Dobbelstrategieë vir toevalsrye

Gambling strategies for random sequences

G Davie
Departement Besluitkunde
Universiteit van Suid-Afrika (Pretoria)
Posbus 392, Unisa, 0003
davieg@unisa.ac.za

George Davie

George Davie (gebore 1967) voltooi in 1994 die graad MSc (Wiskunde) en in 2000 die graad PhD (Wiskunde) aan die Universiteit van Pretoria. Hy is sedert 1997 by die Universiteit van Suid-Afrika in Pretoria, tans in die Departement Besluitkunde.	**George Davie** (born 1967) completed the degree MSc(Math) in 1994 and the degree PhD (Math) at the University of Pretoria in 2000. Since 1997 he has been at the University of South Africa in Pretoria, at present in the Department of Decision Sciences.

ABSTRACT

There is a general consensus that it is not possible to gamble successfully against a random sequence. This consensus is based on results from probability theory that all gambling systems are in some sense futile and the idea that at any stage of the sequence, the next outcome is entirely unpredictable.

Our ideas of a random sequence are epitomised by the consecutive outcomes of the tosses of a fair coin, and we have strong intuitions about such a process. Defining randomness mathematically *however, has been a difficult task. Fairly recently, the concept of* algorithmically random *has been spectacularly successful as a definition of randomness, capturing many of our most important intuitions regarding when something is random. The definition uses the following concept of* Kolmogorov complexity:

Definition *Let U be a universal Turing machine and let x be a finite binary string. The* Kolmogorov complexity *of x, denoted by $C_U(x)$, is the length of the shortest program (for U) that outputs x on the empty input.*[1]

Recall that a universal Turing machine is just an abstracted computer, and the reader is welcome to think of it as a mathematical model of a real computer.

Informally then, an object (binary string) is complex *if no short program gives the string as output and* simple *if some short program gives the string as output. So, for example, a string of a million zeroes is simple ("do 10^6 times: print 0") while the outcomes of a coin tossed a million times is (most probably) complex. The idea is thus that complex strings are their own shortest description. We will often drop the subscript U and write just $C(x)$.*

We would like to define an infinite *sequence as* random *if each initial segment of it of length n is complex, that is, has complexity almost n.*

To make this definition work, we need to restrict slightly the classes of program that we allow. We require them to form a prefix free *set, that is, no program is allowed to be an initial segment of*

another. This seems somewhat artificial but in fact is equivalent to the requirement that a program indicates its own end, which all programs in common program languages do. A prefix machine *P is a universal Turing machine which is defined on a prefix free set and the K-*complexity *of x, denoted by $K_P(x)$, is then the length of the shortest program (for P) that outputs x on the empty input. We can imagine P as fixed and will often drop the subscript to write just $K(x)$. We then have the following:*

Definition *(Chaitin[2], Levin[3]) Fix a universal prefix Turing machine U from $\{0,1\}^*$ to $\{0,1\}^*$. An infinite binary sequence ω is* algorithmically random *if there is a fixed $c \in \mathbb{N}$ such that*

$$K_U(\omega_{1:n}) \geq n - c$$

for all n.

The class of random strings this definition gives us has some very satisfying properties: most binary sequences are random (the set has Lebesgue measure 1), they are normal (that is they contain every subsequence with the expected frequency, e.g. they contain the subsequence 01 *with frequency $\frac{1}{4}$). They are unpredictable in that we cannot successfully predict digits given digits so far and they also satisfy all the important laws of probability theory such as the law of large numbers and the law of the iterated logarithm.*

*Can we gamble against such sequences? Important for us is the following definition, where x*0 *denotes the sequence x followed by a zero.*

Definition *A function d from finite binary strings to the non-negative reals is a* martingale[4] *if*

$$d(\omega_{1:n}) = \frac{d(\omega_{1:n}0) + d(\omega_{1:n}1)}{2}$$

A martingale succeeds *on a binary sequence ω if* $\limsup_n d(\omega_{1:n}) = \infty$.

We can think of a martingale as giving the capital d that we have at every stage when playing a particular gambling strategy against the digits of an infinite sequence. A martingale d succeeds on ω if the amount of money the martingale wins is unlimited. *Also see the recent book of Nies[5].*

The following result can be read as stating the absence of successful gambling strategies against algorithmically random sequences:

Theorem[6] *A binary sequence ω is algorithmically random if and only if no computably enumerable martingale succeeds on ω.*

For any such ω and given martingale strategy there is thus a finite amount k such that we cannot win more than k Rand, denoted from now on by Rk. This is a satisfying result and agrees with our intuition regarding randomness.

One would think of a betting strategy as one that, for a given finite sequence of outcomes so far, instructs us what to bet on the next digit being a 1 *(or a* 0*). This is reflected in the concept of a* computable *martingale. This is one in which the function $d : \mathbb{N} \to \mathbb{R}$ is computable, which means that there is a program for d which can tell us what our capital would be after any initial segment.*

What then are the computably enumerable *martingales of the theorem? When the function $d : \mathbb{N} \to \mathbb{R}$ is* computably enumerable.

The main point is that this class of martingales strictly contains the intuitively appealing computable ones and in fact form a much larger, more powerful class of martingales. The theorem thus states that not even these martingales can succeed against a random sequence, let alone a computable one. However, the following also holds, and this is the main result of this paper:

Theorem *For every algorithmically random sequence ω there is a computable gambling strategy which will allow us to win any predetermined amount n, by starting with an appropriate initial capital, uniformly computable in n. Moreover, we can repeat this strategy at any stage of the formation of ω, as often as we like.*

We think this result runs counter to our intuitions regarding randomness, as epitomised in the idea of a sequence of coin tosses. Since the strategy uses a constant associated with the entire sequence ω (that is, the constant is not determined by any initial segment of ω) we conclude by suggesting that it is perhaps because the definition of algorithmically random treats infinite random sequences as completed objects that we get this counterintuitive result.

KEY CONCEPTS: Randomness, Kolmogorov complexity, Algorithmically random sequence, Martingale.

TREFWOORDE: Willekeurigheid, Kolmogorov-kompleksiteit, Algoritmiese toevalsry, Martingaal.

OPSOMMING

Dit is 'n algemene opvatting dat dit nie moontlik is om suksesvol te dobbel teen 'n toevalsry (of aleatoriese ry) van nulle en ene nie. Ons gebruik idees van Kolmogorov-kompleksiteit om hierdie opvatting te bevraagteken. Ons wys dat alhoewel daar geen algoritme is waarmee ons 'n onbeperkte hoeveelheid geld kan wen teen 'n algoritmiese toevalsry nie[6], is daar wel vir elke so 'n ry 'n dobbelstrategie waarmee ons enige gewenste bedrag kan wen deur met 'n voorafbepaalbare bedrag 'n eenvoudige verdubbelingstrategie uit te voer. Ons kan ook dieselfde strategie gebruik om 'n onbeperkte hoeveelheid geld te wen, in die geval moet ons herhaaldelik geld rentevry kan leen.

1. NOTASIE EN DEFINISIES

Ons gebruik ω om 'n oneindige binêre ry aan te dui en die notasie $\omega_{1:n}$ om die aanvangsegment van die eerste n syfers van ω aan te dui. Ons dui ook die syfers vanaf posisies n na n' in ω aan deur $\omega_{n:n'}$. Dus het ons bv. vir $\omega = 01100011...$ dat $\omega_{1:4} = 0110$ en $\omega_{2:5} = 11000$. Ons dui die versameling eindige woorde oor die alfabet $\{0,1\}$ aan met $\{0,1\}^*$. Ons sal die lengte van 'n eindige string x aandui met $|x|$, dus is bv. $|0111| = 4$. Ons dui met log binêre log aan, bv. $\log 8 = 3$ aangesien $2^3 = 8$.

Ons verduidelik vervolgens wat ons bedoel met 'n *berekenbare* funksie en 'n *berekenbaar aftelbare* funksie $f : \mathbb{N} \to \mathbb{R}$.

'n Funksie vanaf $\mathbb{N}$ na $\mathbb{R}$ is *berekenbaar* as daar 'n berekenbare funksie $f' : \mathbb{N} \times \mathbb{N} \to \mathbb{Q}$ is sodanig dat ons vir alle n,k het dat $|f'(k,n) - f(k)| \leq 2^{-n}$. Dit beteken presies dat ons vir enige k vir $f(k)$ tot enige gewenste akkuraatheid kan vind.

'n Funksie $f : \mathbb{N} \to \mathbb{R}$ is *berekenbaar aftelbaar* as die versameling $\{(n,r) \in \mathbb{N} \times \mathbb{Q} : r < f(n)\}$ berekenbaar aftelbaar is. Dit beteken presies dat ons vir enige n uiteindelik alle getalle kleiner as $f(n)$ kan voortbring. Dit beteken op sy beurt dat ons vir elke n vir $f(n)$ van onder af kan benader. Ons het egter nie noodwendig op enige stadium 'n idee oor hoe na ons aan die werklike $f(n)$ is nie.

2. WAT IS 'N TOEVALSRY?

'n Argetipiese voorbeeld van wat ons as 'n toevalsry sou beskou is dié voortgebring deur opeenvolgende gooie van 'n muntstuk. Ons noteer 'n 1 indien die muntstuk "kop bo" land en 'n 0 indien

die muntstuk "stert bo" land. Omdat ons só 'n muntstukgooi as 'n sinnebeeld van ons intuïsies oor 'n toevalsry beskou, sal ons deurentyd wiskundige idees oor toevalsrye teen die idee van 'n muntstukgooi meet, veral in die gevalle waar die wiskundige idees in stryd is met dié wat ons oor 'n muntstukgooi het.

Ons ondersoek eers verder ons intuïsies aangaande toevalsrye.

Beskou vir eenvoud 'n binêre ry van nulle en ene. Wanneer beskou ons só 'n ry as 'n toevalsry? 'n Primêre beginsel vir 'n toevalsry is die van

onvoorspelbaarheid,

d.w.s. ons voel dat ons nie die syfers van só 'n ry vooruit kan bepaal nie, selfs nie met die syfers wat tot nou verskyn het nie. Verder voel ons ook dat so 'n toevalsry nie voortgebring kan word met 'n eenvoudige *resep* nie, soos bv. "herhaal oneindig keer die patroon 01"nie. Hierdie resep bring natuurlik die ry

01010101...

voort. Gestel nou dat daar vir 'n aanvangsegment van 'n ry, geen resep vir die segment bestaan wat veel korter is as die segment self nie, m.a.w. ons kan nie veel beter doen as om die segment self weer te gee nie. Ons sou só 'n aanvangsegment aanvaar as deel van 'n toevalsry.

3. KOLMOGOROV-KOMPLEKSITEIT EN ALGORITMIESE TOEVALSRYE

Ons ondersoek verder hierdie idee van "geen korter resep as die string self nie", of anders gestel, die idee van

nie-saamdrukbaarheid.

'n Natuurlike interpretasie vir *resep* is die van *algoritme* of rekenaarprogram. Dit is 'n bekende en fundamentele feit dat daar 'n universele program is, wat alle ander programme kan simuleer. Met dié konteks kan ons die volgende definisie vir eindige rye gee:

Definisie 1. [7,1] *Laat U 'n universele Turing-masjien wees en x 'n eindige binêre ry. Die* ***Kolmogorov-kompleksiteit*** *van x, aangedui deur $C_U(x)$, is die lengte van die kortste program vir U wat x as uitvoer gee, op die leë invoer.*

Ons neem nou aan dat ons verder met 'n vaste universele masjien werk en laat die onderskrif U voortaan weg. Ons kan dus die Kolmogorov-kompleksiteit van 'n objek sien as die lengte van 'n kortste "resep" vir of "beskrywing" van die objek.

Natuurlik is "Druk x" (van lengte $|x|$ plus 'n konstante) altyd een so 'n geldige program, dus sal altyd $C_U(x) \leq |x| + k$, k 'n konstante.

Die idee is dat 'n objek **eenvoudig** is indien een of ander beskrywing daarvan kort is.

Voorbeeld 1. $\overbrace{010101...01}^{10^6\ keer}$ *is eenvoudig.*

En **kompleks** is indien *geen* beskrywing daarvan kort is nie.

Voorbeeld 2. *Gooi 'n muntstuk* 10^6 *keer, skryf* 1 *as ons die muntstuk "kop" land en* 0 *indien "stert".*

'n Binêre ry word dan as kompleks beskou indien die Kolmogorov-kompleksiteit daarvan naby is aan die lengte.

Die kompleksiteit van 'n binêre string is nie *berekenbaar* nie, d.w.s daar bestaan geen algoritme wat as invoer enige string neem en as uitvoer die string se kompleksiteit gee nie. Dit is duidelik as mens besef dat enige so 'n algoritme vir ons op 'n kort invoer, 'n lang komplekse string as uitvoer sou kon gee, soos volg: "Maak 'n lys van alle stringe van lengte kleiner as 1000 met kompleksiteit minder as 900. Gee as uitvoer die eerste string wat nie in die lys verskyn nie." Dit is inderdaad bewysbaar dat 'n algoritme wat stringkompleksiteite kan gee, ook die befaamde *halt-probleem* sou kon oplos.[8] D.w.s. dié program sal ook kan besluit of enige gegewe program op enige gegewe invoer sal halt.

'n Naïewe uitbreiding van hierdie idee na oneindige stringe sou wees om 'n *oneindige* ry van nulle en ene as 'n toevalsry te ag indien *elke* aanvangsegment van die ry kompleks is. Ongelukkig werk dié idee nie. 'n Insiggewende manier om dit te sien is om te besef dat elke aanvangsegment van 'n ry (na die eerste 1) 'n stringlengte gee. Beskou bv. die ry

$$101|10110$$

waar die | net vir duidelikheid geplaas is. Ons kan die ry spesifiseer deur net

$$10110$$

te gee en uit dié segment die lengte 5 te kry en dan die lengte in binêr voor by te voeg as 101. Ons kry dus die eerste 3 syfers "verniet". 'n Effense verfyning van hierdie opmerking gee die algemene geval.

Ons let nou op dat enige werklike rekenaarprogram sy eie einde aandui, oftewel *self-afsluitend* is. Dit beteken ook dat die versameling rekenaarprogramme in enige taal 'n *prefiksvrye* versameling vorm. "Prefiksvry" beteken dat geen program 'n aanvangsegment is van 'n ander nie. 'n Program se einde word byvoorbeeld aangedui met 'n instruksie "end" (BASIC) of deur die sluiting van die laaste hakie (Lisp). As ons enigiets hierna byvoeg sal dié deel nie uitgevoer word nie. Dit is nie onmiddellik duidelik dat enige prefiksvrye versameling so sal wees dat elke element van die versameling implisiet sy einde aandui nie, maar dit is wel die geval.[8]

Ons pas dus ons definisie aan om net prefiksvrye programme toe te laat. Daar is, net soos vir gewone programme, ook 'n universele Turing-masjien vir sulke algoritmes, naamlik 'n universele prefiksmasjien. Ons vereis ook van ons universele masjien dat dit enige ander masjien kan simuleer met hoogstens 'n konstante oorhoofse koste. Met dié aanpassing op die versameling toelaatbare programme vir die binêre stringe, kry ons die gewenste veralgemening van kompleks vir oneindige rye en kan ons die volgende definisie gee van 'n (algoritmiese) toevalsry:

Definisie 2. [3,9] *Laat P 'n universele prefiksmasjien wees (invoere vir die masjien is self-afsluitend) vanaf $\{0,1\}^*$ na $\{0,1\}^*$. Vir x 'n arbitrêre eindige binêre string, is die* prefikskompleksiteit $K_P(x)$ *van x die lengte van 'n kortste program vir P wat x as uitvoer gee op die leë invoer.*

Soos by $C(x)$ laat ons gewoonlik die onderskrif weg met dié afspraak dat ons deurgaans met dieselfde universele masjien werk.

Ons merk op dat $K(x) \leq |x| + K(|x|) + c$, c 'n universele konstante vir P. Dit geld aangesien ons 'n algemene program (aldus die universele konstante c) vir P kan skryf wat as invoer 'n self-afsluitende beskrywing neem vir x (van lengte $K(x)$) gevolg deur x self en dan vir x as uitvoer gee. Die self-afsluitende beskrywing van $|x|$ gevolg deur x self vorm dan saam 'n self-afsluitende program vir x soos volg: P lees die eerste $K(|x|)$ syfers wat 'n beskrywing is van $|x|$, om $|x|$ te kry. P lees dan $|x|$ syfers verder om x self te kry.

In die besonder geld dus dat $K(x) \leq 2\log x + k$, met k 'n universele konstante. Dit sal byvoorbeeld geld as ons eers die lengte van x unêr gee, met lengte $\log x$, dan 'n 0 en dan x self (ook van lengte

$\log x$) , bv. vir $x = 1011001$ sal die program 111111101011001 'n (onnodig lank) selfafsluitende program wees vir x. In die prefiksvrye geval is x self ongelukkig nie meer 'n beskrywing van x nie, maar amper!

Ons kan nou die begrip van 'n (algoritmiese) toevalsry definieer:

Definisie 3. [3,2] *'n Oneindige binêre ry ω is 'n* algoritmiese toevalsry *as daar 'n $c \in \mathbb{N}$ bestaan, sodat vir alle $n \in \mathbb{N}$*

$$K(\omega_{1:n}) \geq n - c.$$

Ons eis dus dat *elke* aanvangsegment nie-saamdrukbaar moet wees. Let op dat die versameling toevalsrye nie afhang van die universele masjien U nie. Indien U verander, sal hoogstens die getal c verander. Intuïtief is dit omdat ons vir 'n gegewe universele masjien, 'n ander masjien kan simuleer met net 'n konstante oorhoofse koste. Duidelik, mits daar só 'n c bestaan, sal enige $c' > c$ ook die definisie bevredig. Vir 'n gegewe ω, noem die kleinste c waarvoor die definisie hierbo geld, die *saamdrukbaarheidskonstante*. Dié konstante speel 'n kritieke rol in die vervolg.

Vir 'n definisie van toevalsry het hierdie definisie baie gewenste eienskappe:

- Die aantal toevalsrye het Lebesgue-maat 1, d.w.s. 'n oneindige binêre ry is met waarskynlikheid 1 'n toevalsry.
- Sulke rye is onvoorspelbaar, d.w.s. ons kan nie $\omega_{1:n}$ gebruik om $\omega_{1:n'}$ te voorspel vir n' groter as n nie (dit sou in stryd wees met die nie-saamdrukbaarheid van elke aanvangsegment).
- Sulke rye is ook *normaal*, m.a.w. hulle bevat alle deelrye met die verwagte frekwensie (bv. sal hulle die deelry 00 met frekwensie $\frac{1}{4}$ bevat).
- Laastens bevredig hulle ook alle standaard waarskynlikheidswette soos die wet van groot getalle en die wet van geïtereerde logaritme (ons kan inderdaad die konstante c in Definisie 3 gebruik om dié wette effektief te maak).[10]
- Dit behoort ook duidelik te wees dat sulke rye nie eksplisiete patrone kan bevat nie, aangesien dit ons in staat sou stel om kort resepte (algoritmes) te vind vir van die aanvangsegmente.

4. DOBBELSTRATEGIEË BESTAAN NIE

Die volgende definisie en resultaat is vir ons baie belangrik:

'n Funksie d vanaf eindige binêre stringe na die nie-negatiewe reële getalle is 'n *martingaal*[4,5] as

$$d(\omega_{1:n}) = \frac{d(\omega_{1:n}0) + d(\omega_{1:n}1)}{2}.$$

'n Martingaal *slaag* op 'n binêre ry ω indien

$$\limsup_n d(\omega_{1:n}) = \infty.$$

Mens kan soos volg dink: Ons het een of ander dobbelstrategie sodanig dat ons weddenskappe op elke stadium afhang van die syfers van die binêre ry tot dusver. Ons strategie sê byvoorbeeld om te wed dat die volgende syfer 'n 1 gaan wees as die vorige syfer 'n 0 is, en andersom. Die funksie d gee dan vir elke string σ die kapitaal wat ons sou wen op 'n string σ deur van die eerste tot die laaste syfer van σ die strategie uit te voer. As die funksie d dus net die kapitaal gee, waar is die strategie?

As ons dink aan 'n dobbelspel waar ons ons wedgeld verdubbel as ons reg wed en dit verloor as ons verkeerd wed, dan bevat d inderdaad implisiet ons strategie. Byvoorbeeld as $d(\omega_{1:n}0) = d(\omega_{1:n}) + k$ dan is die strategie om k rand te wed dat die $n+1$de syfer 'n 0 is (aangesien d vergroot met k wanneer dit 'n 0 is). Let op dat die martingaal voorwaarde dan afdwing that $d(\omega_{1:n}1) = d(\omega_{1:n}) - k$ wat beteken dat ons sou verloor het as die volgende syfer 'n 1 was.

Dus bepaal die voorwaarde dat ons verwagte kapitaal dieselfde is na 'n weddenskap as voor. Dit maak sin aangesien ons wed dat 'n spesifieke syfer 'n 1 (of 'n 0 is) en daar presies soveel rye is wat die teenoorgestelde syfer in daardie posisie het, naamlik 'n 0 (of 'n 1). Verder verseker die voorwaarde ons daarvan dat die spel "regverdig" is en nie teen ons "gelaai" is nie (geen casinospel is regverdig nie!). Hierdie voorwaarde word bevredig in die geval hierbo, naamlik waar ons twee maal ons wedgeld terugkry as ons reg wed en ons wedgeld verloor as ons verkeerd wed.

'n Martingaal *slaag* op ω indien ons 'n onbeperkte hoeveelheid geld met die strategie implisiet in d op ω wen. Dit wil sê, as daar geen bogrens is op die hoeveelheid geld wat die strategie wen nie. Ons kan dus vooraf besluit om te stop as ons x rand sê, gewen het, en ons is seker dat dit wel op 'n stadium sal gebeur.

Schnorr het die volgende gewys:

Stelling 1. [6] *'n Binêre ry ω is 'n algoritmiese toevalsry as en slegs as geen berekenbaar aftelbare martingaal op ω slaag nie.*

Vir enige só 'n ω en gegewe strategie is daar dus 'n eindige bedrag k sodat ons nie meer as Rk kan wen nie. Dit is 'n intuïtief bevredigende resultaat en strook met ons intuïsie van willekeurigheid.

Ons bespreek kortliks Schnorr se resultaat: Ons kan aan 'n dobbelstrategie dink as een wat, gegee die ry uitkomste (van nulle en ene) tot dusver, vir ons die volgende syfer voorspel en sê wat om op dié voorspelling te wed. Ons voorspel dalk dat die volgende uitkoms 'n 0 gaan wees en dat ons $R100$ daarop gaan wed. Dié intuïsie word weerspieël in die klas van *berekenbare* martingale. Hierdie is martingale waarvan die funksie $d : \mathbb{N} \to \mathbb{R}$ berekenbaar is. Dit beteken dat daar 'n program is vir d wat vir ons kan vertel wat ons kapitaal op enige aanvangssegment sou wees.

Die *berekenbaar aftelbare* martingale daarteenoor, is dié waarvoor die funksie $d : \mathbb{N} \to \mathbb{R}$ berekenbaar aftelbaar is. Sien afdeling 1 vir die definisie.

Dit is nie so belangrik om presies te verstaan wat 'n berekenbaar aftelbare martingaal is nie, wat wel belangrik is, is dat dié klas martingale die klas van berekenbares streng bevat en 'n baie groter, meer kragtige klas martingale vorm. Schnorr se stelling gee dus dat selfs sulke martingale nie suksesvol kan wees teen 'n algoritmiese toevalsry nie, en dus beslis geen berekenbare martingaal nie.

5. BESTAAN VAN 'N VERALGEMEENDE DOBBELSTRATEGIE

In teenstelling met die resultate hierbo, het ons die volgende:

Stelling 2. *Vir elke algoritmiese toevalsry ω bestaan daar 'n berekenbare dobbelstrategie wat ons in staat stel om enige voorafbepaalde hoeveelheid n te wen, deur met 'n gepaste beginkapitaal $b(n)$ te begin. Beide die beginkapitaal $b(n)$ en die maksimum aantal opeenvolgende uitkomste m waarop ons sal moet wed, is uniform berekenbaar in n. Ons kan die strategie soveel maal herhaal soos wat ons wil, en mag die gewenste kapitaal n ook na willekeur verander.*

Voorbeeld 3. *Gestel byvoorbeeld ons besluit na die* 20*ste syfer van ω dat ons gaan begin wed en $R100$ wil wen. Ons bereken dan ons nodige aanvangskapitaal en maksimum aantal toekomstige syfers waarop ons sal moet wed, sê* 10*. Ons volg dan die dobbelstrategie. Ons is verseker daarvan*

dat ons voor die 30*ste syfer R*100 *maak. Ons kan die prosedure soveel maal herhaal as wat ons wil en die gewenste bedrag n na willekeur verander solank ons ooreenkomstig die gepaste beginkapitaal vir elke nuwe geval gebruik.*

Die leser se aandag word daarop gevestig dat die saamdrukbaarheidskonstante c die kritieke rol speel in die opvolgende bewys. Dié getal se (on)berekenbaarheid word hieronder net voor afdeling 6 bespreek.

Bewys. Ons bewys die volgende, waaruit die stelling volg : Vir elke algoritmiese toevalsry ω bestaan daar 'n berekenbare funksie, $\phi : \mathbb{N} \to \mathbb{N}$ wat op invoer n as uitvoer n' gee met die eienskap dat ten minste een 1 in die fragment $\omega_{n+1:n'}$ sal verskyn.

Laat dus ω 'n algoritmiese toevalsry wees. Volgens die definisie bestaan daar dan 'n kleinste $c \in \mathbb{N}$ só dat $K(\omega_{1:n}) \geq n - c$ vir alle n. Ons gaan c op 'n essensiële manier gebruik.

Ons wil vir enige gegewe n 'n n' kan vind sodanig dat $\omega_{n+1:n'}$ ten minste een 1 moet bevat. As ons so 'n n' kan vind, en besluit hoeveel geld ons wil wen, kan ons genoeg aanvangskapitaal leen om vanaf syfer $n+1$ na syfer n' 'n eenvoudige verdubbelingstrategie uit te voer. Ons is gewaarborg daarvan dat ons voor of by syfer n' ons gewenste geld sal wen.

Hoe vind ons n' ? Ons vind 'n s sodanig dat geen ω met saamdrukbaarheidkonstante c vir die s opeenvolgende posisies vanaf n net nulle kan bevat nie. Ons vind dus s sodat alle stringe van lengte $n' = n+s$ met net nulle tussen $n+1$ en n' Kolmogorov-kompleksiteit minder as $n-c$ het.

Hoe doen ons dit? Let op dat om 'n aanvangsegment $\omega_{1:n'}$ met s nulle vanaf posisie n te beskryf, benodig ons:

- 'n self-afsluitende beskrywing van $\omega_{1:n}$
- 'n self-afsluitende beskrywing van s.

Die samevoeging van hierdie twee beskrywings sal vir ons 'n beskrywing van $\omega_{1:n'}$ gee. ("Skryf $\omega_{1:n}$ neer gevolg deur s nulle".)

Let op dat ons wel die twee beskrywings eenvoudig kan aaneenskakel omdat beide selfafsluitend is. Die masjien wat die twee saamvoeg weet dus waar die eerste een eindig en ook waar die tweede een eindig.

Uit die opmerkings na definisie 2 is $K(\omega_{1:n}) < n + K(n) + d < n + 2\log n + d'$ met d en d' universele konstantes. Verder is $K(s) < 2\log s + d''$, met d'' 'n universele konstante. Laat die lengte van 'n program wat die samevoeging beskryf hierbo, uitvoer, d''' wees. (Ons merk op dat ons die konstantes d, d' , d'' en d''' maklik kan vind vir enige redelike universele prefiks-masjien.)

Indien ons dus kan kry dat

$$n + 2\log n + 2\log s + d' + d'' + d''' < n' - c = n + s - c$$

of, as ons die drie konstantes verenig in 'n nuwe konstante k, dat

$$n + 2\log n + 2\log s + k < n + s - c$$

dan sal ω êrens in die segment van lengte s 'n 1 moet bevat, anders sou geld vir n' dat

$$K(\omega_{1:n'}) < n - c$$

wat teenstrydig is.

Om te kry dat

$$\begin{aligned} n+2\log n+2\log s+k &< n+s-c \text{ of} \\ 2\log n+2\log s &< s-c-k \end{aligned}$$

hoef ons net s soos volg te neem

$$2\log n+2\log s+c<s-k$$

of

$$2\log n+c<s-2\log s-k.$$

Dit is duidelik dat ons effektief die kleinste $s\in\mathbb{N}$ kan vind waarvoor bostaande geld.
Let op dat, rofweg vir 'n vaste c

$$s\sim 2\log n.$$

Ons weet dus dat ω tussen n en n' ten minste een 1 moet bevat. □

Om dus Rk te wen, sou ons die volgende weddenskappe moet maak op die uitkomste $\omega_{n+1},\omega_{n+2}\ldots$:

$$Rk, R2k, R4k, \ldots$$

hoogstens tot by

$$R2^{s-1}k.$$

Ons moet dus

$$k\sum_{i=0}^{s-1}2^i=(2^s-1)k$$

rand by die bank leen.

Veronderstel die eerste 1 is by ω_{n+j+1}, waar ons

$$R2^jk$$

wed. Ons kry dus

$$R2^{j+1}k$$

terug. Tot en met posisie $n+j+1$ het ons

$$k\sum_{i=0}^{j}2^i=(2^{j+1}-1)k \text{ rand}$$

gewed. Op posisie $n+j+1$ kry ons $R2^{j+1}k$ terug. Omdat:

$$\begin{aligned} & 2^{j+1}k-(2^{j+1}-1)k \\ = \;& k \end{aligned}$$

het ons dus Rk wins gemaak. Ons gee die bank sy $R(2^s-1)k$ terug en hou ons Rk.

Ons kan die strategie soveel keer herhaal as wat ons wil. Omdat s grootteorde $2\log n$ het, sou ons verwag dat ons in die algemeen meer sal moet leen as ons na *langer* aanvangsegmente begin dobbel, dit is dus tot ons voordeel om vroeg in die ry te wed.

Ons kan dus enige bedrag waarop ons vooraf besluit, wen deur 'n gepaste bedrag te leen en die verdubbelingstrategie uit te voer. Ook kan ons met *herhalings* van dié strategie 'n *onbeperkte* hoeveelheid wen teen die toevalsry – in teenstelling met die uitspraak in Schnorr se stelling waar ons nie toegelaat word om aan die aanvangskapitaal te verander of by te voeg nie.

Ons kan selde vir 'n gegewe algoritmiese toevalsry, vir c vind, wat nie verbasend is nie, gegee dat sulke rye maksimaal onberekenbaar is. Dit is inderdaad nogal vreemd dat ons vir *enige* algoritmiese toevalsrye c kan vind, maar ons kan in sekere belangrike gevalle dit wel doen[10].

Sonder c kan ons dus nie die algoritme hierbo voltooi nie. Die stelling van Schnorr wat bewys dat geen effektiewe dobbelstrategieë *bestaan* nie, behels egter geensins eise op die effektiewe *vind* van sulke strategieë nie, slegs die bestaan daarvan. Hier wys ons dus dat as ons net 'n vaste bedrag wil wen, of (herhaaldelik) geld mag leen, daar wel sulke strategieë bestaan.

6. BESPREKING

Ons laat dit aan die leser oor of die resultaat strook met die leser se intuïsie oor willekeurigheid. Dit is die skrywer se opinie dat dit in stryd is met sterk intuïsies oor willekeurigheid. Ons bespreek kortliks waarom.

Aangesien ons juis willekeurigheid wil vaspen met die definisie van algoritmiese toevalsry behoort ons aan elke algoritmiese toevalsry ω te kan dink as die uitkoms van 'n oneindige ry van muntstukgooie. In die besonder behoort die beskouing van groeiende aanvangsegmente van ω ons intuïsies oor die ontwikkelende ry van gooie van 'n muntstuk te reflekteer. Ons sou egter baie huiwerig wees om te dink dat ons op enige stadium vir so 'n ry, 'n maksimum aantal opeenvolgende nulle kan bereken. Dit is so omdat ons sterk voel dat ongeag enige uitkomste tot op hede, dat 'n werklike ry van gooie *onafhanklik* daarvan verder sal ontwikkel en dit hou verband met die idee dat só 'n proses op elke stadium *vry* is.

In die skrywer se opinie lê die probleem daarin dat die definisie van algoritmiese toevalsry 'n oneindige ry as *voltooide* objek sien en ons toelaat om oor globale eienskappe soos die saamdrukbaarheidskonstante praat, en inderdaad te *gebruik* in ons dobbelstrategie. Die beperkings wat die saamdrukbaarheidskonstante op die ontwikkeling van die ry plaas sit dan ongemaklik met die sterk intuïsie dat só 'n proses die heeltyd vrylik ontwikkel.

VERWYSINGS

1. Kolmogorov, A.N., Three Approaches to the Quantitative Definition of Information, in *Problems of Information Transmission* (Problemy Peredachi Informatsii), Vol. 1 (1965), 1–7.
2. Chaitin, G., A theory of program size formally identical to information theory, *Journal of the Association for Computing Machinery*, Vol. 22 (1975), 329–340.
3. Levin, L., Laws of information conservation (non-growth) and aspects of the foundation of probability theory, *Problems Informat. Transmission*, Vol. 10 (1974), 206–210.
4. Lévy, P., Théorie de l'Addition des Variables Aléatoires, Gauthier-Villars, Paris, 1937.
5. Nies, A., Computability and Randomness, Oxford University Press, USA, 2009.
6. Schnorr, C. P., A unified approach to the definition of a random sequence, *Mathematical Systems Theory*, Vol. 5 (1971), 246–258.
7. Solomonoff, R., A formal theory of inductive inference, part 1 and part 2, *Information and Control*, Vol. 7 (1964), 224–254.
8. Li, Ming and Vitanyi, P., Kolmogorov Complexity and its Applications, Springer-Verlag, 1993.
9. Chaitin, G. Information-theoretical characterizations of recursive infinite strings, *Theoretical Computer Science*, Vol. 2 (1976), 45–48.
10. Davie, G., The Borel-Cantelli Lemmas, Probability Laws and Kolmogorov Complexity, *The Annals of Probability*, Vol. 29, No. 4 (Oct., 2001), 1426–1434.