad B_j[s]\left(Z_i^{\langle n_0,\ldots,n_{i-1}\rangle}[s]\restriction_{k+1}\right)\downarrow \right\}\right| = l.$$

At this point, and if i has a value that we are still interested in, we will guess that all computations involved in defining $Z_i^{\langle n_0,\ldots,n_{i-1}\rangle}\restriction_k$ that converge, have halted by stage s; hence we will bet on $Z_i^{\langle n_0,\ldots,n_{i-1}\rangle}[s](k) = Z^{(n_k)}(k)$. Under certain conditions, this guess is guaranteed to be correct. In particular the following lemma holds:

Lemma 1. *Suppose that*

(a) $l = \left|\left\{j \in \{0,\dots,i-1\} \quad : \quad B_j \text{ is defined along } Z_i^{\langle n_0,\dots,n_{i-1}\rangle}\right\}\right|$, *and*
(b) m *is sufficiently large.*

Let $k = \langle i, \langle n_0,\dots,n_{i-1}\rangle, l, m\rangle$. *Then there is a stage* s *such that*

$$\left|\left\{j \in \{0,\dots,i-1\} \quad : \quad B_j[s]\left(Z_i^{\langle n_0,\dots,n_{i-1}\rangle}[s]\upharpoonright_{k+1}\right)\downarrow\right\}\right| = l. \tag{8}$$

Moreover, at this stage we have

$$Z_i^{\langle n_0,\dots,n_{i-1}\rangle}[s](k) = Z_i^{\langle n_0,\dots,n_{i-1}\rangle}(k).$$

Proof. We abbreviate $Z_i = Z_i^{\langle n_0,\dots,n_{i-1}\rangle}$.

There are only finitely many $n \in \mathbb{N}$ such that $B_j(Z_i\upharpoonright_n)\downarrow$ for some $j \in \{0,\dots,i-1\}$ such that B_j is *not* defined along Z_i. Let N be the maximal such n. Let s_0 be the first stage such that

$$B_j[s](Z_i\upharpoonright_n)\downarrow \text{ if and only if } B_j(Z_i\upharpoonright_n)\downarrow$$

for all $j \in \{0,\dots,i-1\}$ and all $n \le N$.

Given *(a)*, (8) will hold for s large enough. But note that the larger we take m, the larger k is, and the longer it will take for (8) to hold. So we can take m large enough to have $k > N$ and $s \ge s_0$.

By choice of N, s_0 and m, we have

$$Z_i\upharpoonright_N = Z_i[s]\upharpoonright_N,$$

and $B_j(Z_i\upharpoonright_N 0)\uparrow$ and $B_j(Z_i\upharpoonright_N 1)\uparrow$ for all $j \in \{0,\dots,i-1\}$ such that B_j is *not* defined along Z_i. Hence, when (8) holds, we must have

$$\begin{aligned}\{j \in \{0,\dots,i-1\} \quad &: \quad B_j[s]\left(Z_i[s]\upharpoonright_{k+1}\right)\downarrow\} \\ &\subseteq \{j \in \{0,\dots,i-1\} \quad : \quad B_j \text{ is defined along } Z_i\}\end{aligned}$$

and by *(a)* this is actually an equality. This means that all computations involved in defining $Z_i\upharpoonright_{k+1}$ have halted by stage s, so the guess

$$Z_i[s](k) = Z_i(k)$$

is correct. □

4.3 The Betting Strategy

We are now ready to define the sequence (n_k) and the total injection strategy that will succeed on $Z^{(n_k)}$. We have already defined the computable injection above. Now we partition the initial capital; to every natural number $j = \langle i, \langle n_0,\dots,n_{i-1}\rangle, l\rangle$ we assign a fraction 2^{-j-1} of our starting capital. When we are asked to bet on $k = \langle i, \langle n_0,\dots,n_{i-1}\rangle, l, m\rangle$, we will only use the capital

assigned to the number $\langle i, \langle n_0, \ldots, n_{i-1}\rangle, l\rangle$. In particular, if we are asked to bet on this position k at stage s, then we will put $\frac{3}{4}$ of this capital on the outcome $Z_i^{\langle n_0,\ldots,n_{i-1}\rangle}[s](k)$ and $\frac{1}{4}$ of this capital on the other outcome. Once the capital assigned to some $\langle i, \langle n_0, \ldots, n_{i-1}\rangle, l\rangle$ exceeds 1, we start betting evenly on positions with this value of i, and we say that the *substrategy for i has succeeded*.

Note 2. – The substrategy for i is guaranteed to succeed when betting on $Z_i^{\langle n_0,\ldots,n_{i-1}\rangle}$. Indeed, by Lemma 1, when l has the right value and m is big enough, then at some point we will bet on position $k = \langle i, \langle n_0, \ldots, n_{i-1}\rangle, l, m\rangle$ and this bet is guaranteed to be successful, i.e. to increase the capital assigned to $\langle i, \langle n_0, \ldots, n_{i-1}\rangle, l\rangle$ with a factor $\frac{3}{2}$. So the capital assigned to $\langle i, \langle n_0, \ldots, n_{i-1}\rangle, l\rangle$ will exceed 1 if we wait long enough.

– If the substrategy for i succeeds when betting on $Z_i^{\langle n_0,\ldots,n_{i-1}\rangle}$, and the highest position that the strategy has bet on before succeeding is position k, then the substrategy will run exactly the same, and hence also succeed at the same point, on any sequence Y with $Y\restriction_{k+1} = Z\restriction_{k+1}$. In particular, if

$$k < n_i < n_{i+1} < n_{i+2} < \ldots,$$

then the substrategy for i will run exactly the same on $Z_j^{\langle n_0,\ldots,n_{j-1}\rangle}$ for any $j \geq i$, and also on $Z^{(n_k)}$.

4.4 The Sequence (n_k)

We now recursively define n_k by letting $n_0 = 0$ and taking

$$n_i = 1 + \begin{pmatrix}\text{Highest position that the strategy has bet on after the}\\ \text{substrategies for } 0, \ldots, i-1 \text{ have succeeded when betting}\\ \text{on } Z_i^{\langle n_0,\ldots,n_{i-1}\rangle}\end{pmatrix}.$$

By Note 2, these substrategies indeed all succeed, so the sequence is well-defined. Moreover, the substrategies all succeed on $Z^{(n_k)}$, as well. So the total injection betting strategy succeeds on $Z^{(n_k)}$, as there are infinitely many substrategies, that with disjoint parts of the initial capital, all generate one unit of money. This proves the theorem:

Theorem 1. *There is a PCR sequence, which is not TIR.*

5 The Bigger Picture

The first real notion of algorithmic randomness was Martin-Löf randomness (**MLR**), introduced by Martin-Löf in 1966 [4]. Schnorr argued that the *tests* used to define nonrandom sequences in Martin-Löf randomness are too strong to be considered effective. He proposed two important weaker notions of randomness, now known as Schnorr randomness (**SR**) and computable randomness [9,10]. Computable randomness, as well as all variations introduced in section 3 of this

article, are weaker than MLR but stronger than SR. Two nontrivial implications hold between these notions: TPR is equivalent to CR (See e.g. [3] or [7], 7.6.24) and the partial and total versions of KLR are also equivalent ([5]; see also [2], 7.5.4 or [7], 7.6.25).

PCR and KLR are the variations of computable randomness that have been studied best. Among the most interesting results involving these notions, are the constructions by Nies, Stephan and Terwijn [8] of a sequence that is CR but not PCR, and a sequence that is SR but not CR, where both sequences can be constructed in any high Turing degree. This is the best possible, since in all nonhigh Turing degrees Schnorr randomness implies MLR [8], collapsing all notions of computable randomness.

The most important open problem in this area is whether MLR is strictly stronger than KLR, or whether both notions are equivalent. Injection and permutation randomness were introduced by Miller and Nies [6], in the hope that it would be possible to construct an injection/permutation random sequence that is not MLR, thereby providing a stepping stone towards separating KLR from MLR. This was indeed recently achieved by Kastermans and Lempp [3], who separated MLR from PIR. This is the closest that anyone has come to solving the "MLR versus KLR" question so far.

The relative strength of the different notions of computable randomness was studied in detail by Bienvenu, Hölzl, Kräling and Merkle [1]. They constructed

- a PPR sequence that is not TIR,
- a TIR sequence that is not PCR,
- a PCR sequence that is not PPR,

thereby proving that no implications hold between the different computable randomness notions, other than the ones metioned before, and possibly an implication from PIR to KLR.

All of this is summarized in Figure 1.

Theorem 1 of this article is already implied by the construction in [1] of a PPR sequence that is not TIR. My construction however, is quite different from the construction in [1], in which Kolmogorov complexity is central. I believe that my method is more direct and the idea might be easier to understand. On the other hand, the many approximations up to some stage s make the verification a little messy, and the methods used in [1] seem to be more versatile.

An interesting direction for future research, would be to investigate whether the different constructions from [3], [1] and this article can be adapted to give sequences in any high degree, similar to the results from [8].

References

1. Bienvenu, L., Hölzl, R., Kräling, T., Merkle, W.: Separations of non-monotonic randomness notions. 6th International Conference on Computability and Complexity in Analysis (CCA 2009) (2009)
2. Downey, R.G., Hirschfeldt, D.R.: Algorithmic Randomness and Complexity. Theory and Applications of Computability, Springer (2011)

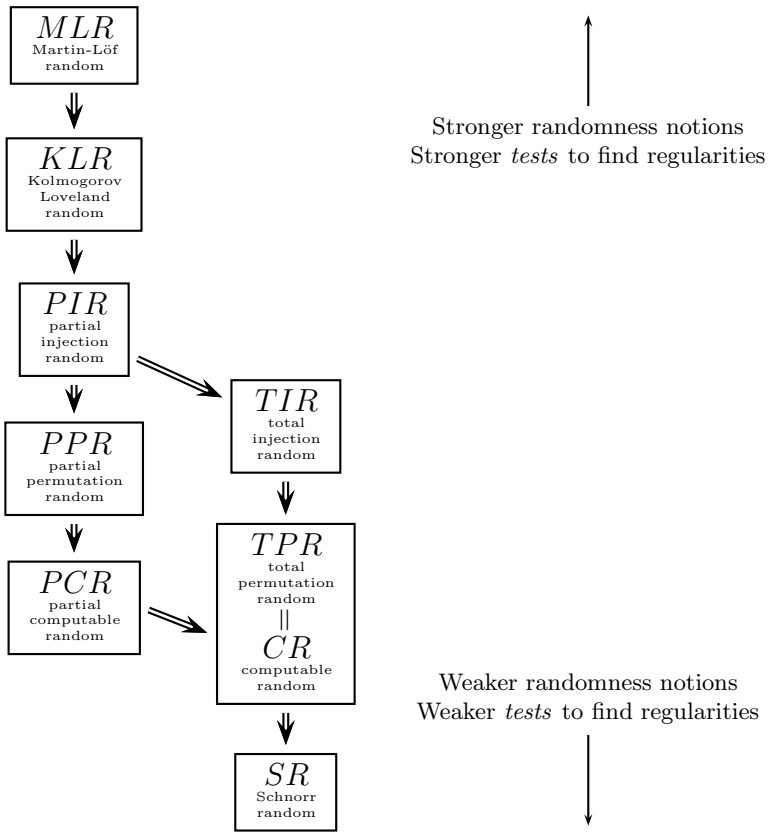

Fig. 1. The relative strength of computable randomness and related notions. No other implications than the ones implied by the figure hold, except for possibly "$PIR \Rightarrow KLR$" or "$KLR \Rightarrow MLR$", but certainly not both.

3. Kastermans, B., Lempp, S.: Comparing notions of randomness. Theoretical Computer Science 411, 602–616 (2010)
4. Martin-Löf, P.: The definition of random sequences. Information and Control 9, 602–619 (1966)
5. Merkle, W.: The Kolmogorov-Loveland stochastic sequences are not closed under selecting subsequences. Journal of Symbolic Logic 68, 1362–1376 (2003)
6. Miller, J.S., Nies, A.: Randomness and computability: open questions. Bulletin of Symbolic Logic 12, 390–410 (2006)
7. Nies, A.: Computability and Randomness. Oxford University Press (2009)
8. Nies, A., Stephan, F., Terwijn, S.A.: Randomness, relativization and Turing degrees. Journal of Symbolic Logic 70, 515–535 (2005)
9. Schnorr, C.P.: A unified approach to the definition of a random sequence. Mathematical Systems Theory 5, 246–258 (1971)
10. Schnorr, C.P.: Zufälligkeit und Wahrscheinlichkeit. Eine algoritmische Begründung der Wahrscheinlichkeitstheorie, Lecture Notes in Mathematics, vol. 218. Springer-Verlag (1971)