Tw (O podstawianiu) Niech $f \in C_k$ a $g_1, \ldots, g_k \in C_n$. Wtedy funkcja $h(x) \simeq f(g_1(x), \ldots, g_k(x)) \in C_n$.

Dowód: Niech $f \in C_k$ będzie obliczana przez program F, a $g_1, \ldots, g_k \in Cn$ będą obliczane przez programy $G_1, \ldots, G_k$ odpowiednio. Wtedy funkcja $h(x)$ jest obliczana przez podany program H. Niech
$m = max\{n, k, \rho(F), \rho(G1), \ldots, \rho(Gk)\}$.
$T(1, m+1)$
$\ldots$
$T(n, m+n)$
$G_1[m+1, \ldots, m+n \to m+n+1]$
$\ldots$
$G_k[m+1, \ldots, m+n \to m+n+k]$
$F[m+n+1, \ldots, m+n+k \to 0]$

Rekursja to sposób zadania kolejnej wartości funkcji poprzez wartości wcześniej obliczone.
Tw (O op rekursji) Niech $f(x) \in C_n$ i $g(x, y, z) \in C_{n+2}$. Wtedy funkcja $h(x, y) : N^{n+1} \to N$ określona układem $h(x, 0) \simeq f(x)$ $h(x, y+1) \simeq g(x, y, h(x, y))$ jest obliczalna $(h \in C_{n+1})$.

Dowód: Niech $f(x) \in C_n$ i $g(x, y, z) \in C_{n+2}$ będą obliczane przez programy F i G odpowiednio. Wtedy funkcja $h(x, y) : N^{n+1} \to N$ jest obliczana przez podany program H. Niech
$m = max(n+2, \rho(F), \rho(G))$. Używać będziemy rejestrów $1, 2, \ldots, m+n+1, m+n+2, m+n+3$ i oznaczenia $t = m+n$
$T(1, m+1)$
$\ldots$
$T(n+1, m+n+1)$
$F[1, 2, \ldots, n \to t+3]$
$q : I(t+2, t+1, p)$
$G(m+1, \ldots, m+n, t+2, t+3 \to t+3]$
$S(t+2)$
$I(1, 1, q)$
$p : T(t+3, 0)$

Minimalizacja
Dla $f(x, y) : N^{n+1} \to N$ połóżmy: $g(x) \simeq \mu y(f(x, y) = 0)$ najmniejsze y takie, że $f(x, z)$ jest określona dla wszystkich $z \leqslant y$ i $f(x, y) = 0$; gdy takie y nie istnieje wartość funkcji pozostaje nieokreślona.

Tw (O op minimalizacji) Jeśli $f \in C_{n+1}$, to $g \in Cn$.
Dowód: Załóżmy, że f jest obliczana przez program F. Wtedy podany program G oblicza funkcję g. Niech $m = max(n+1, \rho(F))$.
$T(1, m+1)$
$\ldots$
$T(n, m+n)$
$p : F[m+1, \ldots, m+n, m+n+1 \to 1]$
$I(1, m+n+2, q)$
$S(m+n+1)$
$I(1, 1, p)$
$q : T(m+n+1, 0)$

Zbiór **R** funkcji częściowo rekurencyjnych definiujemy jako najmniejszy zbiór funkcji rzutowania, który zawiera funkcje bazwoe: zerową, nastepnika i rzuty oraz jest domknięty na operatory podstawiania, rekursji i minimalizacji

Zbiór **PR** oznacz zbiór funkcji pr. rek., które powstają z funkcji bazowych przy uzyciu operatorów podstawiania i rekursji.

MT $M = \langle S, Q, P, q_0, F\rangle$ jest det, jeśli dla każdej pary (stan,litera) (konfiguracji zawierającej takie podsłowo) istnieje co najwyżej jedna instrukcja, którą można do niej zastosować.

Niech M będzie deterministyczną MT zatrzymującą się na wszystkich wejściach. **Czasem działania** lub złożonością czasową M jest funkcja $f : N \to N$, gdzie $f(n)$ jest maksymalną ilością kroków obliczeń M wykonywanych przez nią na każdym wejściu długości n. Mówimy też, że M działa w czasie $f(n)$.

Niech M będzie MT zatrzymującą się na wszystkich wejściach. **Czasem działania** lub złożonością czasową M jest funkcja $f : N \to N$, gdzie $f(n)$ jest maksymalną ilością kroków obliczeń M wykonywanych przez nią na każdym wejściu długości n **w czasie każdego przebiegu obliczeń**. Mówimy też, że M działa w czasie $f(n)$.

Dla **nd MT M** o czasowej złożoności obliczeniowej $f(n)$ takiej, że $n \leqslant f(n)$ dla wszystkich $n \in N$, istnieje det trójtaśmowa MT M1 rozpoznająca ten sam język, której czasowa złożoność obliczeniowa jest rzędu: $O(c^{f(n)})$, dla pewnej liczby $c > 1$.

Idea konstrukcji maszyny M1. Załóżmy, że M w każdej konfiguracji może wykonać co najwyżej dwie instrukcje. Ponieważ jest nd w pewnej konfiguracji ma faktycznie wybór pomiędzy dwiema instrukcjami. Dokonajmy ponumerowania liczbami 0 lub 1 tych instrukcji dla takich konfiguracji. Każdy ciąg obliczeń M na wejściu długości n można wtedy opisać ciągiem zer i jedynek długości co na jwyżej $f(n)$, odpowiadającym wyborowi kolejnej instrukcji. Ciągów takich jest $2^{f(n)+1} - 1$.
Maszynę det M1 konstruuje się tak, aby dla wejścia długości n umieszczanego na pierwszej taśmie wyznaczała na drugiej taśmie kolejny ciąg w porządku leks, a następnie realizowała obliczenia M opisane przez wygenerowany ciąg na taśmie trzeciej. Każde z $O(2^{f(n)})$ obliczeń wymaga: **1**. kopiowania wejścia z taśmy 1. na 3. (ponieważ $n \leqslant O(f(n))$ kroków; **2**. generowania następnego ciągu w $O(f(n))$ krokach; **3**. wykonania obliczeń na taśmie trzeciej w $O(f(n))$ krokach. co razem daje $O(f(n)) \cdot O(2^{f(n)}) = O(4^{f(n)})$ kroków.

Dla k-taśmowej MT M $(k \geqslant 2)$ istnieje 1-taśmowa MT M, która rozpoznaje ten sam język. Jeśli M ma złożoność czasową $O(f(n))$, to M ma złożoność czasową $O(f(n) * f(n))$. Szkic dowodu: Dla k-taśmowej maszyny M jej konfiguracja opisana jest przez zawartość każdej z taśm; czyli ciąg słów $v_i q w_i, i = 1 \ldots k$ zamieniamy na słowo $\#v_1 q w_1 \# v_2 q w_2 \# \ldots v_k q w_k \#$. Konstrukcja M polega na zamianie każdej instrukcji M na ciąg instrukcji, które symulują jej realizację na opisanym słowie. Każde słowo powstające w czasie symulacji obliczeń M na słowie długości n jest długości $O(f(n))$, a zmiana wymaga po prostu jego przeczytania.

Niech $\beta : N \to \Pi$ będzie dowolną numeracją zbioru RAM-programów (czyli funkcją "na"). Mówimy, że β jest efektywna, jeśli istnieją obliczalne funkcje $d : N \to N, kod, A : N^2 \to N$ takie, że dla wszelkich $x \in N$ i $\beta(x) = \langle i0, ..., in\rangle$ mamy: $d(x) = n$ $\forall_{0 \leqslant r \leqslant n} i_r = \langle kod(x, r), A(x, r)\rangle$. Funkcja d określa ilość instrukcji, natomiast kod, A określają kody oraz adresy komórek, którymi one manipulują. $I = \{0, 1, 2, 3\} \times N$ $x \in N, x = 4a + k$ Funkcja $\alpha : N \to I, \alpha(x) = \langle k, a\rangle$ jest bijektywną numeracją zbioru instrukcji. Funkcja: $\mu(x) = \langle \alpha(a_0), \ldots, \alpha(a_n)\rangle$ gdzie $\tau(a_0, \ldots, a_n) = x$. Wyznaczenie przedstawienia x $l(x) = \mu(y \leqslant x)(x < 2^{y+1})$, c - ciąg dzieleń, b - ciąg reszt $l_1(x) = \sum_{y \leqslant l(x+1)} sg(b(y, x+1))$ - ilość '1' w rozw. bin. b_1 - pozycje kolejnych jedynek, $a_1(i+1, x) = b_1(i+1, x) - b(i, x) - 1$ - odpowiednie współczynniki takie, że: $x = \sum_{i=0}^{l_1(x)} 2^{\sum_{j=0}^{i} a_1(j,x)} - 1$ Należy zauważyć, że: $d(x) = l_1(x), kod(x, r) = rm(4, a_1(r, x)), A(x, r) = qt(4, a_1(r, x))$

Tw (O parametryzacji, s-m-n) Dla dowolnych liczb naturalnych $n, m \geqslant 1$ istnieje totalna i obliczalna funkcja $s_n^m : N^{1+m} \to N$ taka, że $\forall_{e \in N, a \in N^m} \varphi_e^{(m+n)}(a, -) \simeq \varphi_{s_n^m(e,a)}^{(n)}(-)$
Szkic dowodu: Dla prostoty założymy, że $n = m = 1$ i będziemy swobodnie używać tych oznaczeń. Niech $e, a \in N$. Przez $\Pi_{e,a}$ oznaczymy program powstały z P_e poprzez modyfikacje: 1. na po czątku umieszczamy ciąg instrukcji T(1,2), Z(1), S(1), ..., S(1); przy czy instrukcja S(1) występuje a-razy; 2. potem następuje tekst programu P_e ze zmodyfikowanymi instrukcjami warunkowymi. Z konstrukcji wynika, że $\varphi_e^{(2)}(a, y) \simeq \varphi_{\Pi_{e,a}}^{(1)}(y)$ Pozostaje zauważyć, że $\mu(m) = \Pi_{e,a} \leftrightarrow s_1^1(e, a) = m$.

Funkcją uniwersalną d la n-argumentowych funkcji RAM-obliczalnych nazywamy funkcję $\Psi_U^{(n)} : N^{n+1} \to N$ taką, że: $\forall_{e \in N, x \in N^n} \Psi_U^{(n)}(e, x) \simeq \varphi^(n)_e(x)$

Każda funkcja uniwersalna jest częściowo-rekurencyjna, a więc i RAM-obliczalna.
Dowód: Niech e będzie dowolną liczbą naturalną. Niech $\sigma_n(e, x, k)$ będzie kodem konfiguracji uzyskanej po k krokach obliczeń P_e na x. Przyjmować będziemy, że konfiguracją następną po końcowej jest ona sama. Wtedy $(\sigma_n(e, x, k))_0$ jest zawartością licznika rozkazów, natomiast $(\sigma_n(e, x, k))_1$ zawartością rejestru zerowego po k krokach obliczeń P_e. Inaczej σ_n zadana jest poprzez schemat rekursji: $f(x) = cd(cod_n(x))$ $g(x, y, z) = C_{(z)_0 < d(e)+1} \bar{\delta}(i_{(z)_0}, z) + C_{(z)_0 \geqslant d(e)+1} z$ Wtedy funkcje: $c_n(e, x, k) = (\sigma_n(e, x, k))_1, j_n(e, x, k) = (\sigma_n(e, x, k))_0$ są prymitywnie rekurencyjne i opisują zawartość rejestru 0 (c) i licznik rozkazów (j) po k krokach obliczeń P_e . Wtedy $\Psi_U^{(n)}(e, x) \simeq c_n(e, x, \mu k(j_n(e, x, k) \geqslant d(e) + 1))$ co oznacza, że funkcja uniwersalna jest częściowo rekurencyjna.

Tw (Rice) Niech $\mathcal{B}$ będzie właściwym i niepustym podzbiorem C_1 . Wtedy problem "$\varphi_x \in \mathcal{B}$" jest nierozstrzygalny; bardziej formalnie zbiór $B = \{x \in N; \varphi_x \in \mathcal{B}\}$ nie jest rekurencyjny.
Ponieważ zbiory N i $\emptyset$ są rekurencyjne otrzymujemy inne sformułowanie tw: Problem "$\varphi_x \in \mathcal{B}$" jest rozstrzygalny wtw, gdy $B = \emptyset$ lub $B = C_1$. Dowód: Niech $f_\emptyset$ będzie funkcją o pustej dziedzine. Funkcja ta jest obliczalna. Załóżmy, że funkcja ta **nie** należy do $\mathcal{B}$. Jeśli tak jest, to prowadzimy dowód dla zbioru $C_1 \setminus \mathcal{B}$ i korzystamy z faktu, że problem "$\varphi_x \in \mathcal{B}$" jest rozstrzygalny wtw gdy problem "$\varphi_x \notin \mathcal{B}$" jest rozstrzygalny. Niech $f \in \mathcal{B}$ będzie ustaloną funkcją. Rozważmy obliczalną funkcję: $g(x, y) = \begin{cases} f(y) & x \in W_x \\ \infty & x \notin W_x \end{cases}$ Możemy do niej zastosować tw. o parametryzacji, niech k będzie totalna i obliczalna: $\forall_{x,y \in N} g(x, y) = \varphi_{k(x)}(y)$ Zależnie of tego czy $x \in W_x$ mamy: $\forall_{x \in N} x \in W_x \leftrightarrow \varphi_k(x) \in \mathcal{B}$ Więc "$\varphi(x) \in \mathcal{B}$" nie może być rozstrzygalny.

Tw (R-S) Niech $\mathcal{A} \in C_1$ będzie takim zbiorem funkcji RAM-obliczalnych, że zbiór indeksów $A = \{x \in N : \varphi_x \in \mathcal{A}\}$ jest r. p. Wtedy dla dowolnej funkcji $f \in C_1$ następujące warunki są równoważne: 1. $f \in \mathcal{A}$ 2. Istnieje skończona funkcja $\theta \subseteq f$ taka, że $\theta \subseteq \mathcal{A}$.
Dowód: (1) pociąga (2): nie wprost. Niech $f \in \mathcal{A}$ będzie taką funkcją, że żadna skończona funkcja $\theta \subseteq f$ nie należy do $\mathcal{A}$. Niech $P = P_e$ będzie programem obliczającym $cc_K, K = x \in N; x \in W_x$. Rozważmy obliczalną funkcję: $g(z, t) = \begin{cases} f(t) & \neg H(e, z, t) \\ \infty & H(e, z, t) \end{cases}$ Zastosujmy tw o parametryzacji, $s \in C_1$, totalna, obliczalna i taka, że: $g(z, t) = \varphi_{s(z)}(t)$ Z określenia $\varphi_{s(z)} \subseteq f$, ale $z \in K$ pociąga, że $\varphi_{s(z)}$ jest skończona, więc $\notin \mathcal{A}$ podobnie, gdy $z \notin K$ Otrzymujemy: $\forall_z z \in \notin K \leftrightarrow s(z) \in A$, więc pierwszy nie jest r.e. co daje sprzeczność z założeniem. (2) pociąga (1): nie wprost. Przypuśćmy, że istnieje obliczalna funkcja $f \notin \mathcal{A}$ dla której $\theta \in \mathcal{A}$ dla pewnej skończonej funkcji $\theta \subseteq f$. Postąpimy podobnie. Niech $g(z, t) = \begin{cases} f(t) & t \in Dom\theta z \in K \\ \infty & w.p.p. \end{cases}$ Najpierw pokażemy, że g obliczalna. $Dom\theta$ jest skończony, więc rekurencyjny. Wtedy zbiór $B = \{(t, z)t \in Dom\theta \in K\}$ jest r.e. Na koniec zauważmy: $g(z, t) = f(t) \cdot cc_B(t, z)$. Zastosujmy s-m-n tw. s totalna, obliczalna i taka, że: $g(z, t) = \varphi_{s(z)}(t)$ dla wszystkich z, t. Wtedy w zależności od tego czy $z \notin K$ ponownie otrzymujemy: $\forall_z \in z \notin K \leftrightarrow s(z) \in A$.