ТЕОРІЯ АЛГОРИТМІВ ТА АВТОМАТІВ

Частково-рекурсивні функції

Нехай $f(x_1, x_2, ..., x_n)$ - функція, що визначена на підмножині N^k множини всіх наборів чисел з розширеного натурального ряду $N = \{0,1,2...\}$, $x_i \in N$, і приймає значення з N (поза цією множиною N^k функція вважається невизначеною). Функції такого типу називаються частково-числовими функціями злічено-значної логіки. Позначимо P_N^r - множину всіх частково-числових (арифметичних) функцій.

Розглянемо найпростіші частково-числові функції:

1) $O(x)=0$ - функція, що перетворює в 0;
2) $S(x)=x+1$ – функція додавання одиниці;
3) $I_m^n(x_1, ..., x_m, ..., x_n) = x_m$, $1 \le m \le n$ функція вибору координати.

На множині P_N^r визначено три операції: C - суперпозиція, $ПР$ – примітивна рекурсія, μ - мінімізація.

1. *Суперпозиція* вводиться так само, як і для попередніх функціональних систем. Нехай задані функції, які $\in P_N^r$: $f_0(x_1,\ldots,x_k), f_1(x_1,\ldots,x_n),\ldots,f_k(x_1,\ldots,x_n)$. Будемо говорити, що функція $\Phi(x_1,\ldots,x_n)$ отримана за допомоги операції суперпозиції з функцій $f_0, f_1,\ldots,f_k$, якщо для $\forall$ набору $x_1,\ldots,x_n$ чисел з N її значення рівні $f_0(f_1,\ldots,f_k)$.

$$\Phi(x_1,\ldots,x_n) = f_0(f_1(x_1,\ldots,x_n),\ldots,f_k(x_1,\ldots,x_n))$$

Приклади:

$$\Phi(x) = O(S(x)) = 0$$

$$\Phi(x) = S(O(x)) = 1\ ,\ S(S(O(x))) = 2\ ,\ \ldots, \underbrace{S(S\ldots S}_{k-разів}(O(x))) = k$$

$$\underbrace{S(S\ldots S(S}_{m-разів}(x))) = x + m$$

2. *Операція примітивної рекурсії*

Функція $f(x)$ з P_N^r отримана за схемою примітивної рекурсії з числа a и функції $h(x,y) \in P_N^r$, якщо виконуються наступні умови:

$$\begin{cases} f(0) = a \\ f(x+1) = h(x, f(x)) \end{cases}$$

Наприклад, розглянемо функцію $f(x)=x= I_1^1(x)$.

Схема примітивної рекурсії:

$$\begin{cases} f(0) = 0 = O(x) \\ f(x+1) = x+1 = f(x)+1 = S(f(x)) = S(I_2^2(x, f(x))) \end{cases}$$

де $h(x, f(x)) = S(I_2^2(x, f(x)))$

Схема примітивної рекурсії для багатьох змінних

Функція $f(x_1,\ldots,x_n,y)$ з P_N^r отримана за схемою примітивної рекурсії з функції $g(x_1,\ldots,x_n)$ и функції $h(x_1,\ldots,x_n,y,z)$, якщо виконуються наступні умови:

$$\begin{cases} f(x_1,\ldots,x_n,0) = g(x_1,\ldots,x_n) \\ f(x_1,\ldots,x_n,y+1) = h(x_1,\ldots,x_n,y,f(x_1,\ldots,x_n,y)) \end{cases}$$

Числову функцію $f(x_1,\ldots,x_n)$ називають примітивно-рекурсивною, якщо вона може бути отримана з найпростіших $\{O(x),\ S(x),\ I_m^n(x_1,\ldots,x_m,\ldots,x_n),\ 1 \le m \le n\}$ за допомоги скінченної кількості операцій суперпозиції і примітивної рекурсії.

Позначимо клас примітивно-рекурсивних функцій $Pnp \subseteq P_N^r$.

Отримаємо функцію $f_1(x,y) = x+y$ за схемою примітивної рекурсії з O, S, I_m^n.

Схема примітивної рекурсії:

$$\begin{cases} f_1(x,0) = x = I_1^1(x) \\ f_1(x,y+1) = x+(y+1) = (x+y)+1 = S(x+y) = S(f_1(x,y)) = S(I_3^3(x,y,f_1(x,y))) \end{cases}$$

$$g(x) = I_1^1(x)$$

$$h(x,y,f_1(x,y)) = S(I_3^3(x,y,f_1(x,y)))$$

Отримаємо функцію $f_2(x,y) = x \times y$ за схемою примітивної рекурсії з O, S, I_m^n.

Схема примітивної рекурсії:

$$\begin{cases} f_2(x,0) = 0 = O(x) \\ f_2(x,y+1) = x\times(y+1) = x\times y + x = f_2(x,y) + x = f_1(f_2(x,y),x) = f_1(I_3^3(x,y,f_2(x,y)), I_1^3(x,y,f_2(x,y))) \end{cases}$$

$$g(x) = O(x)$$

$$h(x,y,f_2(x,y)) = f_1(I_3^3(x,y,f_2(x,y)), I_1^3(x,y,f_2(x,y)))$$

Розглянемо функцію $f_3(x) = Sg(x) = \begin{cases} 0, x = 0 \\ 1, x \neq 0 \end{cases}$. Доведемо, що вона примітивно-рекурсивна.

Схема примітивної рекурсії:

$$\begin{cases} Sg(0) = 0 = O(x) \\ Sg(x+1) = 1 = S(O(x)) \end{cases}$$

Розглянемо функцію $f_4(x)=\overline{Sg}(x)=\begin{cases}1, x=0\\0, x\neq 0\end{cases}$. Доведемо, що вона примітивно-рекурсивна.

Схема примітивної рекурсії:

$$\begin{cases}\overline{Sg}(0)=1=S(O(x))\\ \overline{Sg}(x+1)=0=O(x)\end{cases}$$

Розглянемо функцію $f_5^1(x)=x\dot{-}1=\begin{cases}x-1, x\geq 1\\0, x<1\end{cases}$ Доведемо, що вона примітивно-рекурсивна.

Схема примітивної рекурсії:

$$\begin{cases}f_5^1(0)=0\dot{-}1=0=O(x)\\ f_5^1(x+1)=(x+1)\dot{-}1=x=\mathrm{I}_1^1(x)\end{cases}$$

Розглянемо функцію $f_5(x,y)=x\dot{-}y=\begin{cases}x-y, x\geq y\\0, x<y\end{cases}$ Доведемо, що вона примітивно-рекурсивна.

Схема примітивної рекурсії:

$$\begin{cases}f_5(x,0)=x\dot{-}0=x=I_1^1(x)\\ f_5(x,y+1)=\begin{cases}x-(y+1), x\geq y+1\\0, x<y+1\end{cases}=\begin{cases}(x-y)-1, x-y\geq 1\\0, x-y<1\end{cases}=(x-y)\dot{-}1\end{cases}$$

3. *Операція мінімізації.*

Розглянемо рівняння *F(x,y)=0.* Наприклад: $y^2-xy+5=0$.

$y=\dfrac{x\pm\sqrt{x^2-20}}{2}$ - точний розв'язок, який існує при $|x|\geq\sqrt{20}$, $x,y\in N$.

При $x=0$ $y=\varnothing$

$x=1$ $y=\varnothing$

……

$x=5$ $y=\dfrac{5+\sqrt{5}}{2}$ - не визначена

$x=6$ $y=\dfrac{6+\sqrt{4}}{2}=4$

6 – це мінімальний розв'язок цього рівняння.

Сформулюємо задачу:

Для кожного $x\in N$ ми будемо шукати мінімальне $y\in N$, яке задовольняє співвідношенню *F(x,y)=0* і побудуємо «нову» функцію $y=g(x)=\mu_y(F(x,y)=0)$

Алгоритм: $F(x,0)\neq 0$ не визначена

$F(x,1)\neq 0$ не визначена

……..
$F(x,a)=0$ визначена $\Rightarrow$ $g(x)=a$

Означення: Функція $g(x)$ отримана з функції *F(x,y)* за допомогою операції мінімізації по змінній *y*, якщо:

1. *F(x,y)* визначена для всіх $y \le a$
2. $F(x,y) \ne 0, \forall y < a$
3. *F(x,y)=0* для $y = a$

И будемо позначати її так: $g(x) = \mu_y (F(x,y) = 0)$

Приклади:

1. *F(x,y)=x+y+7*
   x+y+7=0 $\Rightarrow y=-7-x$, не визначена для $x \in N \Rightarrow g(x)$ не визначена.

2. *F(x,y)=-x+y+7*
   $-x+y+7=0 \Rightarrow y=\begin{cases} x-7, x-7 \ge 0 \\ \textit{не визначена}, \text{x - 7} < 0 \end{cases}$
   Але: *F(x,0)=7-x* визначена для $7-x \ge 0 \Rightarrow x \le 7,$ тоді
   $g(x)=\begin{cases} x-7, x=7 \\ \textit{не визначена}, \text{x} \ne 7 \end{cases} = \begin{cases} 0, x=7 \\ \textit{не визначена}, \text{x} \ne 7 \end{cases}$

3. *F(x,y)=xy*
   xy=0 розглянемо *min* , y=0 вже є розв'язком.
   $F(x,0) = x \cdot 0 = 0$ для всіх $x \in N \Rightarrow g(x) = \mu_y (F(x,y) = 0) = 0$

4. *F(x,y)=xy-5*
   xy-5=0 $\Rightarrow$ $\text{y} = \begin{cases} \frac{5}{x}, x=1, \ x=5 \\ \textit{не визначена інакше} \end{cases}$
   F(x,0)=-5 вже не визначена $\Rightarrow g(x)$ - не визначена.

Операція мінімізації функції багатьох змінних

Операція мінімізації по змінній визначається наступним чином:
Нехай задана деяка частково-числова функція $f(x_1,\ldots,x_n),\ n \ge 1$
Означення: функція $g(x_1,\ldots,x_n)$ отримана з $f(x_1,\ldots,x_n)$ мінімізацією по змінній x_n, якщо виконується наступне: $f(x_1,\ldots,x_{n-1},y)=x_n$ (*)

1. Якщо рівняння (*) має розв'язок $y_0 \in N$ и при $\forall\ y \in N$ таких, що $0 \le y < y_0$, функція $f(x_1,\ldots,x_{n-1},y)$ визначена і її значення

відмінні від x_n $(f(x_1,\dots,x_{n-1},y)\neq x_n)$, то припускаємо $g(x_1,\dots,x_n)=y_0$;

2. Якщо рівняння (*) не має розв'язків в цілих невід'ємних числах, то вважаємо, що $g(x_1,\dots,x_n)$ не визначена;
3. Якщо y_0 - найменший цілий невід'ємний розв'язок рівняння (*) и при деякому $y_1\in N, y_1<y_0$ значення $f(x_1,\dots,x_{n-1},y_1)$ не визначене, то припускаємо: $g(x_1,\dots,x_n)$ не визначена.

Про побудовану таким чином функцію $g(x_1,\dots,x_n)$ кажуть, що вона отримана з $f(x_1,\dots,x_n)$ за допомоги мінімізації по змінній x_n и позначають: $g(x_1,\dots,x_n)=\mu_{x_n}(f(x_1,\dots,x_{n-1},y)=x_n)$.

Приклад1. $f(x)=x+1$

Складаємо рівняння $f(y)=x$, тобто $y+1=x; \Rightarrow y=x-1$

для $x=0$ немає розв'язків цього рівняння в області N

для $x=1$ є розв'язки, тобто

$$g(x)=\begin{cases}\text{не визначена при } x=0\\ x-1, x>0\end{cases}$$

Приклад2. $f(x)=\left[\frac{x}{2}\right]$. Складаємо рівняння $\left[\frac{y}{2}\right]=x$. $y=2x$. $g(x)=2x$ (визначена на всіх наборах).

Означення: Функцію $f(x_1,\dots,x_n)$ називають частково-рекурсивною, якщо вона може бути отримана з найпростіших $\{O(x), S(x), I_m^n(x_1,...,x_m,...,x_n),\ 1\le m\le n\}$ за допомоги скінченної кількості операцій суперпозиції, мінімізації і скінченої кількості схем примітивної рекурсії.

Множина $P_{чр}$ всіх функцій, які можна отримати з системи функцій $\{O(x), S(x), I_m^n(x_1,...,x_m,...,x_n),\ 1\le m\le n\}$ за допомоги скінченної кількості операцій суперпозиції, мінімізації і скінченої кількості схем примітивної рекурсії називають класом частково-рекурсивних функцій.
$P_{пр}\subseteq P_{чр}\subseteq P_N^r$. Кожна примітивно-рекурсивна функція буде частково-рекурсивною.

Теореми про примітивно-рекурсивні функції

Теорема 1. Нехай задана функція $f(x_1,x_2,...,x_n)\in P_{пр}$, тоді примітивно-рекурсивними будуть функції

$$g(x_1,x_2,...,x_n)=\sum_{i=0}^{x_n} f(x_1,x_2,...,i)=f(x_1,x_2,...,x_{n-1},0)+f(x_1,x_2,...,x_{n-1},1)+...+f(x_1,x_2,...,x_{n-1},x_n)$$

та

$$h(x_1, x_2, ..., x_n) = \prod_{i=0}^{x_n} f(x_1, x_2, ..., i) = f(x_1, x_2, ..., x_{n-1}, 0) \cdot f(x_1, x_2, ..., x_{n-1}, 1) \cdot ... \cdot f(x_1, x_2, ..., x_{n-1}, x_n)$$ Доведення:

Схема примітивної рекурсії:

$$\begin{cases} g(x_1, ..., 0) = f(x_1, ..., 0) \\ g(x_1, ..., x_{n-1}, y+1) = \sum_{i=0}^{y} f(x_1, x_2, ..., y) + f(x_1, x_2, ..., y+1) = \end{cases}$$

$$g(x_1, x_2, ..., y) + f(x_1, x_2, ..., y+1) = f_1(g(x_1, x_2, ..., y); f(x_1, x_2, ..., y+1))$$

Де $f(x_1, x_2, \ldots, x_{n-1}, y+1) \in P_{пр}$ та $f_1(z_1, z_2) \in P_{пр}$

$h \in P_{пр}$ (д/з)

Теорема 2. Нехай задана примітивно-рекурсивна функція $f(x_1, x_2, \ldots, x_n)$, тоді функція $l(x_1, x_2, \ldots, x_{n-1}, y, z)$ також примітивно-рекурсивна, де

$$l(x_1, x_2, \ldots, x_{n-1}, y, z) = \begin{cases} \sum_{i=y}^{z} f(x_1, x_2, \ldots, x_{n-1}, i), \text{якщо} & y < z \\ f(x_1, \ldots, y), \text{якщо} & y = z \\ 0, \text{якщо} & y > z \end{cases}$$

$$p(x_1, x_2, \ldots, x_{n-1}, y, z) = \begin{cases} \prod_{i=y}^{z} f(x_1, x_2, \ldots, x_{n-1}, i), \text{якщо} & y < z \\ f(x_1, \ldots, y), \text{якщо} & y = z \\ 1, \text{якщо} & y > z \end{cases}$$

Доведення:

Зобразимо функцію у вигляді суперпозиції примітивно-рекурсивних функцій

$$l(x_1, x_2, \ldots, x_{n-1}, y, z) = \sum_{i=0}^{z} f(x_1, x_2, \ldots, x_{n-1}, i) \dot{-} \sum_{i=0}^{y} f(x_1, x_2, \ldots, x_{n-1}, i) + f(x_1, x_2, \ldots, x_{n-1}, y) \cdot \overline{S}g(y \dot{-} z)$$

$p \in P_{пр}$ аналогічно (д/з)

Теорема 3. Якщо задані примітивно-рекурсивні функції $f(x_1, x_2, \ldots, x_n)$, $\alpha(x_1, x_2, \ldots, x_n)$, $\beta(x_1, x_2, \ldots, x_n)$, тоді також примітивно-рекурсивними будуть функції

$$l_1(x_1, x_2, \ldots, x_{n-1}, \alpha(x_1, x_2, \ldots, x_n), \beta(x_1, x_2, \ldots, x_n)) =$$

$$\begin{cases} \sum_{i=\alpha(x_1, x_2, \ldots, x_n)}^{\beta(x_1, x_2, \ldots, x_n)} f(x_1, x_2, \ldots, x_{n-1}, i), \text{якщо для} \forall \text{наборів} (x_1, x_2, \ldots, x_n) & \alpha < \beta \\ 0, \text{якщо} \alpha > \beta \quad \forall (x_1, x_2, \ldots, x_n) \end{cases}$$

$$p_1(x_1, x_2, \ldots, x_{n-1}, \alpha(x_1, x_2, \ldots, x_n), \beta(x_1, x_2, \ldots, x_n)) =$$

$$\begin{cases} \prod_{i=\alpha(x_1, x_2, \ldots, x_n)}^{\beta(x_1, x_2, \ldots, x_n)} f(x_1, x_2, \ldots, x_{n-1}, i), \text{якщо для} \forall \text{наборів} (x_1, x_2, \ldots, x_n) & \alpha < \beta \\ 1, \text{якщо} \alpha > \beta \quad \forall (x_1, x_2, \ldots, x_n) \end{cases}$$

Доведення:

$l_1(x_1, x_2, \ldots, x_{n-1}, \alpha(x_1, x_2, \ldots, x_n), \beta(x_1, x_2, \ldots, x_n)) = l(x_1, x_2, \ldots, x_{n-1}, y, z)$,

де $y = \alpha(x_1, x_2, \ldots, x_n)$, $z = \beta(x_1, x_2, \ldots, x_n)$

тобто l_1 отримана операцією суперпозиції з l, α та β, які $\in P_{пр}$

p_1 аналогічно

Теорема 4. Якщо задані примітивно-рекурсивні функції $f_1(x_1, \ldots, x_n), f_2(x_1, \ldots, x_n), \ldots, f_{k+1}(x_1, \ldots, x_n), \alpha_1(x_1, \ldots, x_n), \alpha_2(x_1, \ldots, x_n), \ldots, \alpha_k(x_1, \ldots, x_n),$ тоді і функція $m(x_1, \ldots, x_n)$ також буде примітивно-рекурсивною, де

$$m(x_1, \ldots, x_n) = \begin{cases} f_1(x_1, \ldots, x_n), якщо \quad \alpha_1(x_1, \ldots, x_n) = 0; \\ f_2(x_1, \ldots, x_n), якщо \quad \alpha_2(x_1, \ldots, x_n) = 0; \\ \ldots \\ f_k(x_1, \ldots, x_n), якщо \quad \alpha_k(x_1, \ldots, x_n) = 0; \\ f_{k+1}(x_1, \ldots, x_n), якщо \quad \alpha_1 \neq 0, \alpha_2 \neq 0, \ldots, \alpha_k \neq 0 \end{cases}$$

причому в нуль обертається лише одна з функцій $\alpha_1, \alpha_2, \ldots, \alpha_k$ (тобто всі умови взаємно виключають одна одну)

Доведення:

$$m(x_1, \ldots, x_n) = f_1(x_1, \ldots, x_n) \cdot \overline{Sg}(\alpha_1(x_1, \ldots, x_n)) + f_2(x_1, \ldots, x_n) \cdot \overline{Sg}(\alpha_2(x_1, \ldots, x_n)) + \ldots + \\ + f_k(x_1, \ldots, x_n) \cdot \overline{Sg}(\alpha_k(x_1, \ldots, x_n)) + f_{k+1}(x_1, \ldots, x_n) \cdot Sg(\alpha_1 \cdot \alpha_2 \cdot \ldots \cdot \alpha_k)$$

тобто зобразили в вигляді суперпозиції примітивно-рекурсивних функцій.

Теорема 5. (Про обмеженість операції мінімізації)

Нехай задані примітивно-рекурсивні функції $f(x_1, x_2, \ldots, x_n, y)$ та $\alpha(x_1, x_2, \ldots, x_n)$

Розглянемо рівняння $f(x_1, x_2, \ldots, x_n, y) = 0$

Якщо для всіх наборів $(x_1, x_2, \ldots, x_n)$ це рівняння має розв'язок $y \leq \alpha(x_1, x_2, \ldots, x_n)$,

Тоді функція $g(x_1, x_2, \ldots, x_n) = \mu_y(f(x_1, x_2, \ldots, x_n, y) = 0)$ буде примітивно рекурсивною.

Доведення:

Побудуємо послідовність:

$f(x_1, x_2, \ldots, x_n, 0); \quad \neq 0$

$f(x_1, x_2, \ldots, x_n, 0) \cdot f(x_1, x_2, \ldots, x_n, 1); \quad \neq 0$

…

$f(x_1, x_2, \ldots, x_n, 0) \cdot \ldots \cdot f(x_1, x_2, \ldots, x_n, y-1); \quad \neq 0$

$f(x_1, x_2, \ldots, x_n, 0) \cdot \ldots \cdot f(x_1, x_2, \ldots, x_n, y); \quad = 0$

…

$f(x_1, x_2, \ldots, x_n, 0) \cdot \ldots \cdot f(x_1, x_2, \ldots, x_n, \alpha(x_1, x_2, \ldots, x_n)); \quad = 0$

…

Помітимо $y \in [0, \alpha]$

$$g(x_1, x_2, \ldots, x_n) = \sum_{i=0}^{\alpha(x_1, x_2, \ldots, x_n)} Sg(f(x_1, x_2, \ldots, x_n, 0) \cdot f(x_1, x_2, \ldots, x_n, 1) \cdot \ldots \cdot f(x_1, x_2, \ldots, x_n, i))$$

Додатки:

1. $\left[\frac{x}{y}\right] = n$ (ціла частина дробу)

Якщо x<y, то функція дорівнює 0

Якщо x>y, то функція дорівнює n

Якщо y=0, то для визначеності функція дорівнює x (частково-числові функції повинні бути визначені на всіх наборах).

$n \le \frac{x}{y} < n+1 \Leftrightarrow ny \le x < y \cdot (n+1)$ $1 \cdot y \dot{-} x, \quad 2 \cdot y \dot{-} x, \quad \ldots, \quad n \cdot y \dot{-} x, \quad (n+1) \cdot y \dot{-} x$

=0 =0 ≤0 >0

$$\left[\frac{x}{y}\right] = \begin{cases} \sum_{i=0}^{x} \bar{S}g(i \cdot y \dot{-} x) & ,для \quad x \ge y \\ 0, \quad для & x < y \end{cases} \Rightarrow \left[\frac{x}{y}\right] \in P_{пр}$$

2.

$$\left[\frac{x}{y}\right] = \begin{cases} n: & ny \le x < y(n+1), y \ne 0 \\ 0: & x < y, \quad y \ne 0 \\ a: & y = 0 \end{cases} = \left(\sum_{i=1}^{x} \bar{S}g(iy \dot{-} x)\right) \cdot Sg(y) + a \cdot \bar{S}g(y)$$

3. $\left[\sqrt{x}\right]$

$\left[\sqrt{x}\right] = y$: $y^2 \le x \le (y+1)^2$

тобто $(y+1)^2 \dot{-} x$ додатне, тоді $\left[\sqrt{x}\right] = y$

Розглянемо рівняння $\bar{S}g((y+1)^2 \dot{-} x) = 0$

$y = \mu_y(\bar{S}g((y+1)^2 \dot{-} x) = 0)$ або $\left[\sqrt{x}\right] = \sum_{y=0}^{x} \bar{S}g(y^2 \dot{-} x) \dot{-} 1$

4. $rest(x, y)$ - залишок від ділення x на y, причому $rest(x, 0) = x$.

$$\frac{x}{y} = \left[\frac{x}{y}\right] + \frac{rest(x, y)}{y}$$

Звідси маємо:

$$x = y \cdot \left[\frac{x}{y}\right] + rest(x, y)$$

Отже,

$$rest(x, y) = x \dot{-} y \cdot \left[\frac{x}{y}\right]$$

Отримали,що $rest(x, y) \in P_{ПР}$ як суперпозиція примітивно-рекурсивних функцій.

Функції простих чисел

1. Розглянемо функцію

$$div(x,y) = \begin{cases} 1, \text{якщо } x \text{ кратно } y \\ 0, \text{якщо } x \text{ не кратно } y \end{cases} = \begin{cases} 1, \text{якщо } rest(x,y) = 0 \\ 0, \text{якщо } rest(x,y) \neq 0 \end{cases} = \overline{Sg}(rest(x,y))$$

$div(x,y) \in P_{ПР}$ як суперпозиція примітивно-рекурсивних функцій.

2. Розглянемо функцію $nd(x)$ - кількість дільників x.

$$nd(x) = \sum_{i=0}^{x} div(x,i) \in P_{ПР}$$

Приклади: $nd(15) = 4$; $nd(0) = 0$; $nd(17) = 2$.

3. Розглянемо функцію $\pi(x)$ – кількість простих чисел, які не перевищують x.

$$\pi(x) = \sum_{i=0}^{x} \overline{Sg}(|nd(i) - 2|)$$

Отже, $\pi(x) \in P_{ПР}$ за теоремою 1. Залишається довести, що $|x - y| \in P_{ПР}$, а це випливає з рівності $|x \quad y| = (x \cdot y) + (y \cdot x)$.

4. $p(n) = p_n$ - просте число, яке серед простих чисел знаходиться на n+1 місці.

Зауважимо, що $\pi(p_n) = n + 1$.

$p_n = \mu_t(\pi(t) = n + 1)$

Для застосування теореми про обмежену операцію мінімізації треба показати, що $p(n) \leq \alpha_n$, де $\alpha_n \in P_{ПР}$. Візьмемо $\alpha_n = 2^{2^n} \in P_{ПР}$.

Доведемо, що $p_n \leq 2^{2^n}$ за допомогою методу математичної індукції:

1) $p_0 = 2 \leq 2^{2^0} = 2$

2) індуктивне припущення $p_s \leq 2^{2^s}$

3) доведемо, що $p_{s+1} \leq 2^{2^{s+1}}$

Нехай $a = p_0 \cdot p_1 \cdot \ldots \cdot p_s + 1$. Тоді серед дільників a немає жодного з чисел $p_0, p_1, \ldots, p_s$. Позначимо деякий дільник a через p_r. Очевидно,що $p_{s+1} \leq p_r \leq a$.

Застосуємо індуктивне припущення:

$$p_{s+1} \leq a \leq 2^{2^0} \cdot 2^{2^1} \cdot \ldots \cdot 2^{2^s} + 1 = 2^{1+2+2^2+\ldots+2^s} + 1 =$$

$$= 2^{\frac{1(1-2^{s+1})}{1-2}} = 2^{2^{s+1}-1} + 1 < 2^{2^{s+1}}$$

Отже, нерівність $p_n \leq 2^{2^n}$ доведено і $p_n = p(n) \in P_{ПР}$.

Функція експоненти

Будь-яке натуральне число можна представити у вигляді:

$$y = p_{i_0}^{\alpha_0} \cdot p_{i_1}^{\alpha_1} \cdot \ldots \cdot p_{i_k}^{\alpha_k}$$

Функція $ex(x,y)$ – це показник степеня числа $p(x)$ у розкладі числа y.

Доведемо, що $ex(x,y) \subset P_{ПР}$.

$$ex(x,y) = \mu_u\left(Sg\left(rest\left(y, P_x^{u+1}\right)\right) = 0\right)$$

За теоремою про обмежену операцію мінімізації залишається довести, що $ex(x,y) \le \alpha(x,y) - y \in P_{ПР}$.

Зворотна рекурсія

Означення. Нехай задані числові функції $\alpha_1(x), \alpha_2(x), \dots, \alpha_s(x)$, які задовольняють наступну умову:

$$\forall x \in \mathbb{N}: \alpha_i(x+1) \le x \quad (i = \overline{1,s})$$

Будемо говорити, що числова функція $f(x_1, x_2, \dots, x_{n+1})$ отримана за допомогою операції зворотної рекурсії із числових функцій $g(x_1, x_2, \dots, x_n)$ та $h(x_1, x_2, \dots, x_n, y, z_1, \dots, z_s)$, якщо для всіх наборів $(x_1, x_2, \dots, x_n, y)$ виконуються наступні умови:

Схема З.Р.
$$\begin{cases} f(x_1, x_2, \dots, x_n, 0) = g(x_1, x_2, \dots, x_n) \\ f(x_1, x_2, \dots, x_n, y+1) = h(x_1, x_2, \dots, x_n, y, \alpha_1(x), \dots, \alpha_s(x)) \end{cases}$$

Теорема про зворотну рекурсію. Функція $f(x_1, x_2, \dots, x_n, y)$, яка отримана за допомогою операції зворотної рекурсії із примітивно-рекурсивних допоміжних функцій $\alpha_i(x+1) \le x \ \forall x \in \mathbb{N}$ та примітивно-рекурсивних функцій $g(x_1, x_2, \dots, x_n)$ та $h(x_1, x_2, \dots, x_n, y, z_1, \dots, z_s)$, також є примітивно-рекурсивною.

Доведення. Розглянемо функцію

$$F(x_1, \dots, x_n, y) = \prod_{i=0}^{y} p_i^{f(x_1, \dots, x_n, 0)}$$

де p_i - просте число, яке серед простих чисел знаходиться на i+1 місці.

Очевидно, що $f(x_1, x_2, \dots, x_n, u) = ex\left(u, F(x_1, \dots, x_n, y)\right) \quad \forall u \le y$.

Таким чином, функція $f(x_1, x_2, \dots, x_n, u) \subset P_{ПР}$ за умови, що $F(x_1, \dots, x_n, y) \in P_{ПР}$. Доведемо це: побудуємо схему примітивної рекурсії.

Схема П.Р.
$$\begin{cases} F(x_1, x_2, \dots, x_n, 0) = p_0^{f(x_1, \dots, x_n, 0)} \\ F(x_1, x_2, \dots, x_n, y+1) = p_0^{f(x_1, \dots, x_n, 0)} \cdot \dots \cdot p_{y+1}^{f(x_1, \dots, x_n, y+1)} - \\ - F(x_1, x_2, \dots, x_n, y) \cdot \\ \cdot p_{y+1}^{h(x_1, \dots, x_n, y+1, ex(\alpha_1(y+1), F(x_1, \dots, x_n, \alpha_1(y+1))), \dots, ex(\alpha_1(y+1), F(x_1, \dots, x_n, \alpha_1(y+1))))} \end{cases}$$

Теорему доведено.

Нумерація пар і кортежів натуральних чисел

Розглянемо функцію $\pi(x_1, x_2)$

x_2 / x_1	0	1	2	3	…

0	0	1	3	6	
1	2	4	7		
2	5	8			
3	9				
⋮					

$\pi(0,0)=0$, $\pi(0,1)=1$, $\pi(1,0)=2$, $\pi(0,2)=3$, $\pi(1,1)=4$, $\pi(2,0)=5$, ⋯

Можна записати трикутником

(0,0) $\pi(0,0)=0$

(0,1) (1,0) $\pi(0,1)=1$, $\pi(1,0)=2$

(0,2) (1,1) (2,0) $\pi(0,2)=3$, ...

… ….

$$0+1+2+...+(x+y+1)=\frac{0+(x+y+1)}{2}\cdot(x+y)$$

... ...

(0, x+y) (1, x+y-1) ... (x, y) ... (x+y, 0)

... ... …

$\pi(x,y)$- називається пеанівською функцією і використовується для нумерації всіх пар (α_1,α_2) чисел з розширеного натурального ряду. Нехай $\pi(x,y)=n$

$x=l(n)$ – ліва компонента

$y=r(n)$ – права компонента

$$\pi(x,y)=\frac{(x+y)\cdot(x+y+1)}{2}+x=\left[\frac{(x+y)\cdot(x+y+1)}{2}\right]+x$$

$\frac{(x+y)\cdot(x+y+1)}{2}$ (оскільки завжди парна та ділиться без остачі)

Ця функція примітивно-рекурсивна (як сума двох примітивно-рекурсивних функцій).

Знайдемо залежність y, x через n.

$$n=\pi(x,y)=\frac{(x+y)\cdot(x+y+1)}{2}+x$$

$2n=(x+y)(x+y+1)+2x$

$2n=(x+y)^2+(x+y)+2x$

$8n=4(x+y)^2+4(x+y)+1-1+8x$

$8n=(2(x+y)+1)^2+8x-1$

$8n+1=(2x+2y+1)^2+8x$

тобто $(2x+2y+1)^2\le 8n+1$

$$2x+2y+1\le\left[\sqrt{8n+1}\right]$$

$$2x+2y+2\le\left[\sqrt{8n+1}\right]+1$$

$$x+y+1\le\frac{\left[\sqrt{8n+1}\right]+1}{2}$$

$1+8n=4(x+y)^2+4(x+y)+1+8x$

$1+8n=4(x+y)^2+12(x+y)-8(x+y)+8x+1$

$1+8n=4(x+y)^2+2\cdot 2(x+y)\cdot 3-8y+9-9+1$

$1+8n=(2(x+y)+3)^2-8y-8$

$1+8n<(2x+2y+3)^2$

$$\left[\sqrt{1+8n}\right]<2x+2y+3$$

$$1+\left[\sqrt{1+8n}\right]<2x+2y+4$$

$$\frac{1+\left[\sqrt{1+8n}\right]}{2}<x+y+2$$

тоді $$x+y+1\le\frac{1+\left[\sqrt{1+8n}\right]}{2}<x+y+2$$

тобто $$x+y+1=\left[\frac{1+\left[\sqrt{1+8n}\right]}{2}\right]$$ (за означенням цілої частини числа)

$$n=\frac{1+(x+y)}{2}\cdot(x+y)+x \Rightarrow$$

$$l(n)=x=n\dot{-}\left[\frac{\left[\sqrt{1+8n}\right]+1}{2}\right]\cdot\left(\left[\frac{\left[\sqrt{1+8n}\right]+1}{2}\right]\dot{-}1\right)\cdot\frac{1}{2}$$

$$r(n)=y=\left[\frac{\left[\sqrt{1+8n}\right]+1}{2}\right]\dot{-}1\dot{-}l(n)$$

l(n) та *r(n)* – примітивно рекурсивні функції.

Довільне відображення N^2 на множину N ($N^2\xrightarrow{\varphi}N$), взаємно однозначне разом з φ_1 та φ_2 $N\xrightarrow{\varphi_1}N$, $N\xrightarrow{\varphi_2}N$ теж взаємно однозначне, називається Геделевскою нумерацією пар, якщо виконується :

$\varphi(x, y) = n$

$x = \varphi_1(n)$

$y = \varphi_2(n)$

$$\varphi(\varphi_1(n), \varphi_2(n)) = n$$
$$\varphi_1(\varphi(x, y)) = x$$
$$\varphi_2(\varphi(x, y)) = y$$

І при цьому

Нумерація трійок:

$\pi_3(x_1, x_2, x_3) = \pi(\pi(x_1, x_2), x_3)$

$\pi_3(x_1, x_2, x_3) = n,$

$x_3 = r(n), \pi(x_1, x_2) = l(n)$

$x_1 = l(l(n))$

$x_2 = r(l(n))$

Нумерація четвірок:

$\pi_4(x_1, x_2, x_3, x_4) = \pi_3(\pi(x_1, x_2), x_3, x_4) = \pi(\pi(\pi(x_1, x_2), x_3), x_4) = n$

$x_4 = r(n)$

$x_3 = r(l(n))$

$x_2 = r(l(l(n)))$

$x_1 = l(l(l(n)))$

$\pi_{n+1}(x_1, \ldots, x_n, x_{n+1}) = \pi_n(\pi(x_1, x_2), x_3, \ldots, x_n, x_{n+1}) = n$

$x_{n+1} = r(n)$

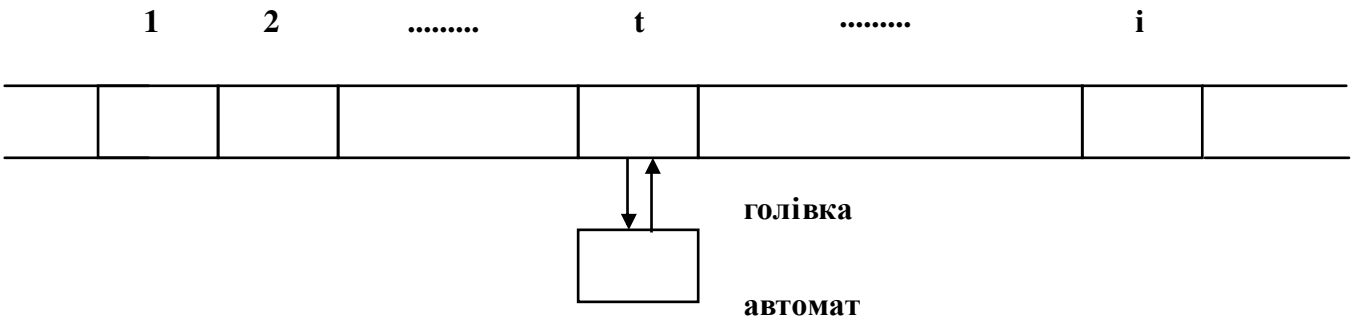
$x_n = r(l(n))$

$\vdots$

$x_2 = r(l(\ldots l(n))$

$x_1 = l(l(\ldots l(n))$

Машина Тюрінга



Машина Тюрінга представляє собою абстрактний пристрій, що складається зі стрічки, зчитувальної голівки і керуючого пристрою.

1) Стрічка припускається потенційно необмеженою в обидві сторони (в кожний момент часу стрічка скінченна (містить скінченну

кількість ланок), але розмір стрічки (кількість її ланок) можна збільшувати). Таким чином стрічка розбита на ланки. В кожній ланці в кожний дискретний момент часу знаходиться рівно один символ з зовнішнього алфавіту.

2) $A = \{a_0, a_1, \ldots, a_{n-1}\}, n \geq 2$, серед яких є порожній символ $\Lambda \in A$ (в якості порожнього символу зазвичай використовують нуль: 0) и * - розділювальний символ.
Найпростіший алфавіт $A = \{0,1\}$, де 0 – порожній символ.

3) Керуючий пристрій в кожний момент часу знаходиться в деякому стані q_j, що належить множині $Q = \{q_0, q_1, \ldots, q_{m-1}\}, m \geq 1$ - множині внутрішніх станів. При цьому q_1 - початковий, а q_0 - кінцевий стан.

4) Зчитувальна (або друкуюча) голівка переміщується вздовж стрічки так, що в кожен момент часу вона «бачить» рівно одну ланку стрічки. Голівка може зчитувати вміст цієї ланки і записувати в неї замість цього символу деякий інший символ з зовнішнього алфавіту (чи той же).

В процесі роботи керуючий пристрій в залежності від стану, в якому воно знаходиться и символу, який воно «бачить», змінює свій(внутрішній) стан (може залишитися в попередньому стані) видає голівці наказ надрукувати в ланці, яку спостерігаємо певний символ з зовнішнього алфавіту і «наказує» голівці або залишатися на місці, або зсунутись на одну ланку вліво, або на одну ланку вправо.

Робота керуючого пристрою (машини Тюрінга) виконується відповідно до програми.

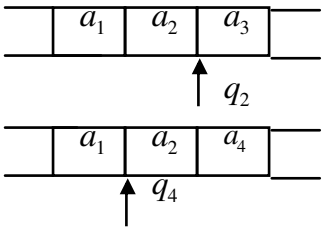
Програма машини Тюрінга складається з скінченої кількості команд. Кожна команда має вигляд п'ятірки :

$q_i a_j \; q_i' a_j' d$

$d = \{S, L, R\}$ - функція руху голівки, де символи S, L, R означають відповідно відсутність руху голівки(стоп), зсування на одну ланку вліво і зсування на одну ланку вправо.

Виконання цієї команди означає наступне:

- якщо голівка в стані q_j розглядає ланку, в якій записаний символ a_j, то відповідно до цієї команди замінюємо символ a_j на a_j', стан голівки q_j на q_j' и голівка зсувається вліво (L), вправо (R) або залишається на місці (S).



$q_2 a_3 \; q_4 a_4 L$

Будемо вимагати, щоб для $\forall$ пари $q_j a_j$ булла в програмі точно одна команда, що починається з цього запису. Якщо стан змінюється на q_0, то будемо говорити що машина зупиняється і жодної команди, що починається з q_0 не пишемо.

Будемо вважати, що:

- МТ починає роботу з стану q_1
- на стрічці на кожному такті тільки скінченна кількість непорожніх символів і серед них можна виділити самий правий і самий лівий символ
- машина в стані q_1 починає на самому лівому непорожньому символі.

Роботу програми МТ будемо ілюструвати за допомоги конфігурації. Слово $a_{j_1} \dots a_{j_{l-1}} q_i a_{j_l} \dots a_{j_s}$ називається конфігурацією машини (в даний момент t).

При l=1 конфігурація має вигляд $q_i a_{j_1} \dots a_{j_s}$.Якщо в момент часу t стічка порожня, то конфігурацією машини буде слово $q_i \Lambda : q_i \Lambda\Lambda\Lambda\Lambda \dots$ Якщо на такті t була конфігурація k_1 , а на наступному k_2, то слід записувати $k_1 \vdash k_2$

Якщо k_1 - початкова конфігурація, а k_m - завершальна, то послідовність $k_1 \vdash k_2 \vdash \dots \vdash k_m$ називається тюринговими розрахунками і говорять, що конфігурація k_m виводиться з k_1.

Якщо $p_1 = a_{j_1} \dots a_{j_s}$ вхідне слово, то машина Т, почавши роботу на слові p_1, або зупиниться через певну кількість кроків, або ніколи не зупиниться.

В першому випадку говорять, що МТ застосовна до слова p_1 і результатом є слово p , що відповідає заключній конфігурації ($p = \mathrm{T}(p_1)$).

В другому випадку, - що машина Т не застосовується до слова p_1.

Часто будуть застосовуватися позначення вигляду P^m для слів вигляду $\underbrace{PP \dots P}_{m \text{ разів}}$.

Розглянемо найпростіший алфавіт *A={0,1}*, де 0 – порожній символ і $(x_1, x_2, \dots x_n)$ довільний набір цілих невід'ємних чисел. Слово $1^{x_1+1}01^{x_2+1}0 \dots 01^{x_n+1}$ називають основним машинним кодом і позначається $K(x_1, x_2, \dots x_n) = K(x_1)0K(x_2)0 \dots K(x_n), \forall n \in N$, зокрема $K(x) = 1^{x+1}$ будемо кодувати блоком з n+1 одиниці.

Далі розглядаються в основному тільки частково-числові функції $f(x_1, \dots, x_n), x_i \in N$.

Означення: Частково-числова функція $f(x_1, \dots, x_n)$ обчислюється за Тюрінгом, якщо існує машина Тюрінга, що обчислює цю функцію, тобто має наступні властивості:

а) якщо $f(x_1, \dots, x_n)$ визначена, то

$T(K(x_1, x_2, \ldots x_n))=K(f(x_1, \ldots, x_n))$

(тобто через скінченну кількість тактів машина прийде до q_0 і на стрічці буде записаний код $K(f(x_1, \ldots, x_n))$

б) якщо $f(x_1, \ldots, x_n)$ невизначена, то або $T(K(x_1, x_2, \ldots x_n))$ не є кодом жодного числа з N, або машина Т не застосовується до слова $K(x_1, x_2, \ldots x_n)$

В подальшому будемо припускати, що в початковий момент часу голівка машини «бачить» саму ліву одиницю слова $K(x_1, x_2, \ldots x_n)$.

Якщо функція f обчислюється за Тьюрінгом за допомогою машини Т, то ми будемо говорити, що машина Т обчислює функцію f.

P_b - клас всіх частково-рекурсивних функцій, що обчислюються на машинах Тьюрінга. $P_{чр} = P_b$

Приклади:

1) $k_1 = q_1 1^n$ $\quad\quad$ $k_0 = q_0 1^n 1^n$

Програма м.т.	**Конфігурації**
$q_1 1 \to q_2 0R$	$q_1 1^n \vdash 0q_2 1^{n-1} \vdash 01q_2 1^{n-2} \vdash \ldots \vdash 01^{n-1} q_2 0 \vdash$
$q_1 0 \to q_0 0R$	$\vdash 01^{n-1} 0q_3 0 \vdash 01^{n-1} 01q_4 0 \vdash 01^{n-1} 0q_5 11 \vdash$
$q_2 1 \to q_2 1R$	$\vdash 01^{n-1} q_5 011 \vdash 01^{n-2} q_6 1011 \vdash \ldots \vdash$
$q_2 0 \to q_3 0R$	$\vdash q_6 01^{n-1} 01^2 \vdash q_1 1^{n-1} 01^2 \vdash 0q_2 1^{n-2} 01^2 \vdash \ldots \vdash$
$q_3 0 \to q_4 1R$	$\vdash 01^{n-2} q_2 01^2 \vdash 1^{n-2} 0q_3 1^2 \vdash 1^{n-2} 01^2 q_3 0 \vdash$
$q_3 1 \to q_3 1R$	$\vdash 1^{n-2} 01^2 1q_4 0 \vdash 1^{n-2} 01^2 11q_5 0 \vdash \ldots q_0 1^{2n}$
$q_4 0 \to q_5 1L$	
$q_5 1 \to q_5 1L$	
$q_5 0 \to q_6 0L$	
$q_6 1 \to q_6 1L$	
$q_6 0 \to q_1 0R$	

2) $k_1 = 1^n q_1 01^m$ $\quad\quad$ $k_0 = 1^m q_0 01^n$

$m \geq 1, n \geq 1$

Програма м.т.	**Конфігурації**
$q_1 0 \to q_2 0L$	$11q_1 01 \vdash 1q_2 101 \vdash 0q_2 1101 \vdash q_2 01101 \vdash$
$q_2 1 \to q_2 1L$	$\vdash q_3 1101 \vdash 0q_4 101 \vdash 01q_4 01 \vdash 010q_5 1 \vdash$
$q_2 0 \to q_3 0R$	$\vdash 0101q_5 0 \vdash 01010q_6 0 \vdash 0101q_7 01 \vdash$
$q_3 1 \to q_4 0R$	$\vdash 010q_8 101 \vdash 01q_8 0101 \vdash 0q_9 10101 \vdash$
$q_4 1 \to q_4 1R$	$\vdash q_9 010101 \vdash q_3 10101 \vdash q_4 0101 \vdash q_5 101 \vdash$

$q_4 0 \to q_5 0R$

$\vdash 1q_5 01 \vdash 10q_6 1 \vdash 101q_6 0 \vdash 10q_7 11 \vdash 1q_7 011$

$\vdash$

$q_5 1 \to q_5 1R$

$\vdash q_8 1011 \vdash q_9 001011 \vdash 0q_3 01011 \vdash q_{10} 1011$

$\vdash$

$q_5 0 \to q_6 0R$ $\vdash 1q_{10} 011 \vdash 1q_0 011$

$q_6 1 \to q_6 1R$

$q_6 0 \to q_7 1L$

$q_7 1 \to q_7 1L$

$q_7 0 \to q_8 0L$

$q_8 1 \to q_8 1L$

$q_8 0 \to q_9 0L$

$q_9 1 \to q_9 1L$

$q_9 0 \to q_3 0R$

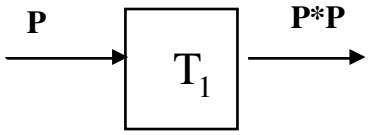
$q_3 0 \to q_{10} 0R$

$q_{10} 1 \to q_{10} 1R$

$q_{10} 0 \to q_0 0S$

Найпростіші програми машини Тьюрінга

1. Програма копіювання



Машина Тьюрінга T_1 починає працювати зі стрічкою, на якій записано слово P з алфавіту A, і через скінченну кількість тактів зупиняється і на стрічці записано слово P*P.

P – слово з алфавіту A= $\{\Lambda, a_1, a_2, \ldots, a_n\}$, тобто скінченна послідовність символів з A. Сукупність всіх слів з алфавіту A позначаємо $\alpha(A)$. Будемо вважати що:

1) що $* \notin A$;

2) що машина починає і закінчує свою роботу на самому лівому непорожньому символі, записаному на стрічці;

3) слово P складається тільки з не порожніх символів, тобто без прогалин (Λ ми можемо винести з слова).

Програма м.т.

$q_1 a_i \to q_2^i \Lambda R$

$q_2^i a_j \to q_2^i a_j R$

$q_2^i * \to q_3^i * R$

$q_2^i \Lambda \to q_3^i * R$

$q_3^i \Lambda \to q_4^i a_i L$

$q_3^i a_j \to q_3^i a_j R$

$q_4^i * \to q_5^i * L$

$q_4^i a_j \to q_4^i a_j L$

$q_5^i a_j \to q_5^i a_j L$

$q_5^i \Lambda \to q_6 a_i R$

$q_6 a_i \to q_2^i \Lambda R$

$q_6 * \to q_7 * L$

$q_7 a_i \to q_7 a_i L$

$q_7 \Lambda \to q_0 \Lambda R$

Наприклад,

$k_1 = a_3 a_2 a_1$ $\qquad$ $k_0 = a_3 a_2 a_1 * a_3 a_2 a_1$

$q_1 a_3 a_2 a_1 \vdash q_2^3 a_2 a_1 \vdash \ldots \vdash a_2 a_1 q_2^3 \Lambda \vdash a_2 a_1 * q_3^3 \Lambda \vdash a_2 a_1 q_4^3 * a_3 \vdash \ldots \vdash \Lambda a_2 q_5^3 a_1 * a_3 \vdash \ldots \vdash q_5^3 \Lambda a_2 a_1 * a_3 \vdash$

$\vdash a_3 q_6 a_2 a_1 * a_3 \vdash a_3 \Lambda q_2^2 a_1 * a_3 \vdash a_3 \Lambda a_1 q_2^2 * a_3 \vdash a_3 \Lambda a_1 * q_3^2 a_3 \vdash a_3 \Lambda a_1 * a_3 q_3^2 \Lambda \vdash a_3 \Lambda a_1 * q_4^2 a_3 a_2 \vdash$

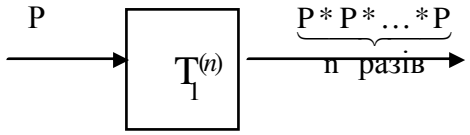
$\vdash \ldots \vdash a_3 \Lambda q_5^2 a_1 * a_3 a_2 \vdash a_3 q_5^2 \Lambda a_1 * a_3 a_2 \vdash a_3 a_2 q_6 a_1 * a_3 a_2 \vdash a_3 a_2 \Lambda q_2^1 * a_3 a_2 \vdash a_3 a_2 \Lambda * q_3^1 a_3 a_2 \vdash \ldots \vdash$

$\vdash a_3 a_2 \Lambda * a_3 a_2 q_3^1 \Lambda \vdash a_3 a_2 \Lambda * a_3 q_4^1 a_2 a_1 \vdash \ldots \vdash a_3 a_2 q_5^1 \Lambda * a_3 a_2 a_1 \vdash a_3 a_2 a_1 q_6 * a_3 a_2 a_1 \vdash a_3 a_2 q_7 a_1 * a_3 a_2 a_1$

$\vdash$

$\vdash \ldots \vdash q_7 \Lambda a_3 a_2 a_1 * a_3 a_2 a_1 \vdash q_0 a_3 a_2 a_1 * a_3 a_2 a_1$

Програма посиленого копіювання



Слово дублюється n раз.

Напишемо програму для n=3

Програма м.т.

$q_1 a_i \to q_2^i \Lambda R$

$q_2^i a_j \to q_2^i a_j R$

$q_2^i * \to q_3^i * R$

$q_2^i \Lambda \to q_3^i * R$

$q_3^i \Lambda \to q_4^i a_i L$

$q_3^i a_j \to q_3^i a_j R$

$q_4^i * \to q_5^i * L$

$q_4^i a_j \to q_4^i a_j L$

$q_5^i a_j \to q_5^i a_j L$

$q_5^i\Lambda \to q_1 a_i R$

$q_1 * \to q_6 * R$

$q_6 a_i \to q_7^i \Lambda R$

$q_7^i a_j \to q_7^i a_j R$

$q_7^i * \to q_8^i * R$

$q_7^i \Lambda \to q_8^i * R$

$q_8^i \Lambda \to q_9^i a_i L$

$q_8^i a_j \to q_8^i a_j R$

$q_9^i * \to q_{10}^i * L$

$q_{10}^i a_j \to q_{10}^i a_j L$

$q_{10}^i \Lambda \to q_6 a_i R$

$q_9^i a_j \to q_9^i a_j L$

$q_6 * \to q_{11} * L$

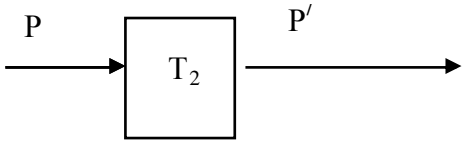
$q_{11} a_i \to q_{11} a_i L$

$q_{11} * \to q_{11} * L$

$q_{11} \Lambda \to q_0 \Lambda R$

2. Програма перекладу

Є два алфавіти: A= $\{a_0, a_1, \ldots, a_k\}$ і B= $\{b_0, \ldots, b_k\}$. Слово P – з A, $P^/$ - з B.



Машина Тюрінга T_2 переводить довільне слово P з A в слово $P^/$ з алфавіту B наступним чином: кожний елемент a_i замінюється на b_i

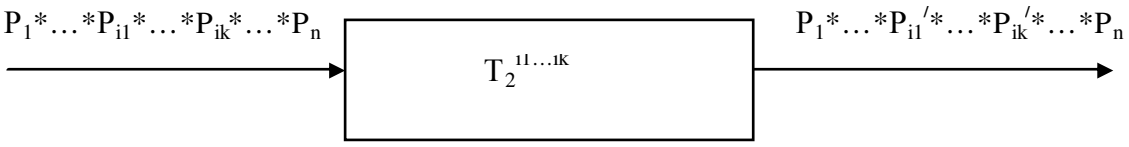
$q_1 a_i \to q_1 b_i R$

$q_1 \Lambda \to q_2 \Lambda L$

$q_2 b_i \to q_2 b_i L$

$q_2 \Lambda \to q_0 \Lambda R$

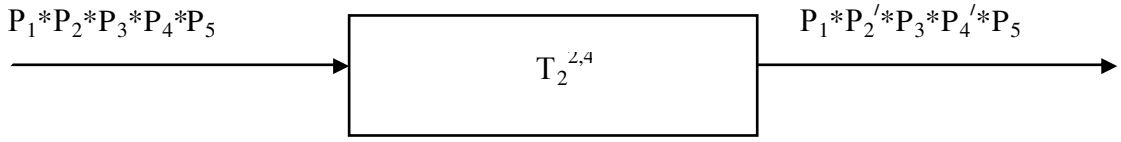
2.1 Програма вибіркового переводу



Нехай на стрічці написано n слів $P_1, \ldots, P_n$, що розділені * з алфавіта A.

Виділимо слова з номерами i1,i2,…ik. Машина $T_2^{i1\ldots ik}$ переводить в алфавіт B слова $P_{i1},\ldots,P_{ik}$ в $P_{i1}',\ldots,P_{ik}'$, а всі інші залишає без змін.

Приклад:



3. Програма сортування

Нехай два неперетинних алфавіти A= $\{a_1,\ldots,a_k\}$ і B= $\{b_1,\ldots,b_k\}$, $\{\Lambda,*\}\notin A\cup B$

Розглянемо $C = A\cup B, A\cap B = \varnothing$

Позначимо через α слово з алфавіту C, $\alpha\in L(C)$.

α(A) – слово, що отримане з α видаленням символів з алфавіту B, що до нього входять.

α(B) – слово, що отримане з α видаленням символів з алфавіту A, що до нього входять.

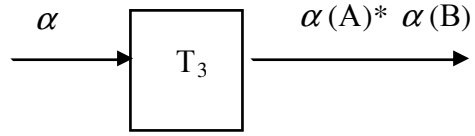
Наприклад:

$\alpha = a_1b_2a_3b_3b_1$

α(A)= a_1a_3

α(B)= $b_2b_3b_1$

Машина Тюрінга T_3 починає зі слова $\alpha\in L(C)$ і через скінченну кількість тактів зупиняється і на стрічці записано слово α(A)* α(B)



Програма м.т.

$q_1a_i \to q_1a_iR$

$q_1b_j \to q_{b_j}*R$

$q_{b_j}b_k \to q_{b_k}b_jR$

$q_{b_k}a_i \to q_{a_i}b_kL$

$q_{a_i}b_j \to q_{a_i}b_jL$

$q_{a_i}* \to q_1a_iR$

$q_{b_k}\Lambda \to q_2b_kL$

$q_2a_i \to q_2a_iL$

$q_2b_j \to q_2b_jL$

$q_2\Lambda \to q_0\Lambda R$

$q_2 * \to q_2 * L$

Операції над програмами

1. Операція суперпозиції
2. Введення заборонених символів
3. Розгалуження за сигнальною буквою (або умовний перехід)
4. Операція проекції
5. Об'єднання двох програм
6. Цикл

1. Операція суперпозиції

$\xrightarrow{P}$ [$T^{(1)}$] $\xrightarrow{f_1(P)}$

$\xrightarrow{P}$ [$T^{(2)}$] $\xrightarrow{f_2(P)}$

Побудуємо машину T_4, яка задане слово $P \in A$ перекладає в слово $f_2(f_1(P))$

$\xrightarrow{P}$ [T_4] $\xrightarrow{f_2(f_1(P))}$

Для цього змінимо в програмі $T^{(1)}$ стан q_0 на q_{m+1}

Програма $T^{(1)}$ $\begin{cases} q_1 \ldots \to \ldots \\ \vdots \\ q_m \ldots \to q_0 \ldots \end{cases}$

А програму $T^{(2)}$ перенумеруємо

Програма $T^{(2)}$ стара: $\begin{cases} q_1 \ldots \to \ldots \\ \vdots \\ q_l \ldots \to q_0 \ldots \end{cases}$; нова: $\begin{cases} q_{m+1} \ldots \to \ldots \\ \vdots \\ q_{m+l} \ldots \to q_0 \ldots \end{cases}$

В програмі $T^{(2)}$ стан q_0 не змінимо.

2. Операція введення заборонених символів

$A = \{a_1, \ldots, a_m\}$, $B = \{b_1, \ldots, b_m\}$, $A \cap B = \varnothing$, $C = A \cup B$, $P \in C$

$\xrightarrow{P \in A}$ [$T^{(1)}$] $\xrightarrow{f(P)}$, $f(P) \in B$

Побудуємо машину T_5, яка задане слово P з алфавіту C перекладає в нове слово, причому з символами з алфавіту A буде працювати так само як і машина $T^{(1)}$, а символи з алфавіту B не помічатиме.

Для цього кожний стан машини $T^{(1)}$ $\{q_0, \ldots, q_k\}$ перепишемо двома станами q_i^R, q_i^L

Наприклад, $q_i a_j \to q_k a_t R$ перепишемо $q_i^R a_j \to q_k^R a_t R$

$q_i a_j \to q_k a_t L$ перепишемо $q_i^L a_j \to q_k^L a_t L$

і доповнимо командами $q_i^R b_j \to q_k^R b_j R$ та $q_i^L b_j \to q_k^L b_j L$

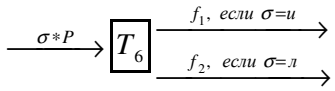
3. Розгалудження за сигнальною буквою

Нехай задані дві машини $T^{(1)}$ та $T^{(2)}$, що підраховують функції f_1 та f_2

$\xrightarrow{P}$ [$T^{(1)}$] $\xrightarrow{f_1(P)}$

$\xrightarrow{P}$ [$T^{(2)}$] $\xrightarrow{f_2(P)}$

Побудуємо машину T_6, яка перекладає слово $\sigma * P$, де σ - спеціальний символ з алфавіту $\{u, л\}$, в нове слово, при цьому якщо $\sigma = u$, то в слово $f_1(P)$, а якщо $\sigma = л$, то в слово $f_2(P)$



Для этого:

1. Перенумеруємо стани $T^{(2)}$ так, щоб не співпадали з станами машини $T^{(1)}$
2. Попередньо запишемо 4 команди

   $q_1 u \to q_2 \Lambda R$

   $q_2 * \to q_4 \Lambda R$

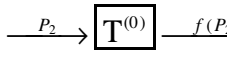
   $q_1 л \to q_3 \Lambda R$

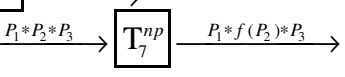
   $q_3 \to q_{k+4} \Lambda R$

   Стани $q_1, q_2, \ldots, q_k$ машини $T^{(1)}$ перенумеруємо через $q_4, q_5, \ldots, q_{k+3}$, q_0 не змінюємо. Стани $q_1, q_2, \ldots, q_m$ машини $T^{(2)}$ перенумеруємо через $q_{k+4}, q_{k+5}, \ldots, q_{m+k+3}$, q_0 не змінюємо.

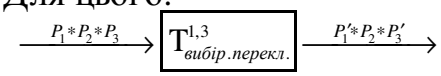
4.Операція проекції

Побудуємо машину $T^7 = T^{np}$, яка отримує з слова $P_1 * P_2 * P_3$, зберігає P_1 та P_3 без зміни, а зі словом P_2 діє за програмою $T^{(0)}$

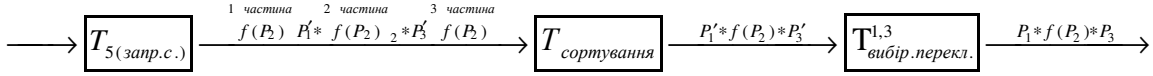


тобто 

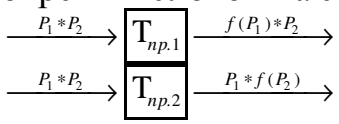
Для цього:



З алфавіту А слово P_1 перекладемо в слово з В, з алфавіту А' слово P_3 перекладемо в слово з алфавіту В', а слово P_2 збережемо без змін

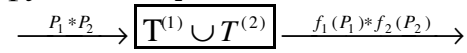


4.1 Розглянемо програму проекції першого слова, яка дозволяє працювати з окремим словом на стрічці, не змінюючи інші:

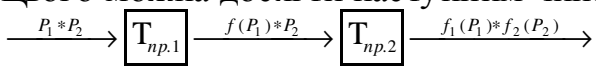


5. Операція об'єднання двох програм

Нехай машина Тюрінга $T^{(1)}$ обчислює функцію $f_1(P)$, а машина $T^{(2)}$ - функцію $f_2(P)$. Тоді об'єднанням двох програм буде:



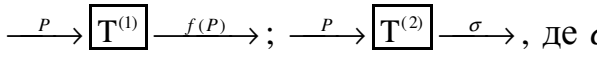
Цього можна досягти наступним чином:



6. Операція циклу

В програмах машини Тюрінга могуть бути цикли. Розглянемо один з них:

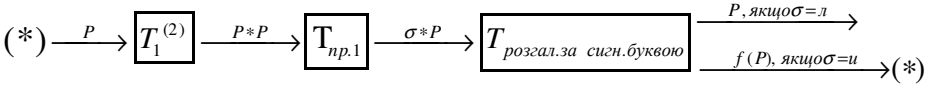
Нехай дана машина Тюрінга $T^{(1)}$, яка обчислює словарну функцію $f(P)$ і машина Тюрінга $T^{(2)}$, яка перевіряє чи задовольняє слово Р деякій властивості.

 , де $\sigma = \{u, л\}$

Можна скласти машину Тюрінга $T_8(T_{циклу})$, яка буде працювати наступним чином:

1. Якщо задане слово P задовольняє вказаній властивості, то машина обчислює значення $f(P)$;
2. Якщо $f(P)$ задовольняє цій же властивості, то машина обчислює $f(f(P))$; і т.д.
3. Якщо на деякому етапі отримане слово не задовольняє властивості, то машина Тюрінга зупиняється і на стрічці залишається це слово без змін.

Цього можна досягти наступним чином:



$T_1^{(2)}$ - програма копіювання (подвоєння слова)

$T_{пр.1}$ - перше слово перекладає по $T^{(2)}$ в символ з $\sigma = \{и, л\}$

Заключний стан програми $T_{розгал.за\ сиги.буквою}$ перенумеруємо на початковий стан машини $T_1^{(2)}$

Теорема Тюрінга

Числова функція є частково-рекурсивною тоді і тільки тоді, коли вона обчислювана на відповідній машині Тюрінга.

Доведення: вестимемо за структурою побудови функції.

Розглянемо найпростіші функції *o(x), S(x)* та $I_m^n(x_1,\ldots,x_m,\ldots,x_n)$. Покажемо, що ці функції обчислюванні.

Розглянемо алфавіт $\{0,1,\Lambda,*\}$. Будемо вважати, що нуль кодується нулем, а число $n \in N$ кодується 1^n, Λ - пустий символ, а * - символ відокремлення, тоді $(x_1,\ldots,x_n) \to 1^{x_1} * 1^{x_2} * \ldots * 1^{x_n}$, $x_i > 0$, $x_i \in N$

1. **програма обчислення функції** *o(x)*≡0
   на стрічці $\Lambda^\infty 1^n \Lambda^\infty$
   $q_1 1 \to q_1 \Lambda R$
   $q_1 \Lambda \to q_0 0 S$
   $q_1 0 \to q_0 0 S$
2. **програма додавання одиниці** *S(x)=x+1*
   $q_1 1 \to q_1 1 L$
   $q_1 \Lambda \to q_0 1 S$
   $q_1 0 \to q_0 1 S$
3. **програма вибору m-тої координати для обчислення функції**
   $I_m^n(x_1,\ldots,x_m,\ldots,x_n) = x_m$
   Програма стирає всі символи до слова 1^{x_m}, рахуючи при цьому *, не змінює слова 1^{x_m} та продовжує стирати до появи порожнього символу.

$q_1 1 \to q_1 \Lambda R$	$q_{m+1} 1 \to q_{m+1} \Lambda R$
$q_1 0 \to q_1 \Lambda R$	$q_{m+1} 0 \to q_{m+1} \Lambda R$
$q_1 * \to q_2 \Lambda R$	$q_{m+1} * \to q_{m+1} \Lambda R$
$q_2 1 \to q_2 \Lambda R$	…
$q_2 0 \to q_2 \Lambda R$	$q_n 1 \to q_n \Lambda R$
$q_2 * \to q_3 \Lambda R$	$q_n 0 \to q_n \Lambda R$
…	$q_n * \to q_{n+1} \Lambda R$
$q_{m-1} 1 \to q_{m-1} \Lambda R$	$q_{n+1} 1 \to q_{n+1} \Lambda R$
$q_{m-1} 0 \to q_{m-1} \Lambda R$	$q_{n+1} 0 \to q_{n+2} \Lambda R$
$q_{m-1} * \to q_m \Lambda R$	$q_{n+1} * \to q_{n+2} \Lambda R$
$q_m 1 \to q_m 1 R$	$q_{n+2} 1 \to q_{n+2} 1 L$
$q_m 0 \to q_m 0 R$	$q_{n+2} 0 \to q_{n+2} 0 L$
$q_m * \to q_{m+1} \Lambda R$	$q_{n+2} \Lambda \to q_0 \Lambda R$

Розглянемо операції С, ПР та μ. Складемо відповідні машини Тюрінга.

1. Операція С(суперпозиції)

   Нехай функція $F(x_1, x_2, \ldots, x_n)$ має вигляд:

   $F(x_1, x_2, \ldots, x_n) = f_0(f_1(x_1, \ldots, x_n), \ldots, f_k(x_1, \ldots, x_n))$, де f_0 - функція к змінних, а $f_1, \ldots, f_k$ - n змінних

   Нехай існують машини Тюрінга $T_0, T_1, \ldots, T_k$, які обчислюють відповідні функції $f_0, f_1, \ldots, f_k$. Тоді, отримаємо функцію $F(x_1, x_2, \ldots, x_n)$ наступним чином:

   $\xrightarrow{x_1 * x_2 * \ldots * x_n} \boxed{T_1^{(k)}} \xrightarrow{(x_1 * x_2 * \ldots * x_n) * \ldots * (x_1 * x_2 * \ldots * x_n)}$ програма копіювання *к* раз

   $\longrightarrow \boxed{np_1 T_1} \xrightarrow{f_1(x_1 * x_2 * \ldots * x_n) * (x_1 * x_2 * \ldots * x_n) * \ldots * (x_1 * x_2 * \ldots * x_n)}$

   $\longrightarrow \boxed{np_2 T_2} \xrightarrow{f_1(x_1 * x_2 * \ldots * x_n) * f_2(x_1 * x_2 * \ldots * x_n) * \ldots * (x_1 * x_2 * \ldots * x_n)} \ldots \longrightarrow$

   $\longrightarrow \boxed{np_k T_k} \xrightarrow{f_1(x_1 * x_2 * \ldots * x_n) * f_2(x_1 * x_2 * \ldots * x_n) * \ldots * f_k(x_1 * x_2 * \ldots * x_n)} \boxed{T_0} \xrightarrow{F(x_1, \ldots, x_n)}$

2. Операція ПР(примітивної рекурсії)

   Нехай функція $f(x)$ отримана за схемою примітивної рекурсії за допомогою константи *a* та функції $g(x, y)$, якщо: $\begin{cases} f(0) = a \\ f(x+1) = g(x, f(x)) \end{cases}$

   Нехай існує програма машини Тюрінга $T^{(0)}$, яка обчислює функцію $g(x, y)$. Мі повинні скласти програму, яка обчислює функцію $f(x)$ за схемою примітивної рекурсії.

   f(0)=a,

   f(1)=g(0,f(0))=g(0,a)

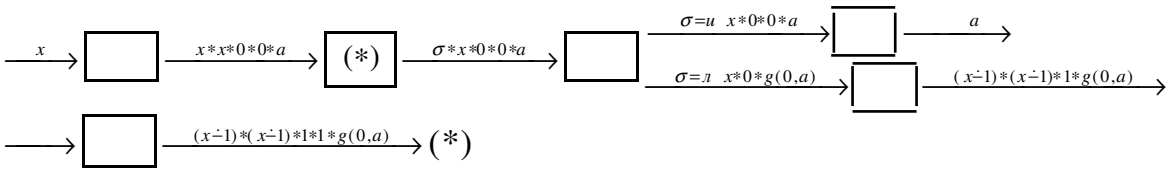
   f(2)=g(1,f(1))

   …

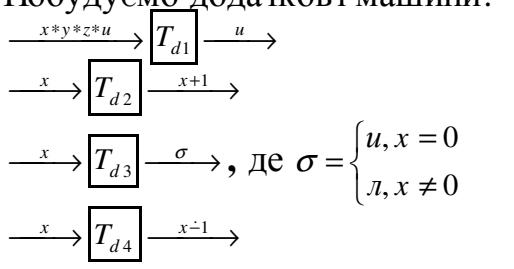
   f(x+1)=g(x,f(x))

   $\xrightarrow{x * y} \boxed{T^{(0)}} \xrightarrow{g(x, y)}$

   Розглянемо алгоритм

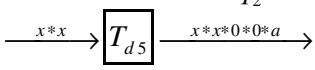


Побудуємо додаткові машини:

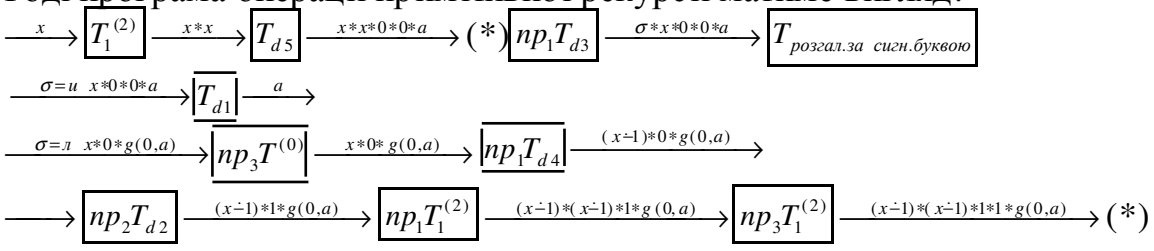


Програма T_{d3}: $q_1 0 \to q_0 u S$; $q_1 1 \to q_1 \Lambda R$; $q_1 * \to q_2 * L$; $q_2 \Lambda \to q_0 л S$

Програма T_{d4}: $q_1 0 \to q_0 0 S$; $q_1 1 \to q_1 \Lambda R$; $q_1 * \to q_2 * L$; $q_2 \Lambda \to q_0 \Lambda S$



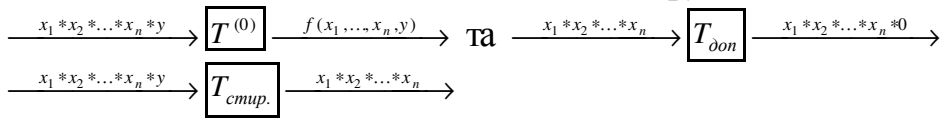
Тоді програма операції примітивної рекурсії матиме вигляд:



3. Операція мінімізації

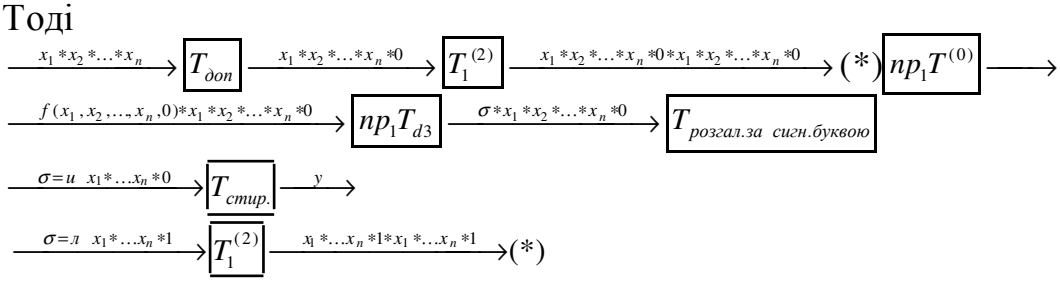
$\mu(f(x_1,\ldots,x_n,y)=0) = g(x_1,\ldots,x_n)$

Нехай задана машина $T^{(0)}$, яка обчислює функцію $f(x_1,\ldots,x_n,y)$.



T_{d3}: $\sigma = \begin{cases} u, f = 0 \\ л, f \neq 0 \end{cases}$

Тоді



Теза Черча: будь-який індуктивно-зрозумілий алгоритм може бути сформульований у вигляді частково рекурсивної функції.

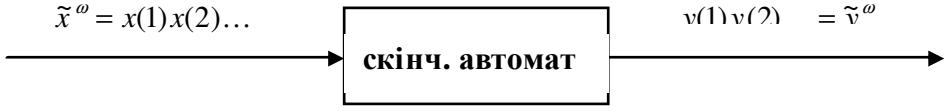
Скінченні автомати

Детерміновані і обмежено-детерміновані функції

Нехай А – непорожній скінченний алфавіт. Словом в алфавіті А називається послідовність, складена з букв алфавіту А. Довжина (число букв) в слові ω позначається $\lambda(\omega)$. Множина всіх слів $\tilde{x}^s = x(1)x(2)\ldots x(s)$ довжини $s \geq 1$ в алфавіті А буде позначатися через A^s. Слово довжини 0 (порожнє слово) будемо позначати символом λ.

A^ω - множина всіх нескінченних слів $\tilde{x}^\omega = x(1)x(2)\ldots$, где $x(t) \in A, t = 1,2,\ldots$.

Розглянемо деякий дискретний пристрій (автомат), що працює в дискретні моменти часу $t = 1,2,\ldots$.На вході цього пристрою в кожен момент часу t подається сигнал $x(t)$, а на виході з`являється сигнал $y(t)$



$x(t) \in A = \{a_1, a_2, \ldots a_k\}$

$y(t) \in B = \{b_1, b_2, \ldots b_l\}$

Тоді, можна вважати, що автомат реалізує l функцій, кожна з яких залежить від k змінних.

$A^\omega = \{\tilde{x}^\omega : x(t) \in A, \forall t\}$

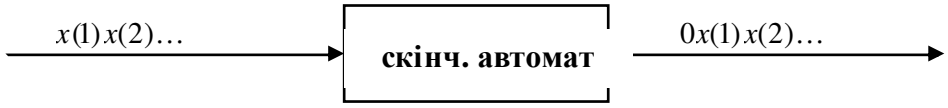
$B^\omega = \{\tilde{y}^\omega : y(t) \in B, \forall t\}$

$A^\omega \xrightarrow[\varphi]{} B^\omega$. Серед всіх відображень φ виділимо клас детермінованих. Відображення $\varphi: A^\omega \to B^\omega$ називається детермінованою функцією, або детермінованим оператором, якщо воно задовольняє наступним умовам:

а) для будь-якого $s \geq 1$ s-й символ $y(s)$ слова $\tilde{y}^\omega = \varphi(\tilde{x}^\omega)$ є однозначною функцією перших s символів $x(1), x(2), \ldots, x(s)$ слова $\tilde{x}^\omega$ (тобто інформації, що отримана на такті t і до такту t, достатньо для однозначного визначення $y(t)$).

Приклади: Нехай $A = \{0,1\}$ і $B = \{0,1\}$, будемо позначати $1^\omega = 111\ldots$, $0^\omega = 000\ldots$

1) $y(t) = \begin{cases} x(t-1), t \geq 2 \\ 0, t = 1 \end{cases}$ - детермінована

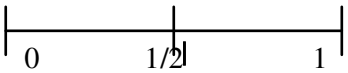


2) $\tilde{y}(t) = \begin{cases} 1^\omega, \textit{если в } \tilde{x}^\omega \textit{нескінченне число нулів} \\ 0^\omega, \textit{в супротивному випадку} \end{cases}$ - не детермінована

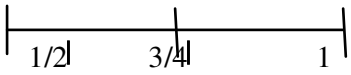
3) $y(t) = x(t+1)$ - не детермінована $\Leftrightarrow \varphi(x(1)x(2)\ldots) = x(2)x(3)\ldots$

4) кожному слову $\tilde{x}^{\omega} = x(1)x(2)\ldots x(t)\ldots$ з $\{0,1\}^{\omega}$ відповідає деяке число $\upsilon(\tilde{x}^{\omega})$ з відрізку [0,1], двійкове розкладання котрого має вигляд $0, x(1)x(2)\ldots x(t)\ldots$. Якщо $a \in [0,1]$, то його двійкове розкладання $0, a_1 a_2 a_3 \ldots$ породжує слово <a> $= x(1)x(2)\ldots x(t)\ldots$.

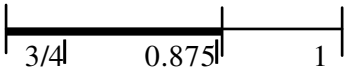
p=0.802



Якщо 0,5<p, то x(1)=1
Якщо 0,5>p, то x(1)=0
тобто <p>=1…



Якщо 0,75<p, то x(2)=1
Якщо 0,75>p, то x(2)=0
тобто <p>=11…



тобто <p>=110…

Побудуємо обернену функцію $\upsilon(\langle p \rangle) = p$

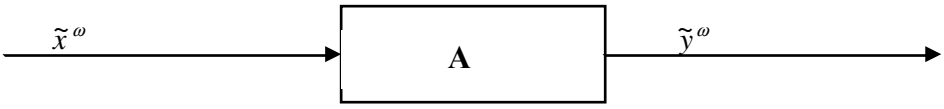
$$p = \upsilon(\tilde{x}) = x(1) * 2^{-1} + x(2) * 2^{-2} + \ldots + x(t) * 2^{-t} + \ldots$$

Розглянемо $\tilde{y} = \left\langle \frac{\upsilon(\tilde{x})}{4} \right\rangle = 0 * 2^{-1} + 0 * 2^{-2} + x(2) * 2^{-3} + \ldots + x(t-2) * 2^{-t} + \ldots$

Інформативні дерева

При розгляданні детермінованих функцій зручно використовувати графічне зображення їх у вигляді нескінченних інформативних дерев.

Розглянемо скінченний автомат з одним входом і одним виходом, де на вихід потрапляє в кожен такт часу t, $t \geq 1$, сигнал $x(t)$ з алфавіту $A = \{a_1, a_2, \ldots, a_k\}$, а на

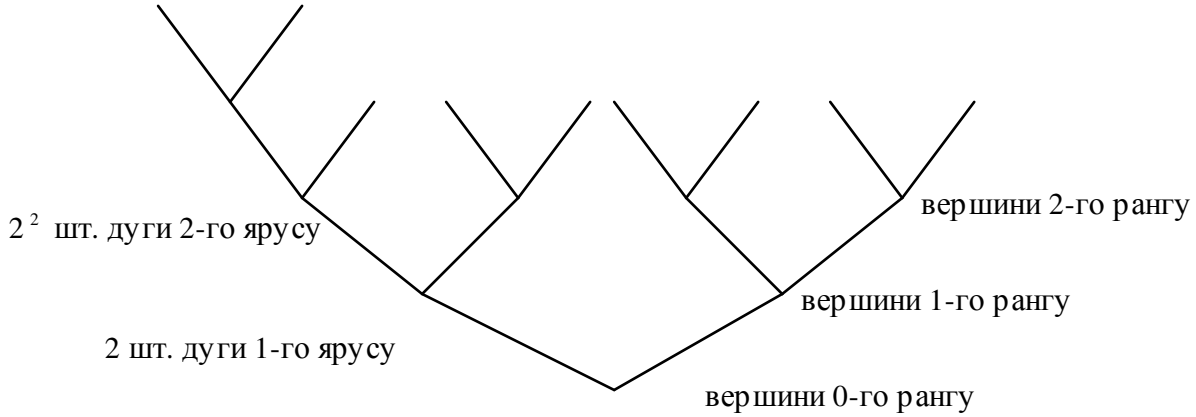


виході з`являється в цей же такт часу t сигнал $y(t)$ з алфавіту $B = \{b_1, b_2, \ldots, b_l\}$. Ми розглядаємо детерміновані відображення $\varphi(\tilde{x}^{\omega})$: з A^{ω} в B^{ω} $A^{\omega} \underset{\varphi}{\longrightarrow} B^{\omega}$. $\varphi(\tilde{x}^{\omega})$ задавали аналітично (наприклад $\varphi(\tilde{x}^{\omega}) = \left\langle \frac{\upsilon(\tilde{x})}{2} \right\rangle$), описана(приклад2) або задавали вихідні сигнали y(1),y(2),…y(t) через x(1),x(2),…x(t)(приклад1).

Розглянемо найпростіший алфавіт $A=\{0,1\}$.Побудуємо інформативне дерево D_A наступним чином :

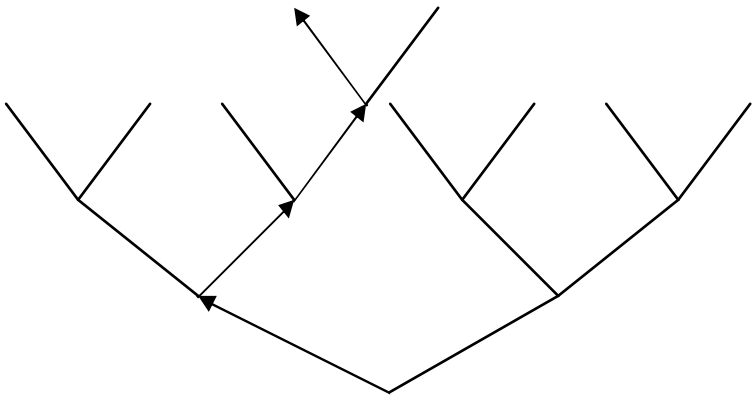
1) З вихідної точки, кореня дерева, випускаємо два ребра (де ліве ребро – інформативний нуль, а праве – інформативна одиниця). Корінь дерева будемо вважати вершиною нульового рангу, а ребра – дугою 1-го ярусу. Ребра не перетинаються і закінчуються в вершинах 1-го рангу.
2) З кожної вершини 1-го рангу проводимо по 2 дуги 2-го ярусу, котрі закінчуються в вершинах 2-го рангу. Ці ребра також не перетинаються ні між собою, ні з попередніми і т.д.
k) В k-тому ярусі ми побудували 2^k дуг і вершин. Символу 0 відповідає ліва дуга, а символу 1 – права.

Зобразимо фрагмент дерева $D_A=\{0,1\}$

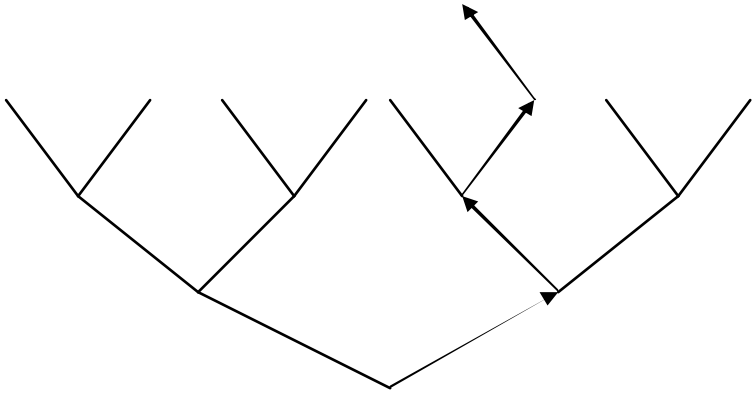


Кожній двійковій нескінченній послідовності $\tilde{x}^{\omega}$можна поставити у відповідність нескінченний двійковий шлях із кореня дерева вгору до нескінченності. І, навпаки, кожен шлях з кореня дерева вгору може бути записаний у вигляді слова $\tilde{x}^{\omega}$.

Наприклад: Розглянемо $\tilde{x}^{\omega}$=0110…, зобразимо відповідний шлях на D_A

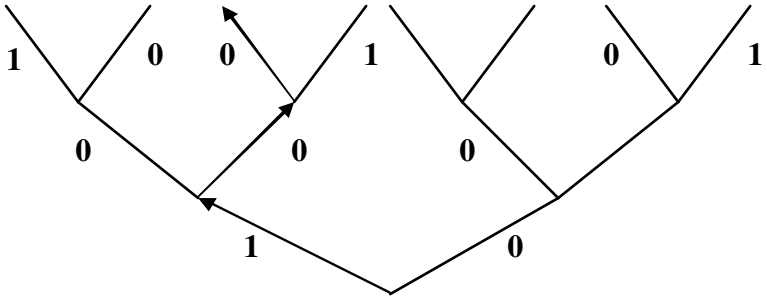


або розглянемо шлях по D_A, йому відповідає слово $\tilde{x}^{\omega}$=1010…



Якщо розглядати всі можливі шляхи з кореня вгору, то будь-які два з них або повністю співпадають або співпадають до деякого місця, а потім розходяться і більше не співпадають. Таким чином, можливих шляхів – незліченна множина.

Означення: Навантажене деревом $D_{A,B}$ отримуємо з дерева D_A шляхом приписування кожній дузі деякого символу з алфавіту B. Розглянемо приклад алфавіту $B=\{0,1\}$. Будь-якому орієнтованому нескінченному ланцюгу в дереві $D_{A,B}$ відповідає слово з B^ω, складене з букв, приписаних дугам цього ланцюга. Тому можна вважати, що нагружене дерево $D_{A,B}$ задає(реалізує) відображення $\varphi: A^\omega \to B^\omega$, що є детермінованою функцією. Наприклад,

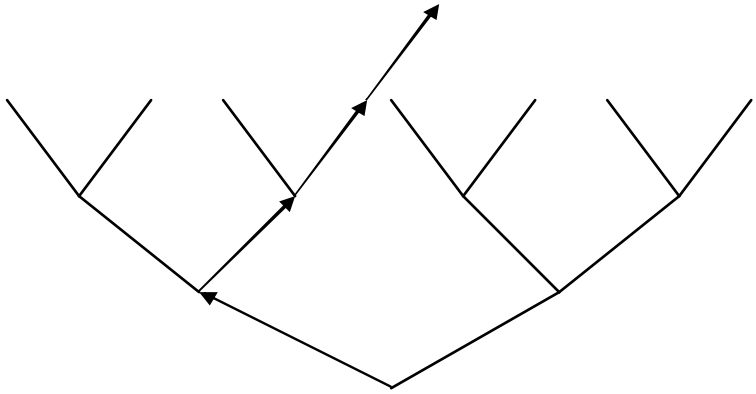


$\tilde{x}^\omega$=010…, $\tilde{y}^\omega$=100…

Твердження: Кожній детермінованій функції $\varphi(\tilde{x}^\omega)=\tilde{y}^\omega$, що відображує $A^\omega \to B^\omega$, відповідає деяке навантажене інформативне дерево $D_{A,B}$ і навпаки.

Доведення:

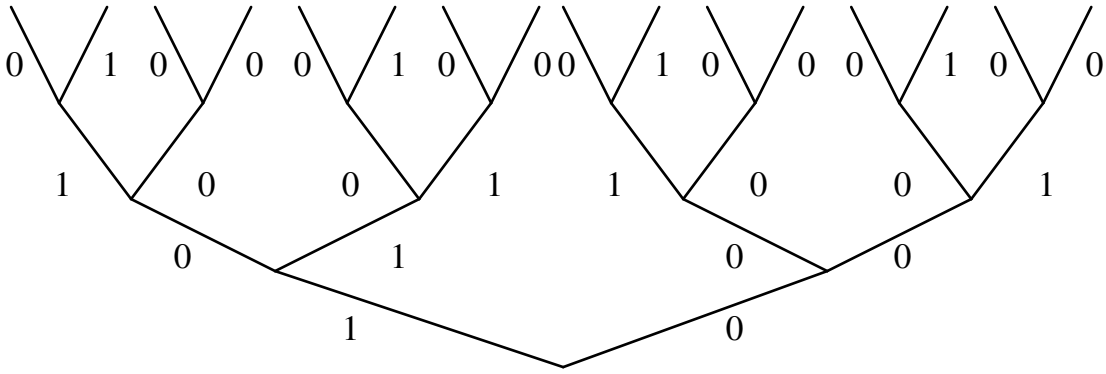
1) Нехай задана детермінована функція $\varphi(\tilde{x}^\omega)=y(1)y(2)\ldots$. Розглянемо алгоритм розстановки міток. Беремо будь-яке ребро і знаходимо шлях з кореня до вершини, в якій закінчується це ребро. Наприклад, $x(1)x(2)x(3)x(4)=0111$. Відповідно до означення детермінованої функції цей кортеж однозначно визначає деякий символ з B. $0111=x(1)x(2)x(3)x(4)$. Його і поставимо за мітку на даному ребрі. І так будемо чинити з кожною дугою(з кожним ребром).



2) Якщо мітки на всіх ребрах, потрібно побудувати детерміновану функцію φ. Для цього беремо довільну послідовність $\tilde{x}^{\omega}$, їй відповідає шлях з кореня до нескінченності. В якості $\varphi(\tilde{x}^{\omega})$ будемо брати послідовність міток вздовж цього шляху.

Приклад:

$$\begin{cases} y(1) = \bar{x}(1) \\ y(2t) = x(2t) * \bar{x}(2t-1) \\ y(2t+1) = x(2t+1) \approx x(2t) \end{cases}$$



Вага дерева. Обмежено-детерміновані функції

Будь-які дві детерміновані функції $\varphi_1(\tilde{x}^{\omega})$ і $\varphi_2(\tilde{x}^{\omega})$ називаються розрізненими, якщо існує вхідне слово $\tilde{x}_0{}^{\omega}$, що перетворюються цими функціями в різні вихідні слова, тобто $\varphi_1(\tilde{x}^{\omega}) = \tilde{y}_1{}^{\omega}$, $\varphi_2(\tilde{x}^{\omega}) = \tilde{y}_2{}^{\omega}$, $\tilde{y}_1{}^{\omega} \neq \tilde{y}_2{}^{\omega}$. Якщо ж $\varphi_1(\tilde{x}^{\omega}) = \varphi_2(\tilde{x}^{\omega})$ при будь-якому вхідному слові $\tilde{x}^{\omega}$, то $\varphi_1(\tilde{x}^{\omega})$ і $\varphi_2(\tilde{x}^{\omega})$ називається еквівалентними(нерозрізненими).

Якщо існує таке слово $\tilde{x}_0^{\,s} = x_0(1)\ \ldots\ x_0(\mathrm{s})$, що $\varphi(\tilde{x}_0^{\,s}\tilde{x}^{\omega}) = \varphi(\tilde{x}_0^{\,s})\,\psi(\tilde{x}^{\omega})$ для будь-якого слова $\tilde{x}^{\omega}$, то оператор ψ називається залишковим оператором оператора φ, породженим словом $\tilde{x}_0^{\,s}$ і позначається $\varphi_{\tilde{x}_0^{\,s}}$.

Нехай $D_{A,B}$ - навантажене дерево, що реалізує детерміновану функцію $\varphi(\tilde{x}^{\omega})$. Залишковому оператору $\varphi_{\tilde{x}_0^{\,s}}\ s \geq 0$ оператора φ відповідає дерево $D_{A,B}(\tilde{x}_0^{\,s})$, що виходить з такої вершини α s-го рангу, в якій закінчується ланцюг, що виходить з кореня і містить рівно s дуг, причому з кореня α_0 в вершину α веде шлях $x_0(1)\ \ldots\ x_0(\mathrm{s})$.

Так як вихідне дерево навантажене мітками, то піддерево, породженому вершиною α, ввести нумерацію ярусів починаючи з 1-го, то йому буде відповідати детермінована функція, що є залишковим оператором $\varphi_{\tilde{x}_0^{\,s}}$ вихідної детермінованої функції φ.

Якщо залишкові оператори $\varphi_{\tilde{x}_0^{\,s1}}$ і $\varphi_{\tilde{x}_0^{\,s2}}$ еквівалентні, то відповідні вершини (α_1 і α_2) і піддерева, що ростуть з них також називаються еквівалентними. Очевидно, що при природному накладанні двох еквівалентних піддерев їх мітки, що зчитуються на кожному ярусі зліва направо, співпадають.

Попарно різних залишкових операторів може бути скінченне і нескінченне число.

Будемо виділяти еквівалентні піддерева одним і тим же станом $q_i\ (i = 0,1\ldots)$.

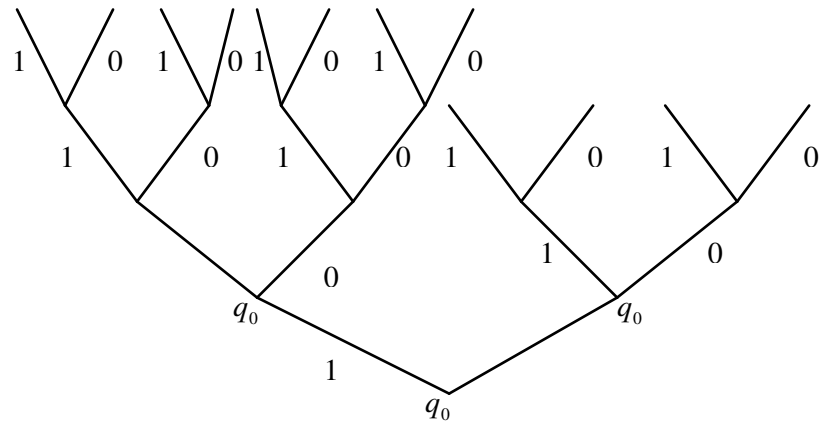
Співвідношення еквівалентності дозволяє в початковому дереві множину всіх піддерев розбити на класи еквівалентності.

Означення. Число r класів еквівалентності, на які розбивається множина всіх піддерев даного дерева, називається вагою дерева і відповідно вагою детермінованої функції.

Інакше кажучи, вага — це максимальне число попарно неекві валентних піддерев. При цьому не виключається випадок, коли всі r нескінченні.

Означення. Детермінована функція $\varphi(\tilde{x}^{w}) = \tilde{y}^{w}$ називається обмежено-детермінованою, якщо вона має скінчену вагу.

Приклади:



1.

$\varphi(\tilde{x}^{w}) = \tilde{y}^{w}$

$y(t) = \bar{x}(t),\ t \geq 1$

$\tilde{x}^{w} = 010101\ldots$

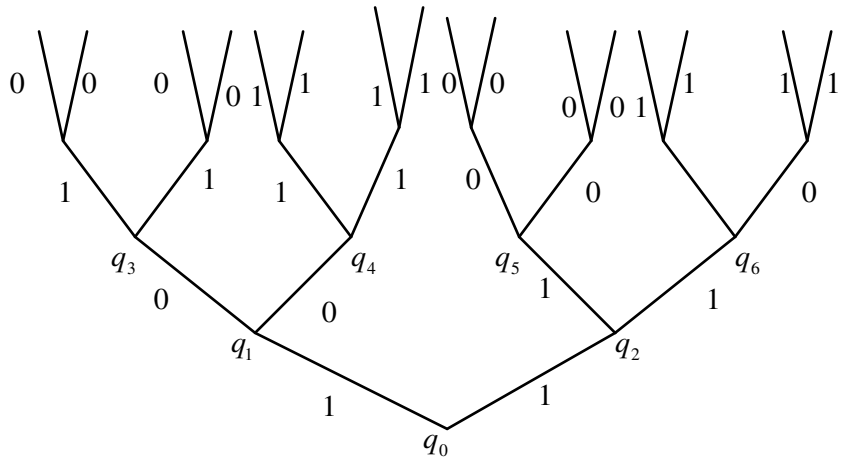
$\tilde{y}^{w} = 101010\ldots$

$\tilde{x}^{w} = 111111\ldots$

$\tilde{y}^{w} = 000000\ldots$

r=1 => φ- обмежено-детермінована функція

2. $\begin{cases} y(1) = 1 \\ y(2t) = x(t),\ t \geq 1 \\ y(2t+1) = \overline{x}(t),\ t \geq 1 \end{cases}$



$t = 1 \quad y(1) = 1$

$y(2) = x(1)$

$y(3) = \overline{x}(1)$

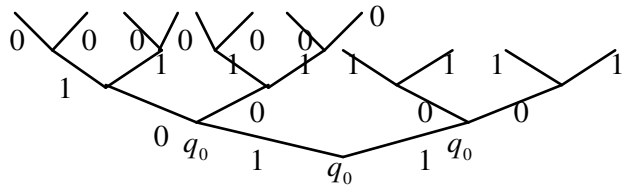
$t = 2 \quad y(4) = x(2)$

$y(5) = \overline{x}(2)$

$t = 3 \qquad \ldots$

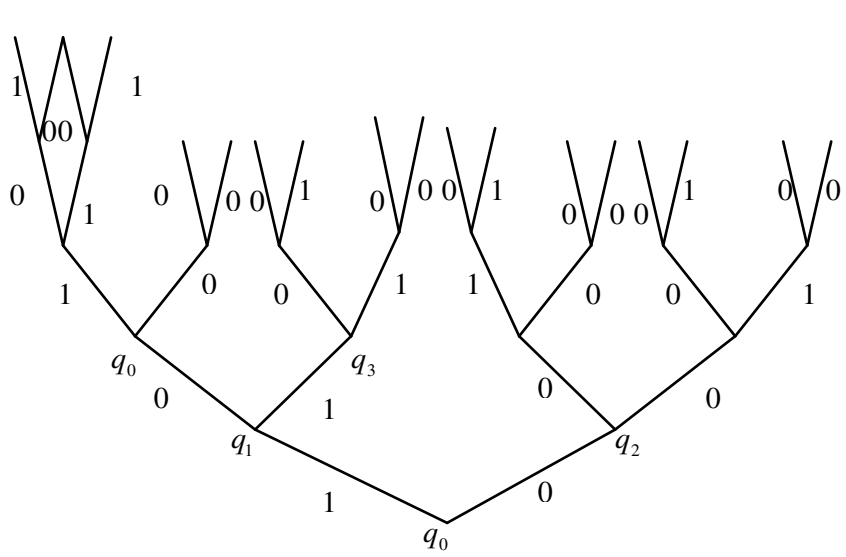
$r = \infty \Rightarrow \varphi$ - не обмежено-детермінована функція

3. $\varphi(\tilde{x}^w) = 1010010001000001\ldots \quad \forall \tilde{x}^w$



$r = \infty \Rightarrow \varphi$ - не обмежено-детермінована функція

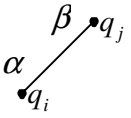
4. $\begin{cases} y(1) = \overline{x}(1) \\ y(2t) = x(2t) \cdot \overline{x}(2t-1) \\ y(2t+1) = x(2t+1) \equiv x(2 \end{cases}$



Отримали чотири різні стани.
Операторів:
$q_0, q_1, q_2, q_3, \Rightarrow$
$r = 4 \Rightarrow \varphi$ - обмежено-детермінована функція

Обмежено-детермінованих функцій скінчене число.
Розглянемо обмежено-детерміновану функцію. Нехай для неї побудовано інформативне навантажене дерево і вершини помічені скінченим числом станів $q_0, q_1, \ldots, q_k$.
Розглянемо дві вершини, помічені одним і тим самим станом q_i $(0 \le i \le k)$, з них виходять еквівалентні піддерева. Якщо йти з цих двох вершин по ребрам, що мають один і той же інформативний шлях, то мітки на цих ребрах будуть однакові, і ми потрапимо в вершини, які будуть відмічені однаковим станом. Отже, мітка однозначно визначається інформативним значенням ребра і станом при вершині.

$q_0, q_1, q_2, q_3, \Rightarrow$



Якщо в момент t ми знаходились в вершині α і стан був q_i, яке ми перевизначимо через q(t), то при надходженні в момент t числа x(t) ми рухаємся по ребру і переходимо в вершину β зі станом $q_j = q(t+1)$, при цьому отримаємо вихідне значення y(t). Таким чином, величини (x(t),q(t)) однозначно визначають величини y(t), q(t).
Тоді y(t) та q(t+1) можуть бути задані канонічними рівняннями оператора φ з початковою умовою q_0:

$$(1)\begin{cases} y(t) = F(x(t), q(t)) \\ q(t+1) = G(x(t), q(t)) \\ q(1) = q_0 \end{cases}$$

де F називають функцією виходів, а G – функцією переходів.
Якщо $\varphi(\tilde{x}^w) = \tilde{y}^w$ обмежено-детермінований оператор, то функції F(x(t),q(t)) та G(x(t),q(t)), а також їх аргументи, від яких вони залежать, приймають скінчене число значень, тому можливо табличне значення обмежено-детермінованого оператора φ з допомогою канонічної таблиці.

x(t)	q(t)	y(t)	q(t+1)
0	q_0	$F(0, q_0)$	$G(0, q_0)$
1	q_0	$F(1, q_0)$	$G(1, q_0)$
0	q_1	…	…
1	q_1		
…	…	…	…
0	q_{r-1}		
1	q_{r-1}	$F(1, q_{r-1})$	$G(1, q_{r-1})$

Замість канонічних рівнянь (1) буває зручно розглядати такі канонічні рівняння, в яких функції виходів і переходів є функціями двійкової логіки. Для отримання відповідного
представлення оператора φ, необхідний стани $\{q_0, q_1, \ldots, q_{r-1}\}$ закодувати впорядкованими кортежами з нулів та одиниць.

q_0 0 0 0 … 0 0

q_1 0 0 0 … 0 1

q_2 0 0 0 … 1 0

q_3 0 0 0 … 1 1

…

q_{r-1} *двійкове представлення числа* $r-1$

Поточний стан q(t) будемо записувати в скалярній формі $q_0(t), q_1(t),...,q_l(t)$, де довжина
кортежу l така, що $2^l \geq r$, тоді $l-1=[\log_2 r]$

Тоді таблиця прийме вигляд

x(t)	$q_1(t)$	…	$q_l(t)$	y(t)	$q_1(t+1)$	…	$q_l(t+1)$
0	0	…	0	…	…	…	…
1	0	…	1	…	…	…	…
…	…	…	…	…	…	…	…

а система канонічних рівнянь (1) буде записана в вигляді (2):

$$(2)\begin{cases} y(t)=F(x(t),q_1(t),...,q_l(t)) \\ q_1(t+1)=G_1(x(t),q_1(t),...,q_l(t)) \\ \dots \\ q_l(t+1)=G_l(x(t),q_1(t),...,q_l(t)) \\ q_1(1)=0 \\ \dots \\ q_l(1)=0 \end{cases}$$

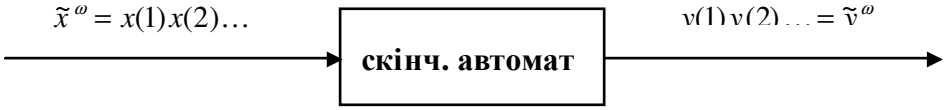
де F - функція виходів, а G_i – функція переходів, $i=\overline{1,l}$, $q_i(1)$ - початкова умова.

Рівняння системи (2) називаються канонічними рівняннями в скалярній формі. Функції F та G_i

можно записати тепер в вигляді ДНФ та КНФ.

Представлення детермінованої функції інформативно навантаженими деревами на алфавітах A і B.

Розглянемо скінченний автомат



де $\begin{matrix} x(t)\in A=\{a_0,a_1,\dots,a_s\} \\ y(t)\in B=\{b_0,b_1,\dots,b_p\} \end{matrix}$. Наприклад, $\begin{matrix} A=\{a_0,a_1,a_2\} \\ B=\{b_0,b_1,b_2,b_3\} \end{matrix}$

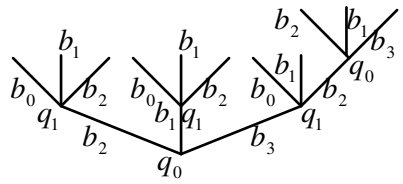
Розглянемо φ: $A^w \to B^w$, яка є детермінованою.

Побудуємо інформативне дерево наступним чином:

1. З початкової точки, кореня дерева випускаємо s ребер, що не перетинаються. Ліве ребро — інформативний символ a_0, наступне — a_1,..., останнє – a_s

2. З кожної вершини першого рангу проводимо по s ребер другого ярусу, де ребра несуть інформацію $(a_0,a_1,a_2,\dots,a_s)$

Далі побудуємо навантажене дерево, шляхом приписування кожній дузі символу з алфавіту B. Наприклад, нехай φ:



Оскільки станів скінченне число, тоді вага r=2 => φ - обмежено-детермінована, її можна задати у вигляді канонічної таблиці:

x(t)	q(t)	y(t)	q(t+1)
a_0	q_0	b_2	q_1
a_1	q_0	b_1	q_1
a_2	q_0	b_3	q_1
a_0	q_1	b_0	q_0
a_1	q_1	b_1	q_0
a_2	q_1	b_2	q_0

Для отримання представлення обмежено-детермінованого оператора □ в вигляді системи канонічних рівнянь в скалярній формі алфавіти A, B і $Q=\{q_0, q_1, ..., q_{r-1}\}$ кодуються векторами, координати яких є кортежами з нулів та одиниць.

$x(t) \to (x_1(t),..., x_m(t)),\ де\ 2^m \geq s$

$y(t) \to (y_1(t),..., y_n(t)),\ де\ 2^n \geq p$

$q(t) \to (q_1(t),..., q_l(t)),\ де\ 2^l \geq r$

Наприклад,

$a_0 = (0,0)$ $b_0 = (0,0)$ $q_0 = (0)$

$a_1 = (0,1)$ $b_1 = (0,1)$ $q_1 = (1)$

$a_2 = (1,0)$ $b_2 = (1,0)$

$b_3 = (1,1)$

Тоді таблиця прийме вигляд

$x_1(t)$	…	$x_m(t)$	$q_1(t)$	…	$q_l(t)$	$y_1(t)$	…	$y_n(t)$	$q_1(t+1)$	…	$q_l(t+1)$
…	…	…	…	…	…	…	…	…	…	…	…

Таблиця задає булеві n функцій виходу та l функцій переходу, які за допомогою ДНФ та КНФ записують у вигляді системи канонічних рівнянь в скалярній формі:

$$(3)\begin{cases} y_i(t) = F_i(x_1(t),..., x_m(t), q_1(t),..., q_l(t)),\ i = \overline{1,n} \\ q_j(t) = G_j(x_1(t),..., x_m(t), q_1(t),..., q_l(t)),\ j = \overline{1,l} \\ q_1(1) = 0,\ \ \ldots\ \ , q_l(1) = 0 \end{cases}$$

Для нашого прикладу канонічна таблиця в скалярній формі:

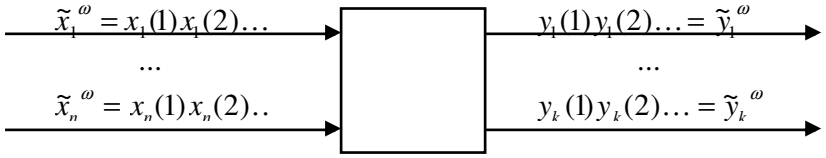
x(t)	q(t)	y(t)	q(t+1)
0	0	0	1
0	1	0	1
1	0	0	1
0	0	1	0

0	1	1	0
1	0	1	0

і система канонічних рівнянь

$$\begin{cases} y_1(t) = \overline{x}_1(t)\overline{x}_2(t)q_1(t) \vee x_1(t)\overline{x}_2(t)\overline{q}_1(t) \vee x_1(t)\overline{x}_2(t)q_1(t) \\ y_2(t) = \overline{x}_1(t)x_2(t)\overline{q}_1(t) \vee x_1(t)\overline{x}_2(t)\overline{q}_1(t) \vee x_1(t)x_2(t)q_1(t) \\ q_1(t+1) = \overline{q}_1(t) \\ q_1(1) = 0 \end{cases}$$

Представлення детермінованих функцій багатьох змінних в вигляді інформативних навантажених дерев



$\tilde{x}_i^{w} \in A_i,\ i = \overline{1,n}$

$\tilde{y}_j^{w} \in B_j,\ j = \overline{1,k}$

Розглянемо

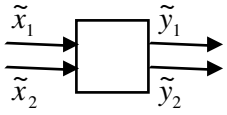
$\varphi_1(\tilde{x}_1^{w},\ldots,\tilde{x}_n^{w}) = \tilde{y}_1^{w}$

…

$\varphi_k(\tilde{x}_1^{w},\ldots,\tilde{x}_n^{w}) = \tilde{y}_k^{w}$

Функція φ_i називається детермінованою, якщо кожний вихідний сигнал $y_i(t)$ на такті t однозначно визначається словами $x_1(1)\ldots x_1(t)\ldots x_n(t)$. Якщо всі $\varphi(t)$ детерміновані, то задане з їх допомогою відображення t вихідних послідовностей називається детермінованою.

Побудуємо інформативне навантажене дерево



$A_1 = A_2 = \{0,1\}$

Випускаємо з кожної вершини починаючи з кореня по 4 ребра:

1е — інформативне 00

2е — інформативне 01

3е — інформативне 10

4е — інформативне 11

За заданим правилом навантажимо дерево мітками.

Розглядаємо всі можливі піддерева, еквівалентні позначаємо станами $q_0, q_1, \ldots$. Якщо

станів скінченне число r, то відображення φ є обмежено-детермінованим, то можна побудувати канонічні системи рівнянь

$x_1(t)$	$x_2(t)$	$q_1(t)$	…	$q_l(t)$	$y_1(t)$	$y_2(t)$	$q_1(t+1)$	…	$q_l(t+1)$
…	…	…	…	…	…	…	…	…	…

$$\begin{cases} y_i(t) = F_i(x_1(t), x_2(t), q_1(t), ..., q_l(t)), \ i = \overline{1,2} \\ q_j(t) = G_j(x_1(t), x_2(t), q_1(t), ..., q_l(t)), \ j = \overline{1,l} \\ q_1(1) = 0, \ \ldots \ , q_l(1) = 0 \end{cases}$$

Нехай $l_1, l_2, ..., l_n$ - кількості символів у відповідних алфавітах $A_1, A_2, ..., A_n$, а $r_1, r_2, ..., r_n$ - кількості символів у відповідних алфавітах $B_1, B_2, ..., B_n$

Позначимо $L = \prod_{i=1}^{n} l_i \quad R = \prod_{i=1}^{k} r_i$

Тоді з довільної вершини, починаючи з кореня, будуть виходити по L інформативних ребер, на яких можуть стояти по R різних груп міток.

Таким чином ми побудуємо інформативно навантажене дерево, що відповідає функціям $\varphi_1, \varphi_2, ..., \varphi_k$, введемо піддерева, що відповідають станам $q_0, q_1, ...$ в випадку, якщо станів скінченне число будемо називати ці функції обмежено-детермінованими і тоді для них можна побудувати канонічні таблиці та канонічні системи рівнянь.

Для того щоб привести таблицю та канонічну систему в векторній формі до скалярної(булевий вигляд) необхідно закодувати двійковими кортежами всі вхідні, вихідні набори та стани

$x_1(t)....x_{n_1}(t)\, q_1(t)...q_l(t)\, y_1(t)...y_{k_1}(t)...q_1(t+1)...q_l(t+1)$

де всі функції F_i та G_j функції всіх виходів та переходів записані у вигляді ДНФ та КНФ

Найпростіші скінченні автомати

Нехай оператор φ обмежено-детермінований, тоді він заданий системою канонічних рівнянь (*), де функції виходів та переходів — всюди визначені функції двійкової логіки.

Нову схему операторів φ будемо зображати в вигляді прямокутника на площині з n_1 входами та k_1 виходами. Вхідні та вихідні канали називаються полюсами.

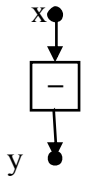
Вважатимемо, що в кожен момент часу t=1,2,... на і-й вхід x_i поступає вхідний символ $x_i(t) \in \{0,1\}$ і в той же такт часу на j-м y_j видається значення

$y_j = F_j(x_1(t), ..., x_i(t), q_1(t), ..., q_l(t))$

Вихідний полюс y_j суттєво залежить від вхідного x_i якщо в системі канонічних рівнянь функція F_j присутня змінна $x_i(t)$, якщо в записі F_j змінна x_i відсутня, то говоритимемо, що полюс y_j залежить з затримкою від x_i.

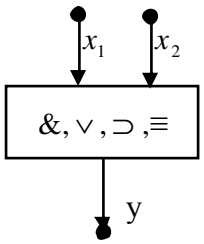
Найпростіші автомати:

1. Заперечення



$$\begin{cases} y(t) = \bar{x}(t), \ t \geq 1 \\ q(t) = q_0(t) = 0 \end{cases}$$

2.



$$\begin{cases} y(t) = x_1(t) \vee x_2(t) \\ q(t) = q_0 = 0, \ t \geq 1 \end{cases}$$

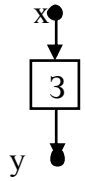
3. Автомат затримки

$$\begin{cases} y(t) = x(t-1) \\ y(1) = 0 \end{cases}$$

Ця функція виконує зсув вхідної послідовності на 1н розряд

x(t)	q(t)	y(t)	q(t+1)
0	0	0	0
1	0	0	1
0	1	1	0
1	1	1	1

$$\begin{cases} y(t) = q(t) \\ q(t+1) = x(t) \\ q(1) = 0 \end{cases}$$



Автомати 1, 2 - вихідний полюс суттєво залежить від вхідних полюсів, а в автоматі 3
вихідний полюс залежить з затримкою від вхідного.

$\psi = \{-, \&, \vee, \supset, \equiv, 3\}$

Візьмемо початковий набір скінчених автоматів φ: $\psi = \{-, \&, \vee, \supset, \equiv, 3\}$

Логічні мережі

Введемо поняття логічної мережі над множиною скінченних автоматів

1) Кожен автомат з множини ψ зображений відповідним чином на площині з написаною системою канонічних рівнянь буде називатися логічною мережею на множині ψ. Системою канонічних рівнянь мережі будемо називати систему канонічних рівнянь даного автомата.
2) Якщо $\sum_1$ і $\sum_2$ логічні мережі на множині ψ, то логічною мережею буде об'єкт, що отриманий з них за допомогою операцій:

O_1 - об'єднання

O_2 - каскаду

O_3 - операція петлі оберненого зв'язку

O_4 - ототожнення вхідних змінних

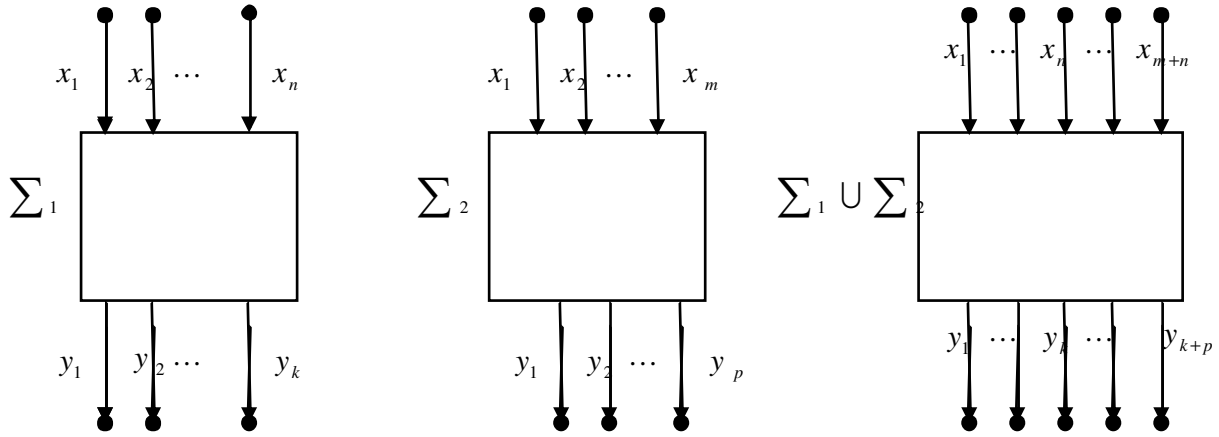
O_5 - ототожнення вихідних змінних

3) Логічною мережею над множеною ψ будуть ті і тільки ті елементи для яких виконується 2 попередніх пункти.

Операції над логічними мережами

1. O_1 - операція об'єднання 2-х мереж

Нехай на площині поданий малюнок, що зображує мережі $\sum_1$ і $\sum_2$ системи канонічних рівнянь. Намалюємо мережі поряд на площині без перетину:



$$\sum_1 : \begin{cases} y_i = F_i^{(1)}(x_1(t)x_2(t)...x_n(t)q_1(t)...q_l(t)), i = \overline{1,k} \\ g_i(t+1) = G_j^{(1)}(x_1(t)x_2(t)...x_n(t)q_1(t)...q_l(t)), j = \overline{1,l} \\ g_i(1) = 0, j = \overline{1,l} \end{cases}$$

$$\sum_2 : \begin{cases} y_i = F_i^{(2)}(x_1(t)x_2(t)...x_m(t)q_1(t)...q_s(t)), i = \overline{1,p} \\ g_i(t+1) = G_j^{(1)}(x_1(t)x_2(t)...x_m(t)q_1(t)...q_s(t)), j = \overline{1,s} \\ g_i(1) = 0, j = \overline{1,l} \end{cases}$$

Для побудови об'єднання даних мереж введемо нумерацію вхідних і вихідних полюсів.

Полюси першої системи залишаємо без змін, а вхідні полюси 2-ї мережі перенумеровуємо $x_1 = x_{n+1}, x_2 = x_{n+2}, \ldots, x_m = x_{n+m}$. Вихідні полюси 2-ї мережі перенумеровуємо $y_1 = y_{k+1}, y_2 = y_{k+2}, \ldots, y_p = y_{k+p}$.

І перепишемо канонічне рівняння другої мережі в нових позначеннях.

$$\begin{cases} y_i = F_i^{(1)}(x_1(t)x_2(t)...x_n(t)q_1(t)...q_l(t)), i = \overline{1,k} \\ g_i(t+1) = G_j^{(1)}(x_1(t)x_2(t)...x_n(t)q_1(t)...q_l(t)), j = \overline{1,l} \\ g_i(1) = 0, j = \overline{1,l+s} \\ y_i = F_i^{(2)}(x_{n+1}(t)x_{n+2}(t)...x_{n+m}(t)q_{l+1}(t)...q_{l+s}(t)), i = \overline{k+1,k+p} \\ g_i(t+1) = G_j^{(1)}(x_{n+1}(t)x_{n+2}(t)...x_{n+m}(t)q_{l+1}(t)...q_{l+s}(t)), j = \overline{l+1,l+s} \end{cases}$$

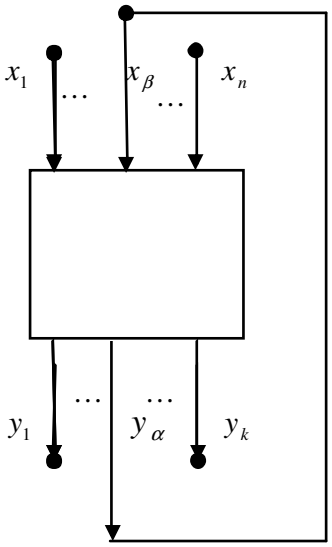
2. O_2 - каскад. Нехай на площині заданий малюнок об'єднання двох мереж і їх системи канонічних рівнянь. Ототожнимо деякий вихідний полюс y_α з вихідним полюсом мережі x_β.

Для цього всі функції виходів і переходів перепишемо наступним чином. Із системи (*) видаляємо рівняння з номером α $y_\alpha = F_\alpha^{(1)}(x_1(t)x_2(t)...x_n(t)q_1(t)...q_l(t))$ і в усіх рівняннях замість $x_\beta(t)$ підставляємо $F_\alpha^{(1)}(x_1(t)x_2(t)...x_n(t)q_1(t)...q_l(t))$.

Отримуємо нову мережу, що називають каскадом або видаленням деякої вихідної змінної. Якщо k=1, то отримуємо автомат без вихода.

3. O_3 - петля оберненого зв'язку.

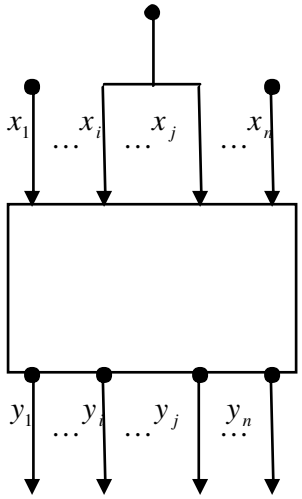
Нехай задана мережа Σ. Виділимо вхідний сигнал x_β і вихідний сигнал y_α.



Операцію можна застосувати тільки у випадку, коли вихідний полюс y_α залежить з запізненням від вхідного x_β. Якщо операцію можна застосувати, то вихідний полюс y_α ототожнює з вихідним x_β. Для цього з системи канонічних рівнянь видаляємо одне рівняння $y_\alpha = F_\alpha^{(1)}(x_1(t)x_2(t)...x_n(t)q_1(t)...q_l(t))$, а в усі інші рівняння входів і виходів замість x_β вставляємо y_α.

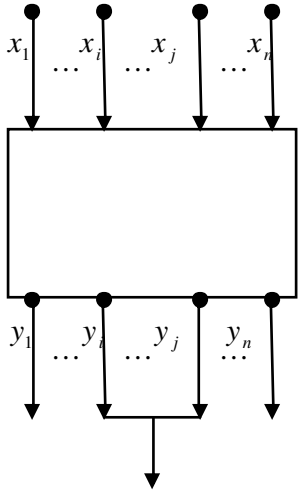
Якщо n=1, то автомат називається без входа.

4. O_4 (Ототожнення вхідних змінних)



Ототожнені полюси x_i і x_j розглядаються, як один вхідний полюс нової схеми. Для цього в усіх рівняннях системи замінити x_j і x_i.

5. O_5 (Ототожнення вихідних змінних)



Якщо є два вихідних полюси y_i і y_j то їх ототожнення полягає у тому, що функцію $F_j^{(1)}$ змінюємо на F_i^1.

Теорема повноти

Набір скінченних автоматів називається повним, якщо для довільної обмежено-детермінованої функції φ, що описана системою канонічних рівнянь в скалярній формі (*) , де функції виходів і переходів записані в ДНФ і КНФ існує така логічна мережа над даним набором скінченних автоматів системи канонічних рівнянь, яка еквівалентна системі (*).

Теорема.

Існують повні набори скінченних автоматів, наприклад $\psi_0 = \{-,\&,\vee,3\}$.

Доведення. Припустимо, що існує система канонічних рівнянь обмежено-детермінованих функцій φ, записана у вигляді (*)

$$\begin{cases} y_i = F_i(x_1(t)x_2(t)...x_n(t)q_1(t)...q_l(t)), i = \overline{1,k} \\ g_j(t+1) = G_j(x_1(t)x_2(t)...x_n(t)q_1(t)..q_l(t)), j = \overline{1,l} \\ g_j(1) = 0, j = \overline{1,l} \end{cases}$$

Будемо вважати, що F_i та G_j представлені у вигляді ДНФ.

$y_i = D_1 \vee D_2 \vee ... \vee D_s$

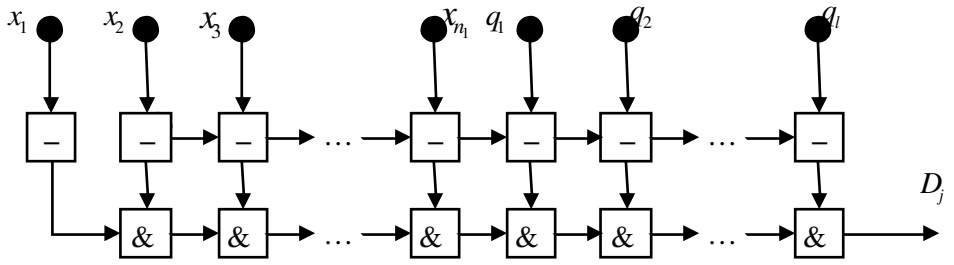
$q_j = D_{s+1} \vee D_{s+2} \vee ... \vee D_f$

Побудуємо логічну мережу, що реалізує елементарну кон'юнкцію D_j

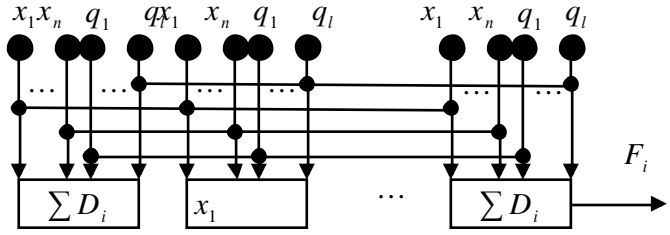
$\sigma = \{0,1\}$

$D_j = x_1^{\sigma_1}(t) \wedge x_1^{\sigma_2}(t) \wedge ... \wedge x_1^{\sigma_n}(t) \wedge x_1^{\sigma_{n+1}}(t) \wedge ... \wedge x_1^{\sigma_{n+l}}(t)$

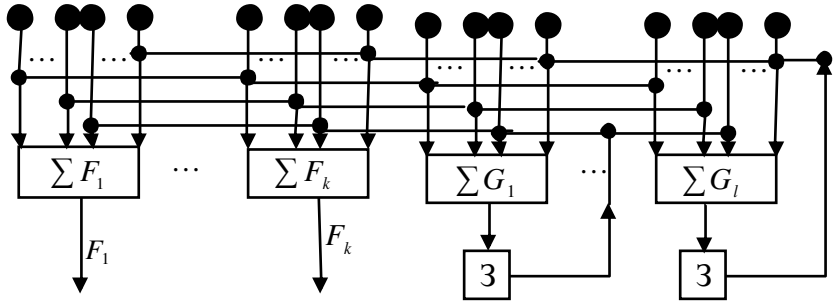
$$x^{\sigma} = \begin{cases} x, & якщо\ \sigma = 1 \\ \bar{x}, & якщо\ \sigma = 0 \end{cases}$$



Якщо змінна в кон'юнкції без заперечення, то подамо вхідний канал одразу на кон'юнкцію. Аналогічно побудуємо логічні мережі $D_1, ..., D_f$. Тепер побудуємо логічні мережі, що реалізують F_i та G_j



Таким чином побудуємо мережі, що реалізують всі функції виходів та переходів, кожна з них має входів та 1н вихід



Отримаємо логічну мережу, що реалізує систему рівнянь (*)

Проблема повноти для скінчених автоматів:

Чи існує скінчений алгоритм, що дозволяє для довільного набору скінчених автоматів визначити повний він чи ні?

Ця задача алгоритмічно невирішена. Не існує скінченого числа дій, які для довільного набору можуть визначити полон він чи ні.