# 67. Nerozhodnutelnost (problém zastavení TS, princip diagonalizace a redukce, Postův korespondenční problém).

## Hranice rozhodnutelnosti

Ať $M = (Q, \Sigma, \Gamma, \delta, q_0, q_f)$ je Turingův stroj. Řekneme, že M je **úplný Turingův stroj**, pokud zastaví pro každý svůj vstup. O takovém stroji řekneme, že **rozhoduje** jazyk L(M).

Ať Σ je abeceda. Třída všech jazyků, které je možné přijmout pomocí úplného Turingova stroje, se nazývá třídou **rekurzivních jazyků** $\mathcal{L}_{\mathbf{REC}} \subset 2^{\Sigma^*}$. Třída všech jazyků, které je možné přijímat pomocí (obecného) Turingova stroje, je třídou **rekurzivně vyčíslitelných jazyků** $\mathcal{L}_{\mathbf{RE}} \subset 2^{\Sigma^*}$. Platí

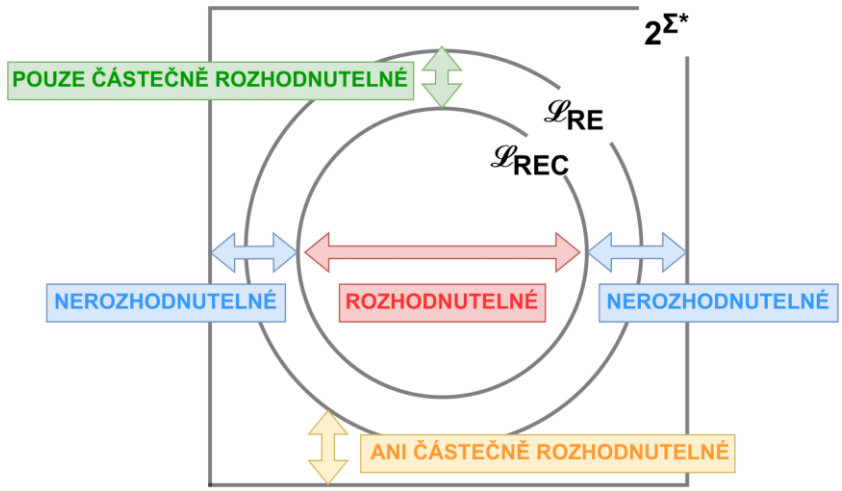
$$\mathcal{L}_{\mathbf{REC}} \subset \mathcal{L}_{\mathbf{RE}} \subset 2^{\Sigma^*},$$

tedy existují jazyky, které nejsou rekurzivně vyčíslitelné (není možné je přijmout žádným Turingovým strojem) a existují jazyky, které nejsou rekurzivní, ale jsou rekurzivně vyčíslitelné (není možné je rozhodovat úplným Turingovým strojem, ale je možné je přijmout obecným Turingovým strojem).

Ať Σ je abeceda. Rozlišujeme jazyky/problémy, které jsou

- **rozhodnutelné**
  - rozhodnutelné jsou ty jazyky, které je možné rozhodovat pomocí nějakého úplného Turingova stroje, rekurzivní jazyky
  - spustíme-li běh úplného Turingova stroje na libovolném řetězci ze Σ*, stroj vždy v konečném počtu kroků zastaví a korektně ohlásí, zda daný řetězec přijímá, nebo odmítá
  - je možné v konečném čase rozhodnout příslušnost/nepříslušnost každého řetězce ze Σ* do daného rozhodnutelného jazyka
- **nerozhodnutelné**
  - neexistuje žádný úplný Turingův stroj, který by takový jazyk rozhodoval
  - pokud je jazyk L nerozhodnutelný, pak buď
    - existuje Turingův stroj, který L přijímá, ale vždy na některých řetězcích, které nepřijímá, cyklí, nebo
    - neexistuje ani žádný Turingův stroj, který L přijímá
- **pouze částečně rozhodnutelné**
  - existuje Turingův stroj, který takový jazyk přijímá, ale neexistuje Turingův stroj, který takový jazyk rozhoduje
  - jde o jazyky, které jsou rekurzivně vyčíslitelné, ale nejsou rekurzivní

- **ani částečně rozhodnutelné**
    - neexistuje žádný Turingův stroj, který takový jazyk přijímá
    - jde o jazyky, které nejsou rekurzivně vyčíslitelné




Uveďme příklady některých jazyků z těchto tříd nad abecedou Σ = {0, 1, #}:

- **rozhodnutelné**
    - **libovolný regulární, bezkontextový, kontextový jazyk**
    - {<M>#<n> | <M> je kód Turingova stroje, <n> je kód přirozeného čísla, kde stroj M má právě n stavů }
    - **SAT: {<φ> | <φ> je kód výrokové formule φ, která je splnitelná}**
    - {<G>#<n> | <G> je kód grafu, <n> je kód přirozeného čísla a G obsahuje kliku o velikosti n}
    - {$<A_1>\#<A_2>$ | $<A_1>$, $<A_2>$ jsou kódy konečných automatů, kde $L(A_1) = L(A_2)$}
- **pouze částečně rozhodnutelné**
    - **HP: {<M>#<w> | <M> je kód Turingova stroje, <w> je kód řetězce a M zastaví na vstupu w}**
    - {<L> | <L> je kód lambda výrazu, který lze pomocí konečné posloupnosti β-redukcí transformovat do normální formy}
    - **PCP: {<S> | <S> je kód Postova systému, který má řešení}**
    - **MP: {<M>#<w> | <M> je kód Turingova stroje, <w> je kód řetězce, kde w ∈ L(M)}**
    - {<A> | <A> je kód zásobníkového automatu, který nepřijímá univerzální jazyk}
    - {<P>#<n> | <P> je kód zdrojového kódu v jazyce C a <n> je kód takového přirozeného čísla, že pro nějaký vstup zdrojového kódu P je dosažitelný řádek n kódu P}
- **ani částečně rozhodnutelné**
    - **co-HP: {<M>#<w> | <M> je kód Turingova stroje, <w> je kód řetězce a M cyklí na vstupu w}**
    - **co-PCP: {<S> | <S> je kód Postova systému, který nemá řešení}**

- **co-MP: {<M>#<w> | <M> je kód Turingova stroje, <w> je kód řetězce, kde w ∉ L(M)}**
- {<$G_1$>#<$G_2$> | <$G_1$>, <$G_2$> jsou kódy bezkontextových gramatik, které generují stejný jazyk}
- {<M> | <M> je kód Turingova stroje, který přijímá právě kódy všech Turingových strojů, které přijímají vlastní kód}
- {<M> | <M> je kód Turingova stroje, který je úplným Turingovým strojem}

# Problém zastavení

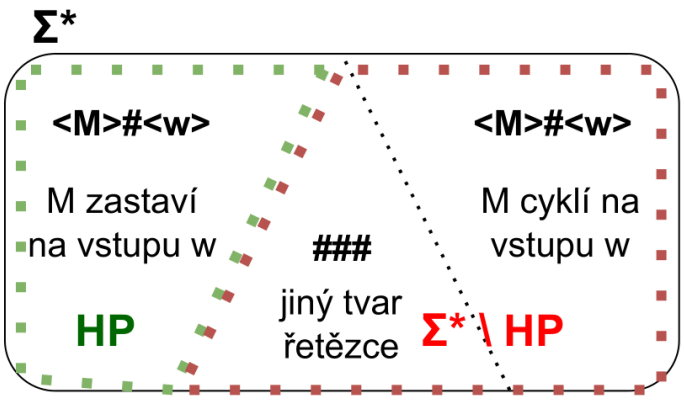
Ať {0, 1} je kódová abeceda. Mějme nějaké kódování Turingových strojů a řetězců libovolné abecedy nad touto kódovou abecedou. Definujeme jazyk **problému zastavení HP** a jazyk **problému nezastavení co-HP** následovně:

**HP** = {<M>#<w> | <M> je kód Turingova stroje M, <w> je kód řetězce w a **M zastaví na w**}

**co-HP** = {<M>#<w> | <M> je kód Turingova stroje M, <w> je kód řetězce w a **M cyklí na w**}

Jazyky HP a co-HP nejsou doplňky z jazykového pohledu, neboť existují i řetězce nad abecedou {0, 1, #}, které nemají ani syntaktický potenciál patřit do HP nebo co-HP, tedy mají úplně jiný tvar než <M>#<w>, kde <M> je kód Turingova stroje a <w> je kód řetězce, takzvané nevalidní instance HP. Množinu {0, 1, #}* tedy můžeme rozložit na třídy:

- **HP**
- **co-HP**
- **nevalidní instance HP**




**Jazyk problému zastavení HP je rekurzivně vyčíslitelný.** Existuje tedy Turingův stroj, který **přijímá** HP. Takový stroj M pracuje následovně:

- Stroj M ověří, zda je jeho vstup w nevalidní instancí HP (není ve tvaru <M'>#<w'>). Pokud tomu tak je, odmítne, jinak pokračuje
- Stroj M spustí simulaci stroje M' na řetězci w'
- Pokud tato simulace skončí, M akceptuje

Stroj M tedy akceptuje všechny řetězce typu <M'>#<w'>, kde stroj M' zastaví na řetězci w', neboť jenom pro ně platí, že simulace spuštěná v druhém bodě skončí a bude možné přejít k dalšímu bodu. Na všech ostatních řetězcích stroj cyklí, nebo je odmítá. HP je tedy rekurzivně vyčíslitelný.

Neexistuje však Turingův stroj, který rozhoduje HP. HP je tedy **pouze částečně rozhodnutelný**. **Jazyk problému zastavení HP není rekurzivní.** Viz následující sekci o diagonalizaci.

Obecně platí, že komplement pouze částečně rozhodnutelného problému (nejedná se o jazykový komplement, viz výše), není ani částečně rozhodnutelný. Tedy **co-HP není ani rekurzivně vyčíslitelný**, není tedy ani částečně rozhodnutelný. Což dává smysl, protože kdyby byl, musel by existovat obecný TS, který by pracoval následovně:

- Stroj M ověří, zda je jeho vstup w nevalidní instancí co-HP (není ve tvaru <M'>#<w'>, stejné jako pro HP). Pokud tomu tak je, odmítne, jinak pokračuje
- Stroj M spustí simulaci stroje M' na řetězci w'
- Pokud tato simulace **cyklí**, M akceptuje.

Takový TS však nemůže existovat, protože není možné identifikovat, že simulace cyklí, aby následně stroj zastavil a akceptoval.

# Diagonalizace

**Diagonalizace** (metoda diagonálního argumentu) je důkazní technika **využívající důkaz sporem**, kterou Georg **Cantor** ukázal **nespočetnost množiny reálných čísel**. Zde ji využijeme k důkazu **nespočetnosti množiny všech jazyků nad abecedou Σ** a k **důkazu nerekurzivity problému zastavení**. Pomocí diagonalizace lze však dokazovat mnohá další tvrzení.

## Důkaz nespočetnosti množiny všech jazyků nad abecedou Σ

Ať Σ je abeceda. Ukážeme, že množina všech jazyků $2^{\Sigma^*}$ nad touto abecedou je nespočetná. Jde o důkaz sporem, tedy budeme předpokládat, že množina $2^{\Sigma^*}$ je spočetná a korektní posloupností logických odvození dojdeme ke sporu, z nějž vyplyne, že tento předpoklad musí být nepravdivý.

- Pro spor předpokládejme, že množina $2^{\Sigma^*}$ je spočetná.
- Potom z definice spočetnosti plyne, že existuje vzájemně jednoznačné zobrazení $f : \mathbb{N} \leftrightarrow 2^{\Sigma^*}$, tzn. existuje bijektivní funkce, která každému přirozenému číslu přiřazuje nějaký jazyk nad abecedou Σ tak, že každý jazyk má přiřazené nějaké přirozené číslo.
- Veškeré jazyky nad abecedou Σ je tedy možné pomocí funkce f seřadit do nekonečné posloupnosti f(0), f(1), f(2), f(3), …
- Veškeré řetězce nad abecedou Σ je možné seřadit lexikograficky do posloupnosti $w_0$, $w_1$, $w_2$, $w_3$, …, neboť množina Σ* je také spočetná
- Nyní tedy můžeme zkonstruovat nekonečnou matici, kde
    - *m.* řádek je popsán jazykem f(m) = $L_m$
    - *n*. sloupec je popsán řetězcem $w_n$
    - každá buňka matice $a_{m,n}$ může nabývat hodnoty 0, nebo 1:

- $a_{m,n} = 1 \Leftrightarrow w_n \in f(m) = L_m$, tedy řetězec popisující daný sloupec patří do jazyka, který popisuje daný řádek
- $a_{m,n} = 0 \Leftrightarrow w_n \notin f(m) = L_m$, tedy řetězec popisující daný sloupec nepatří do jazyka, který popisuje daný řádek

| | $w_0$ | $w_1$ | $w_2$ | $w_3$ | $w_4$ | ····· |
|---|---|---|---|---|---|---|
| $f(0) = L_0$ | $a_{0,0}$ | $a_{0,1}$ | $a_{0,2}$ | $a_{0,3}$ | $a_{0,4}$ | ····· |
| $f(1) = L_1$ | $a_{1,0}$ | $a_{1,1}$ | $a_{1,2}$ | $a_{1,3}$ | $a_{1,4}$ | ····· |
| $f(2) = L_2$ | $a_{2,0}$ | $a_{2,1}$ | $a_{2,2}$ | $a_{2,3}$ | $a_{2,4}$ | ····· |
| $f(3) = L_3$ | $a_{3,0}$ | $a_{3,1}$ | $a_{3,2}$ | $a_{3,3}$ | $a_{3,4}$ | ····· |
| $f(4) = L_4$ | $a_{4,0}$ | $a_{4,1}$ | $a_{4,2}$ | $a_{4,3}$ | $a_{4,4}$ | ····· |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋱ |

- Jelikož jsme řádky uspořádali pomocí zobrazení f, je každý jazyk nad abecedou Σ zastoupen v uvedené matici, tzn. neměl by existovat žádný jazyk ze třídy $2^{\Sigma^*}$, který v matici není obsažen
- Zkonstruujme nyní jazyk $C = \{w_x \in \Sigma^* \mid a_{xx} = 0\}$, tedy jazyk obsahující právě takové řetězce $w_x$, které nepatří do jazyka $f(x) = L_x$. Jinak řečeno, jde o řetězce popisující takové sloupce v uvedené matici, že na diagonální pozici matice je v daném sloupci 0
- Ohodnocení jazyka C v námi uvažované matici by muselo odpovídat komplementované hlavní diagonále matice (0 je nahrazeno za 1, 1 je nahrazeno za 0)

| | | | | | | |
|---|---|---|---|---|---|---|
| **C** | $1-a_{0,0}$ | $1-a_{1,1}$ | $1-a_{2,2}$ | $1-a_{3,3}$ | $1-a_{4,4}$ | ····· |

| | $w_0$ | $w_1$ | $w_2$ | $w_3$ | $w_4$ | ····· |
|---|---|---|---|---|---|---|
| $f(0) = L_0$ | $\mathbf{a_{0,0}}$ | $a_{0,1}$ | $a_{0,2}$ | $a_{0,3}$ | $a_{0,4}$ | ····· |
| $f(1) = L_1$ | $a_{1,0}$ | $\mathbf{a_{1,1}}$ | $a_{1,2}$ | $a_{1,3}$ | $a_{1,4}$ | ····· |
| $f(2) = L_2$ | $a_{2,0}$ | $a_{2,1}$ | $\mathbf{a_{2,2}}$ | $a_{2,3}$ | $a_{2,4}$ | ····· |
| $f(3) = L_3$ | $a_{3,0}$ | $a_{3,1}$ | $a_{3,2}$ | $\mathbf{a_{3,3}}$ | $a_{3,4}$ | ····· |
| $f(4) = L_4$ | $a_{4,0}$ | $a_{4,1}$ | $a_{4,2}$ | $a_{4,3}$ | $\mathbf{a_{4,4}}$ | ····· |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋱ |

- **Takové ohodnocení by se od každého řádku v matici muselo lišit minimálně na pozici na diagonále, která je v tomto ohodnocení komplementovaná.** To znamená, že jazyk C se liší od každého jazyka uvedeného v matici.

- Jazyk C tedy není v matici zastoupen, což je spor, neboť jsme řekli, že v matici jsou uvedeny všechny jazyky nad abecedou Σ
- Spor vychází z předpokladu, že existuje vzájemně jednoznačné zobrazení f : $\mathbb{N} \leftrightarrow 2^{\Sigma^*}$, tedy z předpokladu, že množina $2^{\Sigma^*}$ je spočetná
- **Množina $2^{\Sigma^*}$ je tedy nespočetná.**

Pro lepší ilustraci si můžeme představit, že matici máme již nějak ohodnocenou konkrétními hodnotami (jde pouze pro neformální ukázku, hodnoty 0, 1 jsou zde rozházeny bez jakékoliv logiky). Diagonála je označena **fialově**, komplementovaná diagonála **červeně**. Všimněme si, že pokud přiložíme řádek **C** ke všem ostatním řádkům matice, vždy se liší minimálně na diagonále.

| | $w_0$ | $w_1$ | $w_2$ | $w_3$ | $w_4$ | ⋯ |
|---|---|---|---|---|---|---|
| **C** | 1 | 0 | 0 | 1 | 1 | ⋯ |
| $f(0) = L_0$ | $1^0$ | $0^1$ | $0^0$ | $1^0$ | $1^0$ | ⋯ |
| $f(1) = L_1$ | $1^1$ | $0^1$ | $0^0$ | $1^1$ | $1^1$ | ⋯ |
| $f(2) = L_2$ | $1^0$ | $0^0$ | $0^1$ | $1^0$ | $1^1$ | ⋯ |
| $f(3) = L_3$ | $1^0$ | $0^1$ | $0^1$ | $1^0$ | $1^0$ | ⋯ |
| $f(4) = L_4$ | $1^0$ | $0^0$ | $0^0$ | $1^1$ | $1^0$ | ⋯ |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋱ |

## Důkaz nerekurzivity problému zastavení pomocí diagonalizace

Ukážeme, že problém zastavení HP není rekurzivní. Jde o důkaz sporem, tedy budeme předpokládat, že HP je rekurzivní a korektní posloupností logických odvození dojdeme ke sporu, z nějž vyplyne, že tento předpoklad musí být nepravdivý.

- Pro spor předpokládejme, že jazyk problému zastavení HP je rekurzivní.
- Potom z definice rekurzivity plyne, že existuje úplný Turingův stroj M, který rozhoduje problém zastavení. Turingův stroj M tedy pro každý řetězec na vstupu zastaví, přičemž akceptuje právě tehdy, pokud vstup patří do HP.
- Symbolem $M_x$, kde x ∈ {0, 1}*, označme Turingův stroj
  - s kódem *x*, pokud je *x* validní kód Turingova stroje
  - s kódem *c*, kde *c* je nějaká konstanta odpovídající kódu nějakého triviálního Turingova stroje, která bude přiřazena každému stroji $M_x$, kde *x* není validní kód Turingova stroje
- Můžeme tedy uvažovat o posloupnosti $M_{f(0)} = M_\varepsilon$, $M_{f(1)} = M_0$, $M_{f(2)} = M_1$, $M_{f(3)} = M_{00}$, …, která obsahuje každý Turingův stroj neboť množina všech Turingových strojů je spočetná, jelikož zobrazení f : $\mathbb{N} \leftrightarrow$ {0, 1}* je vzájemně jednoznačné

- Veškeré řetězce nad abecedou Σ je možné seřadit lexikograficky do posloupnosti $w_0$, $w_1$, $w_2$, $w_3$, …, neboť množina Σ* je také spočetná
- Nyní tedy můžeme zkonstruovat nekonečnou matici, kde
    - *m.* řádek je popsán Turingovým strojem $M_{f(m)}$
    - *n*. sloupec je popsán řetězcem $w_n$
    - každá buňka matice $a_{m,n}$ může nabývat hodnoty Z (zastaví), nebo C (cyklí), přičemž:
        - $a_{m,n}$ = Z ⇔ **$M_{f(m)}$ zastaví na řetězci $w_n$**, tedy stroj popisující daný řádek zastaví na řetězci popisujícím daný sloupec
        - $a_{m,n}$ = C ⇔ **$M_{f(m)}$ cyklí na řetězci $w_n$**, tedy stroj popisující daný řádek cyklí na řetězci popisujícím daný sloupec

| | $w_0$ | $w_1$ | $w_2$ | $w_3$ | $w_4$ | ····· |
|---|---|---|---|---|---|---|
| $M_{f(0)}$ | $a_{0,0}$ | $a_{0,1}$ | $a_{0,2}$ | $a_{0,3}$ | $a_{0,4}$ | ····· |
| $M_{f(1)}$ | $a_{1,0}$ | $a_{1,1}$ | $a_{1,2}$ | $a_{1,3}$ | $a_{1,4}$ | ····· |
| $M_{f(2)}$ | $a_{2,0}$ | $a_{2,1}$ | $a_{2,2}$ | $a_{2,3}$ | $a_{2,4}$ | ····· |
| $M_{f(3)}$ | $a_{3,0}$ | $a_{3,1}$ | $a_{3,2}$ | $a_{3,3}$ | $a_{3,4}$ | ····· |
| $M_{f(4)}$ | $a_{4,0}$ | $a_{4,1}$ | $a_{4,2}$ | $a_{4,3}$ | $a_{4,4}$ | ····· |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋱ |

- Zkonstruujme nyní nový Turingův stroj **Conf**, který pracuje následovně:
    - Stroj Conf na vstupu dostane řetězec *w* nad abecedou {0, 1}.
    - Stroj Conf tento řetězec transformuje do tvaru $<M_W>$#w.
    - Stroj Conf spustí simulaci stroje M na řetězci $<M_w>$#w. Připomeňme, že stroj M je úplný Turingův stroj, který dle našeho předpokladu rozhoduje problém zastavení.
    - Pokud stroj M odmítne, stroj Conf akceptuje.
    - Pokud stroj M akceptuje, stroj Conf přejde do nekonečné smyčky.
- Stroj Conf vlastně komplementuje hlavní diagonálu výše uvedené matice, neboť pokud by měl stroj $M_w$ cyklit na vstupu *w*, stroj Conf na vstupu *w* naopak zastaví. Pokud by naopak měl stroj $M_w$ na vstupu *w* zastavit, stroj Conf se na něm naopak zacyklí. Informaci o tom, zda $M_w$ na *w* cyklil, nebo zastavil, nám dá úplný Turingův stroj M rozhodující HP. Conf skutečně komplementuje právě hlavní diagonálu, neboť ze svého vstupu *w* vždy vytvoří řetězec $<M_w>$#w pro stroj M.
- Ohodnocení řádku pro stroj Conf je tedy právě komplementem hlavní diagonály námi uvažované matice (tzn. symbol Z nahradíme symbolem C a naopak).
- **Takové ohodnocení by se od každého řádku v matici muselo lišit minimálně na pozici na diagonále, která je v tomto ohodnocení komplementovaná.** To znamená, že stroj Conf se liší od každého Turingova stroje uvedeného v matici.
- Turingův stroj Conf tedy není v matici zastoupen, ačkoliv jsme řekli, že v matici jsou uvedeny všechny Turingovy stroje.

- Spor vychází z předpokladu, že existuje úplný Turingův stroj M rozhodující HP, tedy že HP je rozhodnutelný.
- **HP je tedy nerozhodnutelný** (tedy **není rekurzivní**).

*Pozn.: Pozor, samotný fakt, že stroj Conf není zastoupen v naší matici, tentokrát nedokazuje, že by snad množina všech Turingových strojů byla nespočetná (jako v případě důkazu nespočetnosti množiny všech jazyků). Množina všech Turingových strojů je spočetná. Ke sporu jsme došli díky nesprávnému předpokladu rekurzivity HP. Pokud by byl HP rekurzivní, pak by totiž určitě muselo být možné sestrojit Turingův stroj Conf, ale my jsme ukázali, že tento stroj nemůže existovat, neboť není zastoupen v matici všech Turingových strojů, přičemž tentokrát víme s jistotou, že v matici jsou zastoupeny všechny Turingovy stroje.*

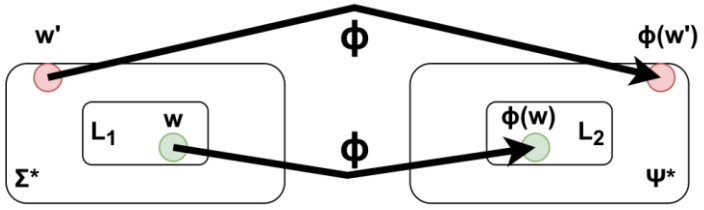
# Redukce

Ať $f : A \rightarrow B$ je zobrazení. Řekneme, že toto zobrazení je **totální**, pokud $\forall a \in A : \exists b \in B : f(a) = b$. Zobrazení f je naopak **striktně parciální**, pokud $\exists a \in A : f(a)$ není definováno.

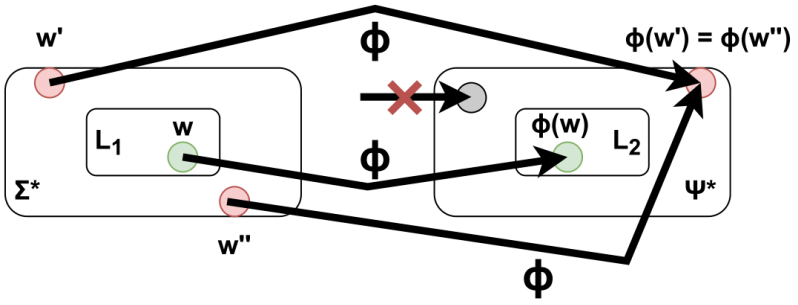
Ať $f : A \rightarrow B$ je zobrazení. Řekneme, že toto zobrazení f je **rekurzivně vyčíslitelné**, pokud je možné je realizovat pomocí nějakého Turingova stroje. Takový Turingův stroj dostane na svůj vstup řetězec $w \in A$
a na své první pásce po dokončení běhu ponechá řetězec *f(w)*, pokud je *f* pro *w* definováno, a akceptuje. Pokud *f(w)* není definováno, odmítne, nebo cyklí.

Ať $L_1$, $L_2$ jsou po řadě jazyky nad abecedami $\Sigma$, $\Psi$. Řekneme, že funkce $\Phi : \Sigma^* \rightarrow \Psi^*$ je **redukcí** jazyka $L_1$ na $L_2$, pokud $\Phi$ je **totální**, **rekurzivně vyčíslitelná** funkce, kde $\forall w \in \Sigma^* : w \in L_1 \Leftrightarrow \Phi(w) \in L_2$ (redukce **zachovává příslušnost do jazyka**).

Vlastnost zachování příslušnosti do jazyka je naznačena na obrázku níže. Funkce $\Phi$ musí být také definovaná pro každý vstup $w \in \Sigma^*$ a musí být možné ji realizovat pomocí Turingova stroje. Jinak řečeno, musí být možné ji realizovat pomocí úplného Turingova stroje (toto současně vyjadřuje, že je totální a rekurzivně vyčíslitelná).




Následující obrázek zdůrazňuje, že redukce $\Phi$ nemusí být ani injektivní, ani surjektivní, tedy může existovat obraz, na nějž se zobrazí více vzorů a může existovat obraz, na nějž se nezobrazí žádný vzor.




Ať $L_1$, $L_2$ jsou jazyky. Pokud lze $L_1$ redukovat na $L_2$, pak píšeme $L_1 <= L_2$, tedy $L_1$ je redukovatelný na $L_2$. Platí následující:

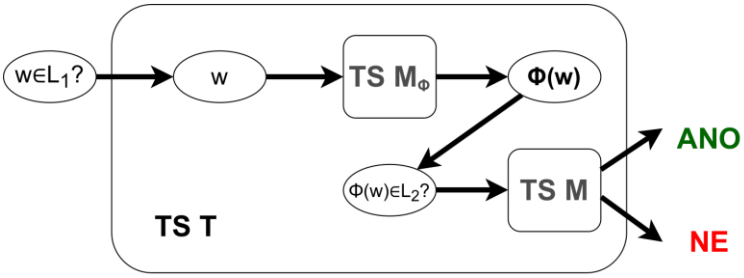
- **Pokud $L_1$ není rekurzivní a $L_1 <= L_2$, pak ani $L_2$ není rekurzivní**
- **Pokud $L_1$ není rekurzivně vyčíslitelný a $L_1 <= L_2$, pak ani $L_2$ není rekurzivně vyčíslitelný**
- Pokud je $L_2$ rekurzivní a $L_1 <= L_2$, pak i $L_1$ je rekurzivní
- Pokud je $L_2$ rekurzivně vyčíslitelný a $L_1 <= L_2$, pak i $L_1$ je rekurzivně vyčíslitelný

Redukci budeme využívat jako důkazní techniku pro **důkaz nerozhodnutelnosti**. Můžeme pomocí ní dokázat, že **jazyk není rekurzivní**, případně že **není rekurzivně vyčíslitelný**. Lze pomocí ní rovněž dokázat, že jazyk je rekurzivní, případně že je rekurzivně vyčíslitelný, ovšem to lze snáze dokázat konstrukcí úplného Turingova stroje, případně konstrukcí obecného Turingova stroje.

Ukažme, že pokud není $L_1$ rekurzivní a $L_1 <= L_2$, pak ani $L_2$ není rekurzivní. Ať $L_1$, $L_2$ jsou po řadě jazyky nad abecedami $\Sigma$, $\Psi$.

- Ať $L_1$ je nerekurzivní a ať existuje redukční funkce $\Phi$, která je redukcí $L_1$ na $L_2$. Tedy existuje úplný Turingův stroj $M_\Phi$ počítající redukci $\Phi$.
- **Pro spor předpokládejme, že $L_2$ je rekurzivní.**
- Z definice rekurzivity plyne, že existuje úplný Turingův stroj M, který rozhoduje $L_2$.
- Zkonstruujme nyní Turingův stroj T, který pracuje následovně:
    - Stroj T má na vstupu řetězec $w \in \Sigma^*$
    - Stroj T využije redukční funkci $\Phi$ a spočte řetězec $\Phi(w)$. Tato procedura jistě skončí v konečném čase, neboť funkci $\Phi$ je možné počítat pomocí úplného Turingova stroje $M_\Phi$ (to plyne z definice redukce)
    - Stroj T spustí simulaci stroje M na řetězci $\Phi(w)$. Tato procedura jistě skončí v konečném čase, neboť M je dle našeho předpokladu úplný Turingův stroj.
    - Pokud stroj M akceptuje, akceptuje i T. Pokud stroj M odmítne, odmítne i T.
- Z definice redukční funkce plyne, že $w \in L_1 \Leftrightarrow \Phi(w) \in L_2$. Jazyk $L_2$ jsme schopni rozhodovat a každý řetězec $w \in \Sigma^*$ jsme schopni pomocí redukce transformovat na řetězec $\Phi(w)$, tedy jsme rovněž schopni pro každý řetězec $w \in \Sigma^*$ rozhodnout, zda $w \in L_1$. Z toho plyne, že stroj T rozhoduje $L_1$.
- To je ovšem spor s faktem, že jazyk $L_1$ je nerozhodnutelný, nemůže být možné jej rozhodovat vůbec nijak, a to ani oklikou přes redukční funkci a rozhodování jiného jazyka.

- Tento spor plyne z předpokladu, že existuje úplný Turingův stroj M rozhodující $L_2$, tedy že $L_2$ je rekurzivní. Stroj M nemůže existovat.
- Z toho plyne, že $L_2$ není rekurzivní, pokud $L_1$ je nerekurzivní a pokud $L_1$ je redukovatelný na $L_2$.




Chceme-li tedy dokázat, že nějaký jazyk $L_2$ není rekurzivní (rekurzivně vyčíslitelný), musíme již znát nějaký referenční jazyk $L_1$, o kterém s jistotou víme, že není rekurzivní (není rekurzivně vyčíslitelný). Následně ukážeme, že $L_1 <= L_2$, z čehož pak vyplyne, že ani $L_2$ není rekurzivní (rekurzivně vyčíslitelný).

## Ukázka důkazu pomocí redukce

Ukažme, že jazyk problému náležitosti (membership problem) MP = {<M>#<w> | <M> je kód Turingova stroje, <w> je kód řetězce, kde w ∈ L(M)} není rekurzivní.

Jazyk MP není rekurzivní, což dokážeme pomocí redukce z HP, tedy ukážeme, že HP <= MP.

- Zkonstruujeme redukční funkci Φ : {0, 1, #}* → {0, 1, #}*, která realizuje redukci jazyku HP na jazyk MP.
- Pokud vstup redukční funkce *x* není validní instancí HP (nemá ani syntaktický potenciál patřit do HP, tzn. není ve tvaru <M>#<w>, kde <M> je kód Turingova stroje a <w> je kód řetězce), pak redukční funkce vrací řetězec <E>#<ε>, kde <E> je kód Turingova stroje s abecedou Σ, který v jednom kroku odmítne každý svůj vstup, tedy platí L(E) = ∅, a <ε> je kód řetězce ε, tedy ε ∉ L(E) a <E>#<ε> ∉ MP
- Pokud nebyl výpočet ukončen v předchozím kroku, pak vstup redukční funkce *x* má tvar <M>#<w>, kde <M> je kód Turingova stroje M a <w> je kód řetězce w.
- Redukční funkce v takovém případě vrátí na výstup řetězec <M'>#<ε>, kde <ε> je kód řetězce ε a M' je Turingův stroj s abecedou Σ = {0, 1, #}, který pracuje následovně:
    - Stroj M' má na vstupu řetězec w' nad abecedou {0, 1, #}
    - Stroj M' tento svůj vstup w' ignoruje
    - Stroj M' spustí simulaci stroje M na řetězci w (kódy <M> a <w> má stroj M' zakódované ve svém stavovém řízení)
    - Pokud tato simulace skončí, M' akceptuje svůj vstup w'. Pokud simulace cyklí, cyklí i M'.
- Pro redukční funkci tedy platí:
    - <M>#<w> ∈ HP ⇒ L(M') = Σ* ⇒ <M'>#<ε> ∈ MP
    - <M>#<w> ∉ HP ⇒ L(M') = ∅ ⇒ <M'>#<ε> ∉ MP
    - <M>#<w> ∈ HP ⇔ Φ(<M>#<w>) ∈ MP

- Zobrazení Φ je tedy totální, rekurzivně vyčíslitelná funkce zachovávající příslušnost do jazyka, je tedy validní redukcí HP na MP
- **Jelikož HP není rekurzivní, pak ani MP není rekurzivní**

V rámci důkazu bylo třeba sestrojit redukční funkci tak, aby korektně obsluhovala všechny řetězce z množiny {0, 1, #}*. Tyto řetězce mohly být

- **nevalidní instance HP**
  - tedy řetězce, které ani po syntaktické stránce nemohou patřit do HP
  - problém detekce nevalidní instance HP je rekurzivní, tedy nevalidní instance je možné jednoduše odhalit v konečném čase
  - redukční funkce tedy nejprve odhalí, zda její vstup není nevalidní instancí HP. Pokud ano, musí na výstup vrátit řetězec, který nepatří do MP, neboť nevalidní instance HP nepatří do HP a vyžadujeme zachování příslušnosti do jazyka
  - z tohoto důvodu vracíme na výstup konstantu <E>#<ε>, kde E je triviální Turingův stroj, který nepřijímá žádný řetězec, tedy <E>#<ε> jistě nepatří do MP, čímž jsme zachovali příslušnost
- **validní instance HP**
  - tedy řetězce, které jsou ve tvaru <M>#<w>, kde <M> je kód Turingova stroje a <w> je kód řetězce
  - tyto řetězce patří do jazyka HP, nebo do jazyka co-HP
  - obecně nelze rozhodnout, zda validní instance HP patří do jazyka HP, nebo do jazyka co-HP, tedy úplný Turingův stroj realizující redukční funkci se nic takového ani nepokouší odhalit
  - redukční funkce musí šikovným způsobem vytvořit svůj výstup tak, aby příslušnost do jazyka byla zachována. Redukční funkce toto musí zajistit dokonce i přes to, že sama nikdy nezjistí, zda její vstup patřil do HP, nebo do co-HP
  - jediné, co odlišuje řetězce z jazyka HP a z jazyka co-HP (řetězce ve tvaru <M>#<w>), je fakt, že v případě jazyka HP stroj M zastaví na w a v případě jazyka co-HP stroj M cyklí na w
  - výstup redukční funkce tedy musí mít v sobě zakódovaný řetězec <M>#<w>, aby mohl provádět simulaci M na w, jelikož to je jediný způsob, jak by mohlo být možné rozlišit řetězce z HP a z co-HP
  - redukční funkce tedy vrací řetězec <M'>#<ε>, přičemž chování stroje M' se bude lišit na základě toho, zda řetězec <M>#<w> patřil do
    - **HP**
      - V tomto případě simulace M na w, kterou spustí stroj M', jistě skončí, neboť <M>#<w> ∈ HP
      - Jakmile simulace skončí, M' akceptuje, což provede pro každý svůj vstup, tedy jistě <M'>#<ε> ∈ MP, tudíž příslušnost do jazyka je zachována
    - **co-HP**
      - V tomto případě simulace M na w, kterou spustí stroj M', jistě cyklí, neboť <M>#<w> ∈ co-HP

        - Simulace cyklí, tedy stroj M' se nikdy nedostane k tomu, aby akceptoval. Cyklí pro každý možný vstup, tedy <M'>#<ε> ∉ MP, tedy příslušnost do jazyka je zachována
    - příslušnost do jazyka je tady zachována, ačkoliv redukční funkce nikdy nezjistí, zda její vstup patři do HP, nebo do co-HP

# Redukce v kontextu rekurzivních jazyků.

Zde představenou redukci není možné použít pro důkaz, do které z podtříd rekurzivních jazyků daný jazyk spadá. Redukce je realizovaná úplným Turingovým strojem, tedy redukční funkce je dostatečně silná na to, aby každý takový jazyk sama rozhodovala.

Pokud jsou tedy $L_1$, $L_2$ rekurzivní jazyky nad abecedou Σ (s výjimkou ∅ a Σ*), pak $L_1 <= L_2$ i $L_2 <= L_1$.

Ať $L_1$, $L_2$ jsou rekurzivní jazyky nad abecedou Σ, přičemž $L_2 \neq \emptyset \wedge L_2 \neq \Sigma^*$. Potom $\exists w^+ \in L_2$, $\exists w^- \notin L_2$, tedy jistě najdeme nějaký řetězec, který do $L_2$ patří a nějaký řetězec, který do $L_2$ nepatří. Redukční funkce Φ, která provádí redukci $L_1$ na $L_2$, pracuje následovně:

- Funkce Φ má na vstupu řetězec w.
- Jelikož $L_1$ je rekurzivní, funkce Φ sama rozhodne, zda $w \in L_1$
- Pokud ano, vrátí řetězec $w^+$
- Pokud ne, vrátí řetězec $w^-$

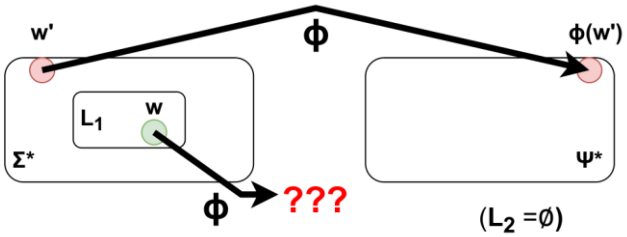
Tato funkce je tedy totální, rekurzivně vyčíslitelná a zachovává příslušnost do jazyka. Redukce v kontextu rekurzivních jazyků nám nedá žádnou užitečnou informaci.

*Pozn.: Existují jiné typy redukcí, například polynomiální redukce, které kladou přísnější omezení na redukční funkci. V případě takových redukcí má smysl na sebe redukovat i rekurzivní jazyky.*

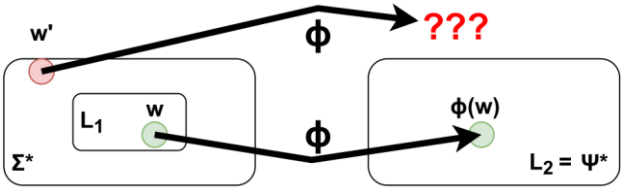
**Redukce v kontextu prázdného a univerzálního jazyka.** Ať Σ je abeceda. Ačkoliv jazyky ∅, Σ* jsou rekurzivní (a tedy redukční funkce je schopná je rozhodovat), na jazyk ∅ je možné redukovat pouze jazyk ∅, zatímco na jazyk Σ* je možné redukovat pouze jazyk Σ* (nebo jiný univerzální jazyk).

V případě jazyka ∅ totiž neplatí, že $\exists w^+ \in \emptyset$, zatímco v případě jazyka Σ* neplatí, že $\exists w^- \notin \Sigma^*$ (v kontextu abecedy Σ). Nelze tedy využít dříve předvedenou argumentaci.

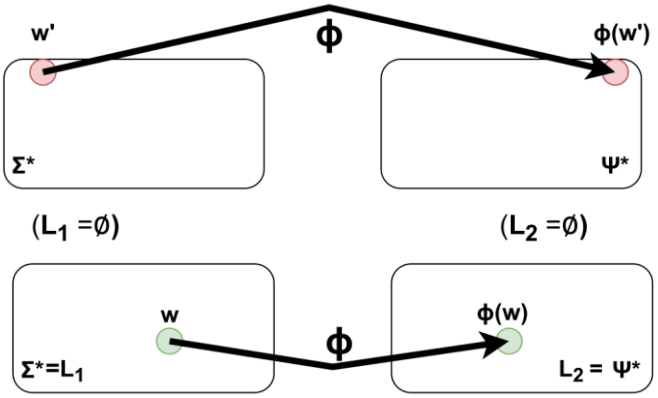
Pokud bychom redukovali libovolný neprázdný rekurzivní jazyk $L_1$ na jazyk ∅, nebylo by možné najít žádný obraz pro řetězce patřící do $L_1$, tedy by nebylo možné zachovat příslušnost do jazyka.




Pokud bychom redukovali libovolný neprázdný rekurzivní jazyk $L_1$ na jazyk Ψ*, nebylo by možné najít žádný obraz pro řetězce nepatřící do $L_1$, tedy by nebylo možné zachovat příslušnost do jazyka.




Prázdný jazyk lze ovšem redukovat na prázdný jazyk, zatímco univerzální jazyk lze redukovat na univerzální jazyk.




# Postův korespondenční problém

Ať Σ je abeceda. Konečný, neprázdný, uspořádaný seznam dvojic neprázdných řetězců nad abecedou Σ

$$S = \langle(\alpha_1, \beta_1), (\alpha_2, \beta_2), (\alpha_3, \beta_3), \ldots, (\alpha_k, \beta_k)\rangle,$$

kde $k >= 1$ a $\forall i \in \{1, \ldots, k\}$: $\alpha_i, \beta_i \in \Sigma^+$, nazveme **Postovým systémem**.

Ať S je Postův systém obsahující *k* dvojic. Neprázdnou n-tici kladných přirozených čísel nižších nebo rovných hodnotě *k* ve tvaru $(a_1, a_2, \ldots, a_n) \in \mathbb{N}_k^n$, kde $n \in \mathbb{N}^+$, nazveme **řešením Postova systému**, pokud

$$\alpha_{a1}\alpha_{a2}\alpha_{a3}\ldots\alpha_{an} = \beta_{a1}\beta_{a2}\beta_{a3}\ldots\beta_{an}.$$

Příkladem by mohl být Postův systém

$$S = \langle (aa, ab), (bab, ab), (ba, baa), (aa, a) \rangle,$$

který má řešení (3, 1, 2, 3, 4), neboť

$$\alpha_3\alpha_1\alpha_2\alpha_3\alpha_4 = ba \cdot aa \cdot bab \cdot ba \cdot aa = baaababbaaa$$
$$\beta_3\beta_1\beta_2\beta_3\beta_4 = baa \cdot ab \cdot ab \cdot baa \cdot a = baaababbaaa$$

Jazyk PCP = {<S> | <S> je kód Postova systému, který má řešení}, nazveme **jazykem Postova korespondenčního problému**.

Jazyk PCP je **nerozhodnutelný**, ale je **částečně rozhodnutelný**. Turingův stroj, který PCP přijímá, pracuje následovně:

- Stroj nejprve zkontroluje, zda má na vstupu korektní instanci Postova systému. Pokud ne, odmítne, jinak pokračuje
- Následně generuje veškeré neprázdné posloupnosti přirozených čísel od 1 do k, kde k je počet dvojic Postova systému. Posloupnosti generuje v lexikografickém pořadí, vzestupně dle délky
- Pro každou posloupnost zkontroluje, zda odpovídá řešení Postova systému. Pokud ano, akceptuje, jinak pokračuje vygenerováním další posloupnosti
- Pokud řešení existuje, bude dříve nebo později nalezeno. Pokud neexistuje, stroj cyklí.

PCP je tedy rekurzivně vyčíslitelný. Nerekurzivitu lze dokázat redukcí z HP.

Postův korespondenční problém (a jeho doplněk) lze využít pro důkaz nerozhodnutelnosti podobně jako HP. Pomocí PCP lze pohodlněji dokazovat nerozhodnutelnost problémů týkajících se gramatik, jako je prázdnost/neprázdnost průniku kontextových gramatik, ekvivalence/neekvivalence bezkontextových gramatik apod.

*Pokud v textu najdete chybu, nebudete něčemu rozumět nebo budete mít dojem, že by bylo vhodné něco doplnit, kontaktujte na discordu uživatele kocotom.*