μ-Rekursive Funktionen

- Wir hatten bisher den **beschränkten μ-Operator** definiert und gezeigt, daß dieser mittels primitiv rekursiver Funktionen simulierbar ist.
- Nun definieren wir den **unbeschränkten μ-Operator**:
  - Dieser Operator führt zu einer *echten* Erweiterung von REK_{pr}.
  - Im speziellen korrespondiert der unbeschränkte μ-Operator zum WHILE-Befehl:
    - Der WHILE-Befehl wiederholt eine Schleife nicht eine festgelegte Anzahl m von Durchläufen, sondern so lange, bis eine Bedingung erfüllt ist.
    - Der unbeschränkte μ-Operator sucht alle Argumente i bis zum kleinsten, der eine Bedingung erfüllt.

$\Rightarrow$ Wir werden zeigen:

- Die Klasse der μ-rekursiven Funktionen ist identisch mit $WHILE$.

Definition: Sei $g : \mathbb{N}^{k+1} \to \mathbb{N}$ eine partielle Funktion.

Dann entsteht die partielle Funktion $f : \mathbb{N}^k \to \mathbb{N}$ aus g durch Anwendung des **μ-Operators**, falls gilt:

$$f(\underline{n}) = \mu i(g(\underline{n}, i) = 0) := \begin{cases} i_0, & \text{falls } \alpha_{i_0} \\ \text{undefiniert, sonst} & \end{cases}$$

wo

(α_{i_0}) $g(\underline{n}, i_0) = 0 \wedge \forall 0 \leq j < i_0$ gilt:
$g(\underline{n}, j)$ ist definiert und $g(\underline{n}, j) \neq 0$.

Wir sagen:

- f entsteht aus g durch Anwendung des **μ-Operators im Normalfall** :$\Longleftrightarrow$
  - g ist total,
  - $f(\underline{n}) = \mu i(g(\underline{n}, i) = 0)$, und
  - $\forall \underline{n}\ \exists j \in \mathbb{N}$ s.d. $g(\underline{n}, j) = 0$.

Definition:

1. Die Klasse REK_μ^{part} der **partiellen μ-rekursiven Funktionen** ist die kleinste Klasse, die
   - die Klasse der primitiv rekursiven Funktionen enthält und
   - die abgeschlossen ist bezüglich
     - simultanes Einsetzen,
     - primitive Rekursion und
     - die Anwendung des μ-Operators.
2. Die Klasse REK_μ der **μ-rekursiven Funktionen** ist analog zu REK_μ^{part} definiert, nur gilt hier:
   - REK_μ ist abgeschlossen bezüglich der Anwendung des μ-Operators im Normalfall (anstatt dem allgemeinen Fall).

Bemerkung:

- Das simulatane Einsetzen für partielle Funktionen ist analog definiert wie für totale Funktionen.

  Es gilt:
  - $f(g(\underline{n}))$ ist undefiniert falls $g(\underline{n})$ undefiniert ist.
- Die primitive Rekursion für partielle Funktionen ist ebenfalls analog dem totalen Fall definiert.

  Hier gilt:
  - für $f(\underline{n}) = g(\underline{n})$ und $g(\underline{n})$ undefiniert, folgt auch daß $f(\underline{n})$ undefiniert ist.

Satz: Es gilt:

1. $REK_\mu \subseteq WHILE$; und
2. $REK_\mu^{part} \subseteq WHILE^{part}$.

D.h., jede totale (bzw. partielle) μ-rekursive Funktion ist eine totale (bzw. partielle) WHILE-berechenbare Funktion.

Beweis: Es gilt:

- jede primitiv rekursive Funktion ist WHILE-berechenbar (da $REK_{pr} = LOOP$ und $LOOP \subseteq WHILE$).
- Weiters ist simultanes Einsetzen und primitive Rekursion angewendet auf Funktionen aus REK_μ WHILE-berechenbar.
  - Dies folgt in analoger Weise wie die Simulation dieser Konstrukte mittels LOOP.

Wir zeigen:

- Der μ-Operator im Normalfall kann mittels WHILE-Befehlen simuliert werden.

Sei P ein WHILE-Programm, das eine Funktion g berechnet, und betrachte $f(\underline{n}) = \mu i(g(\underline{n}, i) = 0)$.

Angenommen P verwende nur Register aus $\{x_1, \ldots, x_\ell\}$.

Das folgende WHILE-Programm simuliert f:

% setze $x_{\ell+1} = g(\underline{n}, 0)$, *mit* $\underline{n} = (n_1, \ldots, n_k)$

$x_{k+1} := 0$; P; $x_{\ell+1} := x_{k+2}$; $x_{k+2} := 0$;

WHILE $x_{\ell+1} \neq 0$ DO

$x_{k+1} := x_{k+1} + 1$;

% setze $x_{\ell+1} = g(\underline{n}, x_{k+1})$

P; $x_{\ell+1} := x_{k+2}$; $x_{k+2} := 0$

END

Dies beweist $REK_\mu \subseteq WHILE$.

Falls f eine partielle Funktion ist, dann gilt:

$\mu i(g(\underline{n}, i) = 0)$ ergibt einen Wert $i_0 \iff$ $g(\underline{n}, i_0) = 0$ und $\forall j < i_0$ ist $g(\underline{n}, j)$ definiert und ungleich 0.

Genau in diesem Fall terminiert auch das obige WHILE-Programm.

$\Longrightarrow REK_\mu^{part} \subseteq WHILE^{part}$. ■

Relationen zu Turing-Maschinen

Bisher wissen wir:

- $REK_\mu \subseteq WHILE$, $REK_\mu^{part} \subseteq WHILE^{part}$,
- $WHILE = GOTO$, $WHILE^{part} = GOTO^{part}$
- $GOTO \subseteq TM$, $GOTO^{part} \subseteq TM^{part}$.

$\Longrightarrow REK_\mu \subseteq TM$ und $REK_\mu^{part} \subseteq TM^{part}$.

Speziell gilt folgendes Resultat:

Satz: Jede partielle oder totale μ-rekursive Funktion wird von einer Turing-Maschine mit Alphabet $\{\#, |\}$ berechnet.

Für die Rückrichtung benötigen wir wieder eine geeignete Gödelisierung für Turing-Maschinen!

$\Longrightarrow$ Wie im primitiv rekursiven Fall verwenden wir auch hier eine Primzahlkodierung!

Dazu machen wir folgende Annahmen:

- Die Zustände einer TM $M = (K, \Sigma, \delta, s)$ seien durchnummeriert, wobei
  - der Startzustand hat stets die Nummer 1; und
  - der Haltezustand hat stets die Nummer $n+1$, mit $n = |K|$.
- Alle betrachteten TM sind auf einem endlichen Teilalphabet eines festen Alphabetes

  $$\Sigma_\infty = \{a_0, a_1, \ldots\}$$

  definiert, mit $a_0 = \#$ und $a_1 = |$.

  Genauer:
  - oBdA sei das Alphabet $\Sigma \subseteq \Sigma_\infty$ einer TM stets eine Menge der Form $\Sigma = \{a_0, a_1, \ldots, a_{m-1}\}$, für ein geeignetes $m > 0$.
- Das zugrundeliegende TM-Modell sei das einer 1-Band Maschine mit zweiseitig unbeschränktem Band.

Definition: Sei $M = (K, \Sigma, \delta, s)$ eine zw-1M mit

- $K = \{q_1, \ldots, q_n\}$, $q_1 = s$, $q_{n+1} = h$;
- $\Sigma = \{a_0, \ldots, a_{m-1}\}$, $a_0 = \#$, $a_1 = |$, $a_m = L$, $a_{m+1} = R$.

Dann sei folgende Gödelisierung definiert:

1. Für $\delta(q_i, a_j) = (q_{i'}, a_{j'})$ sei $\delta_{i,j} := \langle i', j' \rangle$.
2. Für ein Wort $w = a_{i_1} \ldots a_{i_p}$ sei
$$l(w) := \langle i_p, \ldots, i_1 \rangle \qquad \text{und}$$
$$r(w) := \langle i_1, \ldots, i_p \rangle.$$
3. Die **Gödelnummer der TM** M ist
$$\gamma(M) := \langle \langle n, m \rangle, \delta_{1,0}, \ldots, \delta_{1,m-1}, \ldots, \delta_{n,0}, \ldots, \delta_{n,m-1} \rangle$$
4. Die **Gödelnummer einer Konfiguration** $C = q_i, w\underline{a_j}u$ ist
$$\gamma(C) := \langle i, l(w), j, r(u) \rangle$$

Bemerkung: Für die Gödelisierung einer Konfiguration sind die Bandinhalte rechts und links vom Kopf unterschiedlich kodiert:

- Wenn der Schreib-/Lesekopf sich um ein Zeichen bewegt, zeigt er entweder auf das *rechteste* Zeichen von w oder das *linkeste* von u.
  - w wird so kodiert, daß das rechteste Zeichen "unten" in der Primzahlkodierung steht (d.h., zum Exponenten der ersten Primzahl wird),
  - während u wird so kodiert, daß das linkeste Zeichen "unten" steht.

Lemma (Simulationslemma): Es gibt eine primitiv rekursive Funktion $f_U : \mathbb{N}^3 \to \mathbb{N}$ s.d. für jede TM M gilt:

- Ist $C_0, \ldots, C_t$ ($t \in \mathbb{N}$) eine Rechnung von M, so ist
$$f_U(\gamma(M), \gamma(C_o), t) = \gamma(C_t).$$

Beweis: Wir definieren vier Funktionen, die Konfigurationen von Turing-Maschinen berechnen:

$Z(\gamma, \varrho, t) = i \iff$ die TM M mit Gödelnummer $\gamma(M) = \gamma$ erreicht von einer Konfiguration C mit $\gamma(C) = \varrho$ nach t Schritten der Rechnung eine Konfiguration mit Zustand q_i.

$L(\gamma, \varrho, t) = l(w) \iff$ die TM M mit Gödelnummer $\gamma(M) = \gamma$ erreicht von einer Konfiguration C mit $\gamma(C) = \varrho$ nach t Schritten der Rechnung eine Konfiguration wo w der Bandinhalt links vom Kopf ist.

$A(\gamma, \varrho, t) = j \iff$ die TM M mit Gödelnummer $\gamma(M) = \gamma$ erreicht von einer Konfiguration C mit $\gamma(C) = \varrho$ nach t Schritten der Rechnung eine Konfiguration mit a_j unter dem Kopf.

$R(\gamma, \varrho, t) = r(u) \iff$ die TM M mit Gödelnummer $\gamma(M) = \gamma$ erreicht von einer Konfiguration C mit $\gamma(C) = \varrho$ nach t Schritten der Rechnung eine Konfiguration wo u der Bandinhalt rechts vom Kopf ist.

Man kann zeigen: Z, L, A und R sind primitiv rekursiv.

Dies kann wie folgt argumentiert werden.

Wir verwenden zwei Hilfsfunktionen $verl$ und $verk$ die folgende Eigenschaft besitzen:

$$\begin{aligned} verl(\langle (n)_1, \ldots, (n)_k \rangle) &= \langle 0, (n)_1, \ldots, (n)_k \rangle \\ verk(\langle (n)_1, \ldots, (n)_k \rangle) &= \langle (n)_2, \ldots, (n)_k \rangle \end{aligned}$$

$\Longrightarrow$ Die beiden Funktionen sind folgendermaßen bestimmt:

$$verl(n) := \textstyle\prod_{1 \le i \le n} p(i+1)^{(n)_i}$$

$$verk(n) := \begin{cases} 1 & \text{falls } n = 0 \text{ oder } n = 2^j \\ \textstyle\prod_{2 \le i \le n} p(i-1)^{(n)_i} & \text{sonst} \end{cases}$$

$\Longrightarrow$ $verl$ und $verk$ sind primitiv rekursiv.

Die Funktionen Z, L, A und R können mittels simultaner primitiver Rekursion dargestellt werden.

Vorbemerkung: für $|K| = n$ und $|\Sigma| = m$ gilt für die Indizes i', j' mit $\delta(q_i, a_j) = (q_{i'}, a_{j'})$:

$$\langle i', j' \rangle = (\gamma)_{(i-1)\cdot m+j+2}, \qquad \text{mit}$$

$i = Z(\gamma, \varrho, t)$, $j = A(\gamma, \varrho, t)$ und $m = ((\gamma)_1)_2$.

$$\begin{aligned} \Longrightarrow Z(\gamma, \varrho, 0) &= (\varrho)_1 \\ L(\gamma, \varrho, 0) &= (\varrho)_2 \\ A(\gamma, \varrho, 0) &= (\varrho)_3 \\ R(\gamma, \varrho, 0) &= (\varrho)_4 \end{aligned}$$

$$Z(\gamma, \varrho, t+1) = i' = (\langle i', j' \rangle)_1$$

$$L(\gamma, \varrho, t+1) = \begin{cases} L(\gamma, \varrho, t) & \text{für } a_{j'} \in \Sigma \\ verk(L(\gamma, \varrho, t)) & \text{für } a_{j'} = L \\ verl(L(\gamma, \varrho, t)) \cdot 2^{A(\gamma, \varrho, t)} & \\ & \text{für } a_{j'} = R \end{cases}$$

$$A(\gamma, \varrho, t+1) = \begin{cases} j' = (\langle i', j' \rangle)_2 & \text{für } a_{j'} \in \Sigma \\ (L(\gamma, \varrho, t))_1 & \text{für } a_{j'} = L \\ (R(\gamma, \varrho, t))_1 & \text{für } a_{j'} = R \end{cases}$$

$$R(\gamma, \varrho, t+1) = \begin{cases} R(\gamma, \varrho, t) & \text{für } a_{j'} \in \Sigma \\ verk(R(\gamma, \varrho, t)) & \text{für } a_{j'} = R \\ verl(R(\gamma, \varrho, t)) \cdot 2^{A(\gamma, \varrho, t)} & \\ & \text{für } a_{j'} = L \end{cases}$$

Es bleibt z.z. daß die Bedingungen der Fallunterscheidungen primitiv rekursiv beschreibbar sind.

D.h., wir beschreiben primitiv rekursiv ob:

- $a_{j'} \in \Sigma = \{a_0, \ldots, a_{m-1}\}$, oder
- $a_{j'} = a_m = L$, oder
- $a_{j'} = a_{m+1} = R$.

$\Longrightarrow$ Mit $j' = (\langle i', j' \rangle)_2 = ((\gamma)_{(i-1)\cdot m+j+2})_2$ erhalten wir:

$$\begin{aligned} a_{j'} \in \Sigma &\iff ((\gamma)_{(i-1)\cdot m+j+2})_2 \dot{-} (m-1) = 0 \\ a_{j'} = L &\iff |(((\gamma)_{(i-1)\cdot m+j+2})_2, m)| = 0 \\ a_{j'} = R &\iff |(((\gamma)_{(i-1)\cdot m+j+2})_2, m+1)| = 0 \end{aligned}$$

Damit sind die Funktionen Z, L, R und A primitiv rekursiv.

Als gesuchte Funktion $f_U : \mathbb{N}^3 \to \mathbb{N}$ wählen wir:

$$f_U(\gamma, \varrho, t) = \langle Z(\gamma, \varrho, t), L(\gamma, \varrho, t), A(\gamma, \varrho, t), R(\gamma, \varrho, t) \rangle \quad \blacksquare$$

Bemerkungen:

- Mittels f_u kann man die Konfiguration der TM mit Gödelnumer γ nach t Rechenschritten bestimmen.
- ABER:
  - Man kann damit **nicht** bestimmen, was die TM mit Gödelnummer γ berechnet hat!
- Genauer:
  - Wir können mittels primitiv rekursiven Mitteln nicht feststellen, *wann* der Haltezustand erreicht wird.
- Dazu benötigen wir ein Konstrukt, daß über die primitive Rekursion hinausgeht, nämlich den μ-Operator.

  $\Rightarrow$ Wir suchen die kleinste Rechenschritt-Anzahl t_0 s.d. der Haltezustand q_{n+1} erreicht wird, d.h., s.d. $Z(\gamma, \varrho, t_0) = n + 1$.

Satz: Es gilt:

1. $TM \subseteq REK_\mu$; und
2. $TM^{part} \subseteq REK_\mu^{part}$.

D.h., jede totale (bzw. partielle) TM-berechenbare Funktion ist (partiell) μ-rekursiv.

Beweis: Sei $f : \mathbb{N}^k \to \mathbb{N}$ eine totale TM-berechenbare Funktion.

Dann gibt es eine Turing-Maschine M_f die f berechnet, d.h.:

$$f(n_1, \ldots, n_k) = p \iff s, \#|^{n_1}\# \ldots \#|^{n_k}\underline{\#} \vdash^*_{M_f} h, \#|^p\underline{\#}$$

Wir müssen zeigen: $f \in REK_\mu$.

Dazu verwenden wir die Funktionen Z, L, R und A vom Simulationslemma.

Beachte:

- Diese Funktionen haben allerdings Gödelnummern als Argumente, während die f simulierende μ-rekursive Funktion $n_1, \ldots, n_k$ als Argumente haben muß.

$\Rightarrow$ Zunächst müssen wir also zeigen, wie man mittels primitiver Rekursion aus $n_1, \ldots, n_k$ die Gödelnummer der zugehörigen Startkonfiguration berechnen kann.

$\Rightarrow$ D.h., wir suchen primitiv rekursive Funktionen $g_k : \mathbb{N}^k \to \mathbb{N}$ s.d.:

$$g_k(n_1, \ldots, n_k) = \gamma(s, \#|^{n_1}\# \ldots \#|^{n_k}\underline{\#})$$

Laut Definition der Gödelnummer einer Konfiguration gilt:

$\gamma(s, \#|^{n_1}\# \ldots \#|^{n_k}\underline{\#}) = \langle 1, l(\#|^{n_1}\# \ldots \#|^{n_k}), 0, 1\rangle$

da $s = q_1$, $\# = a_0$ und $r(\varepsilon) = 1$.

Somit müssen wir nur noch für jedes k eine primitiv rekursive Funktion $l_k : \mathbb{N}^k \to \mathbb{N}$ definieren mit

$$\begin{aligned} l_k(n_1, \ldots, n_k) &= l(\#|^{n_1}\# \ldots \#|^{n_k}) \\ &= \langle \underbrace{1, \ldots, 1}_{n_k}, 0, \ldots, 0, \underbrace{1, \ldots, 1}_{n_2}, 0, \underbrace{1, \ldots, 1}_{n_1}\rangle, \end{aligned}$$

da $\# = a_0$ und $| = a_1$.

Wir zeigen mittels Induktion nach k daß die Funktionen l_k primitiv rekursiv sind.

$k = 1$: Dann ist

$$l_1(n) = \langle \underbrace{1, \ldots, 1}_{n}\rangle = \prod_{1 \le i \le n} p(i)^1$$

und l_1 ist klarerweise primitiv rekursiv.

$k > 1$: Angenommen alle l_j mit $j < k$ sind primitiv rekursiv. Wir zeigen, daß l_k primitiv rekursiv ist.

l_k kann mittels primitiver Rekursion dargestellt werden:

$$\begin{aligned} l_k(n_1, \ldots, n_{k-1}, 0) &= \langle 0, \underbrace{1, \ldots, 1}_{n_{k-1}}, 0, \ldots, 0, \underbrace{1, \ldots, 1}_{n_1}\rangle \\ &= verl(l_{k-1}(n_1, \ldots, n_{k-1})) \end{aligned}$$

$$l_k(n_1, \ldots, n_{k-1}, n+1) = 2 \cdot verl(l_k(n_1, \ldots, n_{k-1}, n))$$

Im Rekursionsschritt wird mittels Multiplikation mit 2 die Gödelnummer von $\#|^{n_1}\# \ldots \#|^{n_{k-1}}\#|^{n}\#$ in die Gödelnummer von $\#|^{n_1}\# \ldots \#|^{n_{k-1}}\#|^{n}|$ umgewandelt.

Wir setzen mit der Simulation der Arbeit von M_f fort.

Sei $\gamma(M_f) = \gamma_f$ die Gödelnummer von M_f, und sei die Zahl der Zustände von M_f $|K| = n$.

Es gilt:

- Wegen der Totalität von f muß es eine Schrittanzahl t_0 geben s.d. M_f nach t_0 Schritten den Haltezustand $h = q_{n+1}$ erreicht.

$\Rightarrow$ Der Wert p von f steht dann links vom Kopf, d.h., p ist in $L(\gamma_f, g_k(n_1, \ldots, n_k), t_0)$ kodiert.

Wir erhalten:

$$\begin{aligned} & f(n_1, \ldots, n_k) = p \\ \iff\ & s, \#|^{n_1}\# \ldots \#|^{n_k}\underline{\#} \vdash^*_{M_f} h, \#|^{p}\underline{\#} \\ \iff\ & \exists t_0 \Big(Z(\gamma_f, g_k(n_1, \ldots, n_k), t_0) = n+1 \ \wedge \\ & \qquad L(\gamma_f, g_k(n_1, \ldots, n_k), t_0) = \langle \underbrace{1, \ldots, 1}_{p}\rangle \ \wedge \\ & \qquad A(\gamma_f, g_k(n_1, \ldots, n_k), t_0) = 0 \ \wedge \\ & \qquad R(\gamma_f, g_k(n_1, \ldots, n_k), t_0) = 1 \Big). \end{aligned}$$

Den Wert t_0 erhalten wir mittels μ-Operator: wir suchen die kleinste Zahl t_0 s.d.

$$Z(\gamma_f, g_k(n_1, \ldots, n_k), t_0) = n+1 = ((\gamma_f)_1)_1 + 1$$

Um festzustellen, welchen Funktionswert f berechnet hat, muß $L(\gamma_f, g_k(n_1, \ldots, n_k), t_0)$ decodiert werden.

$\Longrightarrow$ Dazu verwenden wir die primitiv-rekursive Dekodier-Funktion Dek, gegeben durch:

$$Dek(j) := \mu_{i \le j}(l_1(i) = j),$$

wo

$$l_1(j) = \prod_{1 \le i \le j} p(i)^1 = \langle \underbrace{1, \ldots, 1}_{j}\rangle.$$

$$\begin{aligned} \Longrightarrow f(n_1, \ldots, n_k) = Dek\Big(& L\big(\gamma_f, g_k(n_1, \ldots, n_k), \\ & \mu i(Z(\gamma_f, g_k(n_1, \ldots, n_k), i) = ((\gamma_f)_1)_1 + 1)\big)\Big) \end{aligned}$$

Es folgt: $f \in REK_\mu$, und somit $TM \subseteq REK_\mu$.

Im Falle einer partiellen TM-berechenbaren Funktion f ändert sich nur, daß der μ-Operator undefiniert ist falls f undefiniert ist.

$\Longrightarrow TM^{part} \subseteq REK_\mu^{part}$. ■

Korollar: $\forall k \in \mathbb{N} \quad \exists h'_k, h_k \in REK_{pr} \quad \forall f : \mathbb{N}^k \to \mathbb{N} \in TM \quad \exists \gamma_f \in \mathbb{N} \quad \forall \underline{n} = (n_1, \ldots, n_k) :$

$$f(\underline{n}) = h'_k\Big(\gamma_f, \underline{n}, \mu i\big(h_k(\gamma_f, \underline{n}, i) = 0\big)\Big).$$

Beweis: Man setze einfach

$$h'_k(\gamma_f, \underline{n}, t) = Dek(L(\gamma_f, g_k(\underline{n}, t)) \quad \text{und}$$
$$h_k(\gamma_f, \underline{n}, i) = (n+1) \dot{-} Z(\gamma_f, g_k(\underline{n}, i))$$

■

Es gilt noch ein schärferes Resultat:

Satz (Kleene's Normalformtheorem):

$\exists U \in REK_{pr} \quad \forall k \in \mathbb{N} \quad \exists T_k \in REK_{pr} \quad \forall f : \mathbb{N}^k \to \mathbb{N} \in TM \quad \exists \gamma_f \in \mathbb{N} \quad \forall \underline{n} = (n_1, \ldots, n_k) :$

$$f(\underline{n}) = U\big(\mu i(T_k(\gamma_f, \underline{n}, i) = 0)\big).$$

Beweis: Die gebundene Variable i des μ-Operators muß hier zwei Werte transportieren:

1. den Zeitpunkt t_0, nach dem der Haltezustand erreicht wird;
2. den Bandinhalt links vom Kopf nach Erreichen des Haltezustandes.

Seien h'_k, h_k wie in obigen Korollar definiert.

Wir definieren:

$$T_k(\gamma_f, \underline{n}, i) := |(i)_1, h'_k(\gamma_f, \underline{n}, (i)_2)| + h_k(\gamma_f, \underline{n}, (i)_2)$$

Dann gilt $T_k(\gamma_f, \underline{n}, i) = 0$ gdw folgende zwei Bedingungen erfüllt sind:

1. $(i)_1 = h'_k(\gamma_f, \underline{n}, (i)_2) = Dek(L(\gamma_f, g_k(\underline{n}), (i)_2))$:
   Die TM mit Gödelnummer γ_f, gestartet mit Input $\underline{n}$, erreicht nach $(i)_2$ Schritten eine Konfiguration, wo der mit Dek dekodierte Inhalt des Bandes links vom Kopf $(i)_1$ ist.
2. $h_k(\gamma_f, \underline{n}, (i)_2) = 0$:
   Die Maschine hält nach $(i)_2$ Schritten.

$\Longrightarrow$ Wir setzen $U(p) := (p)_1$

$$\Longrightarrow f(\underline{n}) = (\mu i(T_k(\gamma_f, \underline{n}, i) = 0)_1$$
$$= U(\mu i(T_k(\gamma_f, \underline{n}, i) = 0)).$$

■

Bemerkung:

- Ein analoges Resultat gilt für partielle Funktionen.