1 Computability theory

1.1 Intuitive idea of computability and Church Thesis

The surest recipe for non computer scientist to be horrified:

A hot debate over the right program language

- All program languages and machine models are „equally powerful“
- In every model there are the same non computable problems

The computability in the intuitive sense

$f : \mathbb{N}^k \to \mathbb{N}$ is called(partial) function.

f is computable if

$\exists$ an effective procedure($=$algorithm) which computes f.

effective procedure $=$ Java-program (, ..., "appropriate" program language)

Input: $(x_1, \ldots, x_k) \in \mathbb{N}^k$

Output: $f(x_1, \ldots, x_k)$

program halts in finitely many steps in case of $(x_1, \ldots, x_k) \in$ domain of f.

infinite loop otherwise.

Examples

input n

repeat

until false

computes the total not defined function Ω

$$f_\pi(n) = \begin{cases} 1 & \text{if } n \text{ is an initial segment in the decimal representation of } \pi. \\ 0 & \text{otherwise} \end{cases}$$

f_π is computable:

Use large number arithmetic, and apply an appropriate approximations "large enough".

Example

$$f(n) = \begin{cases} 1 & \text{if } n \times `7' \text{ appears somewhere in the decimal representation of } \pi \\ 0 & \text{otherwise} \end{cases}$$

Is f computable ? Yes !

If $\forall n : n \times `7'$ occurs: $\forall n : f(n) = 1$

If $`7'$ occurs maximum n_0 times somewhere:

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

Church-Turing thesis

Functions computable by Turing machine

are exactly those computable in the intuitive sense.

Not a proposition but everybody accepted.

The reasons

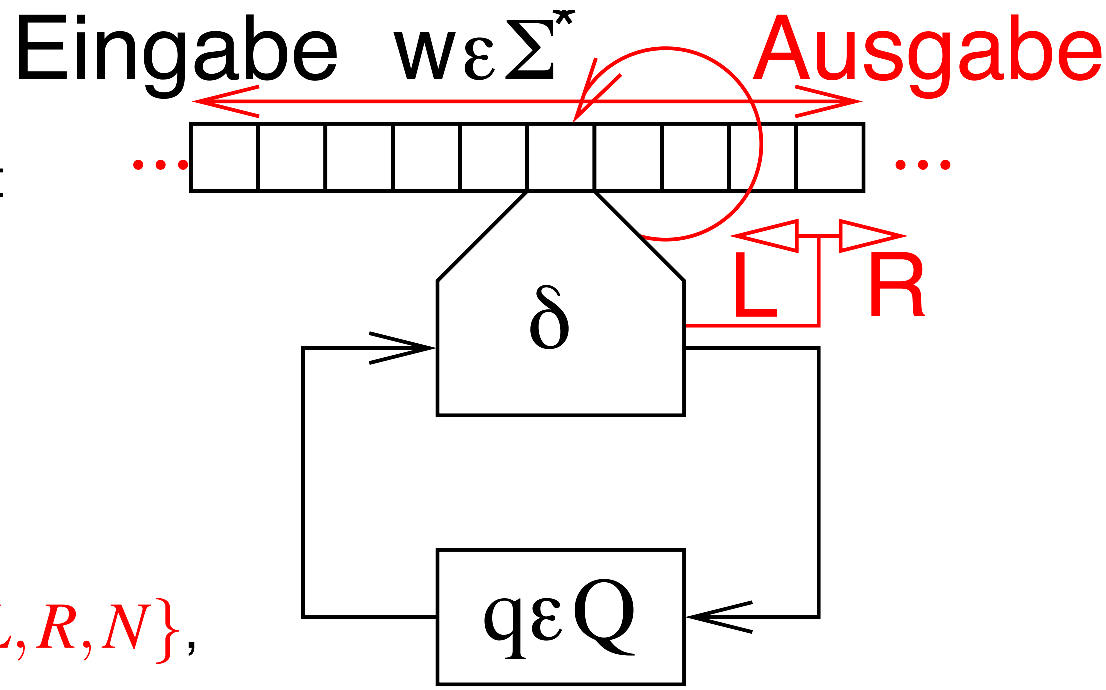
- All known computable models are weaker or equivalent.

  this we can prove

- All „intuitive“ computable known function are Turing-computable.

Deterministic Turing machines (DTM)

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



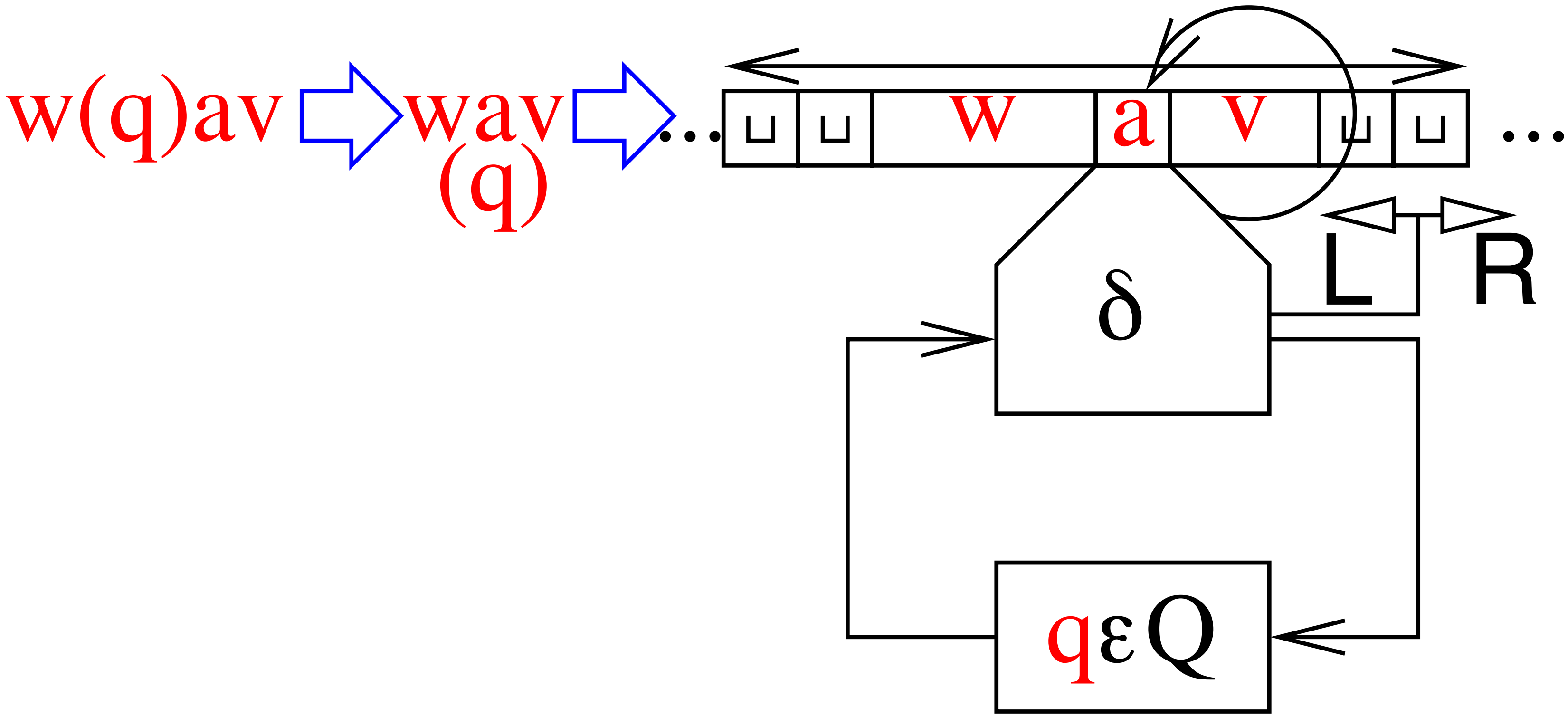
- Q states, Σ input alphabet
- Γ tape alphabet,

  $\sqcup \notin \Sigma$: blank symbol,

  $\Sigma \cup \{\sqcup\} \subseteq \Gamma$
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, N\}$,

  transition function;
- $s \in Q$, initial state
- $F \subseteq Q$, final states

Configuration of a TM



$w, v \in \Gamma^*$, $a \in \Gamma$, $q \in Q$

Functionality of DTM

$$wa(q)bcv \overset{\delta(q,b)=(q',b',N)}{\vdash} wa(q')b'cv$$

$$wa(q)bcv \overset{\delta(q,b)=(q',b',L)}{\vdash} w(q')ab'cv$$

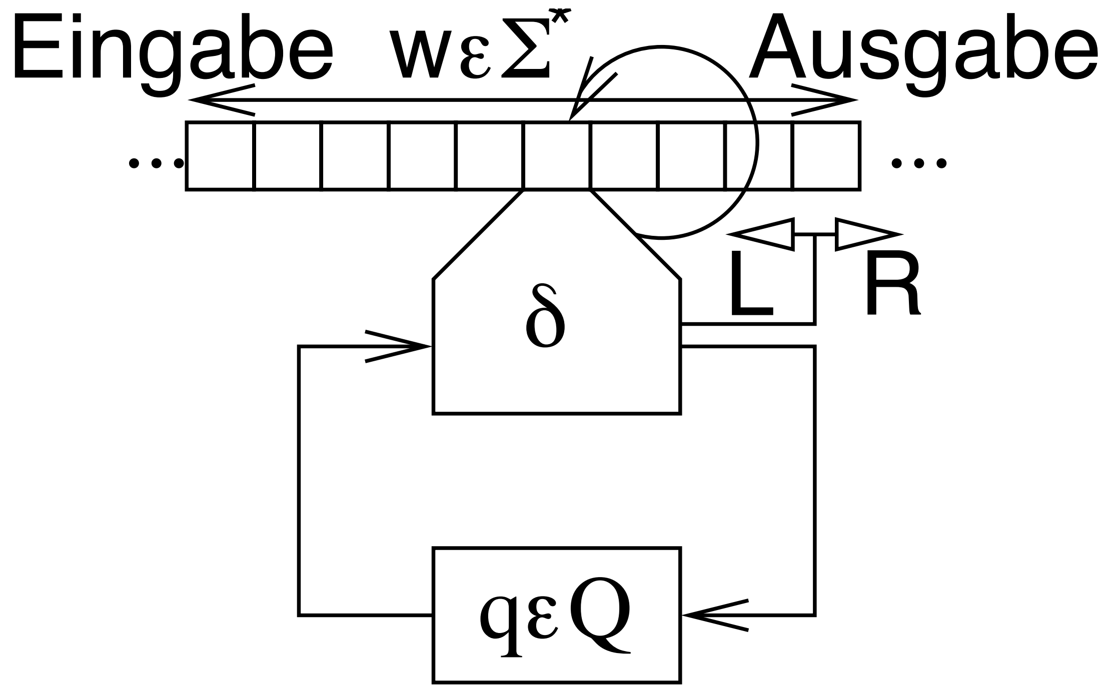
$$wa(q)bcv \overset{\delta(q,b)=(q',b',R)}{\vdash} wab'(q')cv$$

The transition function δ has three issues:

- New state like FA
- New tape symbol — overwrites the old symbol in head position
- Moving directions of the head

Nondeterministic Turing machines (NTM)

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



- Q, states
- Σ, input alphabet
- Γ tape alphabet,
  $\sqcup \notin \Sigma$: blank symbol,
  $\Sigma \cup \{\sqcup\} \subseteq \Gamma$
- $\delta : Q \times \Gamma \to 2^{Q \times \Gamma \times \{L,R,N\}}$,
  transition function;
- $s \in Q$, initial state
- $F \subseteq Q$, final states

How NTM works?

T accepts $w \Leftrightarrow$

$\exists \alpha, \beta \in \Gamma^*, f \in F : (s)w \vdash^* \alpha f \beta$

$L(T) := \{w \in \Sigma^* : T \text{ accepts } w\}$.

The difference between DMT versus NTM:

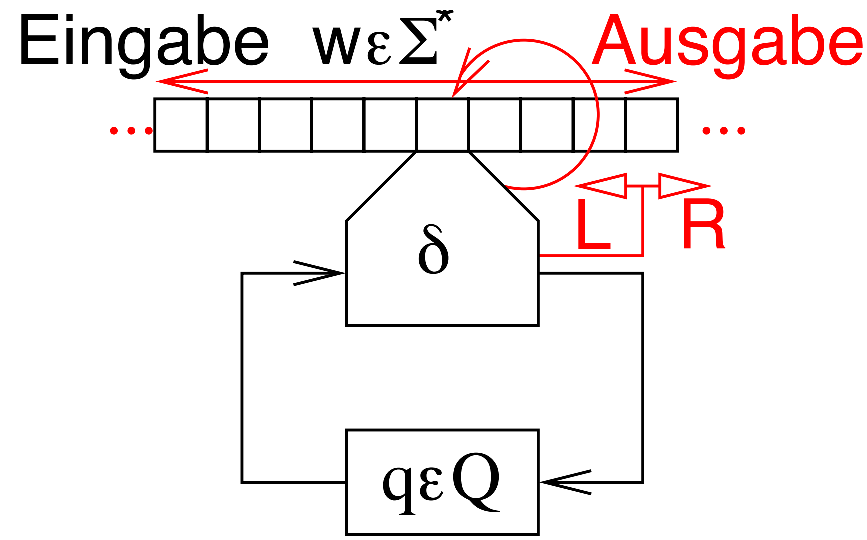
δ defines the transitions between configurations versus

δ admits between configurations .

Turing machines as acceptors

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

$L(T)$?



Definition:

T accepts $w \Leftrightarrow$

$\exists \alpha, \beta \in \Gamma^*, f \in F : (s)w \vdash^* \alpha f \beta$

($\exists$ a sequence of (by δ admitted)

configuration transitions

$(s)w \to \cdots \to x(f)y$ with $f \in F$.)

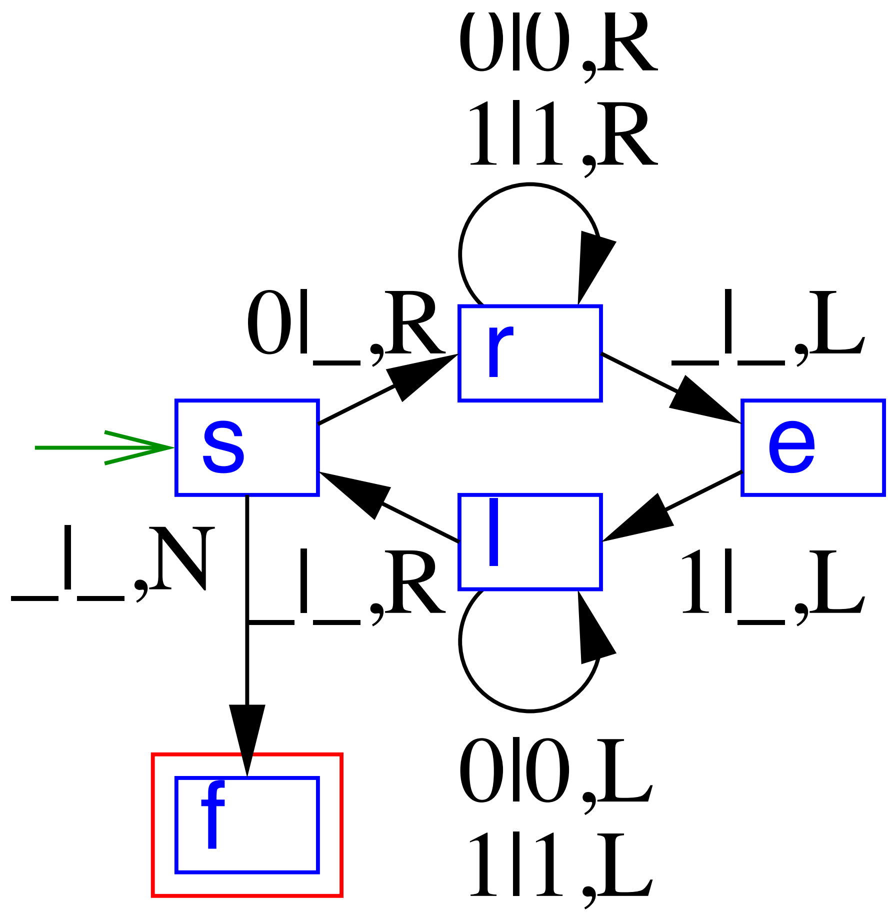
$L(T) := \{w \in \Sigma^* : T \text{ accepts } w\}$.

Decidable languages

A language $L \subseteq \Sigma^*$ is decidable if there is a TM T s.t. $L(T) = L$ and T stops on every input:

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

Example: $\{0^n1^n : n \geq 0\}$.



Properties of the decidable sets

Proposition: $A, B \subseteq \Sigma^*$ decidable $\Rightarrow$

$A \cup B$,

$A \cap B$,

$A \setminus B$,

$A \cdot B$,

A^* are decidable.

Examples : $\Sigma^*, \emptyset$,

Every finite set is definable.

Turing machines compute functions

$T = (Q, \Sigma, \Gamma, \delta, s, F)$ computes the partial function $f_T : \Sigma^* \to \Gamma^* \Leftrightarrow$

$$f_T(w) := \begin{cases} v & \text{if } T \text{ halts by input of } w \text{ with output } v \\ & ((s)w \Rightarrow u(q)v), q \in F \\ \perp = \text{(not defined)} & \text{otherwise} \end{cases}$$

g is Turing computable $\Leftrightarrow \exists T : f_T = g$

Remark: when $g(x) = \perp$, T does not halt.

Decidable languages

Corollary: L is decidable $\Leftrightarrow$ the characteristic function c_L is computable.

$$c_L \colon \Sigma^* \to \{0,1\} \quad \text{with} \qquad c_L(w) = \begin{cases} 1 & \text{if } w \in L \\ 0 & \text{otherwise} \end{cases}$$

Example: $\{0^n 1^n : n \geq 0\}$ is decidable.

Semi-decidable languages

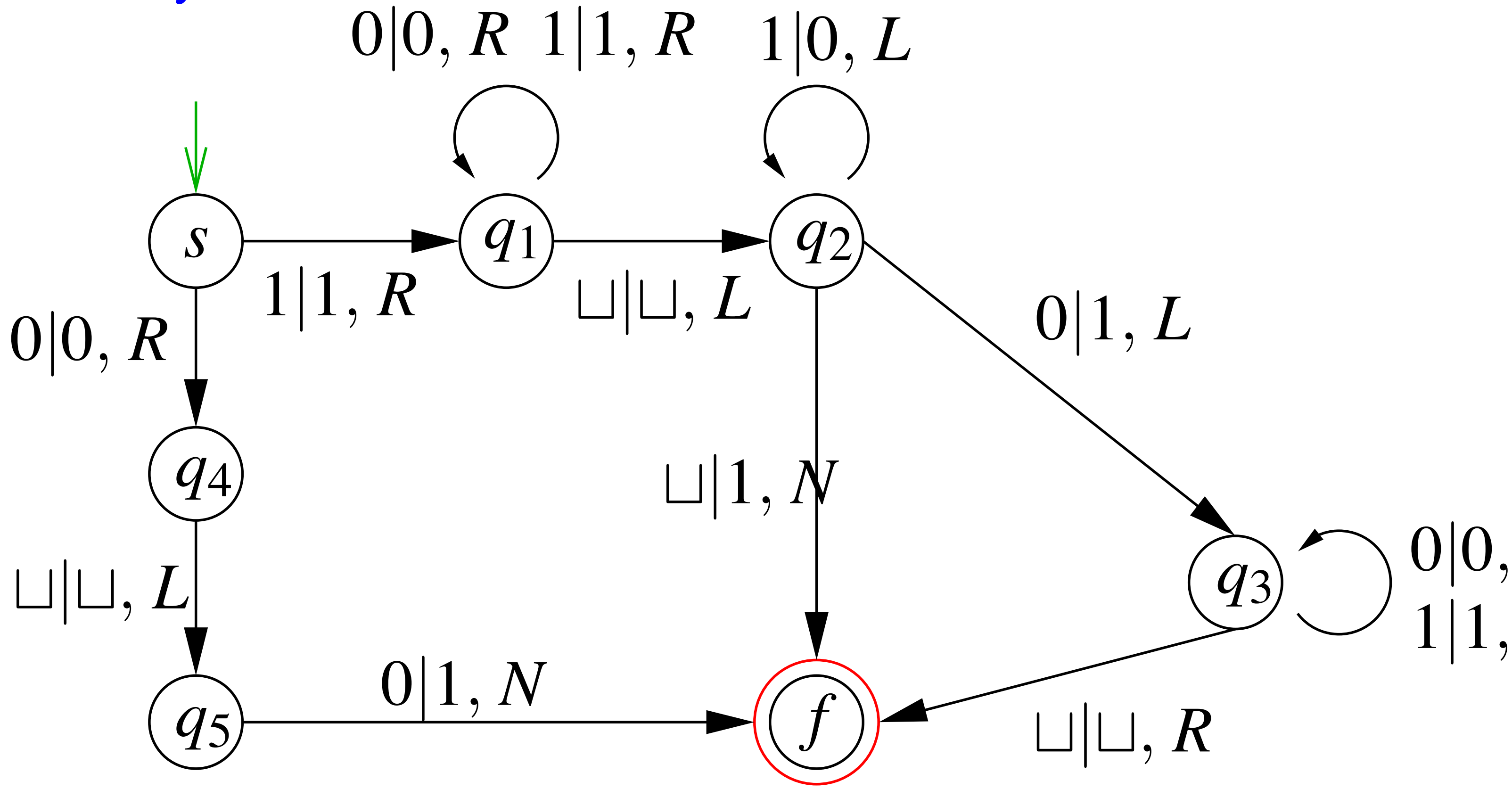
Computability of a function is the main idea.

Instead of acceptor for $L \subseteq \Sigma^*$ consider TM, which computes a „half“ characteristic function

$$\chi_L(w) = \begin{cases} 1 & \text{if } w \in L \\ \perp & \text{otherwise} \end{cases}$$

The semi-decidable languages coincide with the domains of the computable functions.

Example



$$f(w) = \begin{cases} w+1 & \text{if } w \in 0 \cup 1(0 \cup 1)^*, \\ & w \text{ interpreted as binary number} \\ \text{undefined} & \text{otherwise} \end{cases}$$

Remark: Not displayed movement are valid here as infinite loops.

Properties of the semi-decidable sets

Proposition: $A, B \subseteq \Sigma^*$ semi-decidable $\Rightarrow$
$A \cup B$ and $A \cap B$ are semi-decidable.

Let M_1 and M_2 semi-decide A and B

$A \cup B$: **for** $j := 1$ **to** ∞ **do**

  if M_1 accepts w for j steps **then** Accept

  if M_2 accepts w for j steps **then** Accept

$A \cap B$: **for** $j := 1$ **to** ∞ **do**

  if M_1 accepts w in j st. & M_2 accepts w in j st. **then** Accept

Local Variables

Local variable accumulates $x \in A$, ($|A| < \infty$!):

$Q \rightsquigarrow Q \times A$

Example: M is TM, such that memorizes the first symbol of the word and halts if it is not in another place in the word.

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

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

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

Composition

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

Let: $(s)w \vdash^* (r) f_T(w)$ for one $r \in F$ if $f_T(w) \neq \bot$.

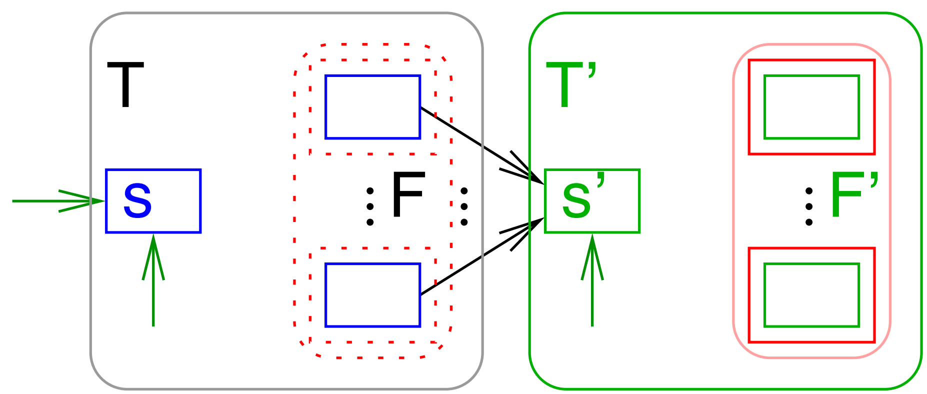
$T' = (Q', \Sigma, \Gamma', \delta', s', F')$.

output: Turing machine $T^\circ = (Q^\circ, \Sigma, \Gamma^\circ, \delta^\circ, s, F')$ for $f_{T'}(f_T(x))$:

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

$$\Gamma^\circ = \Gamma \cup \Gamma'$$

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



a|a,N

If then else

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

Output: Turing machine $T^\circ=(Q^\circ,\Sigma,\Gamma^\circ,\delta^\circ,s,F'\cup F'')$

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

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

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

k **tapes**

k- tapes TM (k- heads): $T = (Q, \Sigma, \Gamma, \delta, s, F)$, where
$\delta : Q \times \Gamma^k \rightarrowtail Q \times \Gamma^k, \{L, R, N\}^k$.
$\delta(q, (a_1, \ldots, a_k)) = (p, (b_1, \ldots, b_k), (C_1, \ldots, C_k))$,
$C_1, \ldots, C_k \in \{L, R, N\}$.
Example: Arithmetical operations of 2 binary numbers,

- replace $a \in \Sigma$ through $(a, 0, \ldots, 0)$ in the input.

Theorem For every TM M with k tapes there is a TM $M'$with one tape which computes the same function. (polynomial).

Theorem For every nondeterministic TM there is a deterministic TM which computes the same function. (exponential)

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

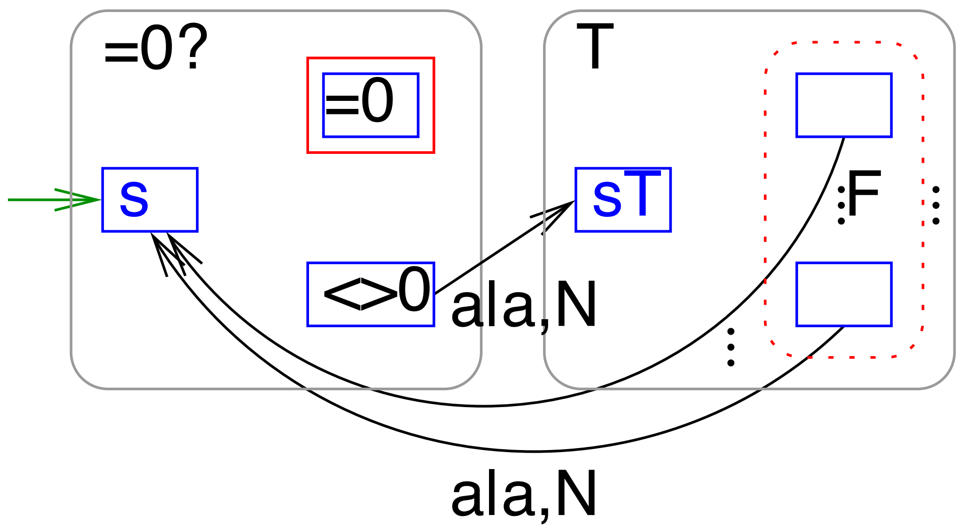
Track i defines a number (unary or binary)

Subprogram: test on track $i = 0$.

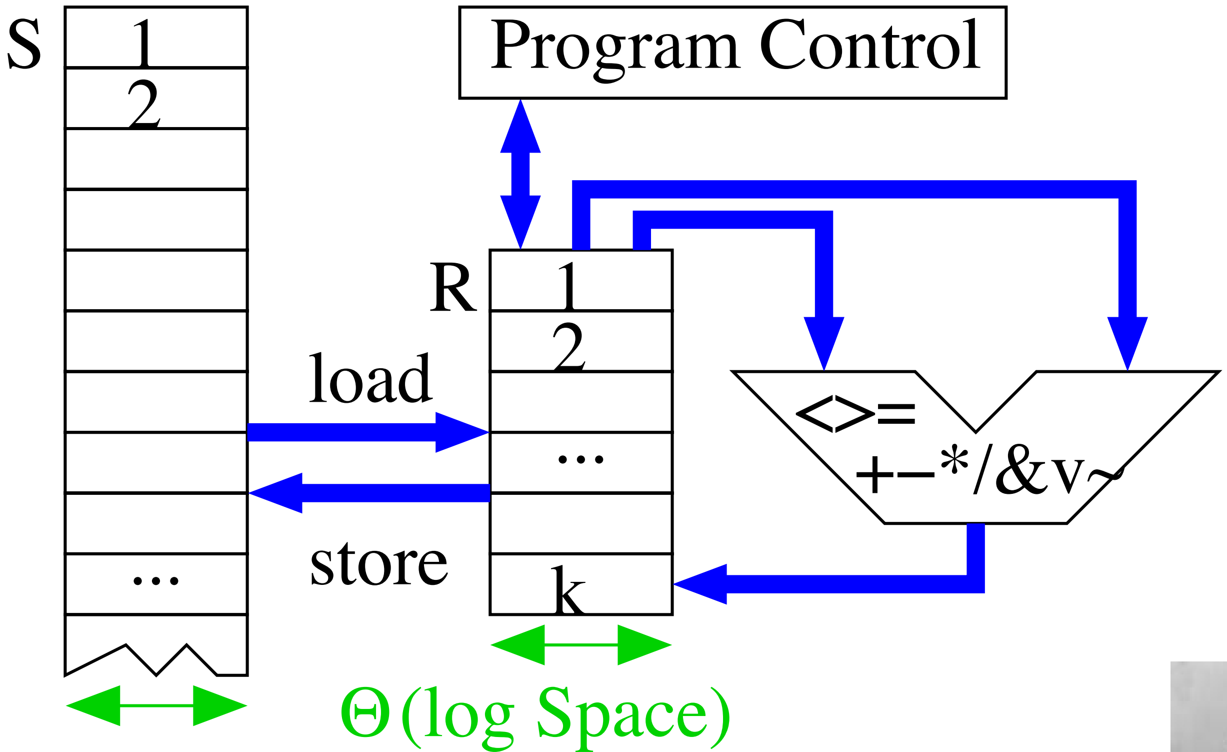
When yes: halt

Leave T moving

back to the start state. (the transition $\delta(f,a) = (s,a,N)$)



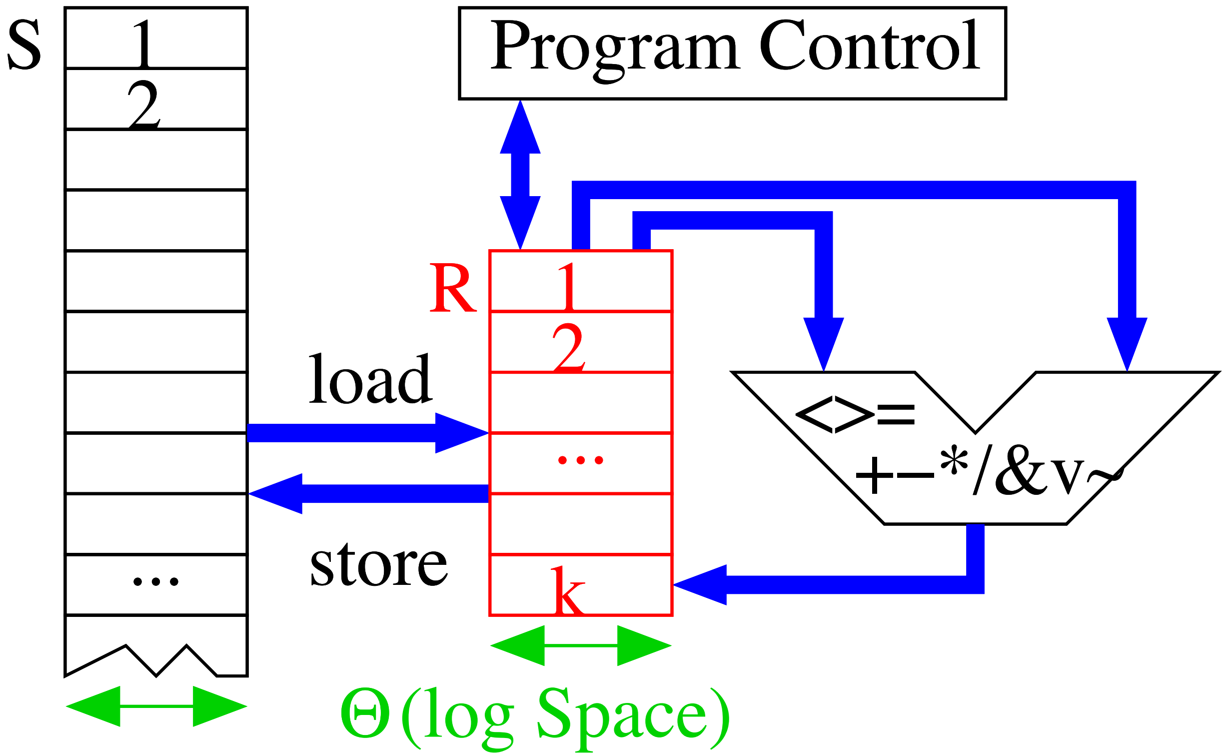
1.2 RAM: Random Access Machine



Modern (RISC) adaptation

of Neumann-models [of Neumann 1945]

Register

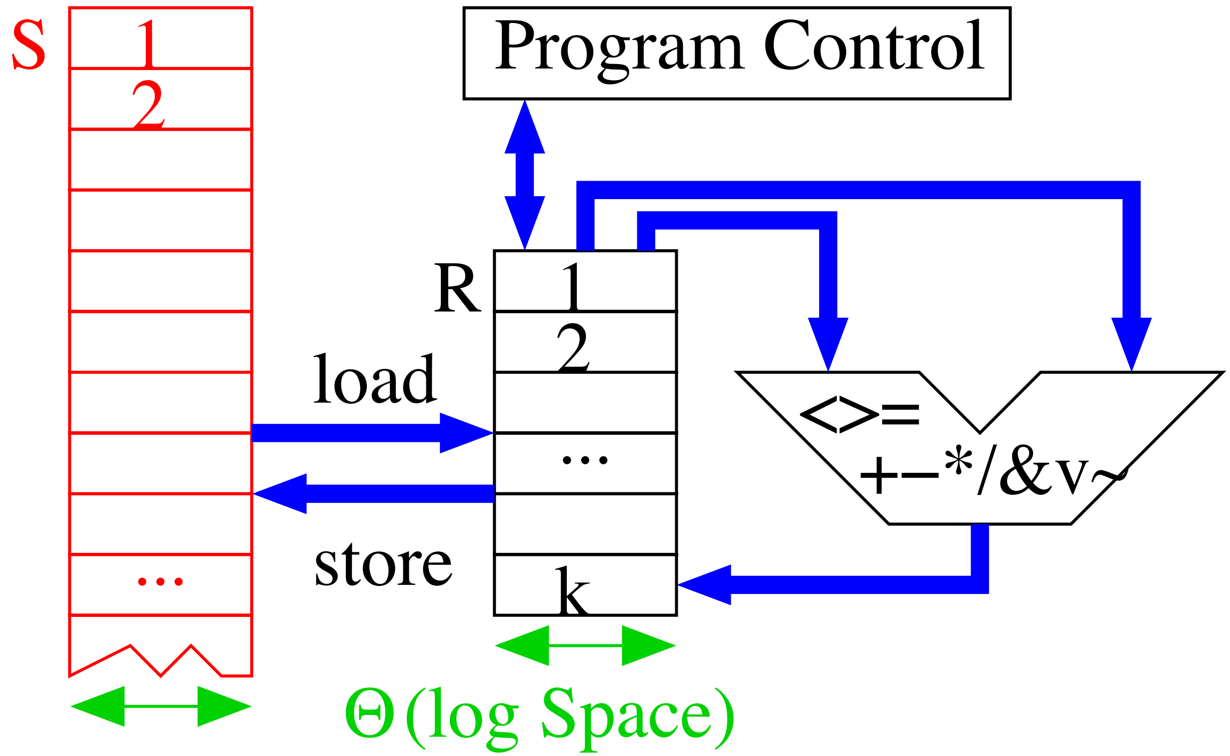


k (any constant) memory

$R_1, \ldots, R_k$ for

(small) integers

The main memory

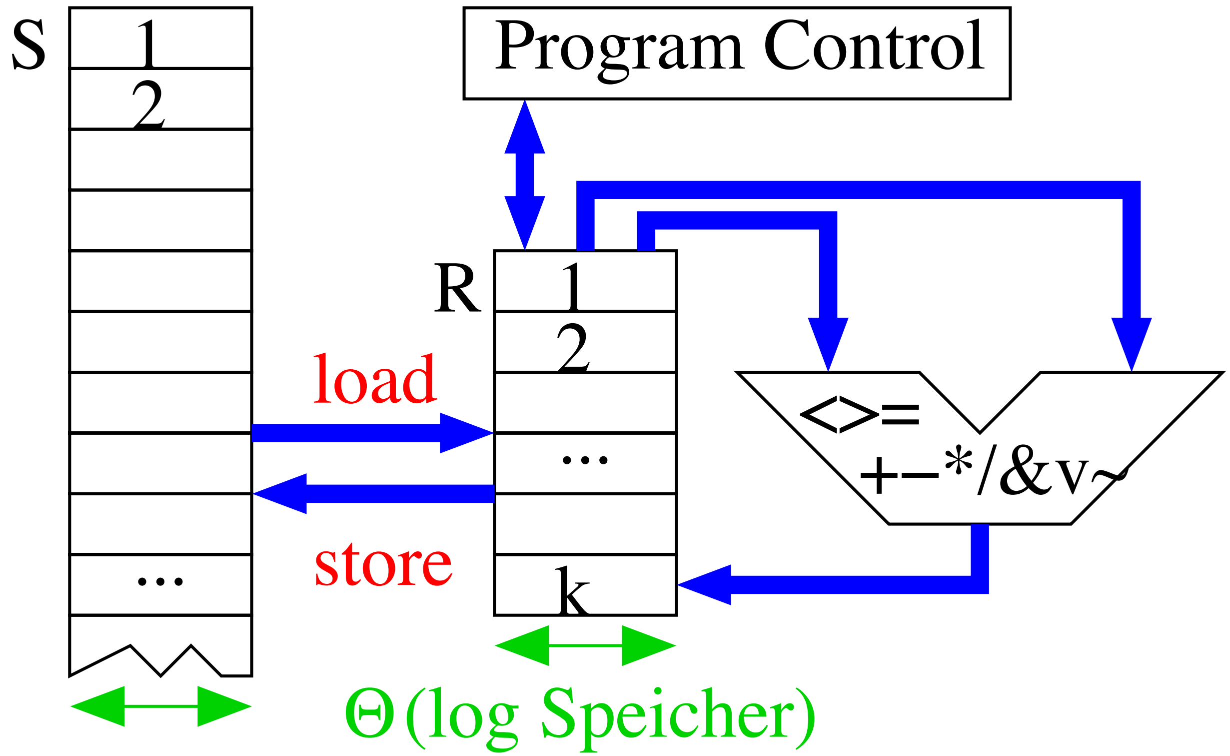


Non bounded supply of memory cells

$S[1], S[2] \ldots$ for

(small) integers

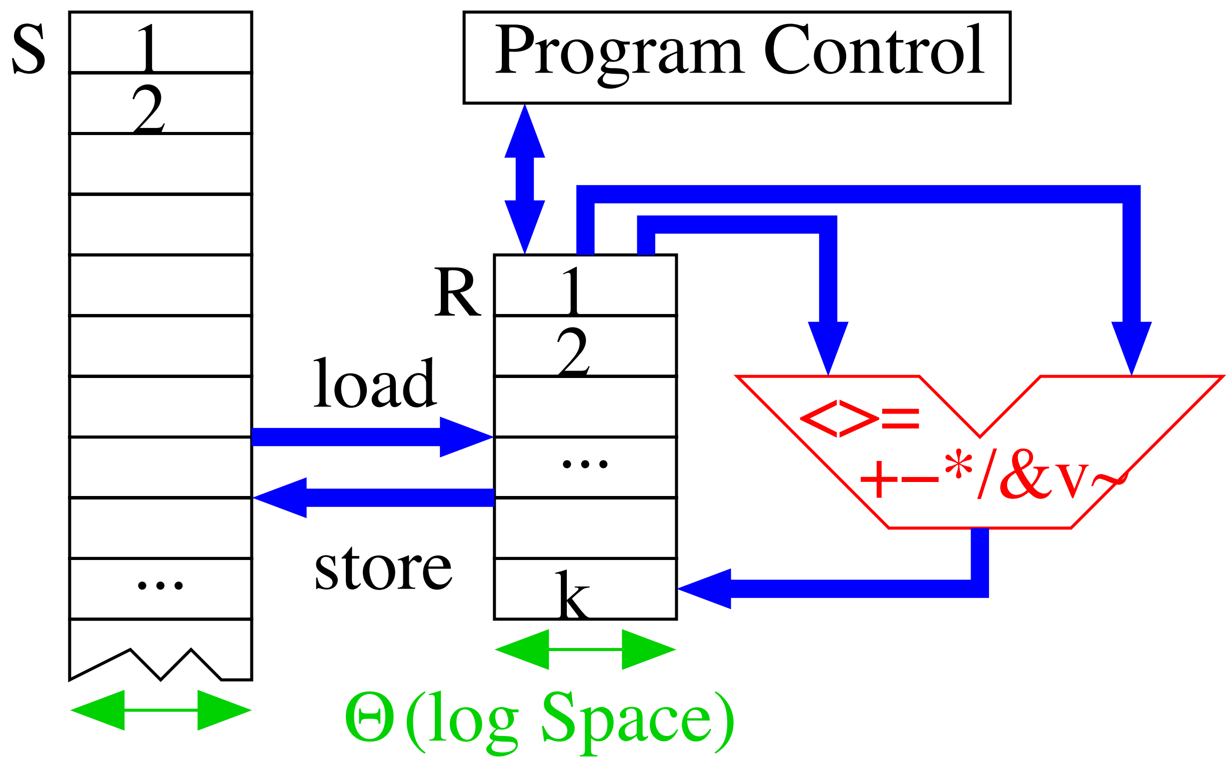
Memory access



$R_i := S[R_j]$ loads the content of the memory cell $S[R_j]$ in Register R_i.

$S[R_j] := R_i$ stores Register R_i in memory cell $S[R_j]$.

Calculation

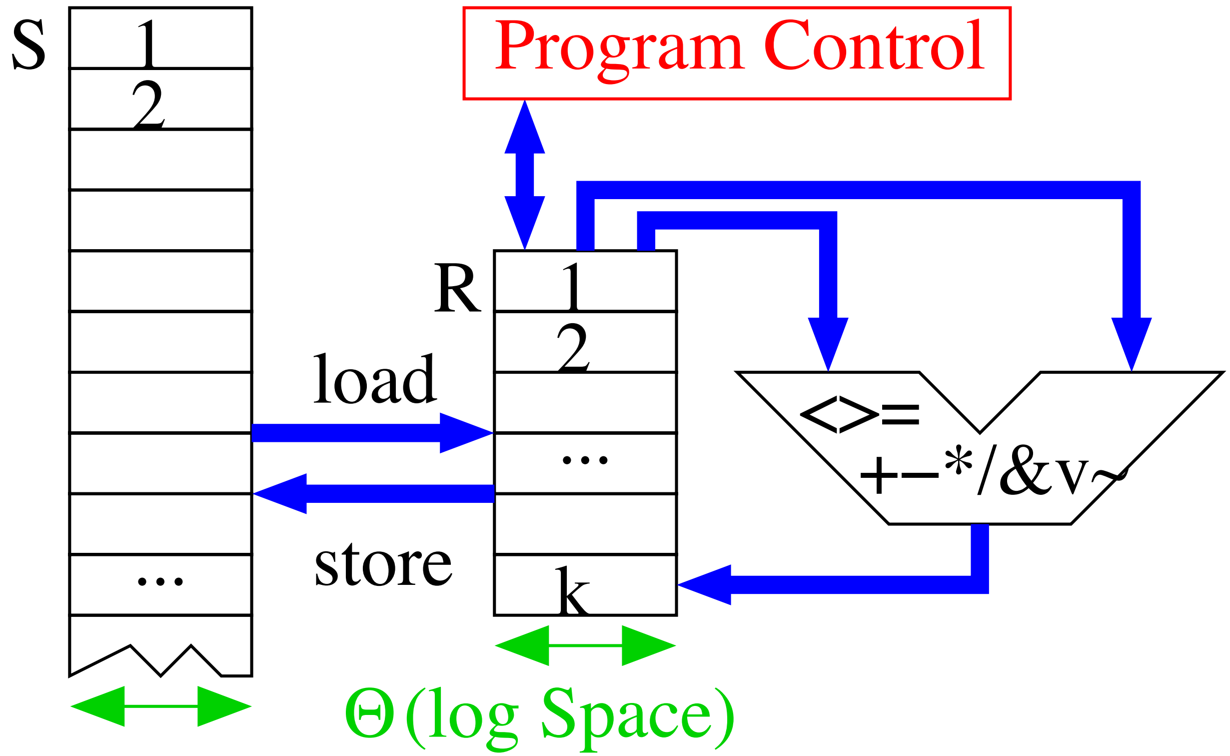


$R_i := R_j \odot R_\ell$ Register arithmetic.

'$\odot$' is a placeholder for a huge number of operations

Arithmetic, Comparison, Logic

Conditional jump



JZ j, R_i Puts the program execution an label j (goto j) if $R_i = 0$

RAM-computability

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

Let M be RAM:

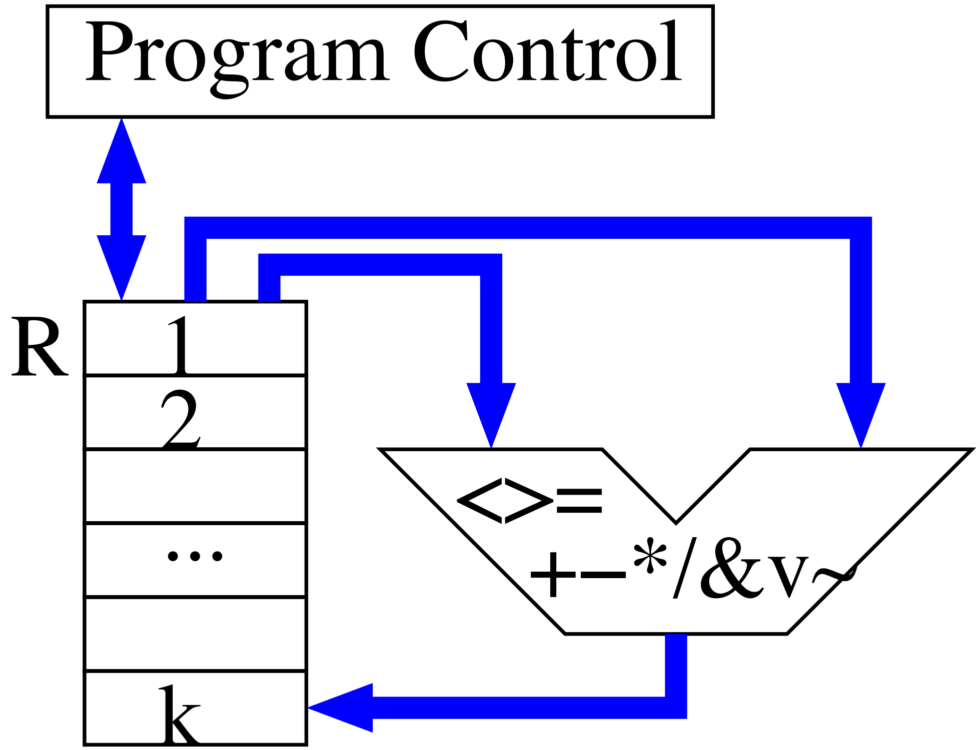
input: $w \in \Sigma^n$ in $S[1]$,. . . ,$S[n]$

output: $f_M(w)$ in $S[1]$,. . . ,$S[|f_M(w)|]$

till HALT- command is executed.

(Unlimited) Register machines

$\approx$ RAM $-$ memory $+$ arbitrary large



$Z(n), S(n), T(m,n), J(m,n,q)$

(Unlimited) Register machines-computability

Configuration: $(q, R_1, \ldots, R_k)$

q is a counter for the program commands

„$\vdash^*$“ we have defined.

$f : \mathbb{N}^{k'} \to \mathbb{N}$, $k' \leq k$ is Register machines computable $\Leftrightarrow$

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

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

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

with PROGRAM$[q]$ =HALT

High level program languages

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

ML, Lisp,. . .

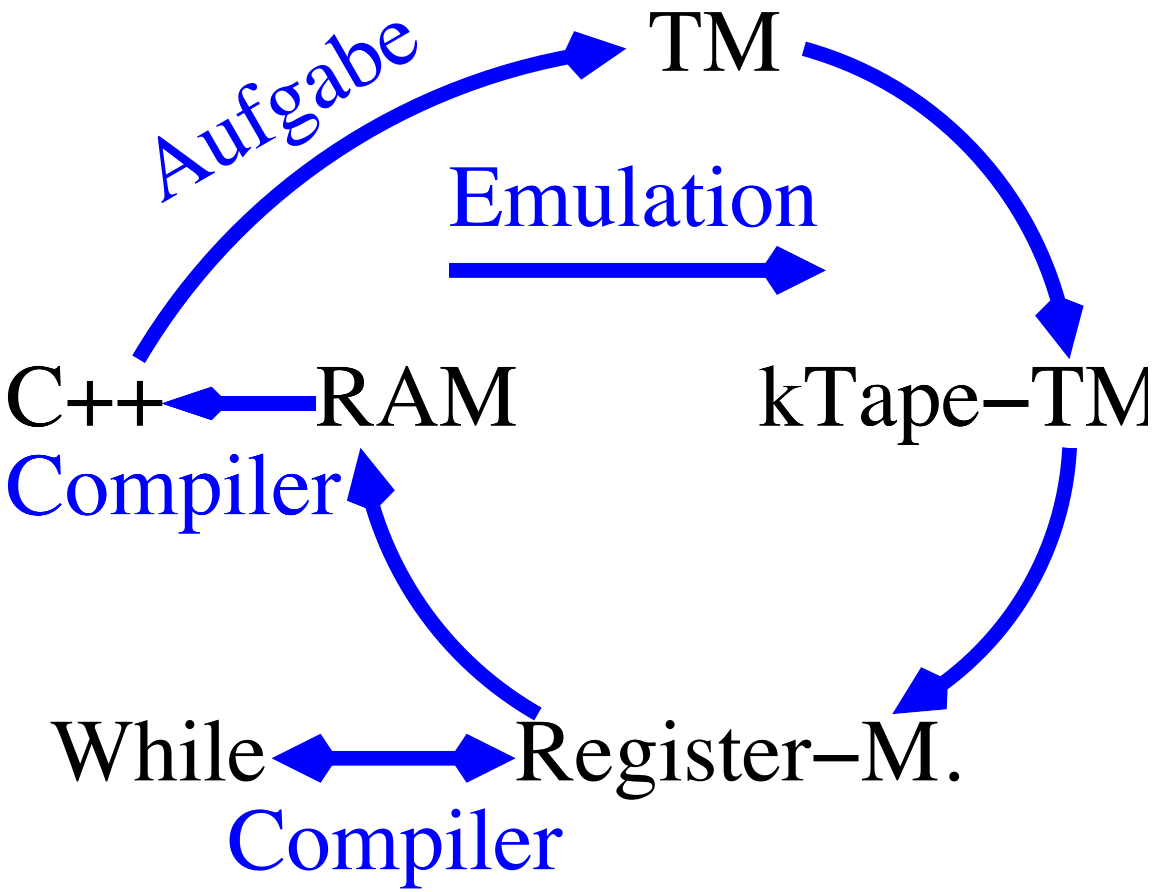
Prolog, Oz,. . .

. . .

are the most popular program models for us.

Compilers translate the programs in RAM Code.

Equivalence of the machine models



Register machine emulates RAM

Idea: an additional register R_S represents the memory:

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

with $b =$ number of RAM bits

$S[i]$ in R_j loading:

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

$S[i] := 0$:

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

R_j in $S[i]$ saving:

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

1.3 Primitive recursive functions

Basic functions

- $O(x) = 0$
- $S(x) = x + 1$
- $I_k^n(x_1, \ldots, x_n) = x_k$, $k \leq n$

Basic operations

- Superposition

  $h(x_1, \ldots, x_n) = f(g_1(x_1, \ldots, x_n), \ldots, g_k(x_1, \ldots, x_n))$

- Primitive recursion

  $h(x_1, \ldots, x_n, 0) = f(x_1, \ldots, x_n)$

  $h(x_1, \ldots, x_n, y + 1) = g(x_1, \ldots, x_n, y, h(x_1, \ldots, x_n, y))$

Primitive recursive functions

A function is primitive recursive if it could be obtained from the basic functions by means of the operations superposition and primitive recursion applied finitely many times.

Examples:

- $x+0=0$ (addition is pr. rec)
  $x+(y+1)=(x+y)+1$
- $x.0=0$ (multiplication is pr rec)
  $x.(y+1)=x.y+x$
- $x^0=1$ (powering is pr. rec)
  $x^{y+1}=x^y.x$

Primitive recursive functions - examples

- $x \dot{-} 1 = 0$ if $x = 0$, $x \dot{-} 1 = x - 1$ if $x > 0$
- $x \dot{-} y = 0$ if $x < y$, $x \dot{-} y = x - y$ if $x \geq y$
- $|x - y| = x \dot{-} y + y \dot{-} x$
- $x \leq y \iff x \dot{-} y = 0$
- $sg(0) = 0$, and $sg(x) = 1$ if $x > 0$
- $\bar{sg}(0) = 1$, and $\bar{sg}(x) = 0$ if $x > 0$

Operations preserving the primitive recursiveness

- if then else

$$h(x) = \begin{cases} f(x) & \text{if } p(x) = 0 \\ g(x) & \text{if } p(x) > 0 \end{cases}.$$

$$h(x) = f(x).\bar{s}g(p(x)) + g(x).sg(p(x)).$$

- bounded sum

$$g(x,y) = \Sigma_{z<y} f(x,z).$$

- bounded minimization

$$h(x,y) = \begin{cases} \mu z_{z<y}[f(x,z) = 0] & \text{if there exists such} \\ y & \text{ow} \end{cases}.$$

Primitive recursive functions - Examples

- $x \mod y$, where $x \mod 0 = x$

  $0 \mod y = 0$

  $$x+1 \mod y = \begin{cases} (x \mod y)+1 & \text{if } (x \mod y)+1 \neq y \\ 0 & \text{ow} \end{cases}.$$

- $x/y, x/0 = 0.$

- $$div(x,y) = \begin{cases} 1 & \text{if } (x \mod y) = 0 \\ 0 & \text{ow} \end{cases}.$$

- $D(x) = \Sigma_{z<x+1} sg(div(x,z))$ the number of factors of x.

Primitive recursive functions - Examples

- $pr(x) = \bar{sg}|D(x) - 2|$- x is a prime number.
- $p(x) = x$th prime number, where $p(0) = 2, p(1) = 3, \ldots$
  $p(0) = 2$
  $p(x+1) = \mu z_{<x!}[z > p(x) \ \& \ pr(z) = 0]$.
- $(x)_y =$ the power of the yth prime number in factoring of x.
  $(x)_y = \mu t_{\leq x}[div(p(y)^{t+1}, x) = 0]$.

Primitive recursive coding

- $\pi(x,y) = 2^x(2y+1) - 1$ pr.rec,
- $L(\pi(x,y)) = x$ and $R(p(x,y)) = y$ decoding functions
  $L(z) = (z+1)_0$
  $R(z) = ((z+1)/2^{(z+1)_0})/2.$
- every natural number is a code of a pair
- the coding is a bijection.

μ-recursive (computable)functions

μ-operation:

$f(x_1,\ldots,x_n) = \mu z[g(x_1,\ldots,x_n,z)=0] \Leftrightarrow$
$(\forall y < z)(g(x_1,\ldots,x_n,y) > 0)\ \&\ g(x_1,\ldots,x_n,z) = 0$

A function is μ-recursive if it could be obtained from the basic functions by means of the operations superposition, primitive recursion and μ-operation applied finitely many times.

Kleene's normal form: f is μ-recursive if there is a primitive rec. functions ρ and L: $f(x) = L(\mu z[\rho(x,n)=0])$.

Example: nowhere defined function $\emptyset(x) = \mu z[S(x)=0]$.

Theorem The class of μ-recursive functions is exactly the class of the computable functions with TM.

1.4 The Ackermann function

[Ackermann 1928, Hermes]

Function $a(x,y)$

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

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

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

Totality of the Ackermann function

Proposition: a is a total, TM-computable function

Proof: Induction on the lexicographical order of (x,y):

Base of induction: $a(0,y) = y+1$

Induction step for $y = 0$:

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

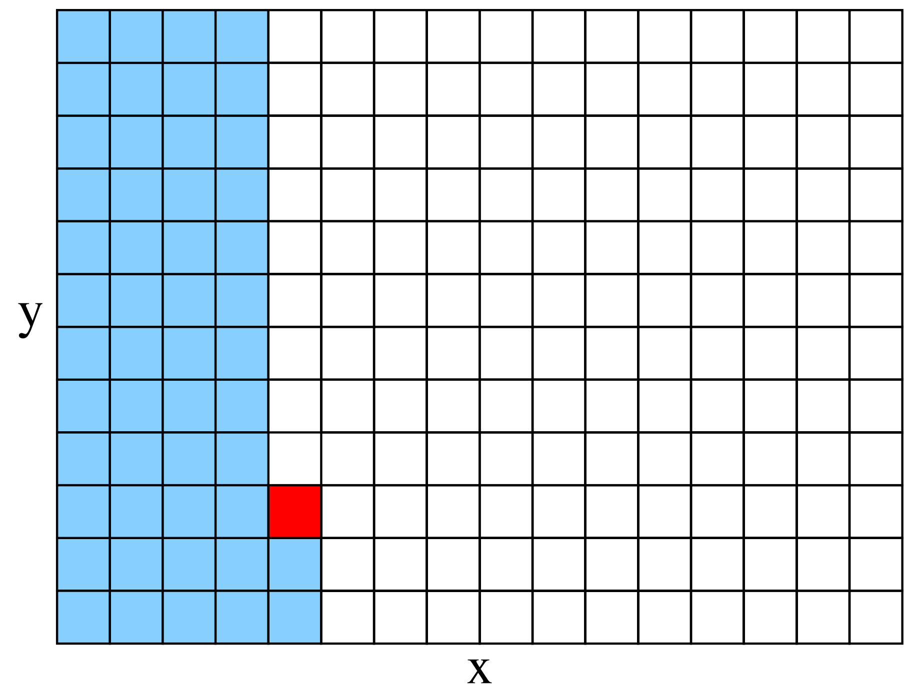
terminates,as $(x-1,1) < (x,0)$

Induction step for $x,y > 0$:

$a(x-1,a(x,y-1))$ terminates as

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

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



The Ackermann function is not primitive recursive

Proof: Assume that a is primitive recursive.

$\longrightarrow a(n,n) = g(n)$ is primitive recursive.

But

$\boxed{\forall \text{primitive recursive } h : \exists k : \forall x_1 \ldots x_n \in \mathbb{N} : h(x_1, \ldots, x_n) < a(k, \max x_1, \ldots, x_n).}$

Then $g(k) < a(k,k)$ A contradiction.

Example

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

Example

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

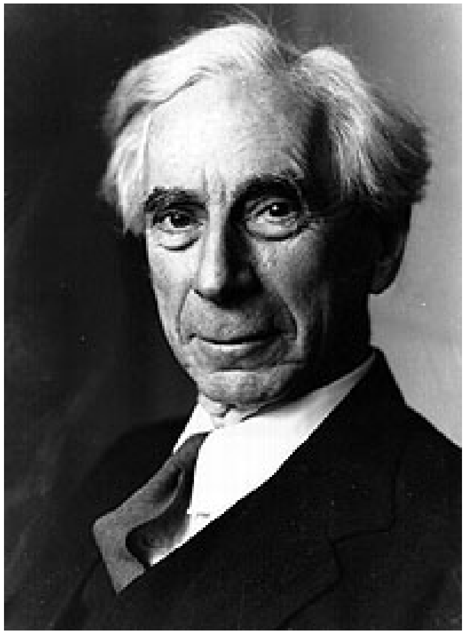
Example

$$
\begin{aligned}
a(3,y) &= 2^{y+3} - 3 \\
a(4,y) &= a(3, a(4, y-1)) = 2^{a(4,y-1)+3} - 3 \\
&= 2^{a(3,a(4,y-2))+3} - 3 = 2^{2^{a(4,y-2)+3}-3+3} - 3 \\
&= 2^{2^{a(3,a(4,y-3))+3}} - 3 = 2^{2^{2^{a(4,y-3)+3}-3+3}} - 3 \\
&= \cdots = 2^{2^{\cdot^{\cdot^{\cdot^{2^{\overbrace{a(4,0)}^{=a(3,1)=2^{1+3}-3}+3}}}}}} - 3 = 2^{\cdot^{\cdot^{\cdot^{2^{16}}}}} - 3 \\
a(4,2) &= 2^{2^{16}} - 3 = 2^{65536} - 3
\end{aligned}
$$

1.5 Halting problem, Undecidability, Reducibility

- Gödel numbering: TMs could processed themselves as an input
- Important example: Universal TM
- Diagonal argument: a undecidable language
- Reductions: it shows that other problems are undecidable.

Paradoxes and Self reference



The barber of a small town
shaves all and only those men
who do not shave themselves.

.

Does the barber shave himself?

Paradoxes and Self reference

Daniel Dösentrieb invented an all-knowing machine.

Yes No

Ones places a yes/no Question and the answer lights up.

Dagobert Duck wants to buy the machine.

But will pay however if only it functions correctly.

It places the Question to the machine:

Will you answer with no?

What happens?

Decidability

$A \subseteq \Sigma^*$ is decidable (computable) if

the characteristic function c_A is computable.

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

Semi-decidability

$A \subseteq \Sigma^*$ is semi-decidable if

the „half“ characteristic function χ_A is computable.

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

Every decidable set is semi-decidable.

Proposition: $A \subseteq \Sigma^*$ decidable $\Leftrightarrow$
A and $\bar{A}$ are both semi-decidable

Proof: Let TM

M_A acceptor for A and

$M_{\bar{A}}$ acceptor for $\bar{A}$

for $s := 1$ **to** ∞ **do**

if M_A halts in s steps **then** Accept

if $M_{\bar{A}}$ halts in s steps **then** Reject

Computably enumerable

$A \subseteq \Sigma^*$ recursively (computably) enumerable if

$A = \emptyset$ or $\exists$total computable function $f : \mathbb{N} \to \Sigma^*$:

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

Proposition: A is computably enumerable $\Leftrightarrow$ A is semi-decidable

Computably enumerable $\longrightarrow$ semi-decidable

Let A is computably enumerable by means of f.

Function $\chi_A(x)$

  for $s := 1$ **to** ∞ **do**

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

Semi-decidable $\longrightarrow$ computably enumerable

- Consider $\pi(k,m) = 2^k(2m+1) - 1$ - a codding function for all pairs of natural numbers.
- Each natural number n is a code of exactly one pair $n = \pi(k,m)$.
- Let $L(\pi(m,k)) = m$ and $R(\pi(m,k)) = k$ be the decoding functions.
- π, L, R are computable functions.
- Consider the sequence of all words in Σ^*:
  $\alpha_0, \alpha_1, \ldots, \alpha_i, \ldots$
  in the following order $|\alpha_i| < |\alpha_{i+1}|$ or $|\alpha_i| = |\alpha_{i+1}|$ and α_i is lexicographically less than α_{i+1}.
- For example: $a, b, aa, ab, ba, bb, \ldots$

Semi-decidable $\longrightarrow$ computably enumerable

Case $A = \emptyset$: trivial.

Otherwise we give one function $f : \mathbb{N} \to \Sigma^*$ with the range A.

$$f(n) = \begin{cases} \alpha_{L(n)} & \text{if } M(\alpha_{L(n)}) \downarrow \text{ for R(n) steps,} \\ a & \text{ow.} \end{cases}$$

Function $f(n)$

a:= some fixed element of A

interpret n as a pair $n = \pi(m,k)$

Consider the word $u = \alpha_m$

if an acceptor M for A accepts u in $\leq k$ steps **then return** u

else return a

Semi-decidable $\longrightarrow$ computably enumerable

- f is total
- f gets only values from A
- $\forall u \in A \exists k : M$ accepts u in k steps
- $f(\pi(m,k)) = \alpha_m$

■

Exercise: Prove that if A is infinite, then A is decidable iff there exists a total computable function $f : \mathbb{N} \to \Sigma^* :$

$A = \{f(0) < f(1) < f(2) < \ldots\}$ in a lexicographical order.

Equivalent statements

- A is computably enumerable
- A is semi-decidable
- $A = L(M)$ for TM M
- χ_A is Turing-, RegM., RAM, ... computable
- A is a domain of one (partial) computable function
- A is a range of a computable function
- $A = \{x \mid \exists n(\rho(x,n) = 0)\}$ for some primitive recursive function ρ

Enumeration of Turing-machines

Consider $T = (Q, \Sigma, \Gamma, \delta, s, F)$. Let:

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

for appropriate constant n

Goödel number $\langle M \rangle$ of Turing machine M

Define the following strings in $\{0,1\}$:

Code $\delta(q,a) = (r,b,d)$ by $0^q\,1\,0^{a+1}\,1\,0^r\,1\,0^{b+1}\,1\,0^d$

where d is the code of the directions: $N = 1, L = 2, R = 3$.

The Turing-machine will be coded by binary numbers:

$$111\text{code}_1 11\text{code}_2 11 \ldots 11\text{code}_z 111,$$

code_i for $i = 1, \ldots, z$: all values of function δ are written in arbitrary order.

Convention:

n is not a Goödel number of a TM,

$\rightarrow n$ describes one TM, which accepts the $\emptyset$

Example

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

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

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

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

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

then $\langle M \rangle$ is:

111010010001010001100010101001000110001001001010001100010001000100100111

Universal Turing machine

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

input: $\langle M \rangle w$

M is the simulated TM, w is the binary coded input.

U simulates M on w.

U accepts $\langle M \rangle w$ if M accepts w.

Universal Turing machine

3 Tapes:

1. $\langle M \rangle$
2. the state q_M of M unary coded
3. the content of the tape w of M

Universal Turing machine

if prefix v of w represents a TM **then** // 111tuple111

move v on the tape $\langle M \rangle$

$q_M := 1$ // the initial state of M

while $q_M \neq 2$ **do** // final state of M

run to the beginning of $\langle M \rangle$

foreach $(q, a, r, b, d) \in \langle M \rangle$ **do** // field by field

if $q = q_M$ **then** // compare with q_M

if input symbol of the tape 3$=a$ **then**

$q_M := r$ // copy on the state tape

put b on the tape 3

the moving on the tape 3 is according to the chosen d

The diagonal language L_d is undecidable

Let M_i is the TM with $\langle M_i \rangle = i$.

Let w_i be the binary representation of i.

L_d:= $\{w_i : M_i$ does not accept $w_i\} = \{w_i \mid M_i(w_i) \uparrow\}$

Proof:

Assume: $L_d = \{w_i : M_i$ does not accept $w_i\}$ is decidable.

$\xrightarrow{\text{Def. „decidable"}} \exists M_i : M_i$ accepts L_d and halts always.

What does M_i do with w_i?

$w_i \in L_d \xrightarrow{\text{Def. } M_i} w_i$ will be accepted. $\xrightarrow{\text{Def. } L_d} w_i \notin L_d$

$w_i \notin L_d \xrightarrow{\text{Def. } M_i} w_i$ will not be accepted. $\xrightarrow{\text{Def. } L_d} w_i \in L_d$

Both lead to a contradiction.

$\bar{L}_d = \{w_i : M_i$ accepts $w_i\}$ is undecidable

Assume: $\bar{L}_d$ is decidable.

$\rightarrow \exists M : M$ accepts $\bar{L}_d$

modify $M \rightsquigarrow M'$ so M' accepts L_d

(Exchange accepts/does not accept for the final state).

A contradiction.

Notice that $\bar{L}_d$ is semi-decidable. Run the universal machine on $\langle M_i \rangle w_i$.

Undecidable problems

Does not exist a program P, such that

$$\text{halts}(\langle P\rangle, X)) = \begin{cases} yes & \text{if } P(X) \text{ halts} \\ no & \text{otherwise} \end{cases}$$

Assume that there is:

D(X)= **if** halts(X,X) **then** loop(X) **else** halt

D($\langle D\rangle$)= **if** halts($\langle D\rangle$,$\langle D\rangle$) **then** loop($\langle D\rangle$) **else** halt

If halts($\langle D\rangle, \langle D\rangle$)=yes, then $\downarrow$D($\langle D\rangle$), but $\uparrow$D($\langle D\rangle$).

If halts($\langle D\rangle, \langle D\rangle$)=no, then $\downarrow$D($\langle D\rangle$), but $\uparrow$D($\langle D\rangle$).

Halting problem

$H := \{w_i v : M_i \text{ halts on } v\}$

Proposition: H is not decidable.

Proof: Assume that H is decidable.

We construct one TM, by which $\bar{L}_d$ will be accepted.

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

$\Leftrightarrow M_i$ accepts w_i.

$\Leftrightarrow w_i w_i \in H$.

This we could do by means of one TM for H and one universal TM.

A contradiction.

Reducibility

Definition Let $L_1, L_2 \subseteq \Sigma^*$. L_1 is reducible to L_2 ($L_1 \leq L_2$) if there is a total computable function f, s.t.

$$w \in L_1 \iff f(w) \in L_2.$$

Lemma Let $L_1 \leq L_2$. If L_1 is not decidable (not semi-decidable) then L_2 is not decidable (not semi-decidable).

Rice theorem

Let $\mathbf{R}$ be the class of all Turing computable functions.

Theorem Let $\mathbf{S}$ be a nontrivial class of Turing computable functions ($S \neq \emptyset, S \neq R$). Then the set

$$C(S) = \{w \mid M_w \text{ computes a function } \in S\}$$

is not decidable.

Proof:

Assume that $C(S)$ is decidable.

Case 1. $\emptyset \notin S$ and $f \in S$. Then there is a Turing machine M_f that computes f.

Let M be a Turing machine and w is a word.

$$T_{M,w}(x) = \begin{cases} M_f(x) & \text{if} \downarrow M(w) \\ \perp & \text{otherwise} \end{cases}$$

Then:

if $\downarrow M(w) \Rightarrow (\forall x) T_{M,w}(x) = f(x)$

if $\uparrow M(w) \Rightarrow T_{M,w}(x) = \emptyset$.

$$T_{M,w} \in S \Leftrightarrow \downarrow M(w).$$

$$\langle T_{M,w} \rangle \in C(S) \Leftrightarrow \langle M \rangle w \in H.$$

A contradiction.

Case 2. $\emptyset \in S$. Consider: $R \setminus S$ is non trivial.

Then $C(\bar{S})$ is undecidable, and hence $C(S)$ is undecidable.