نظریه محاسبه - جلسه هفتم

$NSA_1 = \{ w \mid w = e(T) \text{ , } T \text{ ، } w \text{ را نمی پذیرد} \}$ $\qquad$ $NSA_2 = \{ w \in \{0,1\}^* \mid w \text{ که توصیفش } w \text{ نیست} \}$

$NSA = NSA_1 \cup NSA_2$

قضیه: $NSA \notin RE$

برهان خلف: ماشین T' هست که NSA را می‌پذیرد. $\qquad e(T') = w'$

حالت I) $w' \in NSA \Rightarrow T'$ باید w' را بپذیرد. (چون T' پذیرنده‌ی NSA است.)

$\Downarrow$ با توجه به تعریف NSA ، ماشین با توصیف w' (یعنی T') ، w' را نمی پذیرد. ※

حالت II) $w' \notin NSA \Rightarrow T'$ ، w' را نمی پذیرد. (چون T' پذیرنده‌ی NSA است.)

$\Downarrow$ با توجه به تعریف NSA ، T' ، w' را می پذیرد. ※

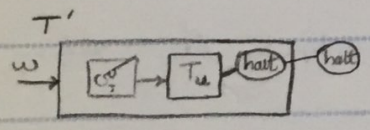
$\Leftarrow$ هر دو حالت به ※ رسید؛ در نتیجه $NSA \notin RE$

$SA = \{ w \mid w = e(T) \text{ , } T \text{ ، } w \text{ را می پذیرد} \}$

قضیه: $SA \notin R$, $SA \in RE$

اگر $SA \in R$ باشد، آن‌گاه $\overline{SA} \in R$. ※ $\Rightarrow SA \notin R$

اثبات $SA \in RE$: T' را می‌سازیم.

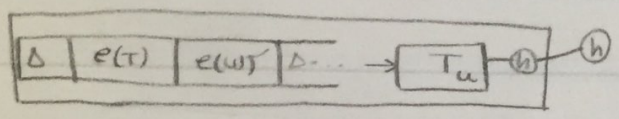


اگر T ، w را بپذیرد، T' هم w را می پذیرد و اگر T ، w را نپذیرد، T' هم w را نمی پذیرد. $\Rightarrow L(T') = SA$

Scanned by CamScanner

نظریه محاسبه - جلسه هشتم

مسئله‌ی توقف: ورودی: توصیف ماشین T و رشته‌ی w - خروجی: آیا T روی w توقف می‌کند؟
halting



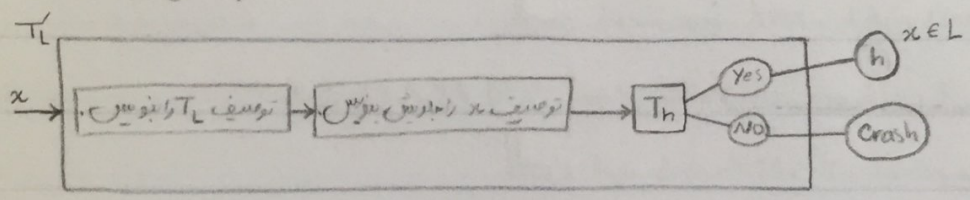
- مسئله‌ی توقف در RE هست.

- مسئله‌ی توقف در R نیست.

برهان خلف: ماشین T_h وجود داشته باشد که مسئله‌ی توقف را تصمیم بگیرد.

یک زبان RE مثل L را با ماشین T_L در نظر بگیرید.

$T_L(x)$ ← توقف در زمان متناهی جواب در زمان متناهی / تا ابد ادامه می‌دهد.



$L(T'_L) = L$ و T'_L در زمان متناهی جواب می‌دهد. ⟸ $L \in R$ ⟸ بنابر فرض خلف هر زبان عضو RE عضو R است.

اما می‌دانیم $SA \in RE$ و $SA \notin R$. ※

تعریف تقلیل: $L_1 \in \Sigma_1^*$ و $L_2 \in \Sigma_2^*$: می‌گوییم L_1 قابل تقلیل (تقلیل پذیر) به L_2 است اگر $f: \Sigma_1^* \to \Sigma_2^*$ قابل محاسبه (باتوریگ) باشد که

$$\forall x \in \Sigma_1^* \quad x \in L_1 \iff f(x) \in L_2$$

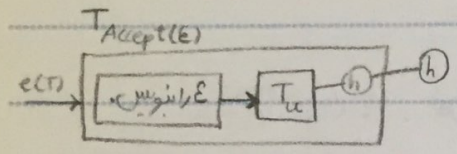
نشان می‌دهیم: $L_1 \leq L_2$ (مثال: $SA \leq Halting$)

قضیه: اگر $L_1 \leq L_2$ و $L_2 \in R$ آن‌گاه $L_1 \in R$ است.

- اگر بخواهیم نشان دهیم $L \notin R$، کافی است L' بیابیم که $L' \leq L$ و $L' \notin R$.

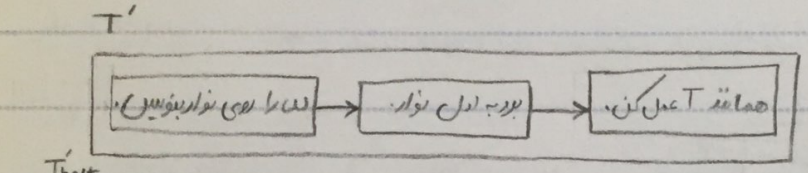
Scanned by CamScanner

مسئله‌ی پذیرفتن $\mathcal{E}$: ورودی: ماشین تورینگ T - خروجی: آیا T ، $\mathcal{E}$ را می‌پذیرد؟
accept($\mathcal{E}$)



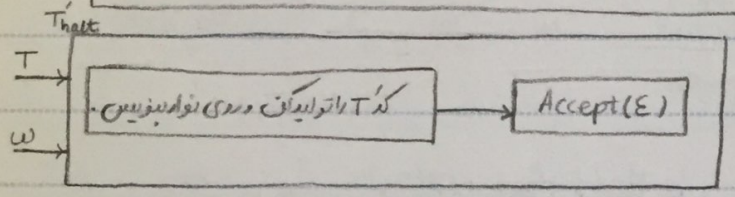
$Accept(\mathcal{E}) \in RE$ -

$Accept(\mathcal{E}) \notin R$ -



T' روی $\mathcal{E}$ توقف می‌کند. $\Leftarrow$ T روی w توقف می‌کند.

برهان خلف:



※ $T'_{halt} \in R \Leftarrow Accept(\mathcal{E}) \in R$

Scanned by CamScanner

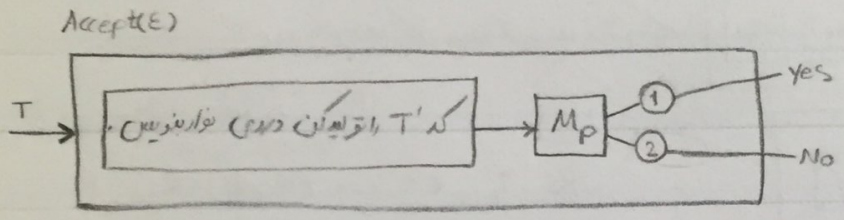
نظریه محاسبه - جلسه دهم

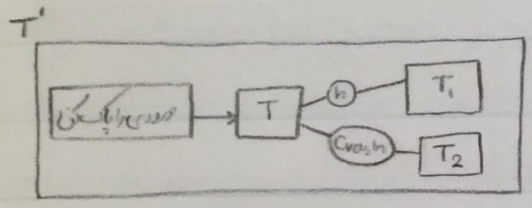
سؤال غیربدیهی: جوابش هم Yes و هم No باشد. $\exists\, T_1, T_2 : e(T_1) \in L,\ e(T_2) \notin L$

قضیه‌ی Rice : هر سوال غیربدیهی که ورودی اش ماشین تورینگ باشد و در مورد زبان ماشین تورینگ باشد، در R نیست.

P: سوال غیربدیهی $(e(T_2) \notin L_P \ ,\ e(T_1) \in L_P)$

فرض خلف: ماشین M_P هست که در زمان متناهی یا توقف می‌کند و یا جواب می‌شود.





اگر $T_\emptyset$ خاصیت P را نداشته باشد، 1= halt و 2= crash می‌باشد و اگر $T_\emptyset$ خاصیت P را داشته باشد، 1=crash و 2= halt خواهد بود.

مسئله‌ی PCP : ورودی: مجموعه‌ای از دومینوها - خروجی: آیا می‌توان تعدادی متناهی از هر کدام به هر تعداد آن دومینوها را به ترتیبی کنار هم چید که رشته‌ی بالا و پایین مساوی شوند؟ از هر کدام هر تعدادی که بخواهیم. (حداقل در کل باید یک دومینو بگذاریم، یعنی جواب غیرتهی می‌خواهیم.)

- $PCP \notin R$, $PCP \in RE$

مسئله: گرامری داریم. آیا مبهم است؟

- تمام ماشین تورینگ هایی را در نظر بگیرید که n حالت (state) دارند (الفبای {0,1}). به ازای هر کدام که روی رشته‌ی 1^n توقف می‌کند، تعداد 1 هایی که روی نوار در انتها نوشته شده است را در نظر بگیرید. ما گفتیم این اعداد را b(n) تعریف می‌کنیم. $(b: \mathbb{N}^0 \to \mathbb{N}^0)$ ← نمی‌توان b را محاسبه کرد. ← اثبات با فرض خلف (M_b وجود دارد) ← T' یک 1 بیشتر می‌نویسد.

Scanned by CamScanner

نظریه محاسبه - جلسه یازدهم

قضیه: $b(n)$ محاسبه پذیر نیست (بازگشتی نیست).

فرض خلف: T_b که $b(n)$ را حساب می کند ← تعداد حالت های $m = T_b$

T_b : $T_b \to T_1$ اجرا تمام شده می رود جلو، سمی یک جای خالی می نویسد، برمی گردد عقب ← m' حالت

$T_b(m') = b(m')$ $\qquad$ $T_b \to T_1\ (m') = b(m')+1$ $\qquad$ ※

m' حالت

توابع مقدماتی (Initial):

1. تابع ثابت: $C_a^k : W^k \to W \qquad k, a \geq 0 \qquad C_a^k(\underbrace{x}_{x_1, \dots, x_k}) = a \qquad k=0 \to \underline{a}$

2. بعدی (successor): $S : W \to W \qquad S(x) = x+1 \qquad x \geq 0$

3. تصویر (projection): $P_i^k(x_1, \dots, x_i, \dots, x_k) = x_i \qquad P_i^k : W^k \to W \qquad k \geq 1 \qquad 1 \leq i \leq k$

تعریف ترکیب تابع:

$f : W^k \to W \quad , \quad 1 \leq i \leq k : \ g_i : W^m \to W$

$h = f(g_1, \dots, g_k) : W^m \to W \qquad h(X) = f(g_1(X), \dots, g_k(X))$

k تایی

h از روی f و $g_1, \dots, g_k$ با ترکیب به دست آمده است.

عملگر بازگشتی ابتدایی (primitive):

$n \geq 0 \qquad g : W^n \to W \qquad h : W^{n+2} \to W \qquad f : W^{n+1} \to W$

$f(X, 0) = g(X) \qquad f(X, k+1) = h(X, k, f(X, k))$

n تایی

f از روی g و h با عملگر بازگشتی ابتدایی به دست می آید.

توابع بازگشتی ابتدایی: 1- همه ی توابع مقدماتی 2- از ترکیب توابع بازگشتی ابتدایی 3- از عملگر بازگشتی ابتدایی با توابع بازگشتی ابتدایی 4- فقط همین ها

Scanned by CamScanner

Subject Date

چندمثال برای تابع بازگشتی مقدماتی:

1. $Add(\overset{x_1}{x}, \overset{x_2}{y}) := f(x,y)$

$f(x,0) = g(x) = P_1^1$

$f(x,k+1) = h(x,k,f(x,k)) = S(P_3^3)$

2. $Mul(\overset{x_1}{x}, \overset{x_2}{y}) := f(x,y)$

$f(x,0) = g(x) = 0 = C_0^0$

$f(x,k+1) = h(x,k,Mul(x,k)) = Add(P_1^3, P_3^3)$

3. $fact(\overset{x_1}{x}) : f(x)$

$f(0) = g() = 1 = C_1^0$

$f(x+1) = h(x, f(x)) = Mul(S(P_1^2), P_2^2)$

Scanned by CamScanner

چند مثال دیگر برای تابع بازگشتی مقدماتی :

1. $pred(x) = \begin{cases} x-1 & x>0 \\ 0 & x=0 \end{cases}$

$pred(0) = g() = 0 \qquad pred(k+1) = h(k,t) = P_1^2$

2. $Sub(x,y) = \begin{cases} x-y & x \geq y \\ 0 & \text{در غیر این صورت} \end{cases}$

$Sub(x,0) = g(x) = P_1^1 = x \qquad Sub(x,k+1) = h(x,k,\overbrace{Sub(x,k)}^{t}) = pred(P_3^3)$

3. $AND(x,y) = pred(Add(P_1^2, P_2^2))$

4. $OR(x,y) = NOT(AND(NOT(P_1^2), NOT(P_2^2)))$

5. $\overset{NOT(x)}{\cancel{NOT(x,y)}} = Sub(C_1^1, P_1^1)$

$\Rightarrow$ هر تابع منطقی روی چند متغیر، ابتدایی بازگشتی است.

- در فرایند تعریف هر تابع بازگشتی مقدماتی، می‌توان از اعمال زیر سود جست :

1. عدد ثابت را می‌توان جایگزین متغیر کرد.
2. متغیر اضافه، اضافه کرد.
3. جایگشت دوری ما
4. متغیر تکرار کرد.

قضیه : اگر به ازای هر x دقیقاً یکی از $P_1, \dots, P_k$ درست باشد و داشته باشیم :

$$f(x) = \begin{cases} f_1(x) & \text{if } P_1(x) \\ f_2(x) & \text{if } P_2(x) \\ \vdots & \\ f_k(x) & \text{if } P_k(x) \end{cases}$$

اگر f_i ها مقدماتی بازگشتی باشند، f هم ابتدایی بازگشتی است.

Scanned by CamScanner

چند مثال دیگر برای تابع بازگشتی مقدماتی:

1. $LT(x,y) = f(Sub(y,x))$ $\qquad f(x) = \begin{cases} 1 & x \geq 1 \\ 0 & x = 0 \end{cases}$

$f(0) = 0 \qquad f(k+1) = h(k, \overset{z}{f(k)}) = 1$

2. $Mod(x,y) = R(y,x) \qquad R(y,0) = 0 \qquad R(y,x) = \begin{cases} R(y,x-1)+1 & R(y,x-1)+1 \neq y \\ 0 & R(y,x-1)+1 = y \end{cases}$

$h(y,k,z) = \begin{cases} z+1 & z+1 \neq y \\ 0 & z+1 = y \end{cases}$

3. $Div(x,y) = Q(y,x) \qquad Q(y,x+1) = h(y,x,\overset{Q(y,x)}{z}) = \begin{cases} x+1 & Mod(x,y) = 0 \\ z & \end{cases}$

نکته: توابع بازگشتی اولیه تام و محاسبه پذیرند.

- تابع روبرو محاسبه پذیر است: ماشین تورینگ T_a روی $\overset{x \text{ ورودی رشته}}{S_x}$ بعد از y مرحله توقف می کند: $H(x,y)$

اما تابع روبرو محاسبه پذیر نیست: (فرق در وجود و حضور) $Halt(x) = \exists y : H(x,y)$

تعریف quantification کران دار: $\overset{\text{گزاره ی منطقی}}{P}$: $(n+1)$ ورودی و ۰ را برگرداند:

quantification کران دار وجودی P: $E_P(x,k-1)$ OR $P(x,k) = E_P(x,k) = (\exists\, 0 \leq y \leq k \quad P(x,y)=1)$

quantification کران دار عمومی P: $A_P(x,k-1)$ AND $A_P(x,k) = A_P(x,k) = (\forall\, 0 \leq y \leq k \quad P(x,y)=1)$

← اگر P، total باشد، E_P و A_P هم total هستند

تعریف کمینه سازی کران دار: $\overset{\text{گزاره ی منطقی}}{P}$: $(n+1)$ ورودی و ۰ را برگرداند:

$m_P : W^{n+1} \to W \qquad m_P(x,k) = \begin{cases} \min\{y \mid \overset{0 \leq y \leq k}{P(x,y)=1}\} \\ k+1 & \text{اگر مجموعه تهی بود} \end{cases}$

$m_P = \mu_y^k [P(x,y)]$

Scanned by CamScanner

قضیه: اگر p بازگشتی اولیه باشد $\mu_y^k[p(x,y)]$ بازگشتی اولیه است.

$$m_p(x,0) = \begin{cases} 0 & p(x,0)=1 \\ 1 & \end{cases}$$

$$m_p(x,k+1) = h(x,k,\overbrace{m_p(x,k)}^{z}) = \begin{cases} z & z \leq k \\ k+1 & z=k+1 \wedge p(x,k+1)=1 \\ k+2 & z=k+1 \wedge p(x,k+1)=0 \end{cases}$$

چند مثال:

1. $Sq(x,y) = (x=y^2) = EQ(Mul(y,y),x)$

2. $PerfectSquare(x) = \mu_y^x[Sq(x,y)]$

3. $Prime(x) = ((\exists\, 0 \leq y \leq x-1 \quad 2 \leq y \wedge Mod(x,y)=0) > x-1)$

4. $prNo(x)$ $\quad prNo(0)=2 \quad prNo(1)=3$

$$prNo(k+1) = h(k,\overbrace{prNo(k)}^{z}) = \mu_y^{prNo(k)!+1}[y>z \wedge prime(y)]$$

Scanned by CamScanner

نظریه محاسبه - جلسه چهاردهم

تعریف کمینه سازی بی‌کران :

$$p : W^{n+1} \to \{0,1\} \qquad \mu_y[p](x,y) = M_p(x) = \min\{y \mid p(x,y)\} \qquad M_p : W^n \to W$$

- اگر در تعریف بالا، $p(x,y)$ به صورت $f(x,y)=0$ باشد، اصطلاحاً $M_p(x)$ را با $M_f(x)$ نشان می‌دهند.

تعریف تابع μ - بازگشتی یا بازگشتی : توابعی که در تعریف آن ها بتوان از اعمال زیر استفاده کرد:

1. تابع مقدماتی 2. ترکیب 3. عمل بازگشتی اولیه 4. کمینه سازی بی‌کران (کلی) توابع total

قضیه : توابع μ - بازگشتی محاسبه پذیرند.

تکنیک μ * - می‌خواهیم نشان دهیم تابع فیبوناتچی بازگشتی اولیه است.

$$f(n) = f(n-1) + f(n-2) \qquad f(0) = f(1) = 1$$

تعریف می‌کنیم: $\forall n \geq 0 : f'(n) = 2^{f(0)} 3^{f(1)} \cdots prNo(n)^{f(n)}$

همچنین برای اینکه بفهمیم توان z در x چند است، تعریف می‌کنیم:

$$Exponent(x,z) = \min_{i \leq x}\{i \mid mod(x, z^i) \neq 0\} - 1$$

حال نشان می‌دهیم که f' بازگشتی اولیه است:

$$f'(0) = 2^{f(0)} = 2$$

$$f'(n) = h(n, \overline{z}, f'(n-1)) = z \times prNo(n-1)^{Exponent(z, prNo(n-1))} \times prNo(n-2)^{Exponent(z, prNo(n-2))}$$

حال با توجه به تساوی روبرو تابع فیبوناتچی هم بازگشتی اولیه است:

$$f(n) = Exponent(f'(n), prNo(n))$$

تکنیک golden numbering : وقتی بخواهیم چند عدد را در یک عدد جای دهیم از $gn(x_0, \dots, x_n)$ استفاده می‌کنیم که به صورت روبرو تعریف می‌شود.

$$gn(x_0, \dots, x_n) = \prod_{k=0}^{n} prNo(k)^{x_k}$$

- نمایش یک configuration با استفاده از gn :

$$\text{نوار} = gn(\overbrace{\text{محتوای خانه‌ی ۱}}^{x_1}, \overbrace{\text{محتوای خانه‌ی ۲}}^{x_2}, \dots) = 2^{x_1} \times 3^{x_2} \times \dots$$

$$\text{پیکره ماشین} = gn(\text{حالت}, \text{جای سر}, \text{نوار}) = 2^{\text{حالت}} \times 3^{\text{جای سر}} \times 5^{2^{x_1} \times 3^{x_2} \times \dots}$$

حالت: $h=0$ ، شروع $=1$

Scanned by CamScanner

هدف: هر تابع محاسبه پذیر، μ-بازگشتی است.

$\underset{\mu\text{-بازگشتی}}{f} \leftarrow TM$

$$f = Result_T \circ f_T \circ InitConfig^{(n)}(x_1, \dots, x_n)$$

1. $InitConfig^{(n)}$ بازگشتی اولیه است. $\quad gn(1,1,t^{(n)}(x_1,\dots,x_n))$

2. $Result_T$ بازگشتی اولیه است. $\quad \min_{i \le x} (Mod(Exponent(x,5), PrNo(i)) \neq 0)$

3. f_T، μ-بازگشتی است. $\quad m$: یک Configuration

$$Symbol(m) = Exponent(Exponent(m,5), PrNo(Exponent(m,3)))$$

$$State(m) = Exponent(m,2)$$

$$Pos^{(n)}(m) = Exponent(m,3)$$

$$Tape(m) = Exponent(m,5)$$

$$\underset{\text{فعل جدول ها}}{Move_T(m)} = \begin{cases} gn(\ ,\ \ , Tape(m)/PrNo(Pos^{(n)}(m))^{\text{عدد قبلی}} \times PrNo(Pos^{(n)}(m))^{\text{عدد جدید}} & state(m)=1 \wedge Symbol(m)=1 \\ \vdots & \vdots \end{cases}$$

$Trace_T(m,k)$: آیا پس از k مرحله T روی m توقف می‌کند؟ ($Trace_T$ = پیکربندی بعد از k مرحله یا توقف)

$$Trace_T(m,k+1) = h(m,k,z) = \begin{cases} z & h = z \text{ حالت} \\ Move_T(z) & h \neq z \text{ حالت} \end{cases}$$

$$\text{تعداد مراحل اجرا} = \min_k (Exponent(Trace_T(InitConfig^{(n)}(x), k), 2) = 0)$$

$$f_T = Trace(x, \text{تعداد مراحل}_T(x))$$

$\Leftarrow$ همه‌ی توابع μ-بازگشتی را می‌توان با یک بار استفاده از کمینه‌سازی بی‌کران ساخت. $\quad$ تابع $\leftarrow$ TM $\leftarrow$ تابع

Scanned by CamScanner

نظریه محاسبه - جلسه پانزدهم

M : ماشین تورینگ

- چندکاری که بلدیم (قابل محاسبه است) به شرح زیر است:

1. $M \leftarrow$ کد رشته‌ای از ۱ و ۰ $\leftarrow$ یک عدد $\langle M \rangle$

2. عدد $\langle M \rangle \leftarrow M$

3. شماره‌گذاری برای ماشین‌ها: $M_0, M_1, \dots$ $\langle M_i \rangle < \langle M_{i+1} \rangle$

4. $\langle M_i \rangle \leftarrow i$

```
t←0
k←0
آیا k کد یک ماشین است؟
t++
if (t=i)
    k را برگردان.
```

5. $\langle M \rangle \leftarrow$ اندیس M

6. شماره‌گذاری برای تمام توابع محاسب‌پذیر $f: W^n \to W$ قابل محاسب است.

$i \to \varphi_i^{(n)}$ که i این تابع محاسب‌پذیر n متغیره ممکن است total نباشد.

total

← درست بالا بی‌نهایت ماشین تورینگ هست که یک تابع را محاسبه می‌کنند.

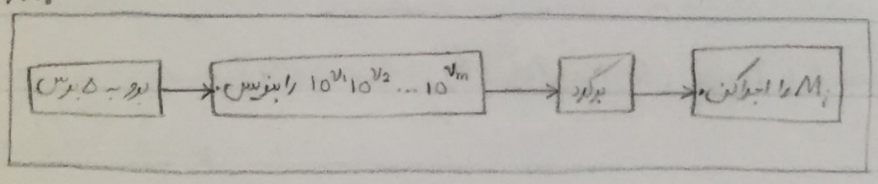
7. ماشین تورینگ جهان‌شمول برای تمام توابع محاسب‌پذیر n متغیره: $U_n = \varphi_i^{(n)}(x)$

قضیه S-m-n: تابع کلی محاسب‌پذیر $S: W^{n+1} \to W$ هست که

$$\varphi_{S(i,v)}^{(n)} = \varphi_i^{(m+n)}(\underbrace{\ }_{n}, \underbrace{v}_{m})$$

$v = (v_1, v_2, \dots, v_n)$

M:



$M(x) \rightsquigarrow M_i(x, v)$

که اندیس $= S(i, v)$

$S(i, v)$:

1. کد M_i را پیدا کن.

2. کد M را تولید کن.

3. اندیس M را برگردان بیاب.

قضیه: سری $f_0, f_1, \dots$ را در نظر بگیرید. اگر $f_i : W \to W$ (محاسب‌پذیر) $(i = 0, 1, \dots)$ باشد و بتوانیم سری را محاسبه کنیم، آن‌گاه وجود دارد تابع کلی محاسب‌پذیر f که:

$$\forall i \quad f_i \neq f$$

کلی محاسب‌پذیر

$$f(i) = f_i(i) + i \qquad \forall i : f \neq f_i$$

نتیجه: اگر $\varphi : W \to W$ ؛ این تابع محاسب‌پذیر باشد، آن‌گاه:

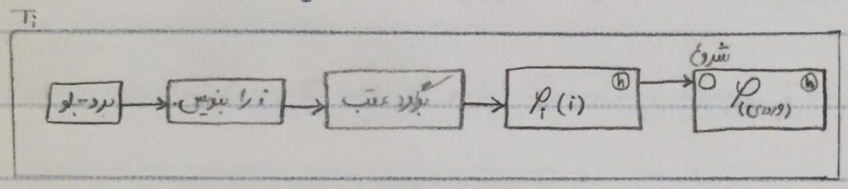
$$\{ n \mid \varphi_n \text{ کلی} \} \notin RE$$

توضیح: مجموعه‌ی توابع کلی شمارا است، اما تابع محاسب‌پذیر از $\mathbb{N}$ به آن وجود ندارد.

لم: اگر $\epsilon : W \to W$ تابعی با دامنه‌ی تهی باشد و $\varphi : W \to W$ تابعی جزئی و قابل محاسبه با دامنه‌ی غیرتهی باشد، آن‌گاه تابع f هست که کلی و قابل محاسبه است و داریم:

$$\varphi_{f^{\varphi}(i)} = \begin{cases} \varphi & i \in \{ j \mid j \in P_j \} \\ \epsilon & \text{در غیر این صورت} \end{cases}$$

تابع f^{φ}:



1. $\langle T_i \rangle$ را محاسبه کن.
2. اندیس T_i را بگیر، $(v =)$

$$\text{رفتار } T_i = \text{رفتار } \varphi_v = \begin{cases} \text{رفتار } \varphi & i \in K \\ \text{رفتار } \epsilon & i \notin K \end{cases}$$

قضیه‌ی Rice: اگر $\emptyset \neq I \subsetneq W$ و I قابل محاسبه باشد، آن‌گاه وجود دارد $i \in I$, $j \notin I$ که: $\varphi_i = \varphi_j$

برهان خلف: قضیه را به ازای حالتی که تابع ϵ خارج از I باشد اثبات می‌کنیم در حالت برعکس آن، می‌توان به این روش اثبات کرد که تکمیل I در قضیه صدق می‌کند در نتیجه خود I هم در قضیه صدق می‌کند.

$$\epsilon \notin I \text{ و } \exists \varphi \in I \text{ , } K = \{ j \mid j \in P_j \} \qquad i \in K \iff \varphi_{f^{\varphi}(i)} \neq \epsilon$$

تابع وقوع I

$$\chi_I \circ f^{\varphi}(i) = \begin{cases} 0 & i \notin K \\ 1 & i \in K \end{cases} \implies \chi_K \text{ محاسب‌پذیر} \to \times$$

نتیجه: اگر $\emptyset \neq I \subsetneq W$ و I وابسته به اندیس باشد، محاسب‌پذیر نیست. وابسته به اندیس: $i \in I, \varphi_i = \varphi_j \iff j \in I$

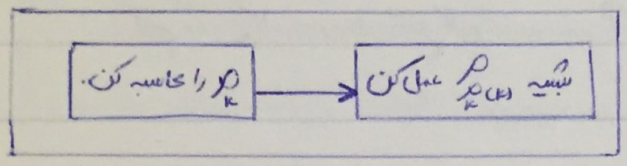
قضیه‌ی بازگشت (fixed point): اگر $f : W \to W$ کلی (تام) و محاسب‌پذیر باشد، آن‌گاه وجود دارد $i \in W$ که:

$$\varphi_i = \varphi_{f(i)}$$

Scanned by CamScanner

اثبات قضیه‌ی بازگشت :

$\rho_{S(k)}$



$$\rho_\nu : k \longrightarrow f(S(k))$$

$$\rho_{S(k)} = \rho_{\rho_k(k)}$$

چون هربار $\rho_{\rho_k(k)}$ موردی بی‌نهایت بیفتد یا توقف کند، $\rho_{S(k)}$ دقیقاً همان کار را می‌کند.

$$\xrightarrow{k=\nu} \rho_{S(\nu)} = \rho_{\rho_\nu(\nu)} \quad ①$$

$$\rho_\nu(k) = f(S(k)) \xrightarrow{k=\nu} \rho_\nu(\nu) = f(S(\nu)) \quad ②$$

$$\xrightarrow{①,②} \rho_{S(\nu)} = \rho_{\rho_\nu(\nu)} = \rho_{f(S(\nu))} \xrightarrow{S(\nu)=i} \rho_i = \rho_{f(i)}$$

Scanned by CamScanner

- می‌خواهیم ماشین تورینگی تولید کنیم که خروجی‌اش کد خودش باشد.

ابتدا دو ماشین تورینگ A و B را در نظر بگیرید که هر کدام کد دیگری را به عنوان خروجی می‌دهد، سپس B را گسترش می‌دهیم.

تابع q: رشته ← ماشین تورینگ (کد ماشین تورینگی که w را می‌نویسد: $q(w)$) $\Leftarrow$ $\langle A\rangle = q(\langle B\rangle)$

- تعریف B: $q(w)$ را حساب کن و بنویس.

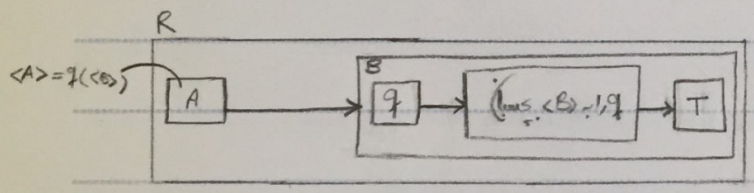
- تعریف A: $\langle A\rangle = q(\langle B\rangle)$

حال تعریف B را گسترش می‌دهیم. ماشین تورینگ M را در نظر بگیرید که ابتدا A را اجرا می‌کند سپس B را اجرا می‌کند.

- تعریف جدید B: $q(w)$ را حساب کن، چند انتقال اضافه کن و به ماشین بگو $\langle B\rangle$ (همان ورودی) اضافه کن و کد نهایی را بنویس.

⇐ حال با همان تعریف قبلی A، ماشین M با خاصیت مورد نظر ابتدایی تولید شد.

قضیه: به ازای هر ماشین تورینگ T با ورودی (w,x)، ماشین تورینگ R وجود دارد که: $R(w) = T(\langle R\rangle, w)$ $\quad T(w, \langle R\rangle)$



Scanned by CamScanner

قضیه: اگر $A_{TM} = \{(\langle M\rangle, w) \mid M$ ، w را می‌پذیرد$\}$ ، A_{TM} تصمیم‌پذیر نیست.

فرض خلف: M_A وجود دارد که زبان‌ش A_{TM} است.

B:

1. کد خودش را محاسبه می‌کند.
2. $b \leftarrow M_A(\langle B\rangle, w)$
3. برعکس b عمل کن. (پذیرش/پذیرش)

$B(w)$:
- می‌پذیرد ← با جواب M_A تناقض
- نمی‌پذیرد ← با جواب M_A تناقض

$\Leftarrow$ ※

قضیه: اگر $MIN_{TM} = \{\langle M\rangle \mid$ بین ماشین‌هایی که همزبان با M هستند، $\langle M\rangle$ کمینه است$\}$ ، آن‌گاه: $MIN_{TM} \notin RE$

فرض خلف: $E_{MIN_{TM}}$ وجود دارد که همه‌ی اعضای MIN_{TM} را تولید می‌کند.

C:

1. کد خودش را محاسبه می‌کند.
2. $E_{MIN_{TM}}$ را اجرا می‌کند تا یک مورد پیدا شود که بزرگتر از $\langle C\rangle$ است. ← D
3. D را اجرا می‌کند.

← ماشین D هم‌رفتار با C است و $\langle C\rangle < \langle D\rangle$ و $\langle D\rangle \in MIN_{TM}$ $\Leftarrow$ ※

قضیه‌ی نقطه ثابت: اگر تابع محاسبه‌پذیر و تام t داشته باشیم، آن‌گاه ماشین F هست که: $L(t(\langle F\rangle)) = L(F)$

F:

1. کد خودت را بساز. $\langle F\rangle$
2. $t(\langle F\rangle)$ را محاسبه کن.
3. $t(\langle F\rangle)$ را روی w اجرا کن.

← رفتار F $\equiv$ رفتار $t(\langle F\rangle)$

- اگر ماشین M_A وجود داشته باشد که در ورودی آن $(\langle M\rangle, w)$ است و اگر زبان M بازگشتی باشد، پاسخ می‌دهد که آیا M، w را می‌پذیرد یا نه. ماشین B را در نظر بگیرید که کد خودش را محاسبه می‌کند و $b \leftarrow M_A(\langle B\rangle, w)$ و سپس برعکس b عمل می‌کند. اگر زبان B بازگشتی باشد به تناقض می‌رسیم. اما نکته‌ای که وجود دارد این است که B به M_A نیاز دارد و M_A به B و در اجرای B به دوری غایت می‌رسیم و در نتیجه زبان B بازگشتی نیست. (با هر ورودی‌ای به دوری غایت می‌رسد)

Scanned by CamScanner

- گزاره‌ها را می‌توان به صورت روبرو نوشت: $Q_1 x_1 Q_2 x_2 \cdots Q_n x_n [\psi]$ $\quad Q_i = \exists$ یا $\forall$

ψ: گزاره‌ی منطقی شامل $\neg, \wedge, \vee, \rightarrow, \leftrightarrow, =, \ldots$ (هر کدام یک رابطه است.)

تعریف $Th(\mathbb{N}, +)$: همه‌ی گزاره‌های صحیح در نظریه اعداد روی اعداد طبیعی با استفاده از رابطه‌های $\neg, \wedge, \vee, \rightarrow, \leftrightarrow, =, +$

تعریف $Th(\mathbb{N}, +, \times)$: همه‌ی گزاره‌های صحیح در نظریه اعداد روی $\mathbb{N}$ با استفاده از $\neg, \wedge, \vee, \rightarrow, \leftrightarrow, =, +, \times$

Scanned by CamScanner

نظریه محاسبه - جلسه بیستم

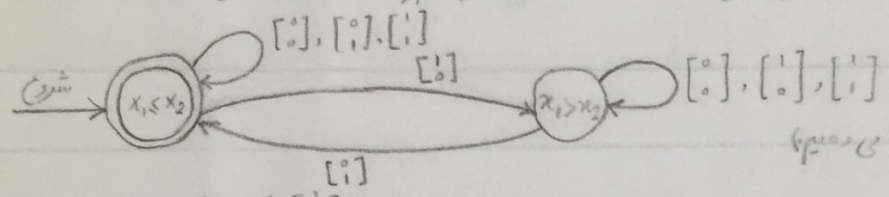
قضیه: $Th(\mathbb{N},+) \in R$

عبارت دبرو را در نظر بگیرید:

$$Q_1 x_1 Q_2 x_2 \dots Q_n x_n \ \psi \in Th(\mathbb{N},+)$$

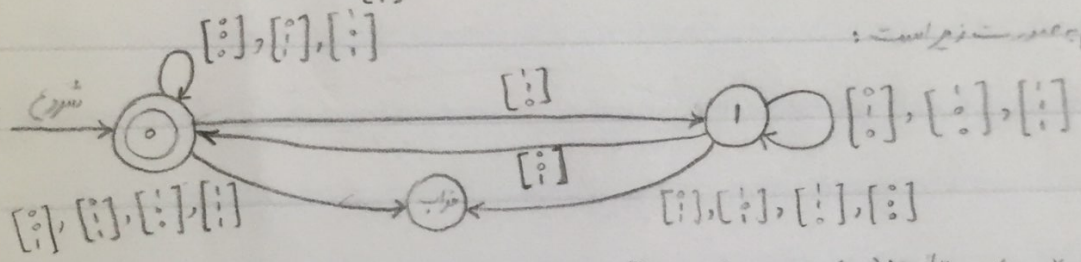
ابتدا ثابت می‌کنیم عبارت منطقی ψ ، اتوماتا دارد:

بدون توضیح برای $\leq$ ، $=$ ، $\wedge$ ، $\vee$ ، $\neg$ می‌توان اتوماتا ساخت؛ مثلاً اتوماتا برای $\leq$ به صورت زیر است:



(بیت‌ها از راست به چپ به اتوماتا می‌دهیم)

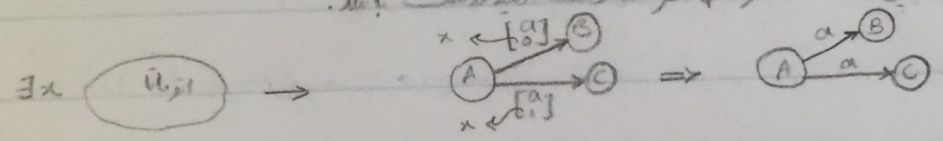
اتوماتا برای $+$ هم به صورت زیر است:



$$+(x_1,x_2,x_3,x_4) \equiv +(x_1,x_2,x_a) \wedge +(x_a,x_3,x_4)$$

$\Leftarrow$ می‌توان برای ψ اتوماتا ساخت.

حال کافی است اگر برای عبارت A ، اتوماتا داریم، برای عبارت های $\exists x\, A$ ، $\forall x\, A$ هم اتوماتا بسازیم. در نهایت عبارت اولیه، یک اتوماتا تبدیل می‌شود که حالت شروع، حالت نهایی است، اگر و تنها اگر عبارت اولیه درست باشد.



$$\forall x\, \psi \equiv \neg \exists x \neg \psi$$

← اگر اتوماتای نهایی را به DFA تبدیل کنیم و آن را کمینه کنیم، به یک ماشین state تبدیل می‌شود.

Scanned by CamScanner

نظریه محاسبه - جلسه بیست و یکم

سری پیکربندی صحیح است.

سری از پیکربندی

لم: موجود دارد تبدیل ($M,w \leftarrow$ گزاره‌ی $\phi_{M,w}$) که: M روی w توقف کند. $\Longleftrightarrow$ $\exists x\ \phi_{M,w}$

اثبات رسمی: برای قضیه‌ی ϕ، $\pi = \langle S_1, \dots, S_n \rangle$ یک اثبات رسمی است.

S_1 ← فرض اولیه

S_2 ← بارش‌های استنتاج

$\vdots$

$S_n = \phi$

یک اثبات رسمی باید:

1- قابل چک کردن با ماشین باشد. M_π ← $M_\pi : \langle \phi, \pi \rangle$ ← آیا π اثباتی برای ϕ هست؟

2- هر چیزی که اثبات دارد، صحیح است.

قضیه: $RE \supseteq L = \{\phi \mid$ اثبات دارد $\phi\}$ $\quad$ L = همه‌ی قضیه‌های اثبات پذیر

همه‌ی: ϕ ← * رشته‌ی بعدی را تولید کن. ← با M_π چک کن آیا اثباتی برای ϕ هست؟ بله ← بپذیر / خیر ← برو به *.

قضیه: $\overbrace{Th(\mathbb{N}, +, \times)}^{Th} \notin R$

فرض خلف: ماشین توریگ M' برای Th هست. ماشین زیر را در نظر بگیرید: (ورودی: M,w)

1- $\phi_{M,w} \leftarrow M,w$ $\quad$ 2- $M' \leftarrow \exists x\ \phi_{M,w}$ ← بله ← بپذیر / خیر ← جواب.

به تناقض رسیدیم: چون ماشین توصیف شده، ماشین halting است که می‌دانیم وجود ندارد.

قضیه: گزاره‌های صحیحی عضو $Th(\mathbb{N}, +, \times)$ وجود دارند که اثبات ندارند.

گزاره‌ی ψ و $\neg\psi$ ای می‌یابیم که هیچ کدام اثبات ندارند $\Longleftarrow$ یکی‌شان صحیح است و اثبات ندارد.

$\psi = \exists x\ \phi_{M,w}$ ← هست M,w که اینکه M روی w توقف می‌کند برایش اثباتی وجود ندارد (و هم چنین برای نقیضش) زیرا در غیر این صورت مسئله‌ی halting جواب می‌داشت.

مثال: ماشین زیر را در نظر بگیرید. (S) گزاره‌ی $\exists x [\phi_{S,\epsilon}]$ و نقیض آن هیچ کدام اثبات ندارند.

S: که سنوش را تولید کند. ← گزاره‌ی $\neg \exists x [\phi_{S,\epsilon}] = \psi$ را بساز. ← چک کن آیا ψ اثبات دارد؟

(*اثبات‌های مختلف را تولید کن. (π) ← آیا π اثباتی برای ψ است؟ بله ← بپذیر / خیر ← *.)

آیا S، ϵ را می‌پذیرد؟ بله ← S، ϵ را نمی‌پذیرد. ← ※ / خیر ← S، ϵ را می‌پذیرد. ← ※

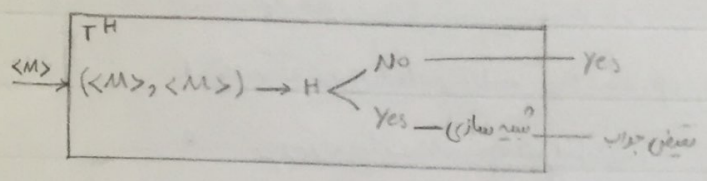
MICRO

Scanned by CamScanner

نظریه محاسبه - جلسه بیست و دوم

اوراکل : فرض می‌کنیم دستگاهی داریم که کاری را انجام می‌دهد که ماشین تورینگ نمی‌تواند انجام دهد و می‌خواهیم ببینیم چه اتفاقی می‌افتد و چه تفاوتی ایجاد می‌شود؟ مثلاً دستگاهی که مسئله‌ی halt را حل می‌کند. به این دستگاه می‌گوییم اوراکل.

مثال: آیا NSA ماشین با اوراکل H دارد؟ بله؛ (دستگاه حل توقف یا halt)



قضیه : $RE \subseteq R^H$ (R^H: R با اوراکل H (halt یا توقف))

همانند مثال بالا عمل می‌کنیم.

- $R^H \subsetneq$ همه‌ی زبان‌ها

راه I: تعداد ماشین تورینگ‌ها با اوراکل H شمارا است. (ماشین تورینگ ۳ حالت برای اوراکل در نظر می‌گیریم. یک حالت H، یک H_a (یعنی اوراکل پذیرفت) و دیگری H_r (یعنی اوراکل نپذیرفت.) برای همین باز هم جدول transition ها متناهی است و در نتیجه تعداد کل ماشین تورینگ‌ها با اوراکل H شمارا است.) $\Leftarrow$ زبان محاسبه‌پذیر با اوراکل H وجود دارد.

راه II: زبان $\{M^H$ خودش را نمی‌پذیرد$\}$ $NSA^H : \{\langle M^H \rangle \mid$ خارج از R^H است؛ اثبات با قطری‌سازی و مثل اثبات $NSA \notin RE$ است.

- $R \subsetneq R^H \subsetneq R^{H^H} \subsetneq R^{H^{H^H}} \subsetneq \dots$

- هر چیزی را نتوانیم توصیف کنیم، شمارا است، اما تعداد زبان‌ها ناشمارا است؛ پس نمی‌توان ماشین تورینگی یافت که همه‌ی زبان‌ها با آن بتواند توصیف شود. یعنی مثلاً اوراکلی وجود ندارد که همه‌ی زبان‌ها را در R آن بتوان جای داد. اگر باشد، ما دستمان به آن نمی‌رسد.

تعریف تورینگ تقلیل پذیر: A تورینگ تقلیل پذیر به B است ($A \leq_T B$) اگر A را بتوان با ماشین تورینگ با اوراکل B حل کرد.

قضیه: اگر $A \leq_T B$ و B تصمیم پذیر باشد، آن‌گاه A تصمیم پذیر است.

Scanned by CamScanner

Kolmogrov Complexity ←

تعریف پیچیدگی رشته: $d(x)$ = کوتاهترین توصیف رشته‌ی x ← $K(x) = |d(x)|$ ($\langle M\rangle, w$ (M را روی w اجرا کنید x را می‌دهد.)

قضیه: $\exists c: K(x) \leq |x| + c$

M: هیچ کاری نکن. ← x ، $\langle M\rangle$ توصیف x هست. ← $K(x) \leq |\langle M\rangle| + |x|$

قضیه: $\exists c: K(xx) \leq K(x) + c$

فرض: $d(x) = \langle M'\rangle, w$ M: دستور را نه‌تا کن. اگر M را اجرا کنیم و سپس M' را اجرا کنیم توصیف xx

به دست می‌آید. پس: $K(xx) \leq |(\langle M' M\rangle, w)| \Rightarrow K(xx) \leq K(x) + c$

قضیه: $\exists c: K(xy) \leq \underbrace{2K(x)}_{\langle M\rangle, w} + \underbrace{K(y)}_{\langle M'\rangle, w'} + c$ یا به طور دقیق‌تر: $\exists c: K(xy) \leq 2\lg K(x) + K(x) + K(y) + c$

w, w' ← رشته و M را روی w و M' را روی w' اجرا می‌کند. ← ماشین : توصیف xy

برای اینکه w و w' را در یک رشته عادی از Δ جدا می‌دهیم: $(|w| = n, \ w = w_1 w_2 \cdots w_n)$

روش I: $w'' = w_1 w_1 w_2 w_2 \cdots w_n w_n 01 w'$

روش II: $w'' = p_1 p_1 p_2 p_2 \cdots p_k p_k 01 w w'$ ($n = (p_1 p_2 \cdots p_k)_2$ ، k = طول n در مبنای 2)

چند تعریف:

1- رشته‌ی x ، c-فشرده‌پذیر است، اگر: $K(x) \leq |x| - c$

2- رشته‌ی x ، فشرده‌ناپذیر با c است، اگر: $K(x) > |x| - c$

3- رشته‌ی x ، فشرده‌ناپذیر است، اگر: $K(x) \geq |x|$

قضیه: وجود دارد رشته‌ی فشرده‌ناپذیر با اندازه‌ی n ، به ازای هر n.

A_k: رشته‌های به طول k $(k \geq 0)$ B_k: رشته‌های به طول حداکثر k $(k \geq 0)$

$|B_{n-1}| = |A_0| + |A_1| + \cdots + |A_{n-1}| = \sum_{i=0}^{n-1} 2^i = 2^n - 1$ ، $|A_n| = 2^n \Rightarrow |B_{n-1}| < |A_n|$

⇐ حداقل یک رشته، عضو A_n هست که نمی‌توان آن را فشرده کرد.

Scanned by CamScanner

تعریف پیچیدگی رشته ها با پایتون: توصیف رشته‌ی x را به صورت (p, w) در نظر بگیرید که p یک برنامه به زبان پایتون است و با ورودی w

می‌آن، x به دست می‌آید: $d_p(x)$ = کوتاه‌ترین توصیف x در زبان پایتون ← $K_p(x) = |d_p(x)|$

قضیه: $\exists C : \quad K(x) \leq K_p(x) + C$ بین هر دو زبان خوبی یک C ثابت وجود دارد.

اثبات: شبیه‌ساز پایتون. و توصیف x با پایتون: (p, w) ← $(\langle M, p \rangle, x)$: توصیف x با توریک ← $K(x) \leq K_p(x) + C$

$$K_p(xy) \leq K_p(x) + K_p(y) + C$$

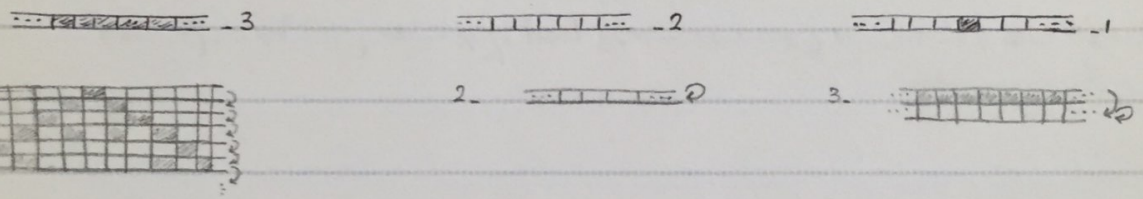
Scanned by CamScanner

نظریه محاسبه - جلسه بیست و سوم

اتوماتای سلولی: تعدادی سلول در نظر بگیرید و تعدادی متناهی حالت که هر سلول در هر لحظه دارای یکی از این حالت هاست و قانونی داریم که طبق آن در هر لحظه حالت سلول ها عوض می شود. سلول ها می توانند در یک خط باشند، یا صفحه ی دو بعدی یا ...

اگر همه ی سلول ها در یک خط چیده شده باشند، قانون به صورت $\delta(q_0, q_1, q_2) \to q$ (۱-بعدی) خواهد بود که Q مجموعه ی حالت ها می باشد و q_0, q_1, q_2 حالت سه سلول کنار هم هستند و طبق این حالت مقدار $\delta(q_0, q_1, q_2)$ برابر حالت سلول وسطی در لحظه ی بعد است. توجه کنید که حالت همه ی سلول ها در لحظه ی اول می دانیم. هم چنین تعداد سلول ها می تواند متناهی یا نامتناهی باشد. (بسته به حالت مسئله ی مورد نظر)

مثال: اتوماتای سلولی ۱-بعدی ای در نظر بگیرید. $Q = \{$ ■ (روشن), □ (خاموش) $\}$ هر سلول اگر دقیقاً یکی از همسایه هایش روشن باشد، روشن می شود؛ حالت بعدی هر یک از حالات زیر را (تا چند مرحله) تعیین کنید:



مثال: با توصیفات مثال قبل آیا حالتی از سلول ها وجود دارد که حالت قبلی نداشته باشد؟ (هم برای تعداد متناهی سلول و هم نامتناهی جواب دهید.)

۱- تعداد متناهی: گرافی جهت دار متناظر با اتوماتا داریم که رأس های آن متناظر با حالت های سلول ها باشد و یال از رأس a به رأس b وجود دارد اگر و تنها اگر حالت متناظر با رأس a به حالت متناظر با رأس b برود. رأس متناظر با حالت همه خاموش را 0 بنامید. تعداد یال های خروجی هر رأس به 0 حداقل 2 است. (یکی از خودش و یکی از همه روشن) فرض کنید تعداد کل سلول های اتوماتا برابر با n باشد.

از هر رأس دقیقاً یک یال خارج می شود. $\Leftarrow$ تعداد رأس ها = تعداد یال های خروجی = تعداد یال های ورودی

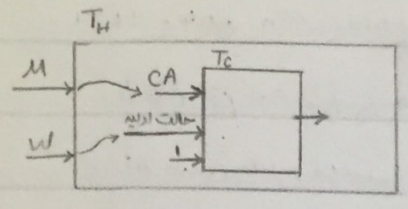
چون تمام یال به 0 وارد می شوند و با توجه به متناهی بودن حداقل یک رأس مثل 0 وجود دارد که یالی به آن وارد می شود.

$\Leftarrow$ حالتی با توصیف سوال وجود دارد.

2- تعداد نامتناهی: حالتی با توصیف سوال وجود ندارد، زیرا برای هر حالت می توان حالت قبلی اش را با این روش یافت. اگر همه ی سلول ها خاموش بودند، حالت قبلی می تواند همه خاموش باشد. اگر حداقل یک سلول روشن وجود داشت، یکی از سلول های روشن را در نظر بگیرید، حالتی می سازیم که حالت قبلی باشد. سلول راست و چپ این سلول را در حالت قبلی به ترتیب روشن و خاموش در نظر بگیرید. از این به بعد می توانیم حالت باقی سلول ها را یکتا مشخص کنیم.

Scanned by CamScanner

- اگر حالت اولیه‌ی اتوماتای معلومی ای داده شده باشد. این مسئله که آیا اتوماتا بعد از بیشتر یا مساوی ۱۰۰ حالت می‌رسد یا خیر عضو RE است. اما عضو R نیست.



فرض خلف: ماشین تورینگ T_C وجود دارد که اتوماتای معلومی، یک حالت اولیه و یک n (تعداد حالت برای تکرار) و خروجی آن این است که آیا اتوماتای مطرح شده با حالت اولیه‌ی داده شده از بعلی به بعد n حالت را تکرار می‌کند یا خیر. ← ماشین T_H جوابی برای مسئله‌ی halt است؛ ابتدا ماشین تورینگ را به یک CA تبدیل می‌کند به این صورت که خانه‌های نوار مثل سلول های CA هستند هر سلول این را تعیین می‌کند که چه سمبلی از کدام خانه نوشته شده است و این که آیا head روی آن خانه است یا خیر؟ اگر هست، حالت ماشین تورینگ در آن لحظه چیست؟ البته قبل از این تبدیل باید تغییراتی روی M ایجاد شود: پس از رسیدن به حالت نهایی، نوار باید از آن پس تغییری نکند و برای جلوگیری از اشتباه گرفتن loop در ماشین تورینگ و رسیدن به halt، به ازای هر transition یک حالت اضافه مطرح می‌کنیم که کار اضافه انجام دهد مثلاً به آن حالت برویم و head به راست برود و سپس از آن حالت به حالت دلخواه برویم و head به چپ برود.

Scanned by CamScanner