2 Теория на изчислимостта

2.1 Функции изчислими интутивно и тезис на Чърч

Ефект

Най-сигурната рецепта да объркате един неинформатик: Ожесточен дебат: кой е най-добрият език за прогармиране

- Всички програмни езици и машинни модели са "'еднакво "' мощни
- Във всеки модел има едни и същи не изчислими проблеми

2.2 Интуитивно понятие за изчислимост

$f : \mathbb{N}^k \to \mathbb{N}$ (частична) функция.

f е изчислима, ако
$\exists$ ефективна процедура (=алгоритъм), която изчислява f.

Ефективна процедура = Java-програма (, ..., “подходящ ” програмен език)
Вход: $(x_1, \ldots, x_k) \in \mathbb{N}^k$
Изход: $f(x_1, \ldots, x_k)$

Програмата завършва за краен брой стъпки, ако
$(x_1, \ldots, x_k) \in$ дефиниционната област на f
и не завършва иначе.

Пример

input n

repeat

until false

изчислява никъде недефинираната функция Ω

Пример

$$f_\pi(n) = \begin{cases} 1 & \text{ако } n \text{ е начален сегмент в десет. представяне на } \pi. \\ 0 & \text{иначе} \end{cases}$$

f_π е изчислима:

Някои апроксимации на π

Архимед: Апроксимации с правилни многоъгълници.

Древните индийци: 1682 преоткрито от Лайбниц

$$\pi = 4\sum_{i=0}^{\infty}\frac{-1^i}{2i+1} = 1-\frac{1}{3}+\frac{1}{5}-\cdots$$

Baile-Borwein-Plouffe 1996:

$$\pi = \sum_{i=0}^{\infty}\frac{1}{16^i}\left(\frac{4}{8i-1}-\frac{2}{8i+4}-\frac{1}{8i+5}-\frac{1}{8i+6}\right)$$

Пример

$$f(n) = \begin{cases} 1 & \text{ако } n \text{ участва някъде в десет. представяне на } \pi. \\ 0 & \text{иначе} \end{cases}$$

Не е известно дали f е изчислима!

Не е известно дали $\forall n : f(n) = 1$ (“нормалност” на π).

Пример

$$f(n) = \begin{cases} 1 & \text{ако } n \times \text{‘7’ участват някъде в десет. предст. на } \pi. \\ 0 & \text{иначе} \end{cases}$$

Дали f е изчислима ?

Пример

$$f(n) = \begin{cases} 1 & \text{ако } n \times \text{‘7’ участват някъде в десет. предст. на } \pi. \\ 0 & \text{иначе} \end{cases}$$

f е изчислима

Ако $\forall n : n \times$ ‘7’ се среща: $\forall n : f(n) = 1$

Ако 7 се среща максимум n_0 пъти:

$$\longrightarrow f(n) = \begin{cases} 1 & \text{ако } n \leq n_0 \\ 0 & \text{иначе} \end{cases}$$

Очевдно е изчислима, защото е константа, за всички n, с изключение на кр. брой!

Пример

LBA: линейно ограничени мащини на Тюринг

DLBA: детерминистични линейно ограничени мащини на Тюринг

$$i(n) = \begin{cases} 1 & \text{if } \forall LBA\ M \exists DLBA\ D : L(M) = L(D) \\ 0 & \text{иначе} \end{cases}$$

Ние не познаваме функцията $i(n)$.

Но, $i(n)$ е константна функция и следователно изчислима.

Неизчислими функции

Нека $r \in \mathbb{R}$ е произволно.

$$f_r(n) = \begin{cases} 1 & \text{ако } n \text{ е начален сегмент в десет. предст. на } r. \\ 0 & \text{иначе} \end{cases}$$

Да допуснем: $\forall r : f_r$ е изчислима.

$\longrightarrow \exists$ съществуват най-малко толкова изчислими функции, колкото са реалните числа.

Противоречие:

- Множеството от всички изчислими функции е изброимо- програмите са изброимо много-крайни низове
- $\mathbb{R}$ не е изброимо.

Неизчислими функции

- Има неизчислими функции
- Множеството от всички изчислими функции е изброимо
  (тъй като програмите, които ги изчисляват са изброимо много-крайни низове от краен брой символи).
- Всички функции са неизброимо много
- Но може ли да посичим конкретна неизчислима функция?

Тезис на Чърч-Тюринг

Изчислимите функции с машини на Тюринг са точно изчислимите в интуитивен смисъл. Не е тероема (не можем да го докажем), но всички го приемат.

Обосновка

- Всички познати изчислителни модели са еквивалентни.

  това можем да го докажем

- Всички известни "‘интуитивно"’ изчислими функции са изчислими по Тюринг.

Изчислими функции с машини на Тюринг

$T = (Q, \Sigma, \Gamma, \delta, s, F)$ изчислява частичната функция
$f_T : \Sigma^* \to \Gamma^* \Leftrightarrow$

$$f_T(w) := \begin{cases} v & \text{ако } T \text{ завършва при вход } w \\ & \text{с изход } v \\ & ((s)w \Rightarrow u(q)v), q \in F \\ \perp = (\text{не е дефинирана }) & \text{иначе} \end{cases}$$

g е изчислима по Тюринг $\Leftrightarrow \exists T : f_T = g$

Забележка: ако $g(x) = \perp$, T не завършва.

Разрешими езици

Един език $L \subseteq \Sigma^*$ е разрешим $\iff$ ако характеристичната функция χ_L е изчислима Тюринг.

$$\chi_L : \Sigma^* \to \{0,1\} \quad \text{където} \quad \chi_L(w) = \begin{cases} 1 & \text{ако } w \in L \\ 0 & \text{иначе} \end{cases}$$

Например: $\{0^n 1^n : n \geq 0\}$ е разрешим.

Лема: Един език $L \subseteq \Sigma^*$ е разрешим, ако има машина на Тюринг T, която завършва винаги, и:

- $\forall w(w \in L \Rightarrow (s)w \vdash^* x(f)y)$ за някое $f \in F$
- $\forall w(w \notin L \Rightarrow (s)w \vdash^* x(q)y)$ за някое $q \notin F$ (Error)

Полуразрешими езици

Един език $L \subseteq \Sigma^*$ е полуразрешим $\iff$ ако полухарактеристичната му функция χ'_L е изчислима Тюринг, където

$$\chi'_L(w) = \begin{cases} 1 & \text{ако } w \in L \\ \perp & \text{иначе} \end{cases}$$

Лема: Всеки разрешим език е полуразрешим.

Изчислими с машини на Тюринг

функции в естествените числа

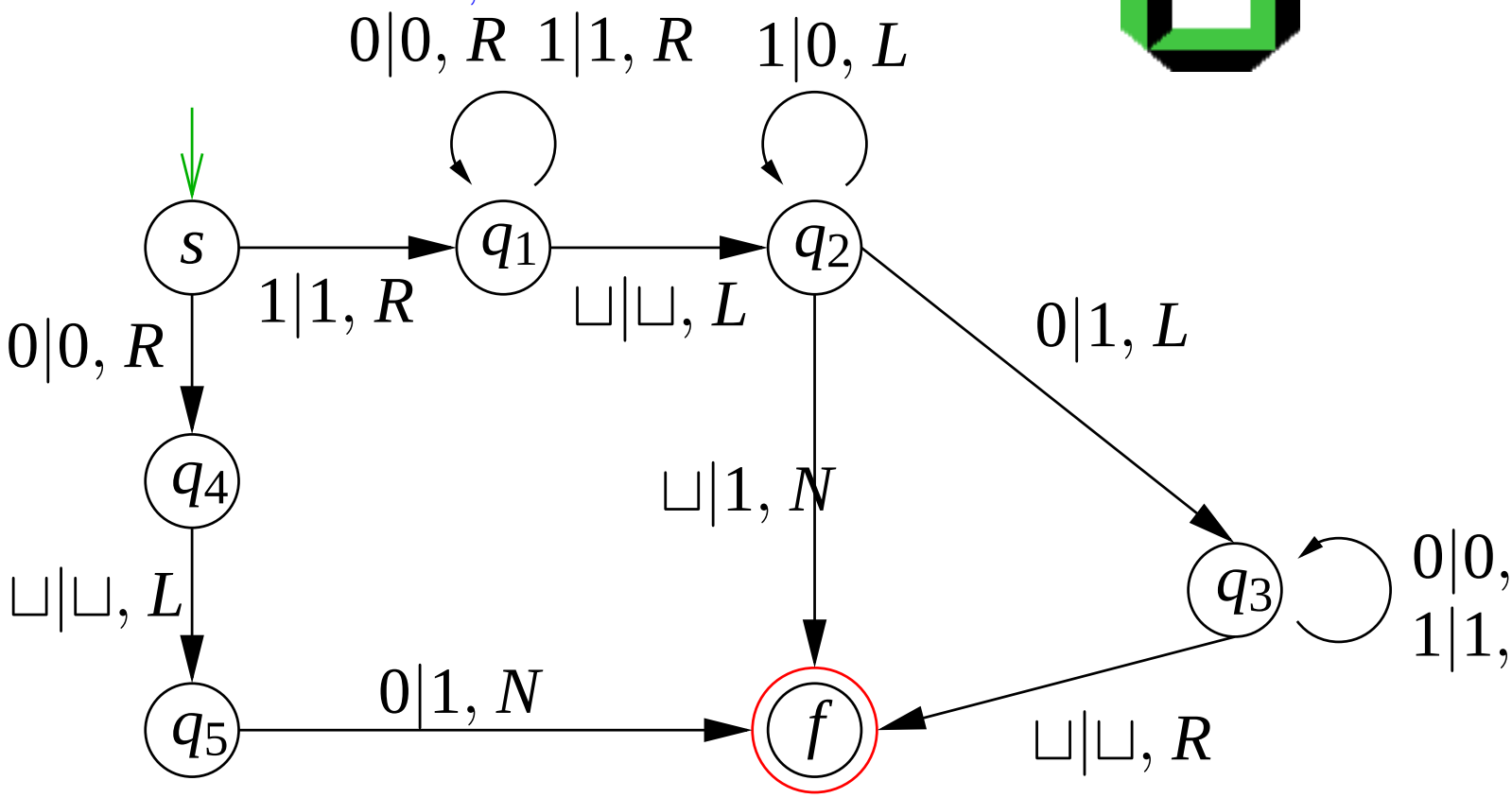
$f : \mathbb{N}^k \to \mathbb{N}$ е изчислима по Тюринг $\Leftrightarrow$

$\exists T = (Q, \Sigma, \Gamma, \delta, s, F) : \forall n_1, \ldots, n_k, m \in \mathbb{N} :$

$f(n_1, \ldots, n_k) = m \Leftrightarrow$

$(s)\text{bin}(n_1)\#\cdots\#\text{bin}(n_k) \vdash^* u(q)\text{bin}(m), q \in F$

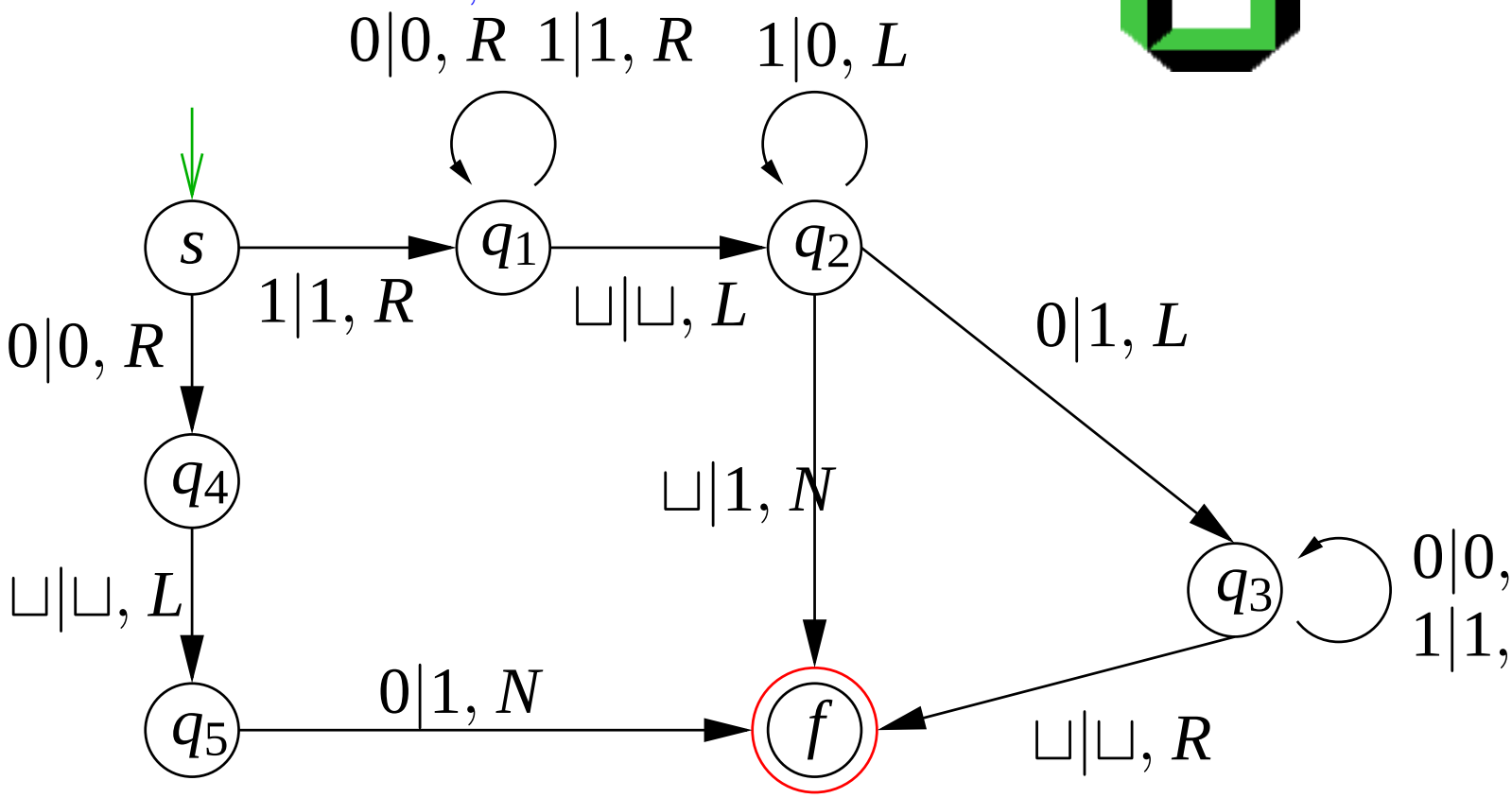
Пример



$$f(w) = \begin{cases} w+1 & \text{ако } w \in 0 \cup 1(0 \cup 1)^*, \\ & w \text{ като двоично число} \\ \text{недефинирана} & \text{иначе} \end{cases}$$

Забележка: Непоказаните дъги са безкрайни цикли.

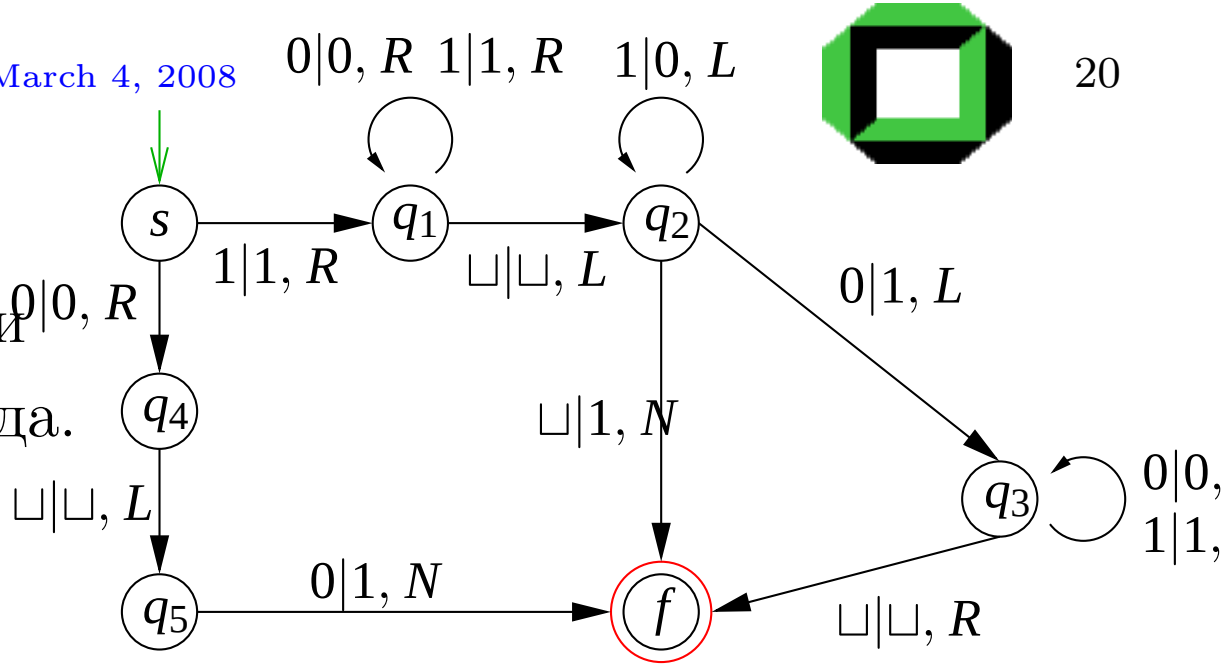
Пример



$(s)11 \vdash 1(q_1)1 \vdash 11(q_1) \vdash 1(q_2)1 \vdash (q_2)10 \vdash (q_2)\sqcup 00 \vdash (f)100$

Разглеждане на случаи по структурата на входа.

Нека $w \in \{0,1\}^*$, $a \in \{0,1\}$, $n \geq 1$.



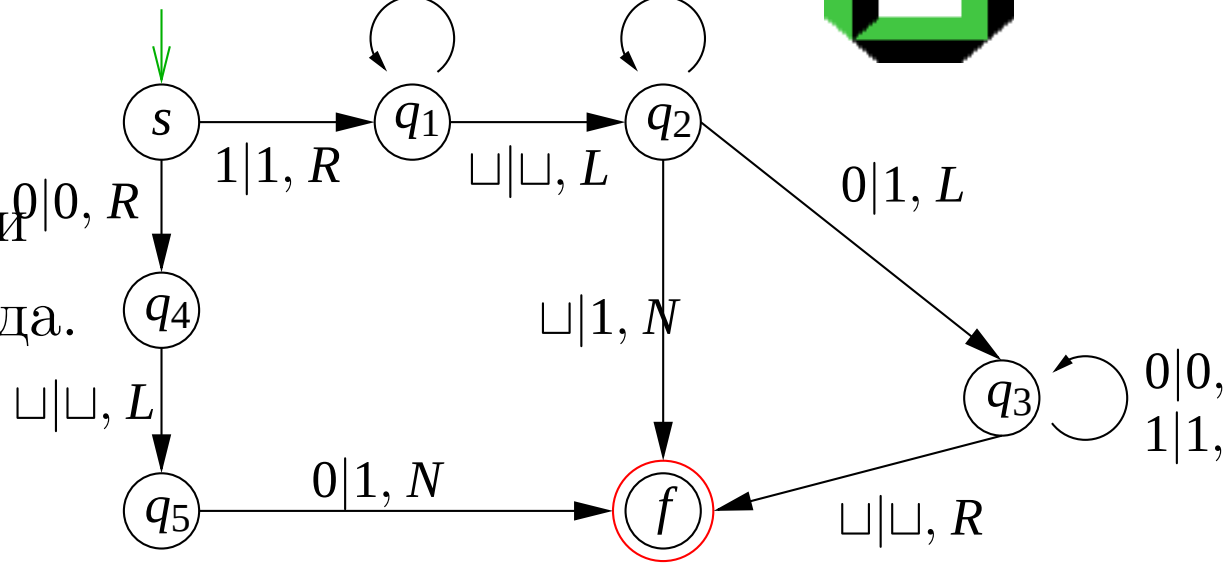
0: $(s)0 \vdash 0(q_4)\vdash (q_5)0 \vdash (f)1$

$0aw$: $(s)0aw \vdash 0(q_4)aw$ не е дефинирана

Разглеждане на случаи по структурата на входа.

Нека $w \in \{0,1\}^*$

$a \in \{0,1\}, n \geq 1$.



1^n: $(s)1^n \vdash 1(q_1)1^{n-1} \overset{n-1}{\vdash} 1^n(q_1) \vdash 1^{n-1}(q_2)1 \overset{n}{\vdash} (q_2)\sqcup 0^n \vdash (f)10^n$

$1w0$: $(s)1w0 \vdash 1(q_1)w0 \overset{|w|+1}{\vdash} 1w0(q_1) \vdash 1w(q_2)0 \vdash 1w(q_3)1 \overset{|w|+1}{\vdash}$
$(q_3)\sqcup 1w1 \vdash (f)1w1$

$1w01^n$:

$(s)1w01^n \vdash 1(q_1)w01^n \overset{|w|+1+n}{\vdash} 1w01^n(q_1) \vdash 1w01^{n-1}(q_2)1 \overset{n}{\vdash}$
$1w(q_2)00^n \vdash 1w(q_3)10^n \overset{|w|+1}{\vdash} (q_3)\sqcup 1w10^n \vdash (f)1w10^n$

Пример:Никъде недефинираната функция

$$T = (\{s\}, \Sigma, \Gamma, \delta, s, \{\})$$

$$\forall a \in \Gamma : \delta(s, a) = (s, a, R)$$

Програмни техники и конструкции на машини на Тюринг

- Запомняне на символ
- Основни машини
- Комбиниране на машини - композиция и разклонение
- While-loops

Запомняне на символ (локални променливи)

Нека $x \in A \subseteq \Gamma$, ($|A| < \infty$!):
$Q \rightsquigarrow Q \times A$.

Пример M е МТ, която запаметява първия символ на входната дума и завършва успешно, ако той не се среща на друго място в думата.

$$\delta([s,\sqcup],0) = ([q,0],0,R) \quad \delta([s,\sqcup],1) = ([q,1],1,R)$$

$$\delta([q,0],1) = ([q,0],1,R) \quad \delta([q,1],0) = ([q,1],0,R)$$

$$\delta([q,0],\sqcup) = ([f,\sqcup,N) \quad \delta([q,1],\sqcup) = (f,\sqcup,N)$$

Кмпозиция и разклонение

Основни машини (за 1 стъпка): за всяко $b \in \Gamma$:

$$a : \delta(s,b) = (f,a,N)$$

$$L : \delta(s,b) = (f,a,L)$$

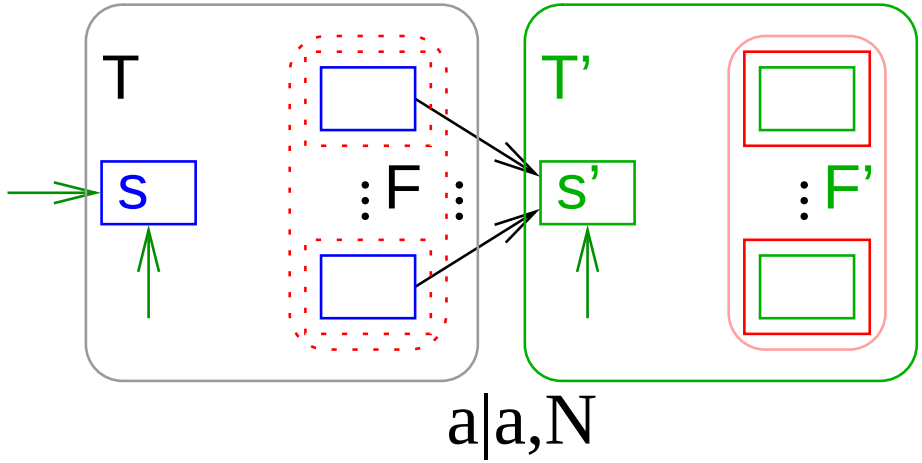
$$R : \delta(s,b) = (f,a,R)$$

Композиция

Дадено: $T = (Q,\Sigma,\Gamma,\delta,s,F)$ и $T' = (Q',\Sigma,\Gamma',\delta',s',F')$.

Изход: Машина на Тюринг $T^\circ = (Q^\circ,\Sigma,\Gamma^\circ,\delta^\circ,s,F')$ за $f_{T'}(f_T(x))$: $Q^\circ = Q \dot\cup Q'$ $\Gamma^\circ = \Gamma \cup \Gamma'$

$$\delta^{\circ}(q,a) = \begin{cases} \delta(q,a) & \text{ако } q \in Q \setminus F \\ (s',a,N) & \text{ако } q \in F \\ \delta'(q,a) & \text{ако } q \in Q' \end{cases}$$



Разклонение

Дадено: $T = (Q, \Sigma, \Gamma, \delta, s, F)$, $T' = (Q', \Sigma, \Gamma', \delta', s', F')$ и $T'' = (Q'', \Sigma, \Gamma'', \delta'', s'', F'')$.

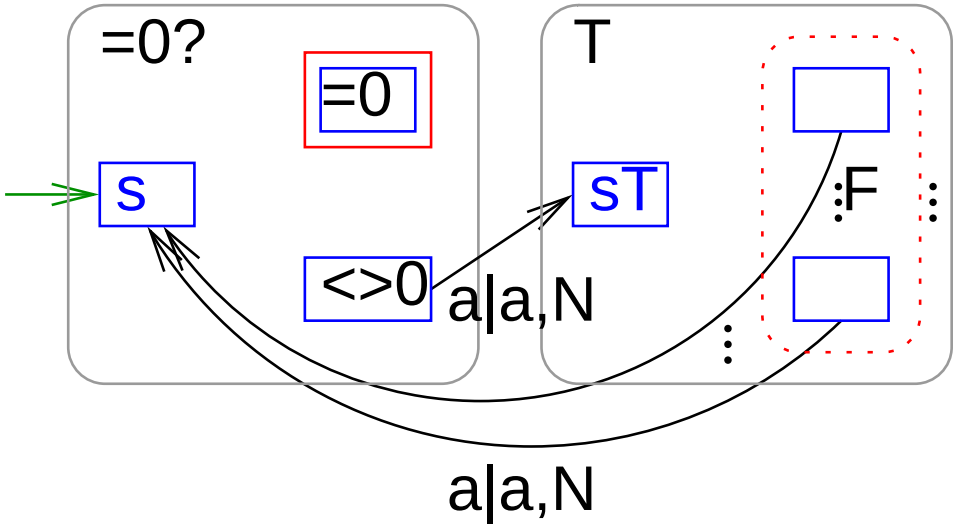
Изход: Машина на Тюринг $T^\circ = (Q^\circ, \Sigma, \Gamma^\circ, \delta^\circ, s, F')$

$Q^\circ = Q \dot\cup Q' \dot\cup Q''$, $\Gamma^\circ = \Gamma \cup \Gamma' \cup \Gamma''$

$$f_{T^\circ}(x) = \begin{cases} f_{T'}(f_T(x)) & \text{ако } f_T(x) = a \\ f_{T''}(f_T(x)) & \text{ако } \downarrow f_T(x) \neq a \end{cases}.$$

$$\delta^\circ(q,b) = \begin{cases} \delta(q,b) & \text{ако } q \in Q \setminus F \\ (s', b, N) & \text{ако } q \in F \ \& \ b = a \\ (s'', b, N) & \text{ако } q \in F \ \& \ b \neq a \\ \delta'(q,b) & \text{ако } q \in Q' \\ \delta''(q,b) & \text{ако } q \in Q'' \end{cases}$$

While-loops: While $i \neq 0$ Do tape:= f_T(tape)



Примери: $R_{\sqcup}$ - сканира надясно докато намери $\sqcup$

(аналогично $L_{\sqcup}$)

Copy: $(q)w\sqcup \vdash (f)w\sqcup w$ - копира думата

Shift R: $(q)\sqcup w\sqcup \vdash (f)w\sqcup w$ - премества думата надясно

Многолентови машини на Тюринг

k- лентова машина на Тюринг (k- глави):

$T = (Q, \Sigma, \Gamma, \delta, s, F)$, където

$\delta : Q \times \Gamma^k \rightarrowtail Q \times \Gamma^k, \{L, R, N\}^k$.

$\delta(q, (a_1, \ldots, a_k)) = (p, (b_1, \ldots, b_k), (C_1, \ldots, C_k))$,

$C_1, \ldots, C_k \in \{L, R, N\}$.

Теорема За всяка k-лентова машина на Тюринг M съществува еднолентова машина на Тюринг M', такава че за всяка дума x на първата лента $M(x)$ завършва с резултат y на първата лента $\Longleftrightarrow$ M' завършва над x с резултат y.

Идея

$\Sigma' = \Sigma \cup (\Sigma \times \{0,1\})^k$.

На i-та позиция единица показва главата на i-тата лента.

- Фаза 1. Замества всеки символ a_i с $(a_i, 0, \sqcup, 0, \cdots \sqcup, 0)$. Отпред $(\sqcup, 1, \sqcup, 1, \cdots \sqcup, 1)$.
- Фаза 2. Симулира изчислението на M върху M' докато M завърши
  (а) сканиране надясно : запомня къде са главите
  (б) сканиране наляво и надясно за симулиране на M
- Фаза 3. Преобразува всеки k-местен символ в унарен символ: $(b_1, 0, \ldots b_k, 0) \rightsquigarrow b_1$. Игнорира другите ленти без първата.

Варианти на машини на Тюринг

k глави: $\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R, N\}^k$

k ленти: т.е. по една глава за лента

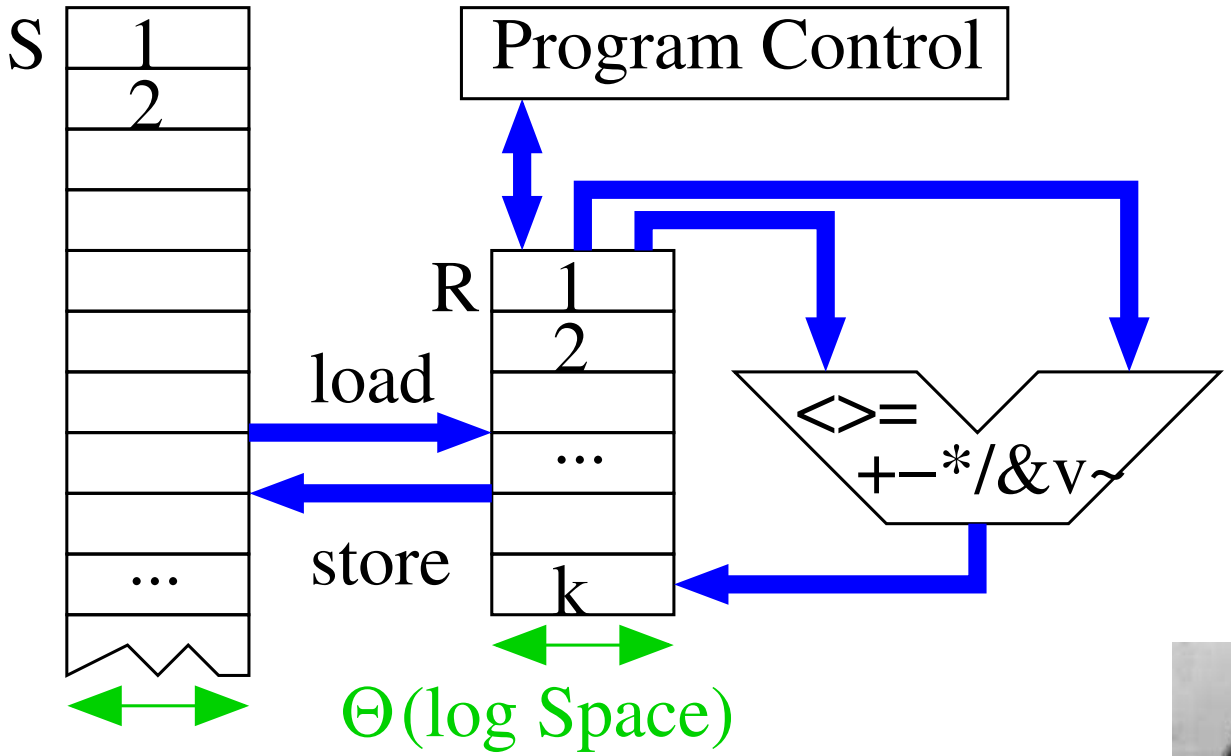
d-размерна лента: например $d = 2$,

$$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, U, D, N\}$$

вероятностни: допълнително инструкции за движение на главата с рандом бит

Времето T за k-лентова-МТ $\to \mathcal{O}\left(T^2\right)$ за едно-лентова-МТ

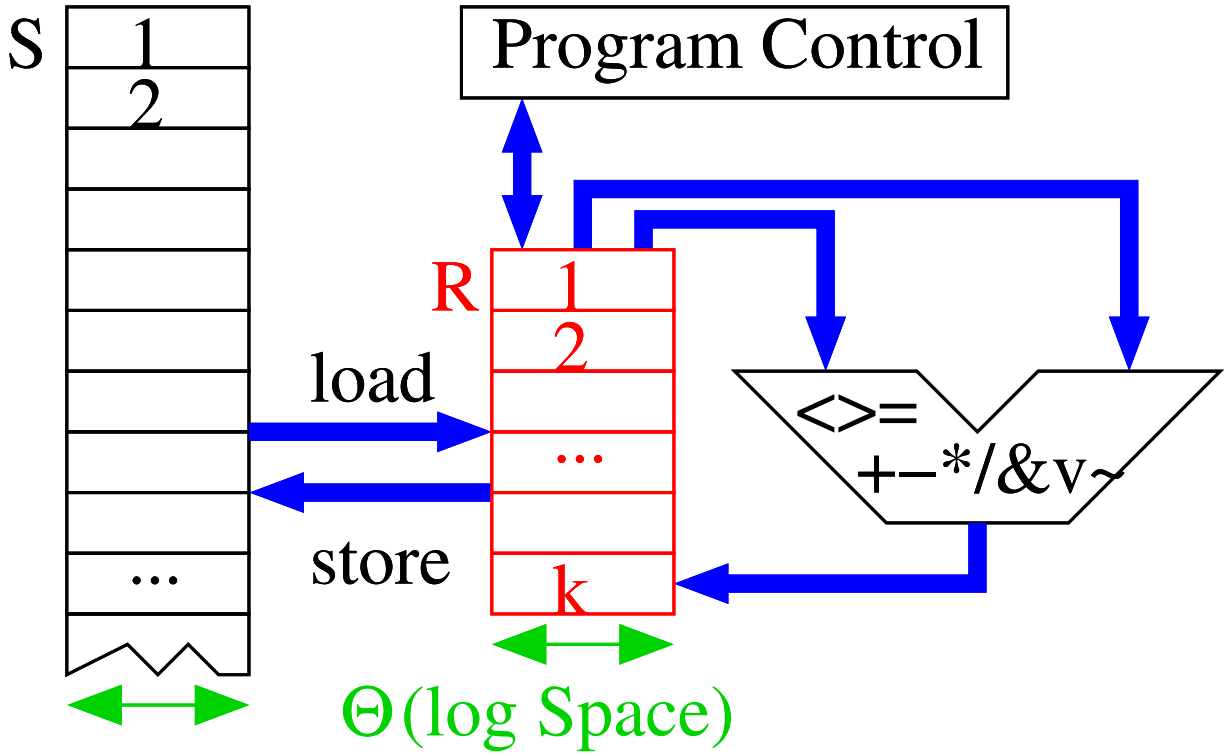
RAM: Машини с директен достъп до паметта (random access memory)



Модерна (RISC)адаптация

на фон Ноймановия модел [1945]

Регистри

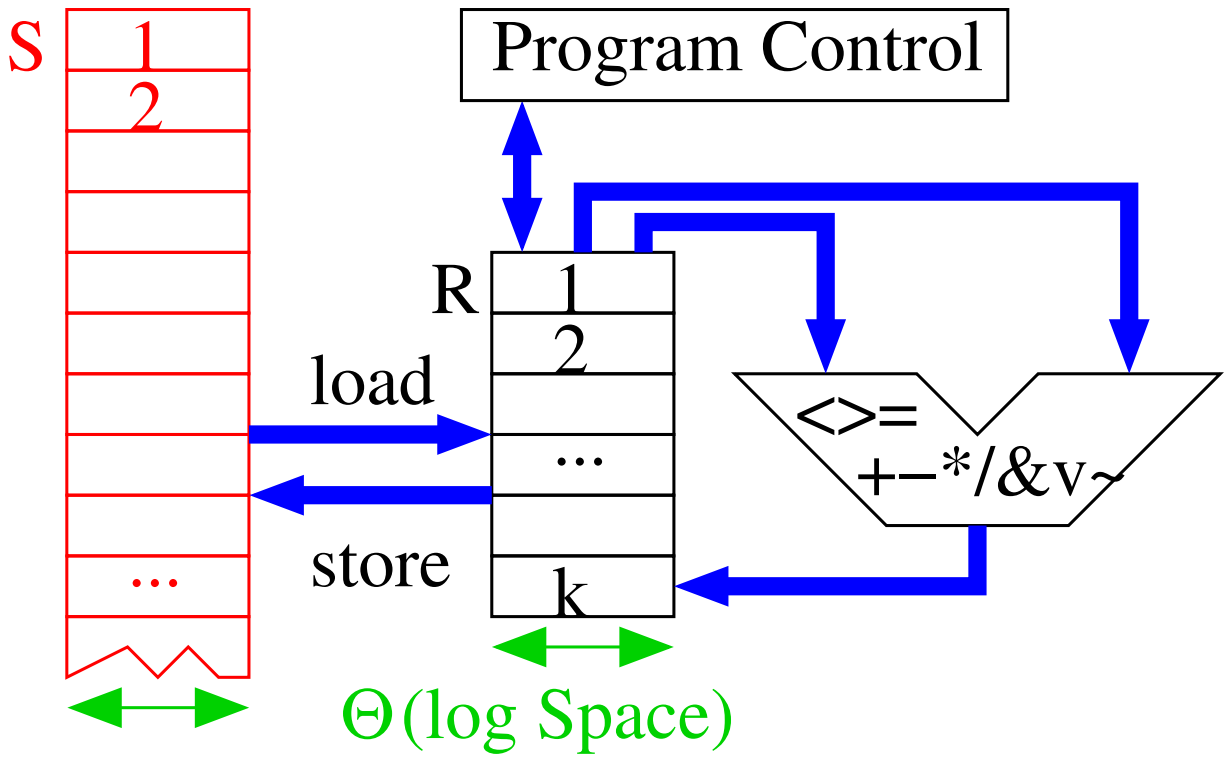


k (всяка константа) регистри

$R_1,\ldots,R_k$ за

(малки) числа

Основната памет

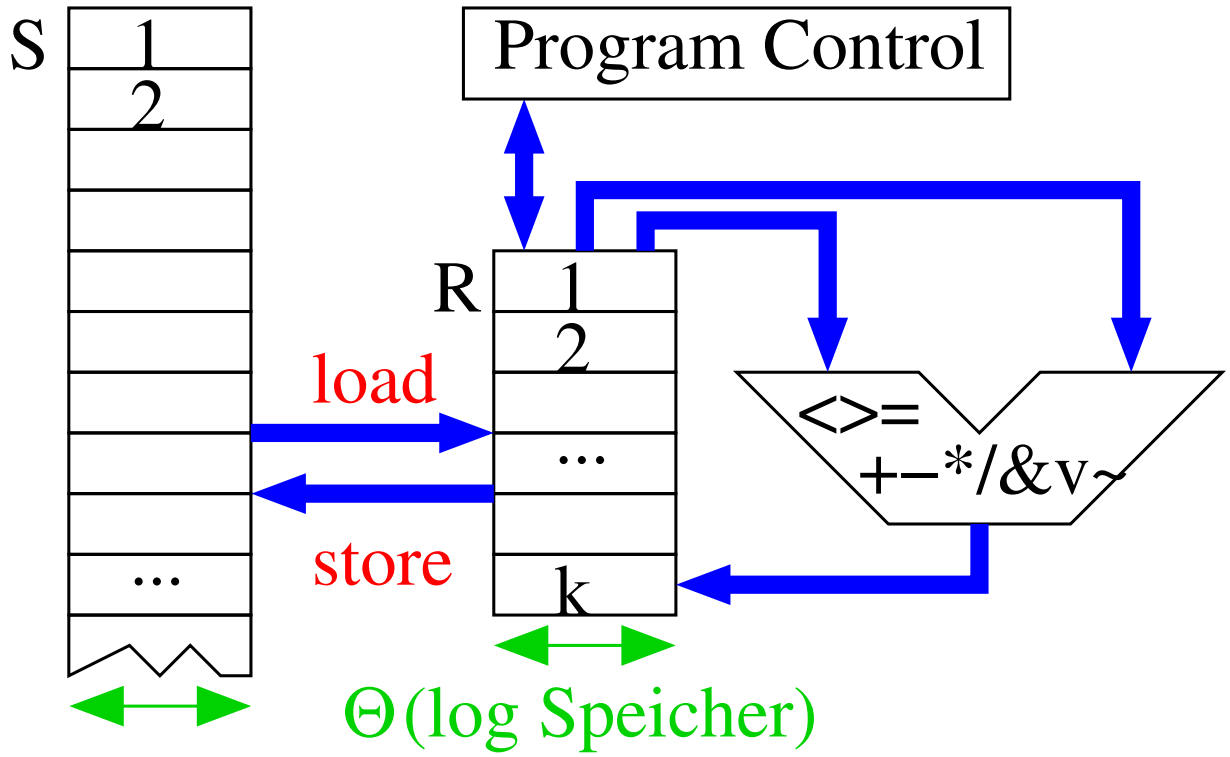


Неограничен брой клетки

$S[1]$, $S[2]\ldots$ за

(малки) числа

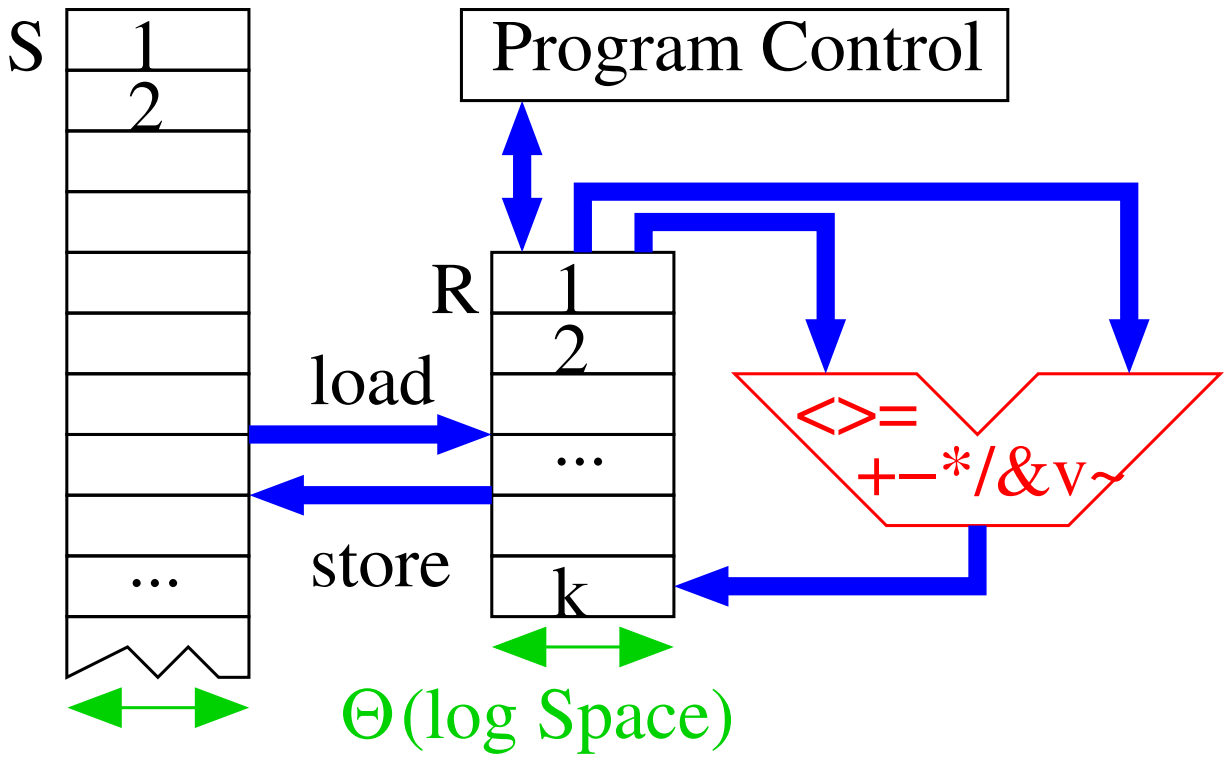
Достъп до паметта



$R_i := S[R_j]$ зарежда съдържанието на клетката от паметта $S[R_j]$ в регистър R_i.

$S[R_j] := R_i$ запомня регистъра R_i в клетката $S[R_j]$.

Изчисление

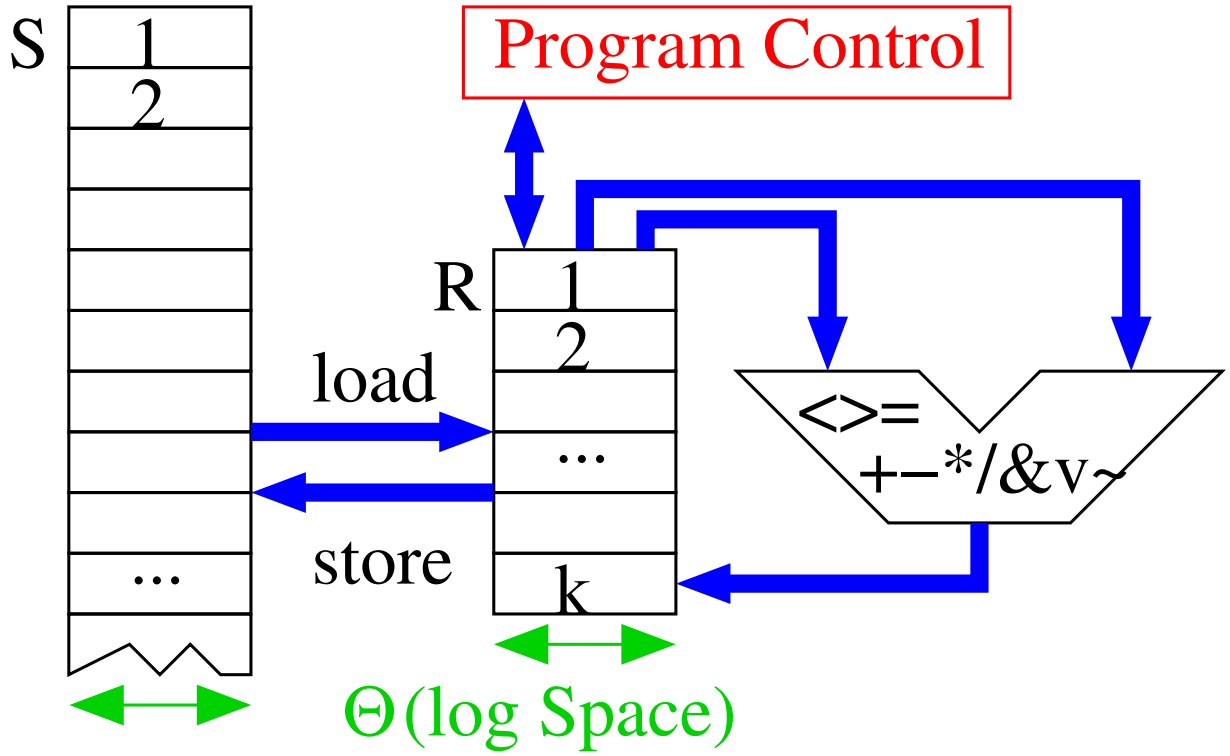


$R_i := R_j \odot R_\ell$ Регистрова аритметика.

‘$\odot$’ означава голямо количество от операции

аритметични, сравнения, логически

Условен преход



JZ j, R_i продължава изпълнението на прогармата с команда с етикет j (goto j) ако $R_i = 0$

"'Малки"' цели числа?

Алтернативи:

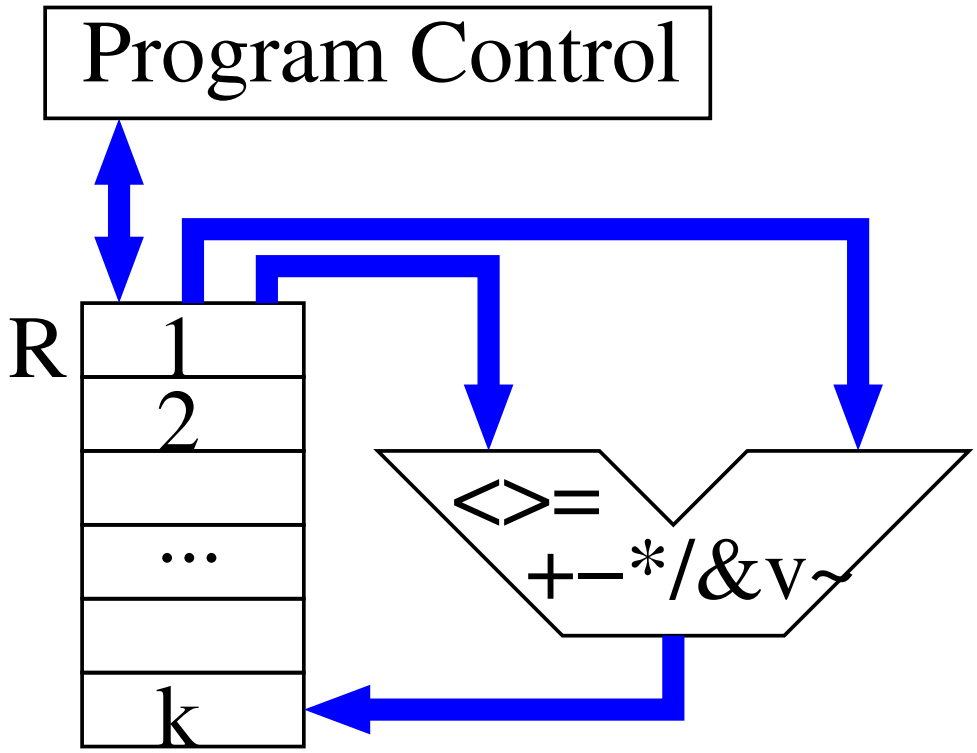
Краен брой битове (64?): теоретически незадоволително, защото се адресира само крайна памет $\rightsquigarrow$ краен автомат

Пройзволен брой: твърде оптимистично

Достатъчно, за да се адресират всички клетки от паметта: Най-добрият компромис.

Машини с Неограничени Регистри (МНР)

$\approx$ RAM $-$ памет $+$ произволно големи числа $+$ произволен (краен) брой (Шефердсон, Стържиц, 1963)



МНР-изчислимост

Конфигурация: $(q, R_1, \ldots, R_k)$

q е брояч на програмните команди

"'$\vdash^*$"' дефинирахме.

$f : \mathbb{N}^{k'} \to \mathbb{N}$, $k' \leq k$ е МНР изчислима $\Leftrightarrow$

$\exists$МНР $M : \forall n_1, \ldots, n_{k'}, m \in \mathbb{N}$:

$$f(n_1, \ldots, n_{k'}) = m \Leftrightarrow$$

$$(1, n_1, \ldots, n_{k'}, 0^{k-k'}) \vdash^* (q, f(n_1, \ldots, n_k), \ldots)$$

with PROGRAM$[q]$ =HALT

RAM-изчислимост

Конфигурация: $(q, R_1, \ldots, R_k, S)$

Нека M е RAM:

вход: $w \in \Sigma^n$ in $S[1],\ldots,S[n]$

изход: $f_M(w)$ in $S[1],\ldots,S[|f_M(w)|]$

когато HALT-командата се изпълни.

Рзаглеждаме естествени числа и трябва да ги кодираме!

Аналогично като MT

Езици за програмиране от високо ниво

Java, C/C++, Pascal,...

ML, Lisp,...

Prolog, Oz,...

...

са най-популярните програмни модели за нас.

Компилаторите транслират програмата в RAM-код.

LOOP-Програми

Най-малкият програмен език за теория на изчислимостта:

$\mathbb{N}$ main($\mathbb{N}x_1, \ldots, \mathbb{N}x_k$){

$\mathbb{N}\ x_0 = 0$; $\mathbb{N}\ x_{k+1} = 0$; $\mathbb{N}\ x_{k+2} = 0$; ...

тяло;

return x_0;

}

В тялото е разрешено да използваме:

Присвояване: $x_i := x_j + c$, $c \in \{-1, 0, 1\}$ $\qquad 0 - 1 := 0$

';': Редица от инструкции

loop x_i: Loop. Изпълни x_i пъти. Началното съдържание на x_i е в сила преди първото изпълнение .

LOOP-Програма

Наблюдение: Loop-програмата винаги звършва.

Дефиниция: f е Loop-изчислима, ако
$\exists$ Loop-програма P, която изчислява f.

Въпрос: кои функции са Loop-изчислими?

Има ли изчислими функции, които не са Loop-изчислими?

Loop-Програми.

$x := x + c$ // $c \in \mathbb{Z}$

$x := y + z$

$x := y \dot{-} z$ // $y \dot{-} z = y - z$ ако $y \geq z$, 0 иначе

$x := y \cdot z$

$x := y \text{ div } z$

$x := y \bmod z$

произволен аритметичен израз

if $x \neq 0$ then ...

Пример: Събиране $x_0 := x_1 + x_2$

$x_0 := x_1$

loop x_2

    $x_0{+}{+}$

Пример: Умножение $x_0 := x_1 \cdot x_2$

loop x_1

$x_0 := x_0 + x_2$

Пример: if $x = 0$ then A

```
y:= 1
loop x
      y−−
loop y
      A
```

While-program

Минимален програмен език за теория на изчислимостта:

$\mathbb{N}$ main($\mathbb{N}x_1, \ldots, \mathbb{N}x_k$){

$\mathbb{N}\ x_0 = 0;\ \mathbb{N}\ x_{k+1} = 0;\ \mathbb{N}\ x_{k+2} = 0;\ \ldots$

тяло;

return x_0;

}

В тялото са разрешени следните оператори:

Присвояване: $x_i := x_j + c,\ c \in \{-1, 0, 1\}$ $\qquad 0 - 1 := 0$

';': Редица от оператори

while($x_i \neq 0$): цикли

WHILE симулира LOOP

loop x do

    P

$\rightsquigarrow$

$y := x$ // y не участва в P

while $y \neq 0$ do

    $y := y - 1$

    P

Може ли While да се симулира с помощта на LOOP?

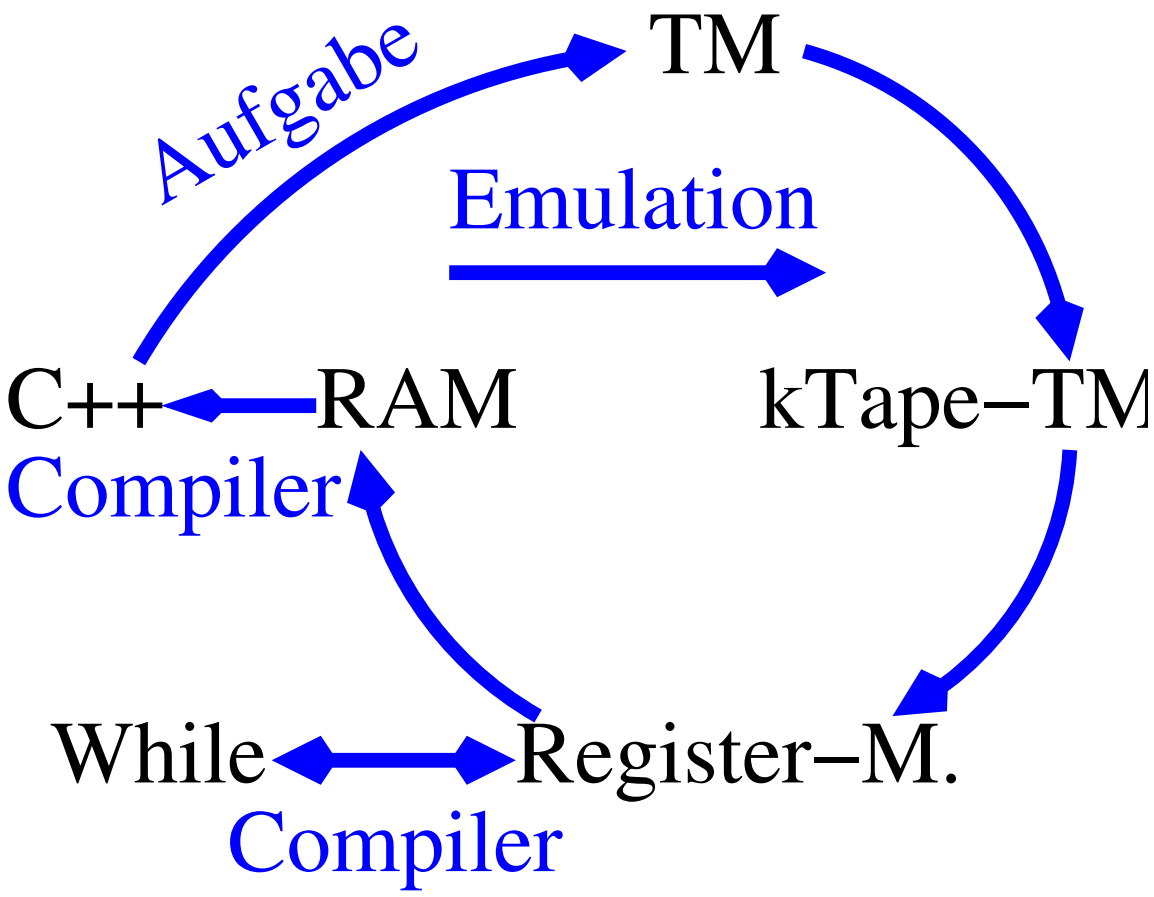
Не !

Защо?

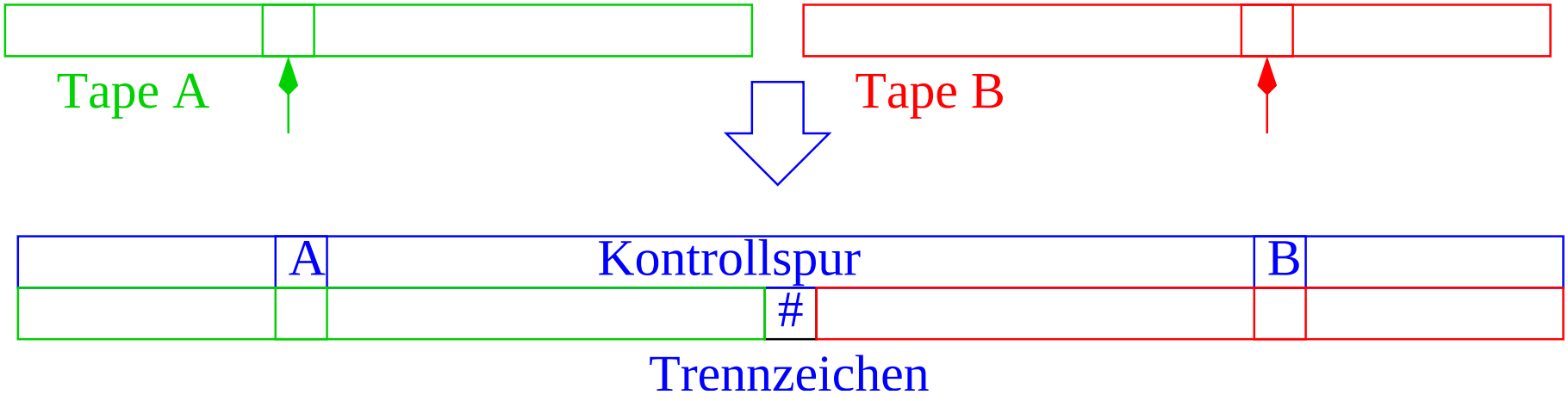
Поне всяка тотална Тюрингово изчислима функция ?

$\rightsquigarrow$ не по-късно

Еквивалентност на машинните модели



Машина на Тюринг емулира k-лентова-МТ



- Непразните думи на лентите ги събираме в една с разделители
- Позициите на главите се маркират
- Едно състояние запеметява $k-1$ символи от лентите

Времето T с k-лентова-МТ $\rightarrow \mathcal{O}\left(T^2\right)$ за 1-л.-МТ.

k-лентова-МТ емулира МНР

- По една лента за регистър (двоичен или унарен формат)
- Отделно състояние за всяка програмна команда
- Подпрограми за аритметиката
- Присвояване $\rightarrow$ Copy

МНР емулира RAM

Идея: допълнителен регистър R_S представящ паметта:

$$R_S = \sum_i S[i] \cdot 2^{bi}$$

с $b =$ броят на RAM битовете

$S[i]$ в R_j зареждане:

$$R_j := \frac{R_S}{2^{bi}} \bmod 2^b \ .$$

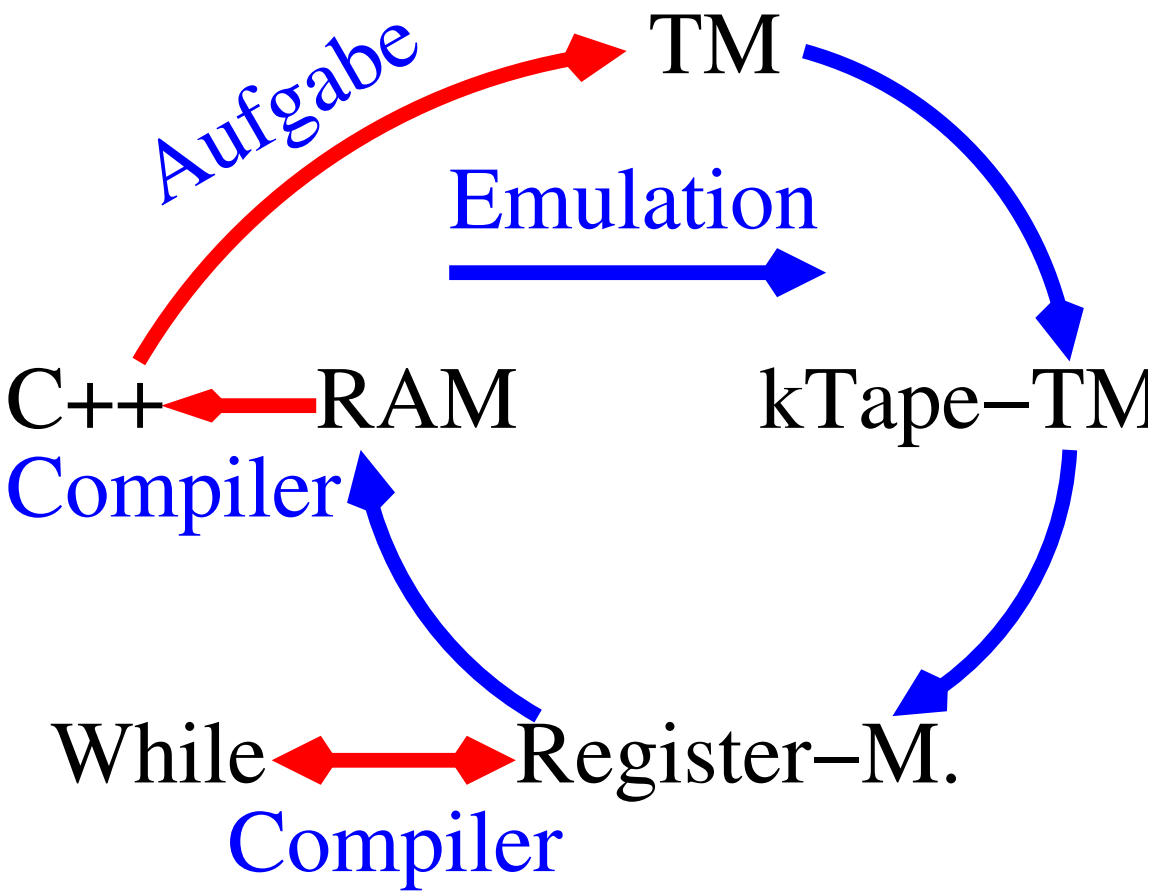
$S[i] := 0$:

$$R_S := R_S - (\frac{R_S}{2^{bi}} \bmod 2^b)2^{bi}$$

R_j в $S[i]$ запазване:

$$S[i] := 0;\ R_S := R_S + R_j \cdot 2^{bi}$$

Еквивалентност на машинните модели



Алгоритми на Марков

Детерминистични правила за заместване.

Дадено: вход $w \in \Sigma^*$

Множество от правила $\Delta \in (\Gamma^* \times \Gamma^*)^*$

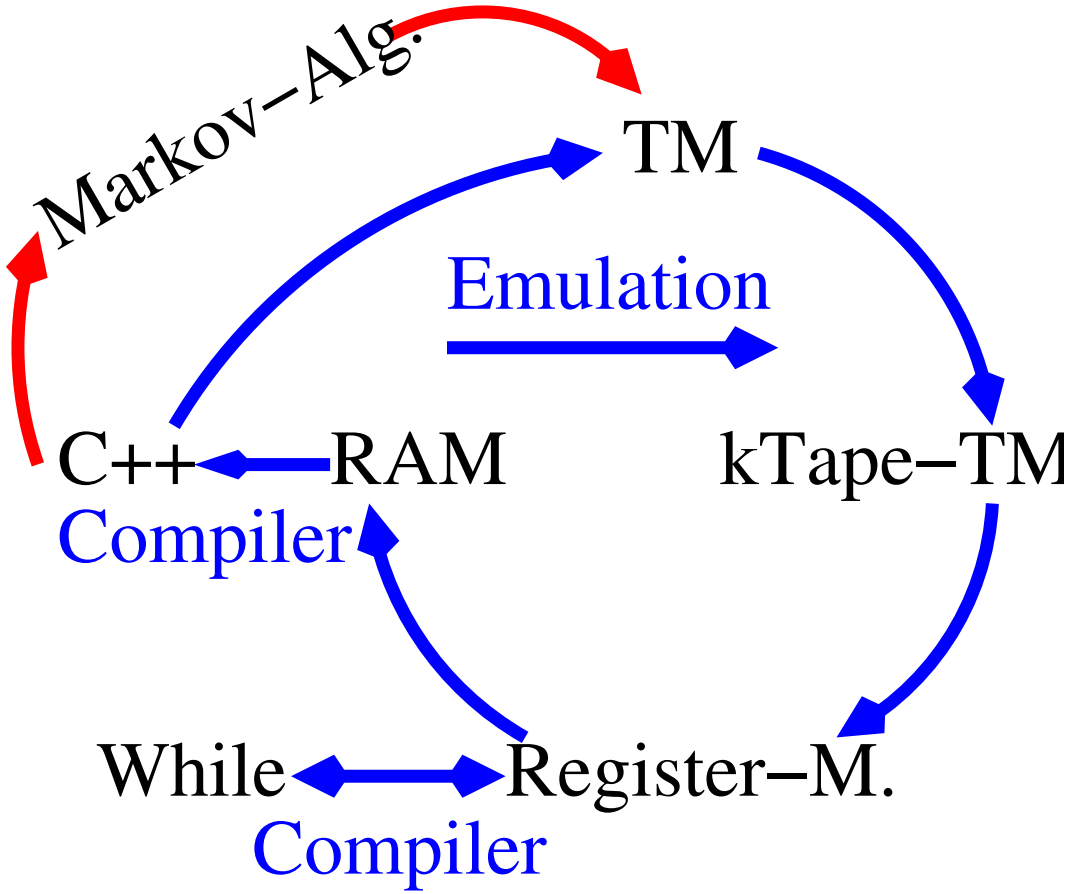
while $\exists(\ell, r) \in \Delta, u, v \in \Gamma^* : w = u\ell v$ do

намери първото правило $(\ell, r) \in R$

и най-късата дума $u \in \Gamma^*$, такава че $w = u\ell v$ за някоя v

$w := urv$

Алгоритми на Марков: Еквивалентност с МТ



Алгоритми на Марков: Еквивалентност с МТ

Дадено: МТ $M = (Q, \Sigma, \Gamma, \delta, s, F)$ с вход w.
Нека наблюдаваме отляво и отдясно max 1 $\sqcup$.

Да разгледаме алгоритъм на Марков за азбуката $Q \cup \Gamma \cup \{(,)\}$.

$$\Delta = \cdots \text{Специални правила за преходите}$$

- $\left\langle ((q)a, (q')a') : \delta(q,a) = (q', a', N) \right\rangle$
- $\left\langle (c(q)a, (q')ca') : \delta(q,a) = (q', a', L) \right\rangle$
- $\left\langle ((q)ac, a'(q')c) : \delta(q,a) = (q', a', R) \right\rangle$

вход: алгоритъм на Марков $\sqcup(s)w\sqcup$

Изход последователност от конфигурациите на МТ!

Полу-Туеви системи

Приличат на недетерминистични алгоритми на Марков.

Също Тюрингово мощни.

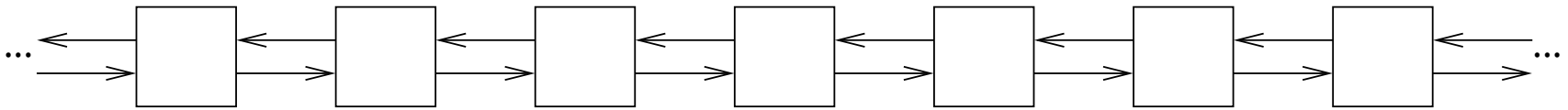
Нашата МТ-симулация има винаги едно приложимо правило.

Клетъчни автомати

Да разгледаме крайният автомат $(\{0,1\},\{0,1\}^2,\delta,\emptyset)$ с

Q	0	1	0	1	0	1	0	1
$L\times R$	$(0,0)$	$(0,0)$	$(0,1)$	$(0,1)$	$(1,0)$	$(1,0)$	$(1,1)$	$(1,1)$
δ	0	1	1	1	0	1	1	0

Свържете безкрайно много такива автомати във верига.



[М. Кук 2002]: Този модел е Тюринг-мощен.

виж също в Wikipedia "‘правило 110 клетъчен автомат"’

Квантов компютър

- Един Qubit запеметява суперпозициите 0 и 1 (и 0 и 1- едновременно).
- Изчисления с n Qubit-а дава суперпозицията на 2^n класически изчисления
- Квантовият компютър може за полиномиално време да факторизира и може да получи дискретни логаритми
- Това дава възможност много криптографски алгоритми да се реализизрат бързо

Квантов компютър: теория на изчислимостта и сложността

- Предположения от теория на сложността:
  - Факторизизрането, DLog не са в P (същото и с рандомизацията)
  - Факторизизрането, DLog не са NP.
- Машините на Тюринг могат да симулират квантовия компютър

Резултат: Квантовите компютри са по-бързи, но не по-мощни от класическите компютри

2.3 Примитивно-рекурсивни и μ-рекурсивни функции

не тук

2.4 Функция на Акерман

[Акерман 1928, Хермес]

Function $a(x,y)$

if $x = 0$ then return $y+1$

if $y = 0$ then return $a(x-1,1)$

return $a(x-1,a(x,y-1))$

Твърдение: a е тотална, МТ-изчислима функция

Д-во:

рекурсия $\rightsquigarrow$ стек с RAM $\rightsquigarrow$ МТ

Тоталност на функцията на Акерман

Д-во: Индукция по лексикографската наредба на (x,y):

База: $a(0,y) = y+1$

Индулционна стъпка за $y = 0$:

$a(x,0) = a(x-1,1)$,

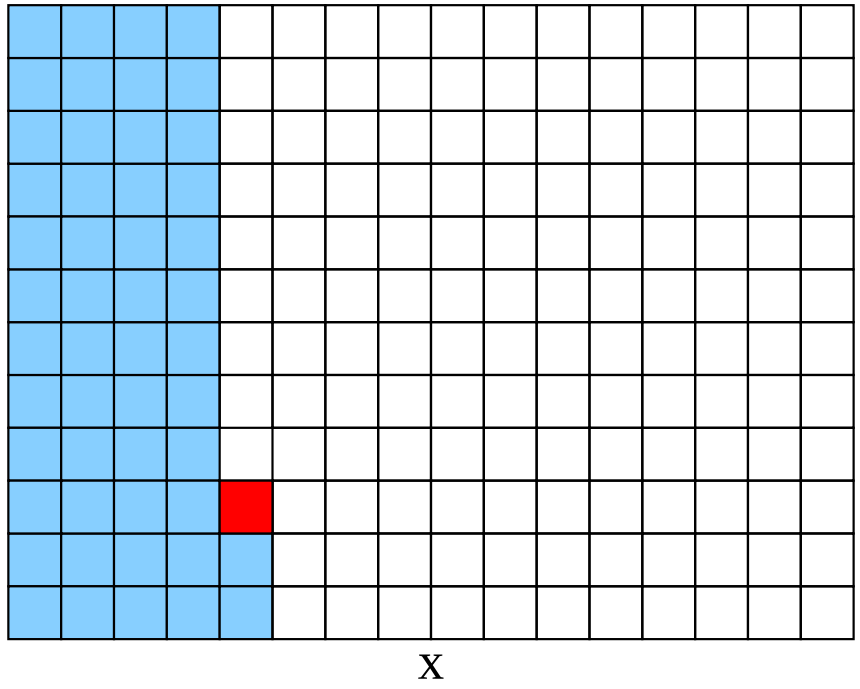
завършва по ИП $(x-1,1) < (x,0)$

Индукционна стъпка за $x,y > 0$:

$a(x-1,a(x,y-1))$ завършва

$(x,y-1) < (x,y)$ и

$(x-1,a(x,y-1)) < (x,y)$



Колко големи числа може да пресмята една Loop програма?

Дефиниция:
Нека за една Loop програма P
$\mathbf{x} = (x_0, x_1, \ldots)$ входният вектор от променливи.
Произволни!
$\mathbf{x}' = (x'_0, x'_1, \ldots)$ векторът на промренливите, с която програмата завършва.

$$f_P(\mathbf{x}) := \sum_{i \geq 0} x'_i$$

$$f_P(n) := \max\left\{ f_P(\mathbf{x}) : \sum_{i\geq 0} x_i \leq n \right\}$$

функцията на Акерман не е Loop-изчислима

Д-во: Да допуснем, че a е Loop-изчислима.

$\longrightarrow a(n,n) = g(n)$ е изчислима с Loop-програма G.

$\longrightarrow a(n,n) = g(n) \leq f_G(n)$

Но ние ще покажем:

Лема Е: $\forall$Loop-програма $P : \exists k : \forall n \in \mathbb{N} : f_P(n) < a(k,n)$.

Противоречие.

Пример

$$a(0,y) = y+1$$

$$\begin{aligned} a(1,y) &= a(0,a(1,y-1)) = a(1,y-1)+1 = \\ &\quad a(0,a(1,y-2))+1 = a(1,y-2)+2 = \cdots = \\ &\quad a(1,0)+y = y+a(0,1) = y+2 \end{aligned}$$

$$\begin{aligned} a(2,y) &= a(1,a(2,y-1) = 2+a(2,y-1) = \cdots \\ &\quad = 2y+a(2,0) = 2y+a(1,1) = 2y+3 \end{aligned}$$

Пример

$$
\begin{aligned}
a(2,y) &= 2y+3 \\
a(3,y) &= a(2,a(3,y-1)) = 2a(3,y-1)+3 \\
&= 2a(2,a(3,y-2))+3 = 4a(3,y-2)+3(1+2) \\
&= 4a(2,a(3,y-3)+3(1+2) = 8a(3,y-3)+3(1+2+4) \\
&= \cdots = 2^y \underbrace{a(3,0)}_{=5} + 3(\underbrace{1+2+\cdots+2^{y-1}}_{=2^y-1}) \\
&= 2^{y+3}-3
\end{aligned}
$$

Пример

$$a(3,y) = 2^{y+3} - 3$$

$$a(4,y) = a(3,a(4,y-1)) = 2^{a(4,y-1)+3} - 3$$

$$= 2^{a(3,a(4,y-2))+3} - 3 = 2^{2^{a(4,y-2)+3}-3+3} - 3$$

$$= 2^{2^{a(3,a(4,y-3))+3}} - 3 = 2^{2^{2^{a(4,y-3)+3}-3+3}} - 3$$

$$= \cdots = 2^{\overset{2}{\vdots}^{\overbrace{a(4,0)}^{=a(3,1)=2^{1+3}-3}+3}} - 3 = 2^{\overset{2^{16}}{\vdots}} - 3$$

$$a(4,2) = 2^{2^{16}} - 3 = 2^{65536} - 3$$

Монотонност на функцията на Акерман

Лема A: $y < a(x,y)$

Лема B: $a(x,y) < a(x,y+1)$

Лема C: $a(x,y+1) \leq a(x+1,y)$

Лема D: $a(x,y) < a(x+1,y)$

Лема BD: $a(x,y) \leq a(x',y')$ ако $x \leq x'$ и $y \leq y'$

Д-во: Упражнение. Индукция,...

Лема Е: $\forall$Loop-program $P : \exists k : \forall n \in \mathbb{N} : f_P(n) < a(k,n)$.

Д-во: Индукция по дефиницията на Loop-програма.

База на индукцията:

$f_{\text{emptyprogram}}(n) = n < n+1 = a(0,n)$.

$f_{x:=\ y+c}(n) \leq 2n+1 < 2n+3 = a(2,n)$

Индукционна стъпка за $P = P_1; P_2$:

По ИП $\exists k_1, k_2 : f_{P_1}(n) < a(k_1, n) \wedge f_{P_2}(n) < a(k_2, n)$.

Нека $k_3 = \max\{k_1 - 1, k_2\}$. Тогава:

$$
\begin{aligned}
f_P(n) &\leq f_{P_2}(f_{P_1}(n)) && \text{Деф. } f_P \\
&< a(k_2, f_{P_1}(n)) && \text{ИП} \\
&< a(k_2, a(k_1, n)) && \text{ИП,монотонност} \\
&\leq a(k_3, a(k_3 + 1, n)) && \text{монотонност} \\
&= a(k_3 + 1, n + 1) && \text{Деф. } a \\
&\leq a(k_3 + 2, n) && \text{Лема B}
\end{aligned}
$$

Индукционна стъпка за $P = \textbf{loop}\ x_i\ \text{do}\ Q$:

Нека x_i не е от Q (напр. копирана в нова променлива).

По ИП $\exists k : f_Q(n) < a(k,n)$.

Нека $\mathbf{x}$ е един вход за $f_P(\mathbf{x})$ и е увеличен с $\sum_j x_j \leq n$.

Нека $m \leq n$ да е стойността на x_i в $\mathbf{x}$

$$
\begin{aligned}
f_P(n) &= f_P(\mathbf{x}) \\
&\leq \underbrace{f_Q(f_Q(\cdots f_Q}_{m \text{ пъти}}(n-m)\cdots)) + m && \text{Деф. } m \\
&\leq a(k, \underbrace{f_Q(f_Q(\cdots f_Q}_{m-1 \text{ пъти}}(n-m)\cdots))) + m - 1 && \text{ИП}\cdots \\
&\leq \underbrace{a(k, a(k, \cdots a(k}_{m \text{ пъти}}, n-m)\cdots)) + m - m && \text{ИП}
\end{aligned}
$$

Индукционна стъпка за $P = \textbf{loop}\ x_i\ \text{do}\ Q$:

$$
\begin{aligned}
f_P(n) \leq & \underbrace{a(k, a(k, \cdots a(k}_{m \text{ пъти}}, n-m) \cdots)) \\
< & \underbrace{a(k, a(k, \cdots a(k}_{m-1 \text{ пъти}}, a(k+1, n-m) \cdots)) && \text{монотонност} \\
= & \underbrace{a(k, a(k, \cdots a(k}_{m-2 \text{ пъти}}, a(k+1, n-m+1) \cdots)) && \text{Деф.}a \\
= \cdots = & a(k+1, n-1) && \text{Деф.}a \\
\leq & a(k+1, n) && \text{монотонност}
\end{aligned}
$$

qed.

По-бързо растящи функции

$k \mapsto$
$\max\{f_P(0) : P$ завършваща While-прогарма с k инструкции.$\}$

Busy Beaver

$\Sigma(n)$: $\max_\delta$ # единици, които са на лентата след като DMT $(\{1,\ldots,n,Z\},\emptyset,\{0,1\},\delta,1,\{Z\})$ завършва (празен вход).

$S(n)$: $\max_\delta$ #преходи, които една DMT $(\{1,\ldots,n,Z\},\emptyset,\{0,1\},\delta,1,\{Z\})$ прави преди да завърши(празен вход).

Busy Beaver

$\Sigma(n)$, $S(n)$ са тоатлни не изчислими функции.

n	$\Sigma(n)$	$S(n)$
1	1	1
2	4	6
3	6	21
4	13	107
5	$\geq$4 098	$\geq$47 176 870
6	$> 1.29 \cdot 10^{865}$	$> 3 \cdot 10^{1730}$

[http://www.drb.insel.de/~heiner/BB/],

[http://www.logique.jussieu.fr/~michel/ha.html]

Busy Beaver

n: 6 5 4 3 2

q/in 0 1 0 1 0 1 0 1 0 1

A: B1R F0L; B1R C1L; B1R B1L; B1R Z1L; B1R B1L

B: C0R D0R; C1R B1R; A1L C0L; B1L C0R; A1L Z1R

C: D1L E1R; D1R E0L; Z1R D1L; C1L A1L;

D: E0L D0L; A1L D1L; D1R A0R;

E: A0R C1R; Z1R A0L; n=1:

F: A1L Z1R; H1N ---

Бавно растящи функции

Обратна функция на Акерман

$$\alpha(m,n) := \min\{i \geq 0 : a(i, \lfloor m/n \rfloor) > \log_2 n\}$$

За всеки реалистичен случай $\alpha(m,n) \leq 5$.

Но $\alpha(m,n) \notin O(1)$.

Важна структура от данни има сложност $m\Theta(\alpha(m,n))$ за m операции и n обекта:

Union-Find Data Structure

Class UnionFind($n : \mathbb{N}$) // Дава разделяне на $1..n$

Function find($i : 1..n$) : $1..n$

assert $\forall i, j \in \{1, \ldots, n\}$:

find(i) = find(j) $\Leftrightarrow$ i, j са в една част

Procedure union($i, j : 1..n$)

A:= частта с $i \in A$

B:= частта с $j \in B$

обедини A и B в една част

Приложения: например алгоритъм на Крускал за minimal spanning tree

Class UnionFind($n : \mathbb{N}$)

parent=$[1, 2, \ldots, n]$: Array $[1..n]$ of $1..n$

gen=$[0, \ldots, 0]$: Array $[1..n]$ of $0..\log n$ // генерация на лидър

Function find($i : 1..n$) : $1..n$

if parent$[i] = i$ then return i

else $i' :=$ find(parent$[i]$)

parent$[i] := i'$ // компресия на път

return i'

Procedure link($i, j : 1..n$)

assert i и j са лидъри на различни множества

if gen$[i] <$ gen$[j]$ then parent$[i] := j$ // баланс

else parent$[j] := i$; if gen$[i] =$ gen$[j]$ then gen$[i]$++

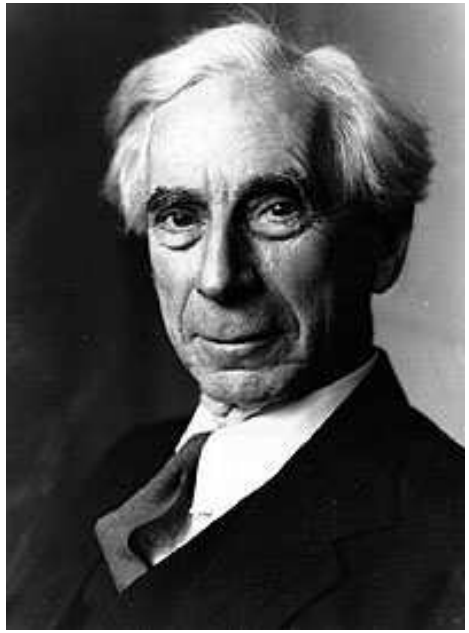
Procedure union($i, j : 1..n$)

if find$(i) \neq$ find(j) then link(find(i), find(j))

2.5 Проблемът за завършване, Неразрешимост, Сводимост

- Гьоделова номерация: МТ могат ад оперират над себе си като вход
- Важен пример: Универсална МТ
- Диагонален метод: един неразрешим език
- Сводимост: показва, че и други проблеми са неразрешими.

Парадокси и Self reference



Бръснарят в един малък град
бръсне тези и само тези хора,
които не се бръснят сами.

.

Кой бръсне бръснаря?

Парадокси и Self reference

Даниел Дйосентриб решил да изобрети една машина която всичко знае.

Да	Не

Задавате да/не въпрос и отговорът светва.

Дагоберт Дъг иска да купи машината.

Но е решил да провери дали работи правилно.

Задава въпрос на машината:

Ще ми отговориш ли с не?

Какво се случва?

Разрешимост

$A \subseteq \Sigma^*$ е (разрешимо), ако

характеристичната му функция χ_A е изчислима.

$$\chi_A(w) = \begin{cases} 1 & \text{ако } w \in A \\ 0 & \text{ако } w \notin A \end{cases}$$

Полуразрешимост

$A \subseteq \Sigma^*$ е полуразрешимо, ако
"'полу"' характеристичната му функция χ_A е изчислима.

$$\chi_A(w) = \begin{cases} 1 & \text{ако } w \in A \\ \bot & \text{ако } w \notin A \end{cases}$$

Твърдение: $A \subseteq \Sigma^*$ разрешимо $\Leftrightarrow$
A и $\overline{A}$ са полуразрешими

Д-во: Нека МТ

M_A полуразрешава A и

$M_{\overline{A}}$ полуразрешава $\overline{A}$

for $s := 1$ to ∞ do

if M_A завършва за s стъпки then Accept

if $M_{\overline{A}}$ завършва за s стъпки then Reject

Рекурсивна номеруемост

$A \subseteq \Sigma^*$ рекурсивно номеруемо, ако

$A = \emptyset$ или $\exists$тотална изчислима функция $f : \mathbb{N} \to \Sigma^*$:

$$A = \{f(1), f(2), f(3), \ldots\}$$

Твърдение: A е рекурсивно номеруемо $\Leftrightarrow$ A е полуразрешимо

Рекурсивна номеруемост $\longrightarrow$ полуразрешимост

Нека A е рекурсивно номеруемо с помощта на f.

Function $\chi'_A(x)$

for $s := 1$ to ∞ do

if $f(n) = x$ then return 1

Рекурсивна номеруемост $\longrightarrow$ полуразрешимост

- Нека $\pi(k,m) = 2^k(2m+1) - 1$ - е кодираща функция на всички двойки естествени числа.
- Всяко естествено число n е код на точно една двойка числа $n = \pi(k,m)$.
- Нека $L(\pi(m,k)) = m$ и $R(\pi(m,k)) = k$ са декодиращите функции.
- π, L, R са изчислими функции.
- Да разгледаме редицата на всички думи в Σ^*: $\alpha_0, \alpha_1, \ldots, \alpha_i, \ldots$, наредени така: $|\alpha_i| < |\alpha_{i+1}|$ или $|\alpha_i| = |\alpha_{i+1}|$ и α_i е лексикографски по-малко от α_{i+1}.
- Например: $a, b, aa, ab, ba, bb, \ldots$

Полуразрешимо $\longrightarrow$ рекурсивно номеруемо

Нека M е МТ полуразрешаваща A.

Ако $A = \emptyset$: тривиално.

Иначе дефинираме функция $f : \mathbb{N} \to \Sigma^*$ с област на стойностите A.

Function $f(n)$

a:= фиксиран елемент на A

Интерпретираме n като код на двойката $n = \pi(m, k)$

Да разгледаме думата с номер m: $u = \alpha_m$

if M завършва при u за $\leq k$ стъпки then return u

else return a

Полуразрешимо $\longrightarrow$ рекурсивно номеруемо

- f е тотална
- f приема само стойности от A
- $\forall u \in A \exists k : M$ приема u за k стъпки
- $f(\text{Coding}(k,u)) = u$

■

Задача: Да се докаже, че ако $A \subseteq \Sigma^*$ е безкрайно, то A е разрешимо тогава и само тагава, когато съществува тотална изчислима функция $f : \mathbb{N} \to \Sigma^*$:

$A = \{f(0) < f(1) < f(2) < \ldots\}$ в лексикографски ред по големина.

Еквивалентни определения

- A е рекурсивно номеруемо
- A е полу-разрешимо
- A е от Чомски тип 0
- $A = L(M)$ за МТ M
- χ'_A е Тюринг-, While-, MHP, RAM, ... изчислима
- A е дефиниционна област на една (частична) изчислима функция
- A е област от стойности на изчислима функция

2.6 Неразрешими Проблеми

Номерация на машините на Тюринг

Да разгледаме $T = (Q, \Sigma, \Gamma, \delta, s, F)$. Нека:

- $Q = \{1, \ldots, n\}$
- $\Sigma = \{0, 1\}$
- $\Gamma = \{0, 1, \sqcup\}$, $\sqcup = 2$
- $s = 1$
- $F = \{2\}$

за подходяща константа n

Пример

Нека $M = (\{1,2,3\}, \{0,1\}, \{0,1,\sqcup\}, \delta, 1, \{2\})$, with

$\delta(1,1) = (3,0,R)$

$\delta(3,0) = (1,1,R)$

$\delta(3,1) = (2,0,R)$

$\delta(3,\sqcup) = (3,1,L)$

$\langle M \rangle$ е тогава:

111010010001010001100010101001000110001001001010001100010001000100100111

Диагонален език L_d

Нека M_i е МТ с $\langle M_i \rangle = i$.

Нека w_i двоичното представяне на i.

$L_d := \{w_i : M_i \text{ не приема } w_i\}$

Твърдение: L_d е неразрешим

Д-во:

Да допуснем:

$L_d = \{w_i : M_i$ не приема $w_i\}$ е разрешим.

Деф. "‘разрешим"’ $\overrightarrow{\text{Деф.}}$ $\exists M_i : M_i$ разпознава L_d и винаги завършва.

Как работи M_i над w_i?

$w_i \in L_d \overset{\text{Деф. } M_i}{\rightarrow} w_i$ ще се приеме от M_i. $\overset{\text{Деф. } L_d}{\rightarrow} w_i \notin L_d$

$w_i \notin L_d \overset{\text{Деф. } M_i}{\rightarrow} w_i$ няма да се приеме от M_i. $\overset{\text{Деф. } L_d}{\rightarrow} w_i \in L_d$

И в двата случая имаме противоречие.

Следствие:

$\overline{L}_d = \{w_i : M_i \text{ приема } w_i\}$ е неразрешим

Да допуснем: $\overline{L}_d$ е разрешим.

$\rightarrow \exists M : M$ приема $\overline{L}_d$

Променяме $M \rightsquigarrow M'$ така M' приема L_d

(Заменяме: приема/не приема за заключителните състояния).

Противоречие.

Универсална машина на Тюринг

$$U = (Q_u, \{0,1\}, \{0,1,\sqcup\}, \delta_u, s_u, F_u)$$

вход: $\langle M \rangle w$

M е симулираната МТ, w е двоично кодираният вход.

U симулира M on w.

U приема $\langle M \rangle w$, ако M приема w

Универсална машина на Тюринг

3 ленти:

1. $\langle M \rangle$
2. състоянието q_M на M (унарно кодирано)
3. съдържанието на лентата w на M

Универсална машина на Тюринг

if prefix v на w е код на една МТ then // 111tuple111

преместваме v на лентата $\langle M \rangle$

$q_M := 1$ // начлното състояние на M

while $q_M \neq 2$ do // крайно сътояние на M

преместваме се до началото на $\langle M \rangle$

foreach $(q, a, r, b, d) \in \langle M \rangle$ do // поле по поле

if $q = q_M$ then // сравняваме с q_M

if входният символ на лента 3$= a$ then

$q_M := r$ // копираме на лента 2

слагаме b на лента 3

Движението по лента 3 е съгласно d

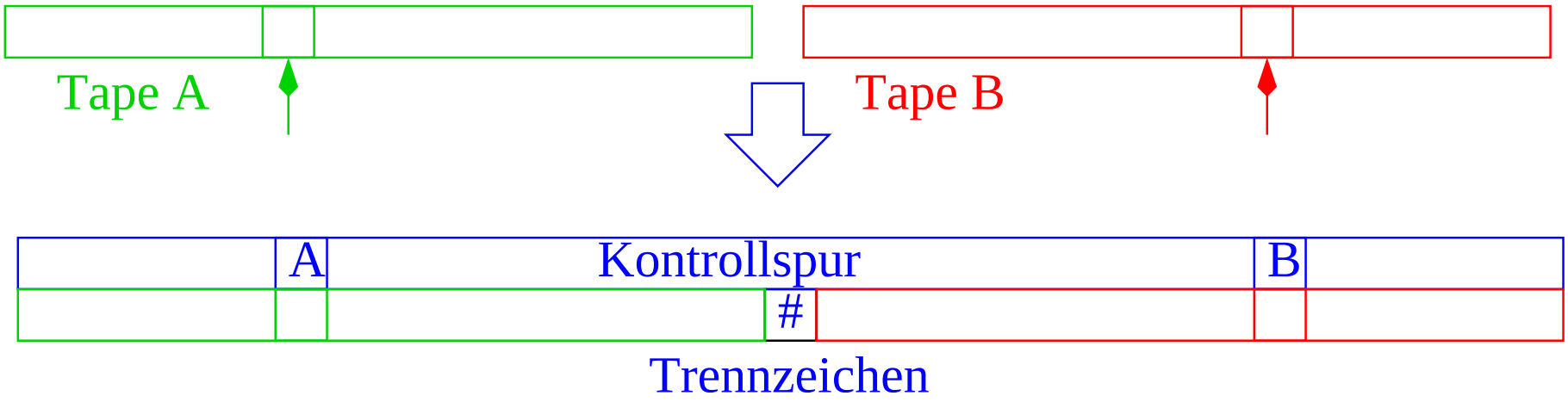
Универсална машина на Тюринг: 3ленти→1лента

Всъщност ние знаем как работи.

Проблем: лентовата азбука независимо от M но $> \{0,1\}$

Кодираме лентовата азбука с фиксиран брой $\{0,1\}$.

Проблем: входът трябва да се кодира също.



Проблемът за завършване

$H := \{w_i v : M_i$ завършва при $v\}$

Твърдение: H не е разрешимо.

Д-во: Да допуснем, че H е разрешимо.

Конструираме една МТ, която ще разпознава $\overline{L}_d$.

$w_i \in \overline{L}_d$?

$\Leftrightarrow M_i$ приема w_i.

$\Leftrightarrow w_i w_i \in H \wedge M_i$ приема w_i.

Това може да направим с помощта на една МТ за H и универсалната МТ.

Противоречие.

Ограниченият проблем за завършване

Твърдение:

$\{w_i v \# w_j : M_i$ завършва при v за най-много j стъпки$\}$

е разрешимо.

Д-во:

Нека универсалната МТ U работи над $w_i v$

за j симулиращи стъпки.

Други неразрешими проблеми

Дадено: машина на Тюрингs T, T'

- $L(T) = \emptyset$? празнота
- $|L(T)| = \infty$? безкрайност
- $L(T) = \Sigma^*$? пълнота
- $L(T) = L(T')$? еквивалентност

Неразрешимост на празнотата

Да допуснем, че M приема $\{i : L(M_i) = \emptyset\}$

Ще покажем, че $\overline{L_d}$ ще стане разрешим.

$w_i \in \overline{L_d} = \{w_i : M_i \text{ приема } w_i\}$?

Констуираме една машина на Тюринг $T(i)$:

> изтрий входа
>
> изпълни M_i над w_i
>
> if state$(M_i) \neq 2$ then безкраен цикъл

Now е $L(T(i)) \neq \emptyset$ ако $w_i \in \overline{L_d}$.

Следователно $\overline{L_d}$ е разрешим.

Противоречие.

Неразрешимост на пълнотата

$L(T) = \Sigma^*$?

подобно доказателство като празнотата! Тук $T(i)$ игнорира входа си!

Метапрограмиране

Доказателството на празнотата взима една програма и я трансформира в друга.

Важна програмна техника.

Post Correspondence Problem (PCP)

Дадено: карйни редици от двойки думи

$$K = (x_1, y_1) \cdots (x_n, y_n) \in (\Sigma^+ \times \Sigma^+)^*$$

Въпрос:

$$\exists i_1, \ldots, i_k \in \{1, \ldots, n\} : x_{i_1} \ldots x_{i_k} = y_{i_1} \ldots y_{i_k}$$

?

Пример

- $K = ((1,111),(10111,10),(10,0))$ има решение $(2,1,1,3)$, имаме:

$$x_2x_1x_1x_3 = 1011111110 = 1011111110 = y_2y_1y_1y_3$$

- $K = ((10,101),(011,11),(101,011))$ няма решение:
  $(133\cdots)$

Пример [Mirko Rahn]

- $K = ((0,011),(001,1),(1,00),(11,110))$

  има на-кисо решение сдължина 595:

  12112121121121121203212112130321203311213111212031212121121312103212112103212130321202111120331121321212131112121121111203203121203212112121213131203213032120320321031213033112131302103201112121112100210121212121203212112121212021203203213032121120321331303303212130303121131130320010321211121211312123032121203212103011032130323021212303331011203131021212131210203203120213131213211032111121212021111212121203213212121211203213031120321121213033121320311212032121310210103213032102021230330213210113021121132031212101032021231321103111212312120303213303003

PCP е полуразрешим

Алгоритъм:

Procedure PCP$((x_1,y_1)\cdots(x_n,y_n))$

for $k := 1$ to ∞ do

foreach $i_1\cdots i_k \in \{1..n\}^k$ do

if $x_{i_1}\cdots x_{i_k} = y_{i_1}\cdots y_{i_k}$ then

output $i_1\cdots i_k$

return

PCP е неразрешим

Д-во виж Schöning.

Идея: допускаме, че е разрешим → проблемът за завършване става разрешим

$x_{i_1} \dots x_{i_k} = y_{i_1} \dots y_{i_k} = (s)w\#\cdots\#u(f)v$

описва една приемаща последователност от МТ-конфигурации

10. проблем на Хилберт — Диофантови уравнения

Дадено:
полином на на много променливи p
с цели коефициенти.
Въпрос [Hilbert 1900]:

$$\exists x_1, \ldots, x_n \in \mathbb{Z} : p(x_1, \ldots, x_n) = 0?$$

[Matiyasevich 1970]: Проблемът е неразрешим.

Затвореност на разрешимите езици

Затворени относно

$\cup$

$\cap$

$\bar{\cdot}$

Затвореност на полуразрешимите езици

Затворени относно

$\cup$

$\cap$

Не са затворени относно

$\bar{\cdot}$

Затвореност на полуразрешимите езици относно обединение

Нека M_1 и M_2 полуразрешават L_1 и L_2

Полуразрешаваща процедура за $L_1 \cup L_2$:

for $j := 1$ to ∞ do

    if M_1 приема w след j стъпки then Accept

    if M_2 приема w след j стъпки then Accept

Незатвореност на полуразрешимите езици относно допълнение

Да допуснем : че са затворени относно допълнение.
Нека M полуразрешава L_d, $\overline{M}$ полуразрешава $\overline{L_d}$

Function isInLd(w)

for $j := 1$ to ∞ do

if M приема w след j стъпки then return true

if $\overline{M}$ приема w след j стъпки then return false

Поне едната от двете завършва.

$\rightarrow L_d$ разрешимо.

Противоречие.

Приложение за паралелна реализация

Няколко алгоритъма $A_1, \ldots, A_k$, решават трудна задача (бавно, бързо, никога).

Зареждаме всички алгоритми едновременно (псевдо).

– Ако са едновременно бързи губим k цената на изчислението

\+ Ако някой не завършва, то сме спечелили безкрайно много.

\+ С различно време за изпълнение можем средно да спечелим.

\+ Можем да използваме паралелни процесори.

\+ Често можем да спестим част от излишната работа.

Паралелна реализация

Приложение: Доказазване на теореми,
Програмен/Хардуерен-верификатор, трудно планиране
и оптимизационни задачи

Пример: Изпълнимост на предикатни формули за време
$\mathcal{O}\big((4/3)^{\#\text{Променливи}}\big)$.
[U. Schöningh, A Probabilistic Algorithm for k-SAT and Constraint Satisfaction Problems, FOCS, 1999]