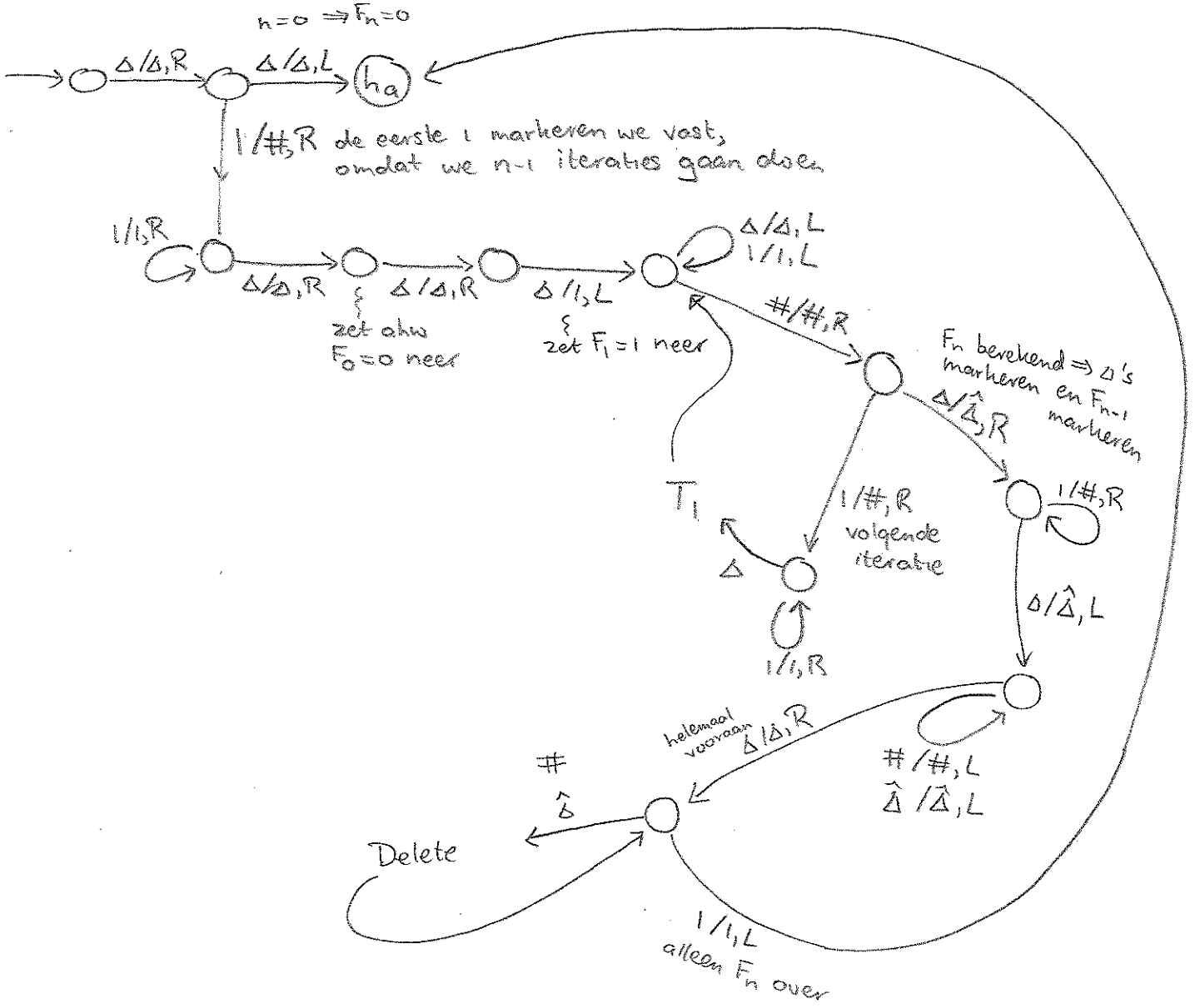
21:06

1) Achter de 1'en van n gaat T_2 F_n berekenen.
In eerste instantie zetten we $\Delta 0 \Delta 1$ (unair) achter n. ← Dit zijn F_0 en F_1
Vervolgens roepen we n-1 keer T_1 aan (voor elke 1 in n minus 1), zodat we $\Delta F_{n-1} \Delta F_n$ achter n hebben staan.
De gebruikte 1'tjes van n markeren we als #.
Tenslotte roepen we herhaaldelijk Delete aan om de #'s van n en F_{n-1} te verwijderen. Voordat we dat kunnen doen moeten we wel de twee Δ's voor F_{n-1} en F_n vervangen door $\hat{\Delta}$. En de 1'en in F_{n-1} vervangen we ook door #, zodat we makkelijk F_n kunnen herkennen.



21:26

2(a)

T_2 moet dit via een breadth-first-search doen: eerst de korte paden en dan de langere paden.
Dan weet je zeker dat àls er een pad in T_1 is dat naar ha leidt (voor een bepaalde invoer x) (ofwel: àls $x \in L(T_1)$), dat dat ook gevonden wordt in T_2.
Stel namelijk dat T_2 een depth first search uitvoert en op een pad komt dat oneindig doorloopt, dan zal het nooit een ander pad naar ha vinden, terwijl dat wel zou kunnen bestaan

21:33

(b) De functie van de \$ aan het begin van de derde tape, is om te voorkomen dat T_2 bij het simuleren van T_1 van de tape afloopt en zou crashen. We willen namelijk voorkomen dat T_2 crasht, omdat er een pad in T_1 zou kunnen zijn dat naar ha leidt en nog niet door T_2 onderzocht/gesimuleerd is

We komen \$ tegen
* als T_1 bij het gesmuleerde pad van de tape af zou lopen
* als we na afloop van het simuleren van een bitsting de derde tape schoon willen vegen om weer opnieuw met $\triangleright x$ te beginnen. Aan de \$ zien we dat we bij het schoonvegen niet verder naar links hoeven

21:42

(c) De functie van de bitstrings op de vierde tape is om te herkennen dat we eventueel kunnen stoppen met zoeken naar een pad naar ha, omdat we weten dat we zo'n pad nooit zullen vinden
We bekijken de bitstrings op de vierde tape, vóórdat we beginnen met simuleren met de eerste bitstring van de volgende lengte: $\underbrace{00\ldots0}_{n+1}$ voor zekere $n \geq 1$.

Dan kijken we eerst of op de vierde tape toevallig alle bitstrings van lengte n staan. Als dat zo is, dan weten we dat T_1 bij alle mogelijke paden in $\leq n$ stappen reject, crasht of links van de tape loopt. Het heeft dan geen zin om nog langere paden ($n+1, n+2, \ldots$ stappen) te proberen, omdat T_1 ook bij die paden binnen $\leq n$ stappen al reject, crasht of links van de tape loopt. T_2 kan stappen met zoeken en zelf naar hr gaan.

21:53

3.

We gebruiken de hint.

We beginnen met de productie:

$$S \to LM_1 1 M_2 R$$

(naar rechts)

Dan produceert L een boodschapper A die door F_{n-2} en F_{n-1} loopt en voor elke 1 die hij daarbij tegenkomt een B naar rechts stuurt. De B springt over M_2 heen en verandert daar in een 1. Zo krijgen we tussen M_2 en R precies $F_{n-2} + F_{n-1} = F_n$.

Om te voorkomen dat er een tweede boodschapper A achter de eerste aan gaat lopen veranderen we L tijdelijk in L_1

$$L \to L_1 A$$
$$A1 \to 1AB \qquad AM_1 \to M_1 A \qquad AM_2 \to M_2$$
$$B1 \to 1B \qquad BM_1 \to M_1 B \qquad BM_2 \to M_2 1$$

Vervolgens produceert R een nieuw symbool M_2 direct links van zichzelf, met daarvoor een boodschapper C die naar links loopt en onderweg de oude M_2 in M_1 verandert, de oude M_1 verwijdert en dan als D verder naar links gaat, waarbij hij F_{n-2} ook verwijdert, en tenslotte L_1 terugverandert in L:

$$1R \to C M_2 R$$
$$1C \to C1 \qquad M_2 C \to C M_1 \qquad M_1 C \to D$$
$$1D \to D \qquad L_1 D \to L$$

Nu kan L weer een boodschapper A produceren, enzovoort, totdat we gaan afronden op het moment dat we de string $L 1^{F_n} M_1 1^{F_{n+1}} M_2 R$ hebben.

Dan stuurt L een boodschapper E naar rechts, die de M_1 verwijdert en als F verder naar rechts gaat, waarbij hij ook F_{n+1}, M_2 en R verwijdert

$$L \to E$$
$$E1 \to 1E \qquad EM_1 \to F$$
$$F1 \to F \qquad FM_2 R \to \Lambda$$

Dan houden we alleen 1^{F_n} over.

Omdat we het Fibonacci getal tussen L en M_1 overhouden, kan dit ook $1^{F_0} = \Lambda$ zijn.

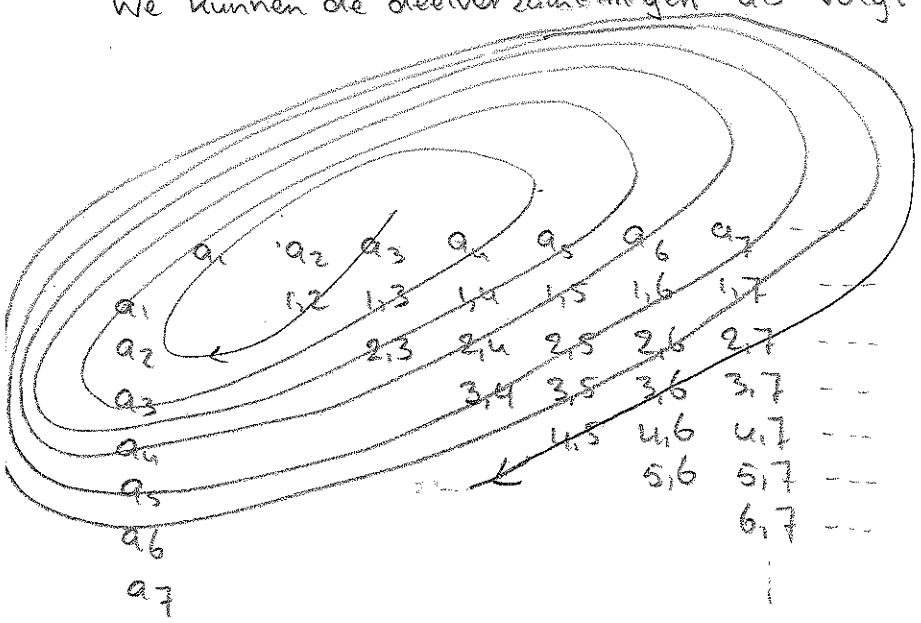
22:15

08:43

4 (a)

De deelverzamelingen van S met twee elementen vormen een deelverzameling van $S \times S$ (Ok, niet helemaal correct, want de deelverzamelingen zijn ongeordende paren, terwijl $S \times S$ geordende paren bevat).

We kunnen de deelverzamelingen als volgt aflopen:



We lopen dus alleen de rechter boven driehoek van $S \times S$ af.
De diagonaal levert namelijk deelverzamelingen met één element, en de linker onder driehoek levert deelverzamelingen die we ook al uit de rechter boven driehoek krijgen

Omdat we de deelverzamelingen van S met twee elementen dus op deze wijze kunnen aflopen ('in een rijtje zetten'), zijn het er maar aftelbaar veel.

08:55

(b)

L_1 = de verzameling van alle strings over een alfabet Σ: aftelbaar

L_2 = de verzameling van alle talen over een alfabet Σ: overaftelbaar (zelfs als Σ maar één element / symbool bevat)

L_3 = de verzameling van alle recursief-opsombare talen over een alfabet Σ: aftelbaar, (vanwege relatie met Turing machines en dus met bitstrings)

L_4 = de verzameling van alle niet-recursief-opsombare talen over een alfabet Σ: overaftelbaar (want $L_4 = L_2 - L_3$)

L_5 = de verzameling van alle mogelijke alfabetten: aftelbaar.
(want $L_5 = \bigcup_{k=1}^{\infty}$ { alfabetten met k elementen }

08:59 Diverse toelichtingen gaan bij 4(b)

En voor elke $k \geq 1$ zijn er maar aftelbaar veel alfabetten (deelverzamelingen van S) met k elementen ('analoog' aan (a))

5 (a)

Om aan te tonen dat $P_1 \leq P_2$ moet je aantonen

* dat je op algoritmische wijze
* een willekeurige instantie I van P_1 kunt omzetten in een instantie $F(I)$ van P_2
* zó dat geldt:

  I is een ja-instantie van $P_1 \iff$
  $F(I)$ is een ja-instantie van P_2

22:55

(b) Aan te tonen: Accepts-Λ $\leq$ WritesSymbol

instanties: T_1 (T_2, σ)

Gegeven een willekeurige Turing machine T_1 (instantie van Accepts-Λ). Dan kiezen we voor σ een symbool dat niet in het tape alfabet van T_1 zit (en ook niet Δ is...).

T_2 ontstaat dan uit T_1 door elke transitie naar h_a in T_1 te vervangen door een transitie naar een nieuwe toestand q^* en vanaf q^* transities naar h_a toe te voegen, waarbij σ wordt geschreven en de leeskop blijft staan.

met dezelfde beweging van de leeskop en dezelfde schrijfactie

Ofwel:

in T_1

vervangen we door

in T_2

voor elke mogelijke $\hat{\sigma}$ in het tape alfabet van T_1.

23:05

Er geldt:

* T_2 en σ zijn op algoritmische wijze uit T_1 te verkrijgen
* T_1 is ja-instantie van Accepts-$\Lambda \iff$

  T_1 bereikt h_a bij invoer $\Lambda \iff$

  T_2 bereikt q^* bij invoer $\Lambda \iff$

  T_2 schrijft een keer σ op tape bij invoer $\Lambda \iff$

  (T_2, σ) is ja-instantie van WritesSymbol.

Merk op dat als de richting D er voor zou zorgen dat T_1 links van de tape loopt, dat dan in T_1 de h_a feitelijk niet bereikt wordt $\Rightarrow$ T_1 is dan nee-instantie van Accepts-Λ

23:10

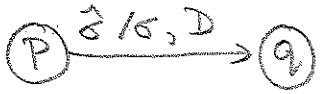
(c)

Aan te tonen : WritesSymbol $\leq$ Accepts-Λ

instanties : (T_2, σ) T_1

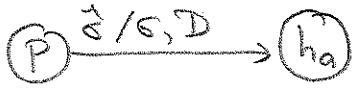
Gegeven een willekeurige instantie (T_2, σ) van WritesSymbol

Dan ontstaat T_1 uit T_2 door (1) iedere transitie van de vorm

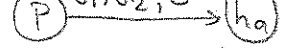


voor willekeurige tapesymbool $\hat{\sigma}$ te vervangen

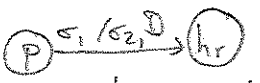
door



en (2) iedere (andere) transitie in T_2 van de vorm



te vervangen door



(dus $\sigma_2 \neq \sigma$),

om te voorkomen dat T_1 Λ accepteert, terwijl T_2 helemaal geen σ schreef bij invoer Λ

23:17

23:22

6 (a)

$$gn(x_0, x_1, x_2, \ldots, x_k) = 2^{x_0} * 3^{x_1} * 5^{x_2} * \ldots * (Pr N_0(k))^{x_k}$$

23:25

(b)

Voor $k \geq 1$ is dus $F(k+1) = F(k-1) + F(k)$

$$gn(F(0), F(1), \ldots, F(k)) = 2^{F(0)} * 3^{F(1)} * \ldots * Pr N_0(k)^{F(k)}$$

$$\Rightarrow F(k-1) = Exponent(k-1, gn(F(0), F(1), \ldots, F(k)))$$

$$F(k) = Exponent(k, gn(F(0), F(1), \ldots, F(k)))$$

$$\Rightarrow$$

$$F(k+1) = Exponent(k-1, gn(F(0), F(1), \ldots, F(k))) + Exponent(k, gn(F(0), F(1), \ldots, F(k)))$$

Inderdaad een functie van k en $gn(F(0), F(1), \ldots, F(k))$

23:29

(c) De functie h moet $F(k+1)$ beschrijven voor elke $k \geq 0$

$$F(k+1) = \begin{cases} 1 & \text{als } k = 0 \quad (\text{want } F(1) = 1) \\ Exponent(k-1, gn(F(0), F(1), \ldots, F(k))) + \\ Exponent(k, gn(F(0), F(1), \ldots, F(k))) & \text{als } k \geq 1 \end{cases}$$

Dit moet gelijk zijn aan $h(k, gn(F(0), F(1), \ldots, F(k)))$

Dat betekent dat

$$h(x_1, x_2) = \begin{cases} 1 & \text{als } x_1 = 0 \\ \text{Exponent}(x_1 - 1, x_2) + \text{Exponent}(x_1, x_2) & \text{als } x_1 \geq 1. \end{cases}$$

23:37