Základy složitosti a vyčíslitelnosti

NTIN090

Petr Kučera

2024/25 (1. přednáška)

Dnešní plán

- Úvod
- Lehký úvod do teorie algoritmů
- Turingovy stroje

Sylabus

1. Turingovy stroje a jejich varianty. Churchova-Turingova teze
2. Halting problém a další nerozhodnutelné problémy
3. RAM a jeho ekvivalence s Turingovými stroji. Algoritmicky vyčíslitelné funkce
4. Rozhodnutelné a částečně rozhodnutelné jazyky a jejich vlastnosti
5. m-převoditelnost a m-úplné jazyky
6. Riceova věta
7. Nedeterministické Turingovy stroje, základní třídy složitosti, třídy P, NP, PSPACE, $\mathrm{EXPTIME}$
8. Savičova věta
9. Věty o deterministické prostorové a časové hierarchii
10. Polynomiální převoditelnost problémů, pojmy NP-těžkosti a NP-úplnosti
11. Cookova-Levinova věta, příklady NP-úplných problémů, důkazy NP-úplnosti
12. Třídy $\mathrm{co\text{-}NP}$ a $\#\mathrm{P}$
13. Parametrizované algoritmy, třída FPT
14. Hypotézy o exponenciálním čase (ETH, SETH) a podmíněné dolní odhady
15. Složitost a kryptografie

Literatura

Obojí

- Sipser, M. *Introduction to the Theory of Computation.* Vol. 2. Boston: Thomson Course Technology, 2006.
- Mé poznámky na stránce k předmětu (http://ktiml.mff.cuni.cz/~kucerap/NTIN090/)

Vyčíslitelnost

- Demuth O., Kryl R., Kučera A.: *Teorie algoritmů I, II.* SPN, 1984, 1989
- Soare R.I.: *Recursively enumerable sets and degrees.* Springer-Verlag, 1987
- Odifreddi P.: *Classical recursion theory*, North-Holland, 1989

Složitost

- Garey, Johnson: *Computers and intractability — a guide to the theory of NP-completeness*, W.H. Freeman 1978
- Arora S., Barak B.: *Computational Complexity: A Modern Approach*. Cambridge University Press 2009.

Motivační otázky

1. Co je to algoritmus?
2. Co všechno lze pomocí algoritmů spočítat?
3. Dokáží algoritmy vyřešit všechny úlohy a problémy?
4. Jak poznat, že pro řešení zadané úlohy nelze sestrojit žádným algoritmus?
5. Jaké algoritmy jsou „rychlé“ a jaké problémy jimi můžeme řešit?
6. Jaký je rozdíl mezi časem a prostorem?
7. Které problémy jsou lehké a které těžké? A jak je poznat?
8. Které parametry jsou příčinou toho, že je daný problém těžký?
9. Jak využít toho, že některé problémy neumíme rychle vyřešit?

Lehký úvod do teorie algoritmů

Programátorská soutěž

Soutěž

Každý, kdo nám pošle program, který vypíše `Hello, world!`, dostane od nás klíčenku!

První program

Jako první dostaneme následující program (v jazyce C).

helloworld.c

```c
#include <stdio.h>

int main(int argc, char *argv[])
{
    printf("Hello, world\n");
    return 0;
}
```

Zřejmě správný program, posíláme klíčenku.

Druhý program, který jsme dostali

helloworld2.c

```
#include <stdio.h>

int exp(int x, int n)
/* Vrátí n-tou mocninu x (x^n)*/
{
   int pow, j;
   pow=1;
   for (j=1; j<=n; ++j)
   {
      pow *= x;
   }
   return pow;
}
```

Druhý program, který jsme dostali

```
int main(int argc, char *argv[]) {
    int n, total, x, y, z;
    scanf("%d", &n);
    total=3;
    while (1) {
        for (x=1; x<=total-2; ++x) {
            for (y=1; y<=total-x-1; ++y) {
                z=total-x-y;
                if (exp(x,n)+exp(y,n)==exp(z,n)) {
                    printf("Hello, world\n");
                    return 0;
                }
            }
        }
        ++total;
    }
}
```

Za jakých podmínek vypíše program `helloworld2` jako prvních dvanáct znaků na výstup `Hello, world` a zastaví se?

…právě když $x^n + y^n = z^n$ pro nějaké $x, y, z \geq 1$.

…právě když `scanf` načte číslo $n \in \{1, 2\}$.

Pro $n > 2$ program `helloworld2` neskončí.

K důkazu potřebujeme velkou Fermatovu větu!

Automatické vyhodnocování soutěže

Kontrola programů je těžká.

Napišme si program, který je provede za nás!

Jaký problém za nás má ten program řešit?

Problém Helloworld

Helloworld

Instance: Zdrojový kód programu P v jazyce C a vstup I.

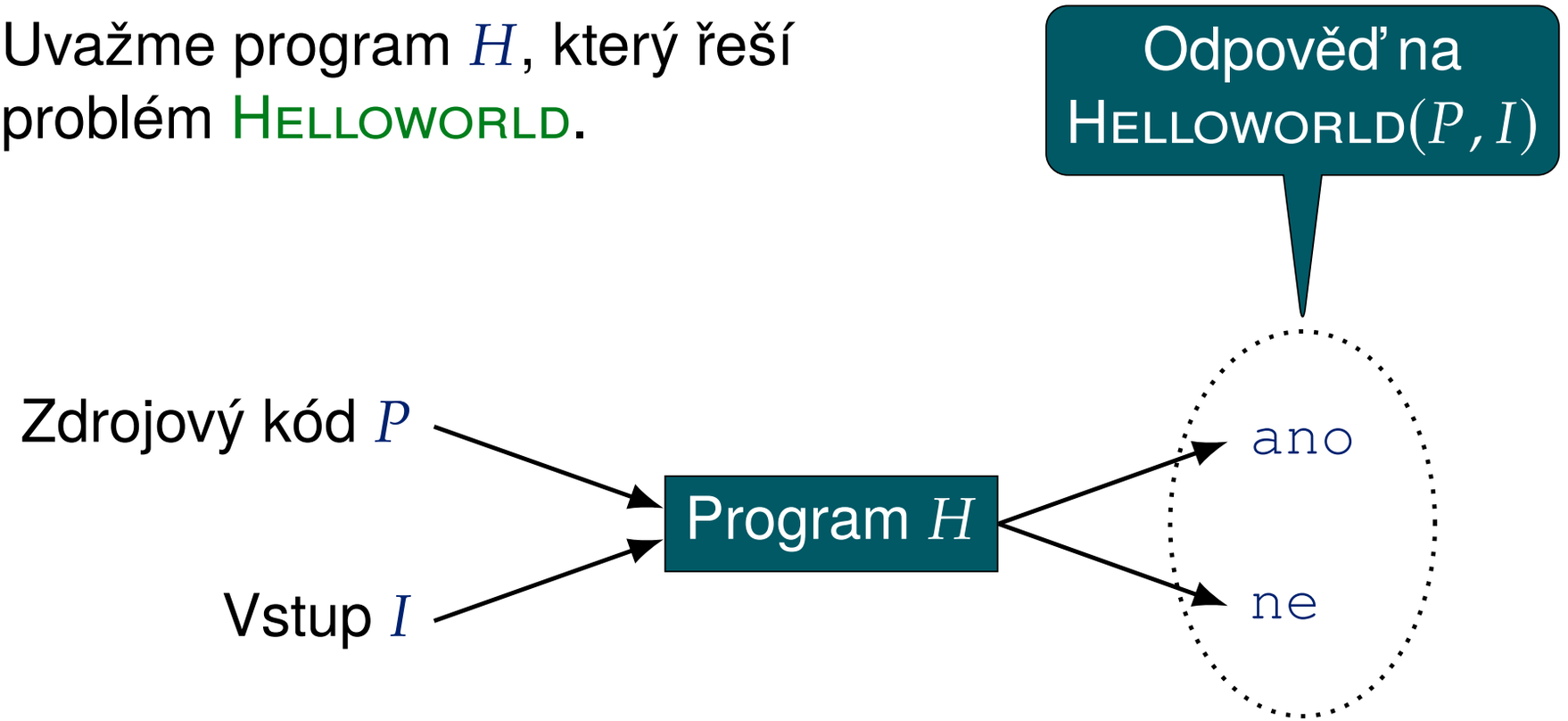
Otázka: Vypíše P se vstupem I jako prvních 12 znaků svého výstupu `Hello, world`? (Nevyžadujeme zastavení.)

Lze napsat program v jazyce C, který ověří, zda dvojice P, I splňuje podmínku problému Helloworld?

Ukážeme si, že nikoli.

Nerozhodnutelnost HELLOWORLD

Uvažme program H, který řeší problém HELLOWORLD.



Zjednodušující předpoklady

- Vstup je předáván programu P i H na standardní vstup a je čten výhradně funkcí `scanf`
- Výstup je realizován na standardní výstup, a to výhradně voláním funkce `printf`

Pozdrav místo odmítnutí

Upravíme program H na H_1 tak, aby místo `ne` psal `Hello, world`.

- Vypíše-li H jako první znak `n`, víme, že nakonec vypíše `ne`
- Odpovídající `printf` upravíme tak, aby rovnou vypsalo `Hello, world`

Co řekne H_1 o sobě?

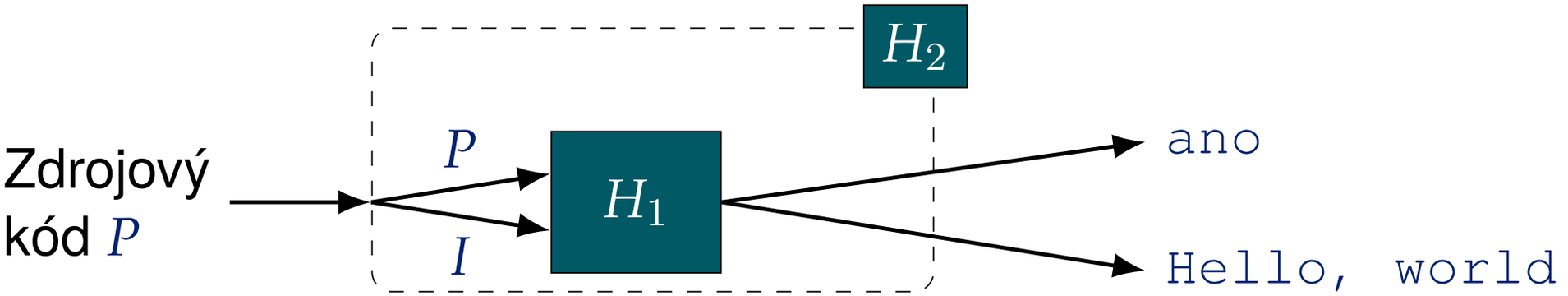
Co je program H_1 schopen říci sám o sobě?

H_1 očekává na vstupu kódy programů s jedním vstupním souborem, ale H_1 sám očekává dva vstupní soubory.

- H_1 upravíme tak na H_2, aby očekával jen jeden vstupní soubor P
- P je použit i jako vstup I v H_2

Dva vstupy v jednom

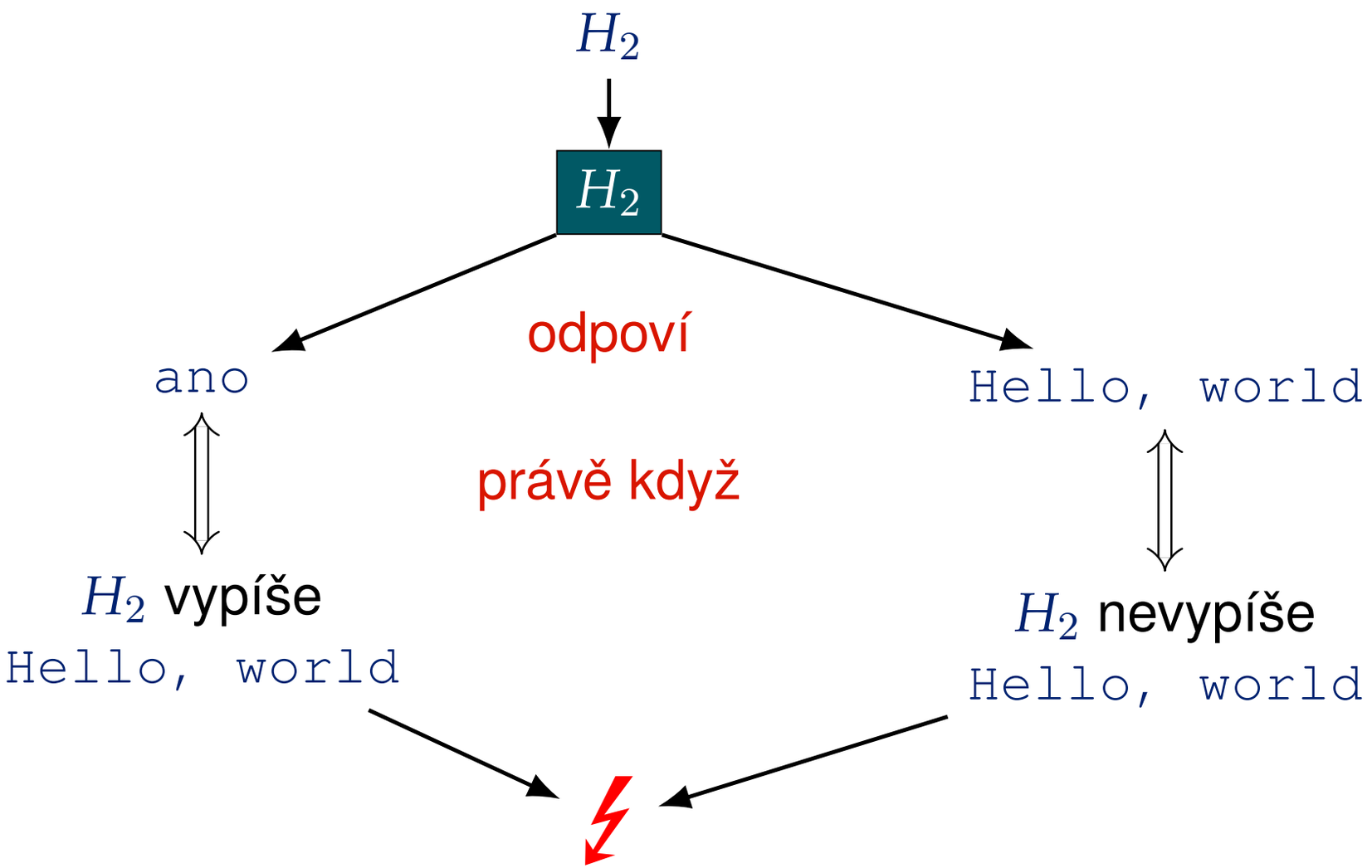
Program H_2 očekává jeden vstupní soubor, který předloží programu H_1 jako oba vstupní soubory, tedy jako zdrojový kód P i jako vstup I.



1. Program H_2 nejprve načte celý vstup a uloží jej v poli A, které alokuje v paměti (např. pomocí `malloc`).
2. Poté program H_2 simuluje práci H_1, přičemž:
   a. Ve chvíli, kdy H_1 čte vstup (pomocí `scanf`), H_2 místo čtení přistoupí do pole A (tj. nahradí `scanf` pomocí čtení z A).
   b. Pomocí dvou ukazatelů do pole A si H_2 pamatuje, kolik z P a I program H_1 přečetl (`scanf` čte popořadě).

Pokud se H_2 zamyslí sám nad sebou

Dostane-li H_2 na vstupu zdrojový kód H_2



Co z toho vyplývá?

- $\Rightarrow$ Program H_2 nemůže existovat.
- $\Rightarrow$ Tedy ani program H_1 nemůže existovat.
- $\Rightarrow$ Tedy ani program H nemůže existovat.
- $\Rightarrow$ Problém HELLOWORLD nelze vyřešit žádným programem v jazyku C (a je tedy algoritmicky neřešitelný).

Programy si budeme muset kontrolovat ručně…

Volání funkce `foo`

Volání funkce `foo`

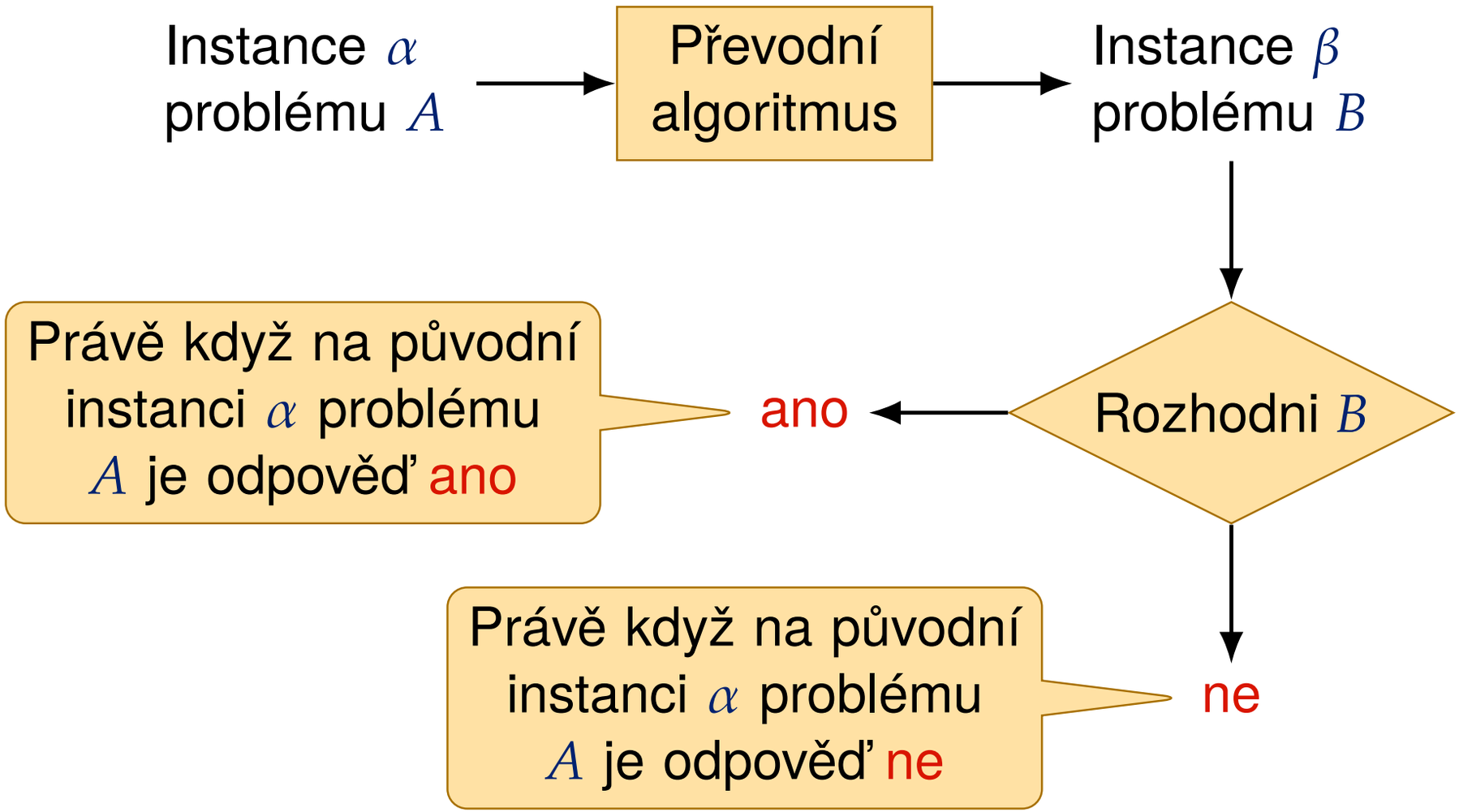
Instance: Zdrojový kód programu Q v jazyce C a vstup V.

Otázka: Zavolá program Q při běhu nad vstupem V funkci jménem `foo`?

- Chceme ukázat, že problém Volání funkce `foo` je algoritmicky nerozhodnutelný.
- Ukážeme, že kdybychom uměli rozhodnout problém Volání funkce `foo`, uměli bychom rozhodnout i problém Helloworld.

Lehký úvod do převoditelnosti

Jsme-li pomocí problému B schopni vyřešit problém A, říkáme, že A je převoditelný na B.



Pozdrav voláním

- Převedeme HELLOWORLD na VOLÁNÍ FUNKCE `FOO`.
- Popíšeme, jak převést
  - instanci HELLOWORLD (program P a vstup I)
  - na instanci VOLÁNÍ FUNKCE `FOO` (program Q a vstup V).
- Musíme přitom zabezpečit, aby platilo, že

program P se vstupem I jako prvních dvanáct znaků svého výstupu vypíše `Hello, world`,

právě když

program Q se vstupem V zavolá funkci jménem `foo`.

Problém VOLÁNÍ FUNKCE `FOO` je algoritmicky nerozhodnutelný.

Jak převést pozdrav na volání

Vstupem převodního algoritmu je program P a vstupní soubor I.

1. Je-li v P funkce `foo`, přejmenujeme ji i všechna její volání na dosud nepoužité jméno (refaktoring, nový program nazveme P_1)
2. K programu P_1 přidáme funkci `foo`, funkce nic nedělá a není volána ($\rightarrow P_2$)
3. Upravíme P_2 tak, aby si pamatoval prvních dvanáct znaků, které vypíše a uložil je v poli A ($\rightarrow P_3$)
4. Upravíme P_3 tak, že použije-li příkaz pro výstup, zkontroluje pole A, je-li prvních dvanáct znaků rovno `Hello, world`. Pokud ano, zavolá funkci `foo`
5. Tím jsme zkonstruovali výsledný program Q, vstup $V = I$

Nevýhody jazyka C pro teorii algoritmů

- Jazyk C je příliš komplikovaný
- Museli bychom definovat výpočetní model (tj. zobecněný počítač), který bude programy v jazyce C interpretovat
- V době vzniku teorie nebyly procedurální jazyky k dispozici, proto je teorie v literatuře obvykle popisovaná tradičnějšími prostředky
- Potřebujeme výpočetní model dostatečně jednoduchý, aby jej bylo lze snadno popsat, současně dostatečně silný, aby byl schopen zachytit to, co intuitivně chápeme pod pojmem algoritmus

Trocha historie …

10. Hilbertův problém

V roce 1900 zformuloval David Hilbert 23 problémů, desátý z nich lze zformulovat takto:

Existuje postup, který by po konečném počtu operací zjistil, zda polynom více proměnných s celočíselnými koeficienty má celočíselný kořen?

Aby bylo možné zodpovědět tuto otázku, je potřeba mít formální definici pojmu algoritmu a efektivní vyčíslitelnosti.

Intuitivně: Algoritmus je konečná posloupnost jednoduchých instrukcí, která vede k řešení zadané úlohy.

Churchova teze

V roce 1934 navrhl Alonzo Church následující tezi:

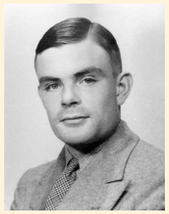
Efektivně vyčíslitelné funkce jsou právě ty, které jsou definované v λ-kalkulu.

Tuto tezi později (1936) upravil na

Efektivně vyčíslitelné funkce jsou právě částečně rekurzivní funkce.

Turingova teze

V roce 1936 publikoval Alan Turing následující tezi



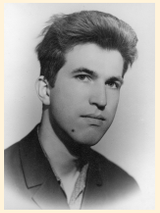
Ke každému algoritmu v intuitivním smyslu existuje ekvivalentní Turingův stroj.

- Zmíněné výpočetní modely (λ-kalkulus, částečně rekurzivní funkce, Turingovy stroje) jsou navzájem ekvivalentní co do výpočetní síly.
- Obvykle se této tezi říká Churchova-Turingova.

10. Hilbertův problém

Existuje postup, který by po konečném počtu operací zjistil, zda polynom více proměnných s celočíselnými koeficienty má celočíselný kořen?

V roce 1970 dal na tuto otázku Yuri Matijasevič negativní odpověď.



Neexistuje algoritmus, který by zjistil, zda daný polynom více proměnných s celočíselnými koeficienty má celočíselný kořen.

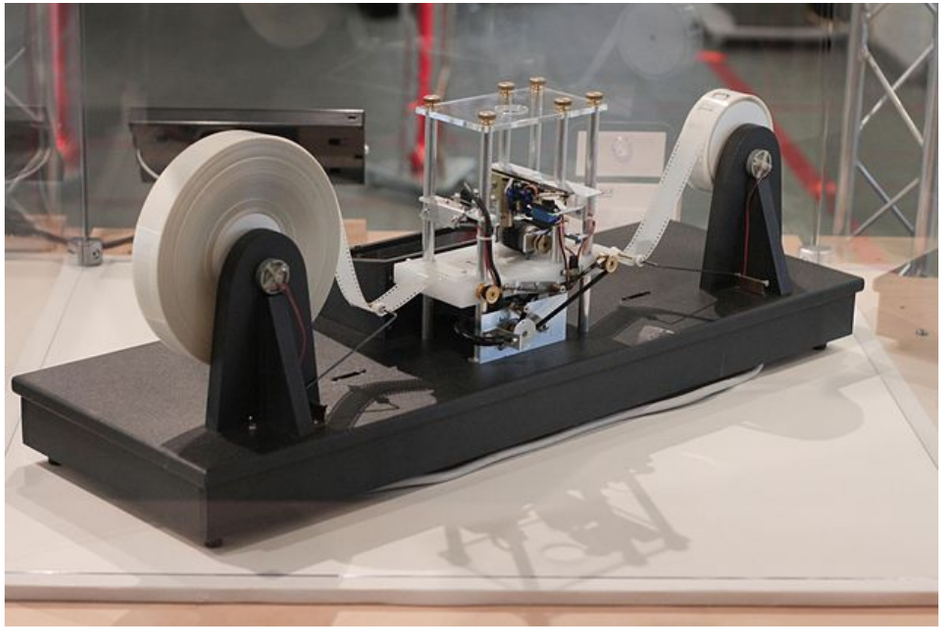
Ekvivalentní modely

Podle Churchovy-Turingovy teze je algoritmus ekvivalentní …

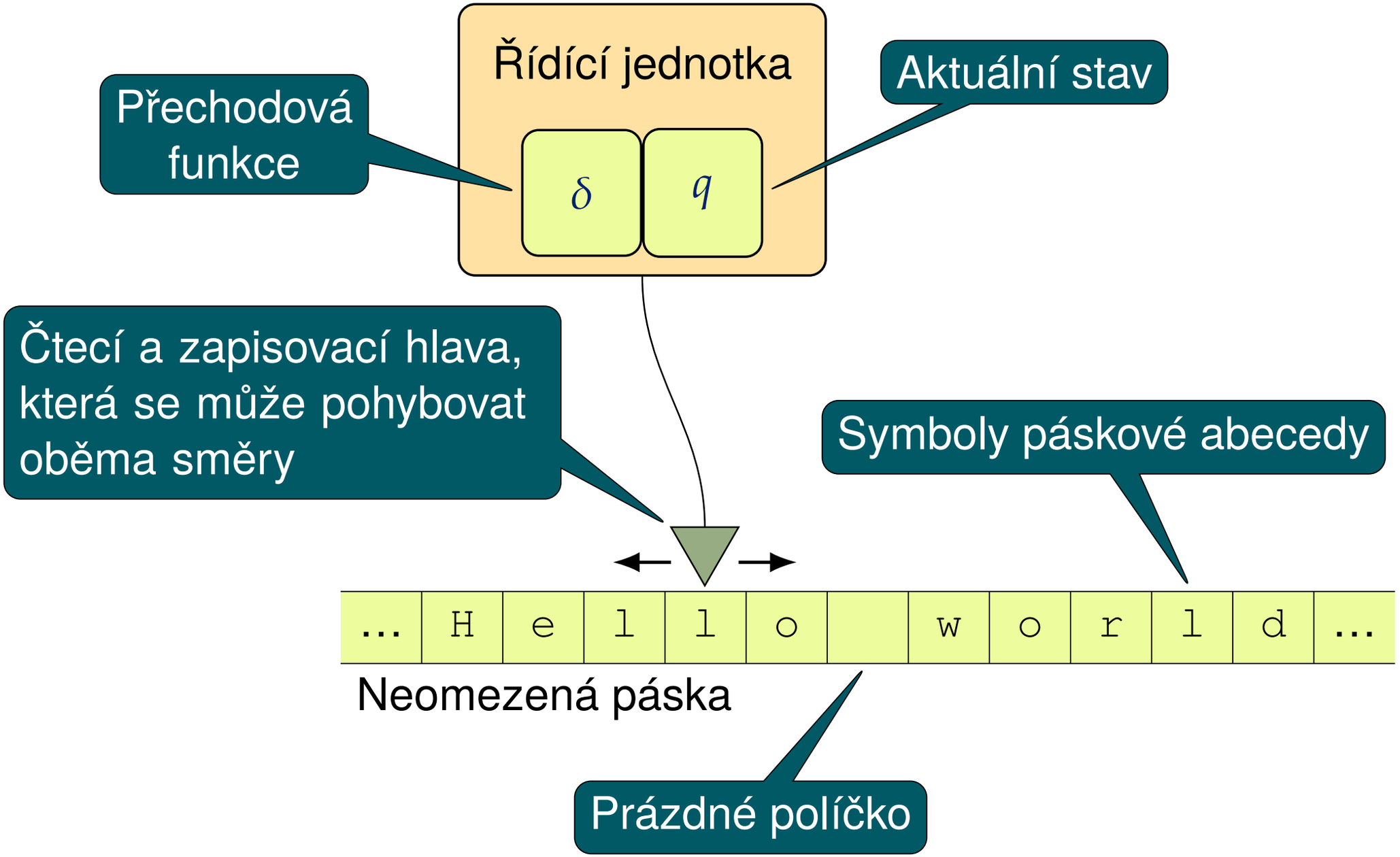
- popisu Turingova stroje,
- programu pro RAM,
- odvození částečně rekurzivní funkce,
- odvození funkce v λ-kalkulu,
- programu ve vyšším programovacím jazyce, jako je C, Pascal, Java, Basic apod.,
- programu ve funkcionálním jazyce jako je Lisp, Haskell apod.

Ve všech těchto modelech jsme schopni počítat tytéž funkce, řešit tytéž problémy a úlohy.

Turingovy stroje



Turingův stroj



Turingův stroj (definice)

(Jednopáskový deterministický) Turingův stroj (TS) M je pětice

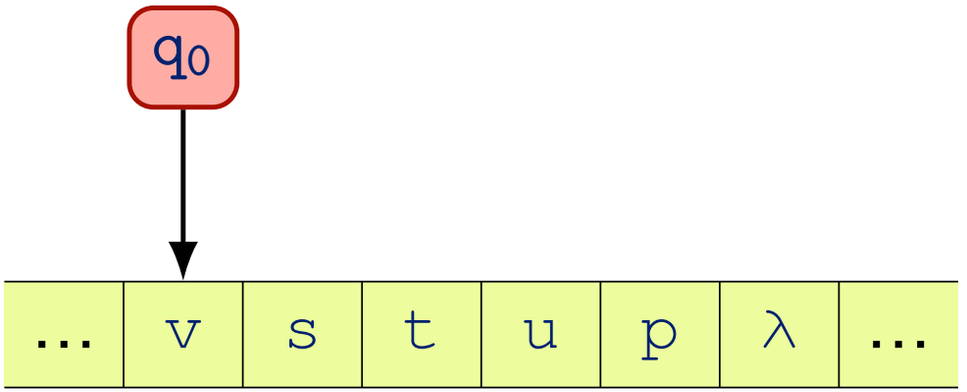
$$M = (Q, \Sigma, \delta, q_0, F)$$

- Q je konečná množina stavů
- Σ je konečná pásková abeceda, která obsahuje znak λ pro prázdné políčko
  - Často budeme neformálně rozlišovat páskovou (vnitřní) a vstupní (vnější) abecedu
- $\delta : Q \times \Sigma \to Q \times \Sigma \times \{R, N, L\} \cup \{\perp\}$ je přechodová funkce, kde $\perp$ označuje nedefinovaný přechod.
- $q_0 \in Q$ je počáteční stav
- $F \subseteq Q$ je množina přijímajících stavů

Počáteční konfigurace

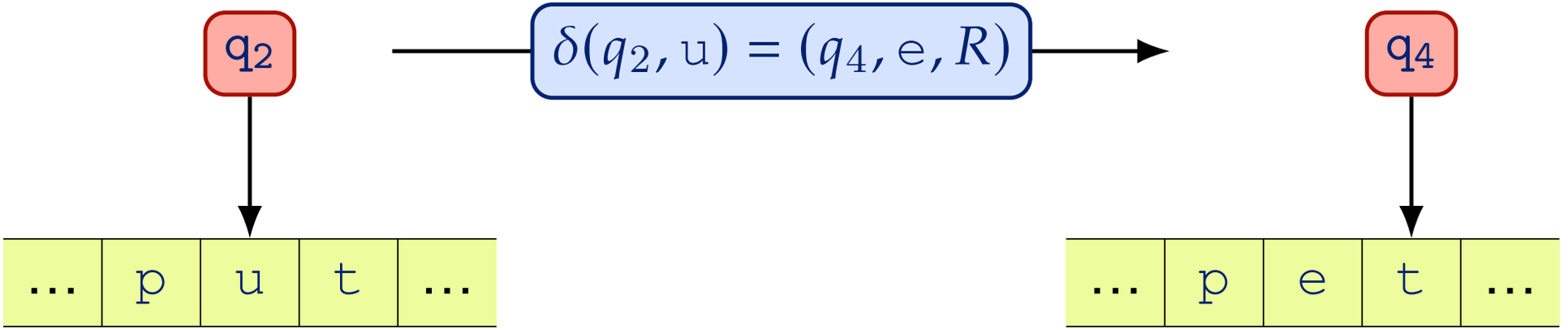
Výpočet zahajuje TS M v počáteční konfiguraci

- Počáteční stav q_0
- Na pásce je zapsané vstupní slovo
  - *nesmí obsahovat prázdné políčko*
- hlava nad nejlevějším symbolem vstupu



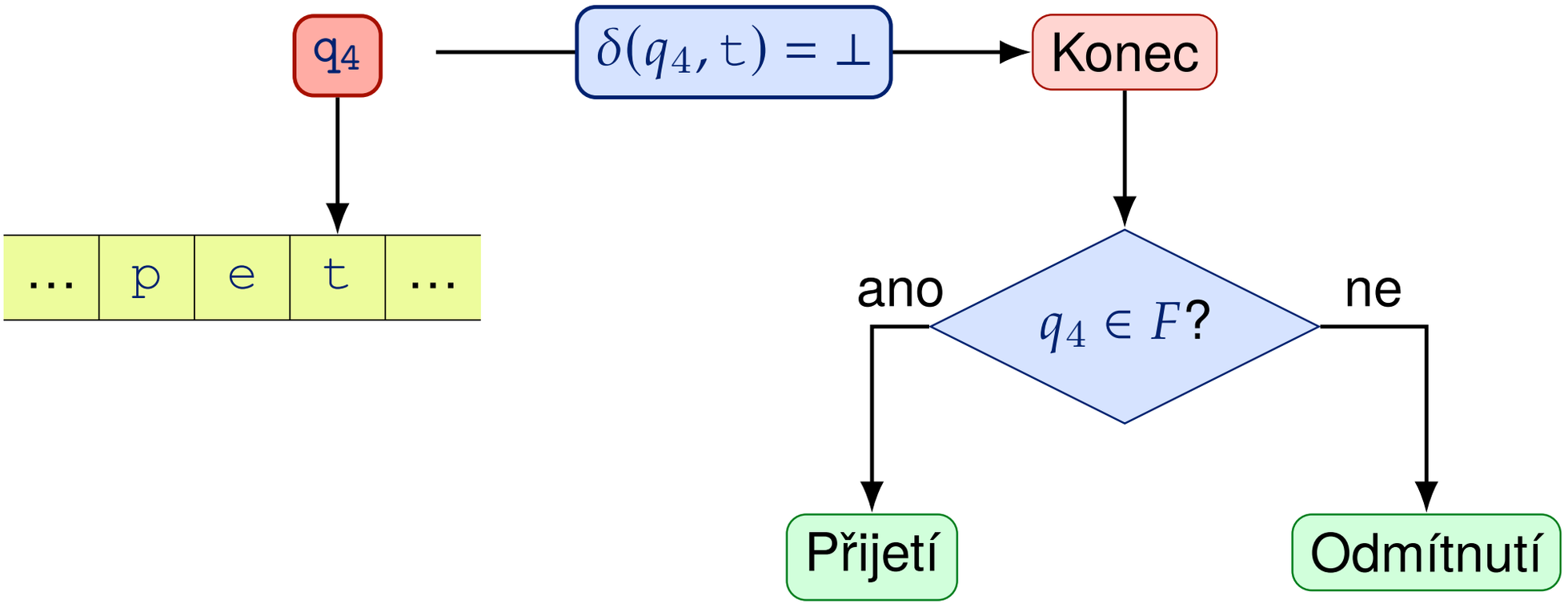
Krok Turingova stroje

- Nechť M je ve stavu $q \in Q$ a pod hlavou je symbol $a \in \Sigma$:
- je-li $\delta(q, a) = (q', a', Z)$, kde $q' \in Q$, $a' \in \Sigma$ a $Z \in \{L, N, R\}$, pak M
  - přejde do stavu q',
  - zapíše na pozici hlavy symbol a' a
  - pohne hlavou doleva (pokud $Z = L$), doprava ($Z = R$), nebo hlava zůstane na místě ($Z = N$).



Konec výpočtu

- Nechť M je ve stavu $q \in Q$ a pod hlavou je symbol $a \in \Sigma$:
- je-li $\delta(q, a) = \bot$, pak výpočet M končí
  - Je-li $q \in F$, je výpočet přijímající
  - Jinak je výpočet zamítající



Jazyk palindromů

$$\mathrm{PAL} = \{w \in \{\mathtt{a},\mathtt{b}\}^* \mid w = w^R\}$$

- Jazyk je množina slov nad konečnou abecedou Σ
  - zde $\Sigma = \{\mathtt{a},\mathtt{b}\}$
- w^R označuje zrcadlové otočení řetězce w
  - například $(\mathtt{abbabab})^R = \mathtt{bababba}$
- Chceme sestavit Turingův stroj M, který rozhoduje PAL
- Výpočet M se vstupem w má být
  - přijímající, právě když $w = w^R$ a
  - zamítající, právě když $w \neq w^R$

Algoritmus TS rozhodujícího jazyk palindromů

$$\mathrm{PAL} = \{w \in \{\mathtt{a},\mathtt{b}\}^* \mid w = w^R\}$$

Práce TS M se vstupem $w = w_1 \ldots w_n$

1 **if** $w = \varepsilon$ **then** // ε označuje prázdný řetězec
2 přijmi
3 Zapamatuj si ve stavu první znak w_1
4 Přesuň hlavu na poslední políčko vstupu w_n
5 **if** $w_1 \neq w_n$ **then**
6 odmítni
7 Smaž poslední znak
8 Přesuň hlavu na první políčko vstupu
9 Smaž první znak (pokud nějaký zbyl)
10 Pohni hlavou doprava
11 **goto** 1 // zbývá $w_2 \ldots w_{n-1}$

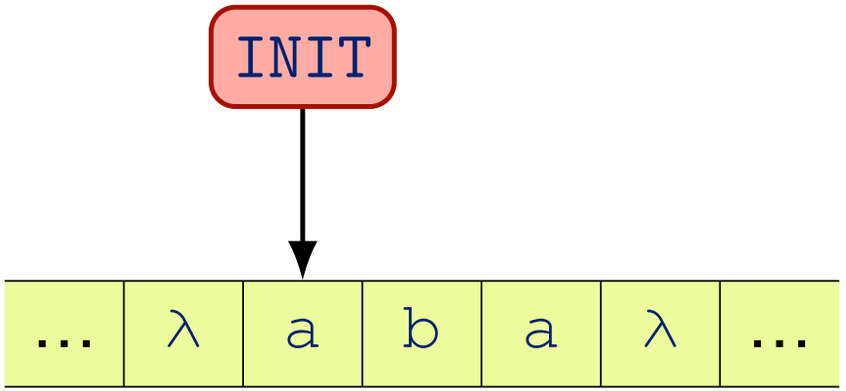
Palindromy — Turingův stroj

- Stavy $Q = \{\texttt{INIT}, \texttt{MEM\_A}, \texttt{MEM\_B}, \texttt{CHECK\_A}, \texttt{CHECK\_B}, \texttt{BACK}, \texttt{ERASE}, \texttt{ACCEPT}\}$
- Abeceda $\Sigma = \{\lambda, \texttt{a}, \texttt{b}\}$
- Počáteční stav $\texttt{INIT}$
- Přijímající stav $\texttt{ACCEPT}$

q, c	$\rightarrow$	q', c', Z
$\texttt{INIT}, \lambda$	$\rightarrow$	$\texttt{ACCEPT}, \lambda, N$
$\texttt{INIT}, \texttt{a}$	$\rightarrow$	$\texttt{MEM\_A}, \texttt{a}, R$
$\texttt{INIT}, \texttt{b}$	$\rightarrow$	$\texttt{MEM\_B}, \texttt{b}, R$
$\texttt{MEM\_A}, \texttt{a}$	$\rightarrow$	$\texttt{MEM\_A}, \texttt{a}, R$
$\texttt{MEM\_A}, \texttt{b}$	$\rightarrow$	$\texttt{MEM\_A}, \texttt{b}, R$
$\texttt{MEM\_A}, \lambda$	$\rightarrow$	$\texttt{CHECK\_A}, \lambda, L$
$\texttt{MEM\_B}, \texttt{a}$	$\rightarrow$	$\texttt{MEM\_B}, \texttt{a}, R$
$\texttt{MEM\_B}, \texttt{b}$	$\rightarrow$	$\texttt{MEM\_B}, \texttt{b}, R$
$\texttt{MEM\_B}, \lambda$	$\rightarrow$	$\texttt{CHECK\_B}, \lambda, L$

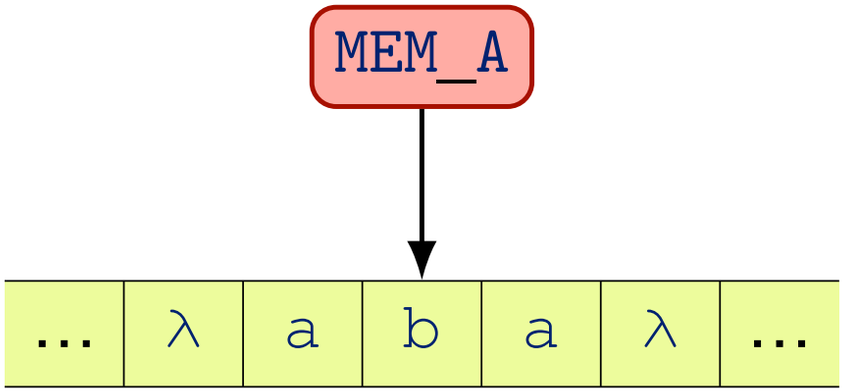
q, c	$\rightarrow$	q', c', Z
$\texttt{CHECK\_A}, \texttt{a}$	$\rightarrow$	$\texttt{BACK}, \lambda, L$
$\texttt{CHECK\_B}, \texttt{b}$	$\rightarrow$	$\texttt{BACK}, \lambda, L$
$\texttt{BACK}, \texttt{a}$	$\rightarrow$	$\texttt{BACK}, \texttt{a}, L$
$\texttt{BACK}, \texttt{b}$	$\rightarrow$	$\texttt{BACK}, \texttt{b}, L$
$\texttt{BACK}, \lambda$	$\rightarrow$	$\texttt{ERASE}, \lambda, R$
$\texttt{ERASE}, \texttt{a}$	$\rightarrow$	$\texttt{INIT}, \lambda, R$
$\texttt{ERASE}, \texttt{b}$	$\rightarrow$	$\texttt{INIT}, \lambda, R$
$\texttt{ERASE}, \lambda$	$\rightarrow$	$\texttt{ACCEPT}, \lambda, N$

Simulace M se vstupem aba



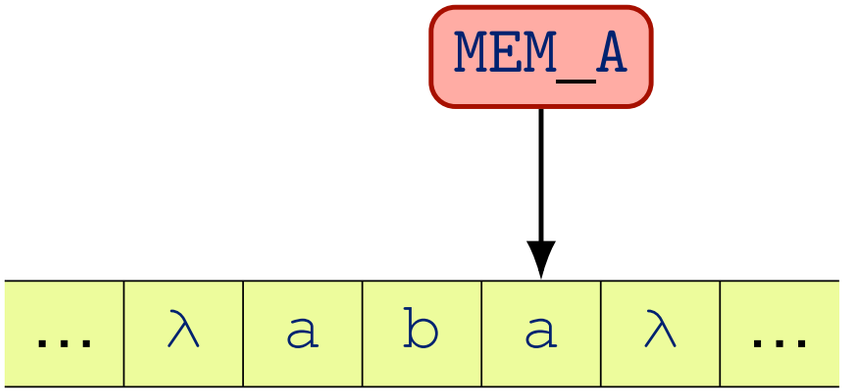
INIT, λ	→	ACCEPT, λ, N
INIT, a	→	MEM_A, a, R
INIT, b	→	MEM_B, b, R
MEM_A, a	→	MEM_A, a, R
MEM_A, b	→	MEM_A, b, R
MEM_A, λ	→	CHECK_A, λ, L
MEM_B, a	→	MEM_B, a, R
MEM_B, b	→	MEM_B, b, R
MEM_B, λ	→	CHECK_B, λ, L
CHECK_A, a	→	BACK, λ, L
CHECK_B, b	→	BACK, λ, L
BACK, a	→	BACK, a, L
BACK, b	→	BACK, b, L
BACK, λ	→	ERASE, λ, R
ERASE, a	→	INIT, λ, R
ERASE, b	→	INIT, λ, R
ERASE, λ	→	ACCEPT, λ, N

Simulace M se vstupem aba



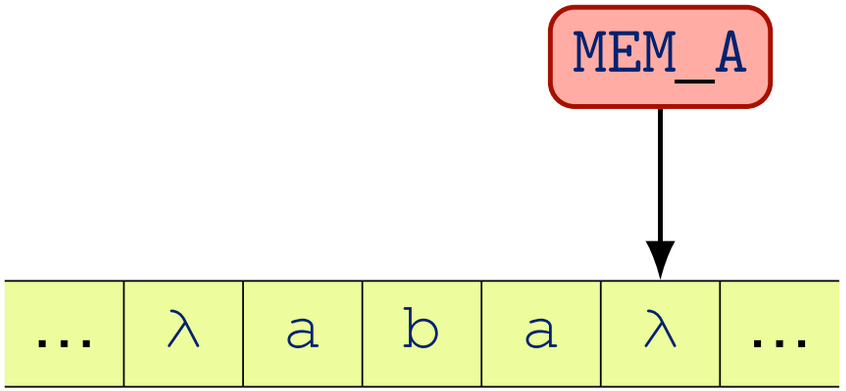
INIT, λ	→	ACCEPT, λ, N
INIT, a	→	MEM_A, a, R
INIT, b	→	MEM_B, b, R
MEM_A, a	→	MEM_A, a, R
MEM_A, b	→	MEM_A, b, R
MEM_A, λ	→	CHECK_A, λ, L
MEM_B, a	→	MEM_B, a, R
MEM_B, b	→	MEM_B, b, R
MEM_B, λ	→	CHECK_B, λ, L
CHECK_A, a	→	BACK, λ, L
CHECK_B, b	→	BACK, λ, L
BACK, a	→	BACK, a, L
BACK, b	→	BACK, b, L
BACK, λ	→	ERASE, λ, R
ERASE, a	→	INIT, λ, R
ERASE, b	→	INIT, λ, R
ERASE, λ	→	ACCEPT, λ, N

Simulace M se vstupem aba



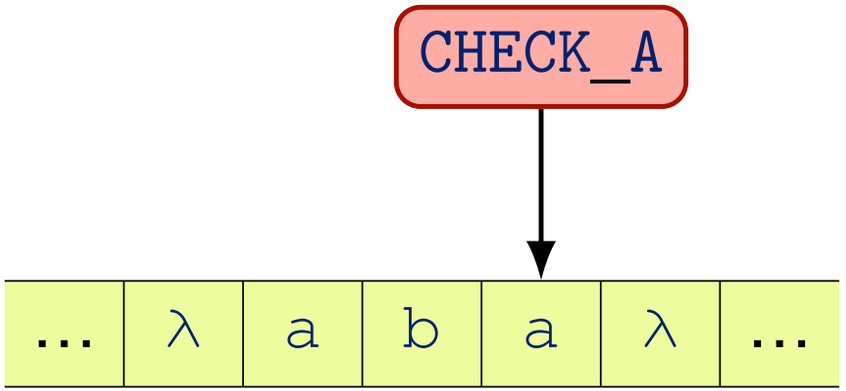
INIT, λ	→	ACCEPT, λ, N
INIT, a	→	MEM_A, a, R
INIT, b	→	MEM_B, b, R
MEM_A, a	→	MEM_A, a, R
MEM_A, b	→	MEM_A, b, R
MEM_A, λ	→	CHECK_A, λ, L
MEM_B, a	→	MEM_B, a, R
MEM_B, b	→	MEM_B, b, R
MEM_B, λ	→	CHECK_B, λ, L
CHECK_A, a	→	BACK, λ, L
CHECK_B, b	→	BACK, λ, L
BACK, a	→	BACK, a, L
BACK, b	→	BACK, b, L
BACK, λ	→	ERASE, λ, R
ERASE, a	→	INIT, λ, R
ERASE, b	→	INIT, λ, R
ERASE, λ	→	ACCEPT, λ, N

Simulace M se vstupem aba



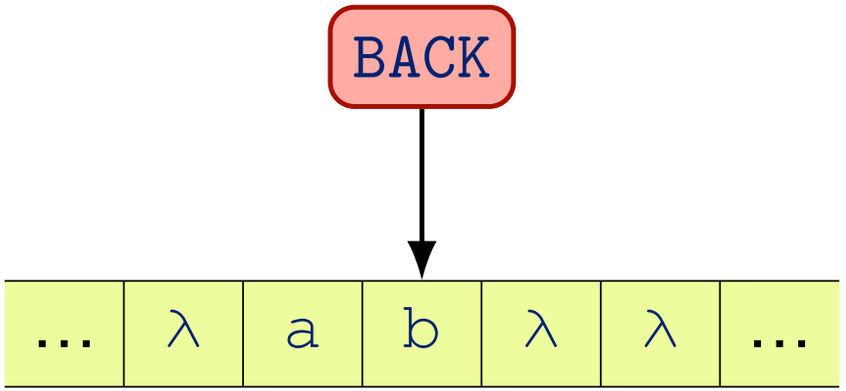
INIT, λ	→	ACCEPT, λ, N
INIT, a	→	MEM_A, a, R
INIT, b	→	MEM_B, b, R
MEM_A, a	→	MEM_A, a, R
MEM_A, b	→	MEM_A, b, R
MEM_A, λ	→	CHECK_A, λ, L
MEM_B, a	→	MEM_B, a, R
MEM_B, b	→	MEM_B, b, R
MEM_B, λ	→	CHECK_B, λ, L
CHECK_A, a	→	BACK, λ, L
CHECK_B, b	→	BACK, λ, L
BACK, a	→	BACK, a, L
BACK, b	→	BACK, b, L
BACK, λ	→	ERASE, λ, R
ERASE, a	→	INIT, λ, R
ERASE, b	→	INIT, λ, R
ERASE, λ	→	ACCEPT, λ, N

Simulace M se vstupem aba



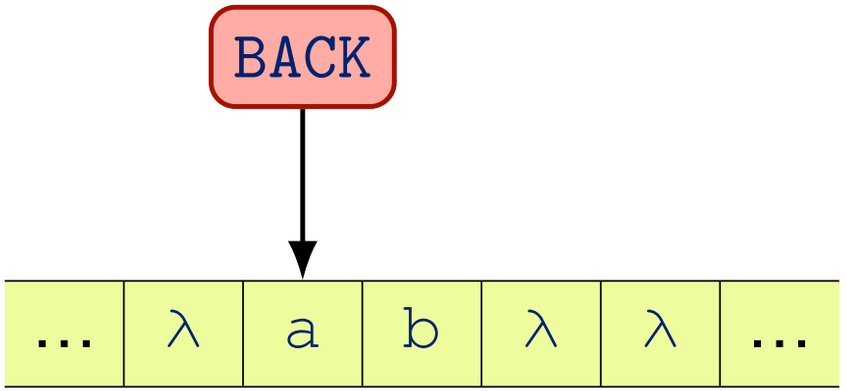
INIT, λ	→	ACCEPT, λ, N
INIT, a	→	MEM_A, a, R
INIT, b	→	MEM_B, b, R
MEM_A, a	→	MEM_A, a, R
MEM_A, b	→	MEM_A, b, R
MEM_A, λ	→	CHECK_A, λ, L
MEM_B, a	→	MEM_B, a, R
MEM_B, b	→	MEM_B, b, R
MEM_B, λ	→	CHECK_B, λ, L
CHECK_A, a	→	BACK, λ, L
CHECK_B, b	→	BACK, λ, L
BACK, a	→	BACK, a, L
BACK, b	→	BACK, b, L
BACK, λ	→	ERASE, λ, R
ERASE, a	→	INIT, λ, R
ERASE, b	→	INIT, λ, R
ERASE, λ	→	ACCEPT, λ, N

Simulace M se vstupem aba



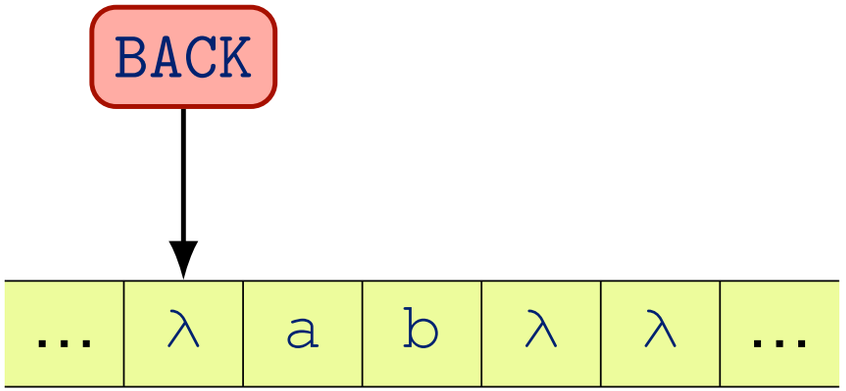
INIT, λ	→	ACCEPT, λ, N
INIT, a	→	MEM_A, a, R
INIT, b	→	MEM_B, b, R
MEM_A, a	→	MEM_A, a, R
MEM_A, b	→	MEM_A, b, R
MEM_A, λ	→	CHECK_A, λ, L
MEM_B, a	→	MEM_B, a, R
MEM_B, b	→	MEM_B, b, R
MEM_B, λ	→	CHECK_B, λ, L
CHECK_A, a	→	BACK, λ, L
CHECK_B, b	→	BACK, λ, L
BACK, a	→	BACK, a, L
BACK, b	→	BACK, b, L
BACK, λ	→	ERASE, λ, R
ERASE, a	→	INIT, λ, R
ERASE, b	→	INIT, λ, R
ERASE, λ	→	ACCEPT, λ, N

Simulace M se vstupem aba



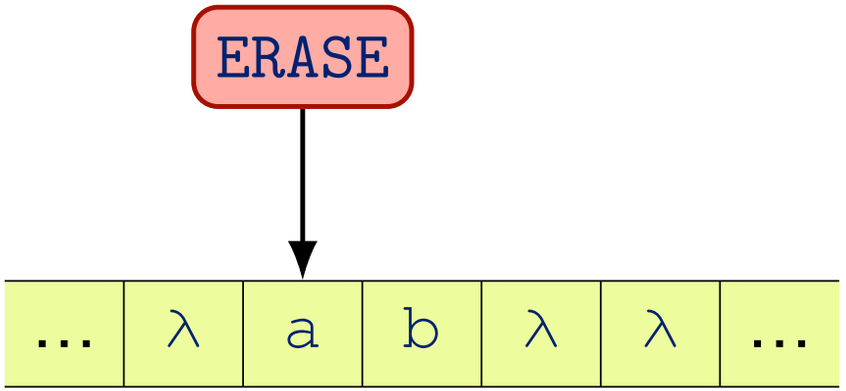
INIT, λ	→	ACCEPT, λ, N
INIT, a	→	MEM_A, a, R
INIT, b	→	MEM_B, b, R
MEM_A, a	→	MEM_A, a, R
MEM_A, b	→	MEM_A, b, R
MEM_A, λ	→	CHECK_A, λ, L
MEM_B, a	→	MEM_B, a, R
MEM_B, b	→	MEM_B, b, R
MEM_B, λ	→	CHECK_B, λ, L
CHECK_A, a	→	BACK, λ, L
CHECK_B, b	→	BACK, λ, L
BACK, a	→	BACK, a, L
BACK, b	→	BACK, b, L
BACK, λ	→	ERASE, λ, R
ERASE, a	→	INIT, λ, R
ERASE, b	→	INIT, λ, R
ERASE, λ	→	ACCEPT, λ, N

Simulace M se vstupem aba



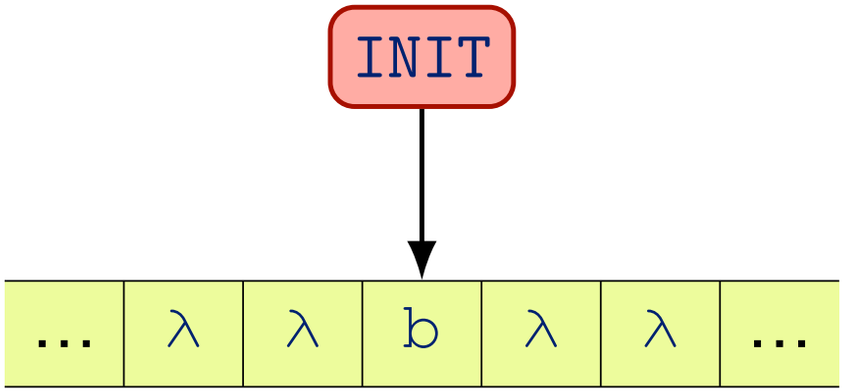
INIT, λ	→	ACCEPT, λ, N
INIT, a	→	MEM_A, a, R
INIT, b	→	MEM_B, b, R
MEM_A, a	→	MEM_A, a, R
MEM_A, b	→	MEM_A, b, R
MEM_A, λ	→	CHECK_A, λ, L
MEM_B, a	→	MEM_B, a, R
MEM_B, b	→	MEM_B, b, R
MEM_B, λ	→	CHECK_B, λ, L
CHECK_A, a	→	BACK, λ, L
CHECK_B, b	→	BACK, λ, L
BACK, a	→	BACK, a, L
BACK, b	→	BACK, b, L
BACK, λ	→	ERASE, λ, R
ERASE, a	→	INIT, λ, R
ERASE, b	→	INIT, λ, R
ERASE, λ	→	ACCEPT, λ, N

Simulace M se vstupem aba



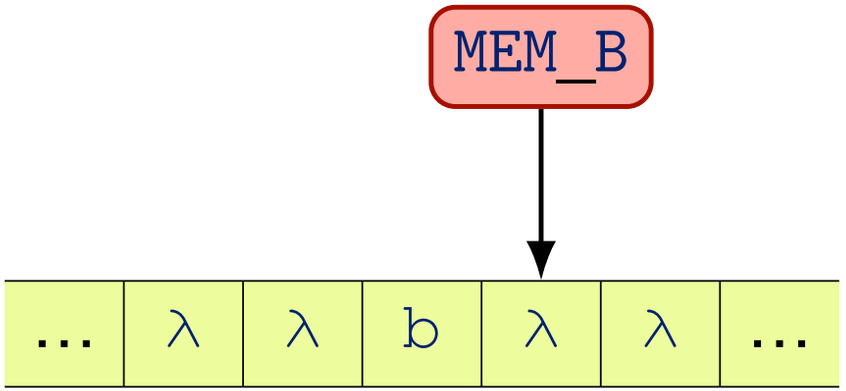
INIT, λ	→	ACCEPT, λ, N
INIT, a	→	MEM_A, a, R
INIT, b	→	MEM_B, b, R
MEM_A, a	→	MEM_A, a, R
MEM_A, b	→	MEM_A, b, R
MEM_A, λ	→	CHECK_A, λ, L
MEM_B, a	→	MEM_B, a, R
MEM_B, b	→	MEM_B, b, R
MEM_B, λ	→	CHECK_B, λ, L
CHECK_A, a	→	BACK, λ, L
CHECK_B, b	→	BACK, λ, L
BACK, a	→	BACK, a, L
BACK, b	→	BACK, b, L
BACK, λ	→	ERASE, λ, R
ERASE, a	→	INIT, λ, R
ERASE, b	→	INIT, λ, R
ERASE, λ	→	ACCEPT, λ, N

Simulace M se vstupem aba



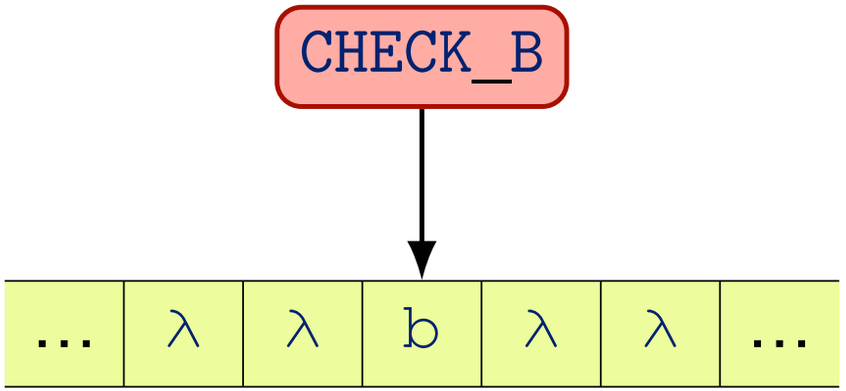
INIT, λ	→	ACCEPT, λ, N
INIT, a	→	MEM_A, a, R
INIT, b	→	MEM_B, b, R
MEM_A, a	→	MEM_A, a, R
MEM_A, b	→	MEM_A, b, R
MEM_A, λ	→	CHECK_A, λ, L
MEM_B, a	→	MEM_B, a, R
MEM_B, b	→	MEM_B, b, R
MEM_B, λ	→	CHECK_B, λ, L
CHECK_A, a	→	BACK, λ, L
CHECK_B, b	→	BACK, λ, L
BACK, a	→	BACK, a, L
BACK, b	→	BACK, b, L
BACK, λ	→	ERASE, λ, R
ERASE, a	→	INIT, λ, R
ERASE, b	→	INIT, λ, R
ERASE, λ	→	ACCEPT, λ, N

Simulace M se vstupem aba



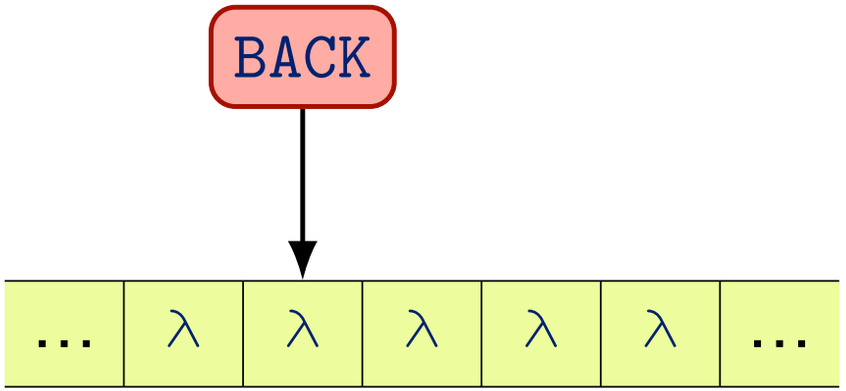
INIT, λ	→	ACCEPT, λ, N
INIT, a	→	MEM_A, a, R
INIT, b	→	MEM_B, b, R
MEM_A, a	→	MEM_A, a, R
MEM_A, b	→	MEM_A, b, R
MEM_A, λ	→	CHECK_A, λ, L
MEM_B, a	→	MEM_B, a, R
MEM_B, b	→	MEM_B, b, R
MEM_B, λ	→	CHECK_B, λ, L
CHECK_A, a	→	BACK, λ, L
CHECK_B, b	→	BACK, λ, L
BACK, a	→	BACK, a, L
BACK, b	→	BACK, b, L
BACK, λ	→	ERASE, λ, R
ERASE, a	→	INIT, λ, R
ERASE, b	→	INIT, λ, R
ERASE, λ	→	ACCEPT, λ, N

Simulace M se vstupem aba



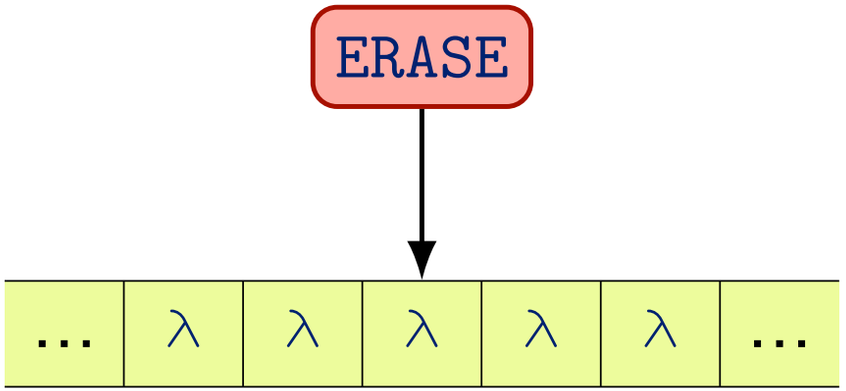
INIT, λ	→	ACCEPT, λ, N
INIT, a	→	MEM_A, a, R
INIT, b	→	MEM_B, b, R
MEM_A, a	→	MEM_A, a, R
MEM_A, b	→	MEM_A, b, R
MEM_A, λ	→	CHECK_A, λ, L
MEM_B, a	→	MEM_B, a, R
MEM_B, b	→	MEM_B, b, R
MEM_B, λ	→	CHECK_B, λ, L
CHECK_A, a	→	BACK, λ, L
CHECK_B, b	→	BACK, λ, L
BACK, a	→	BACK, a, L
BACK, b	→	BACK, b, L
BACK, λ	→	ERASE, λ, R
ERASE, a	→	INIT, λ, R
ERASE, b	→	INIT, λ, R
ERASE, λ	→	ACCEPT, λ, N

Simulace M se vstupem aba



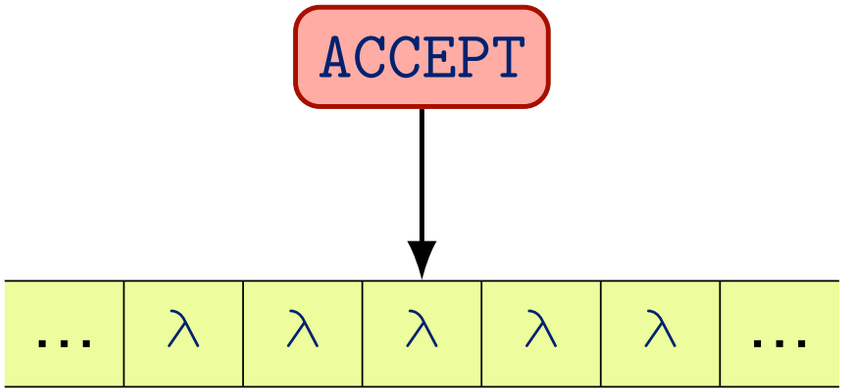
$$
\begin{array}{rcl}
\texttt{INIT},\lambda & \to & \texttt{ACCEPT},\lambda,N \\
\texttt{INIT},\texttt{a} & \to & \texttt{MEM\_A},\texttt{a},R \\
\texttt{INIT},\texttt{b} & \to & \texttt{MEM\_B},\texttt{b},R \\
\texttt{MEM\_A},\texttt{a} & \to & \texttt{MEM\_A},\texttt{a},R \\
\texttt{MEM\_A},\texttt{b} & \to & \texttt{MEM\_A},\texttt{b},R \\
\texttt{MEM\_A},\lambda & \to & \texttt{CHECK\_A},\lambda,L \\
\texttt{MEM\_B},\texttt{a} & \to & \texttt{MEM\_B},\texttt{a},R \\
\texttt{MEM\_B},\texttt{b} & \to & \texttt{MEM\_B},\texttt{b},R \\
\texttt{MEM\_B},\lambda & \to & \texttt{CHECK\_B},\lambda,L \\
\texttt{CHECK\_A},\texttt{a} & \to & \texttt{BACK},\lambda,L \\
\texttt{CHECK\_B},\texttt{b} & \to & \texttt{BACK},\lambda,L \\
\texttt{BACK},\texttt{a} & \to & \texttt{BACK},\texttt{a},L \\
\texttt{BACK},\texttt{b} & \to & \texttt{BACK},\texttt{b},L \\
\texttt{BACK},\lambda & \to & \texttt{ERASE},\lambda,R \\
\texttt{ERASE},\texttt{a} & \to & \texttt{INIT},\lambda,R \\
\texttt{ERASE},\texttt{b} & \to & \texttt{INIT},\lambda,R \\
\texttt{ERASE},\lambda & \to & \texttt{ACCEPT},\lambda,N
\end{array}
$$

Simulace M se vstupem aba



INIT, λ	→	ACCEPT, λ, N
INIT, a	→	MEM_A, a, R
INIT, b	→	MEM_B, b, R
MEM_A, a	→	MEM_A, a, R
MEM_A, b	→	MEM_A, b, R
MEM_A, λ	→	CHECK_A, λ, L
MEM_B, a	→	MEM_B, a, R
MEM_B, b	→	MEM_B, b, R
MEM_B, λ	→	CHECK_B, λ, L
CHECK_A, a	→	BACK, λ, L
CHECK_B, b	→	BACK, λ, L
BACK, a	→	BACK, a, L
BACK, b	→	BACK, b, L
BACK, λ	→	ERASE, λ, R
ERASE, a	→	INIT, λ, R
ERASE, b	→	INIT, λ, R
ERASE, λ	→	ACCEPT, λ, N

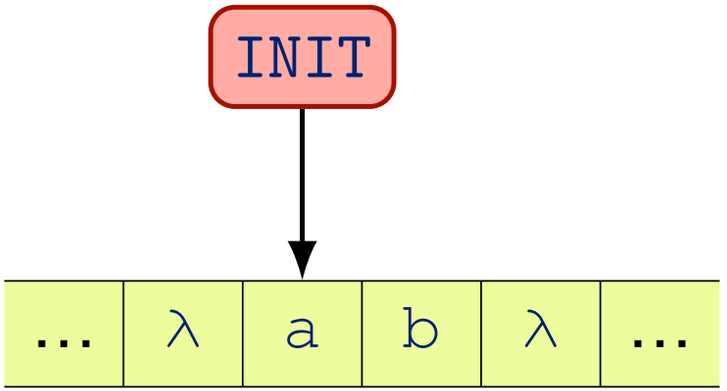
Simulace M se vstupem aba



$$
\begin{array}{rcl}
\texttt{INIT},\lambda & \to & \texttt{ACCEPT},\lambda,N \\
\texttt{INIT},\texttt{a} & \to & \texttt{MEM\_A},\texttt{a},R \\
\texttt{INIT},\texttt{b} & \to & \texttt{MEM\_B},\texttt{b},R \\
\texttt{MEM\_A},\texttt{a} & \to & \texttt{MEM\_A},\texttt{a},R \\
\texttt{MEM\_A},\texttt{b} & \to & \texttt{MEM\_A},\texttt{b},R \\
\texttt{MEM\_A},\lambda & \to & \texttt{CHECK\_A},\lambda,L \\
\texttt{MEM\_B},\texttt{a} & \to & \texttt{MEM\_B},\texttt{a},R \\
\texttt{MEM\_B},\texttt{b} & \to & \texttt{MEM\_B},\texttt{b},R \\
\texttt{MEM\_B},\lambda & \to & \texttt{CHECK\_B},\lambda,L \\
\texttt{CHECK\_A},\texttt{a} & \to & \texttt{BACK},\lambda,L \\
\texttt{CHECK\_B},\texttt{b} & \to & \texttt{BACK},\lambda,L \\
\texttt{BACK},\texttt{a} & \to & \texttt{BACK},\texttt{a},L \\
\texttt{BACK},\texttt{b} & \to & \texttt{BACK},\texttt{b},L \\
\texttt{BACK},\lambda & \to & \texttt{ERASE},\lambda,R \\
\texttt{ERASE},\texttt{a} & \to & \texttt{INIT},\lambda,R \\
\texttt{ERASE},\texttt{b} & \to & \texttt{INIT},\lambda,R \\
\texttt{ERASE},\lambda & \to & \texttt{ACCEPT},\lambda,N
\end{array}
$$

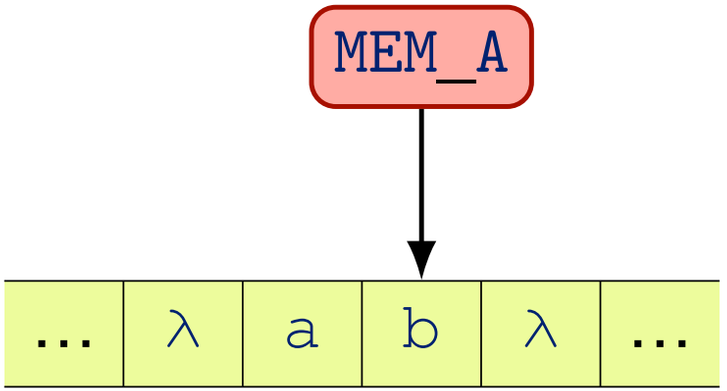
Výpočet skončil, vstup byl přijat

Simulace M se vstupem ab



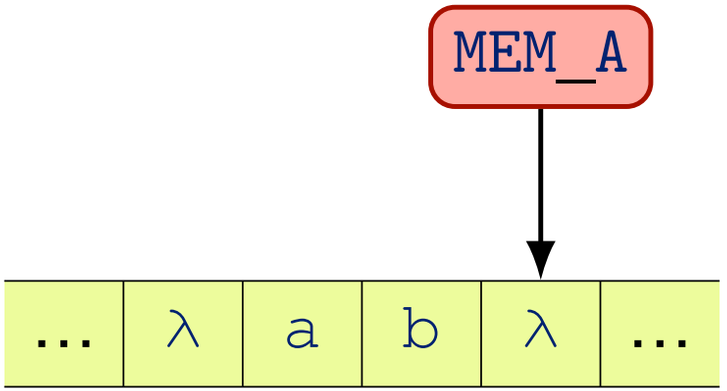
INIT, λ	→	ACCEPT, λ, N
INIT, a	→	MEM_A, a, R
INIT, b	→	MEM_B, b, R
MEM_A, a	→	MEM_A, a, R
MEM_A, b	→	MEM_A, b, R
MEM_A, λ	→	CHECK_A, λ, L
MEM_B, a	→	MEM_B, a, R
MEM_B, b	→	MEM_B, b, R
MEM_B, λ	→	CHECK_B, λ, L
CHECK_A, a	→	BACK, λ, L
CHECK_B, b	→	BACK, λ, L
BACK, a	→	BACK, a, L
BACK, b	→	BACK, b, L
BACK, λ	→	ERASE, λ, R
ERASE, a	→	INIT, λ, R
ERASE, b	→	INIT, λ, R
ERASE, λ	→	ACCEPT, λ, N

Simulace M se vstupem ab



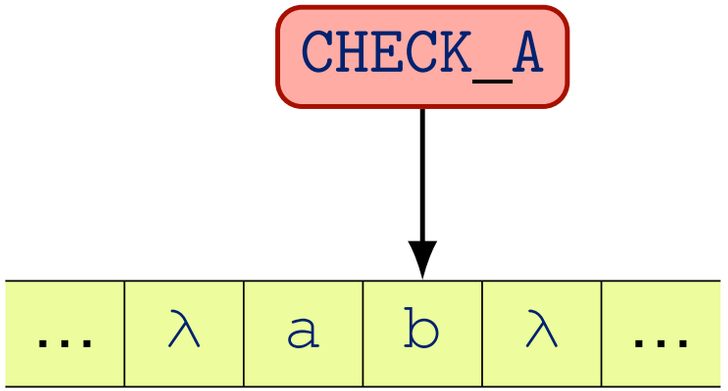
INIT, λ	→	ACCEPT, λ, N
INIT, a	→	MEM_A, a, R
INIT, b	→	MEM_B, b, R
MEM_A, a	→	MEM_A, a, R
MEM_A, b	→	MEM_A, b, R
MEM_A, λ	→	CHECK_A, λ, L
MEM_B, a	→	MEM_B, a, R
MEM_B, b	→	MEM_B, b, R
MEM_B, λ	→	CHECK_B, λ, L
CHECK_A, a	→	BACK, λ, L
CHECK_B, b	→	BACK, λ, L
BACK, a	→	BACK, a, L
BACK, b	→	BACK, b, L
BACK, λ	→	ERASE, λ, R
ERASE, a	→	INIT, λ, R
ERASE, b	→	INIT, λ, R
ERASE, λ	→	ACCEPT, λ, N

Simulace M se vstupem ab



INIT, λ	→	ACCEPT, λ, N
INIT, a	→	MEM_A, a, R
INIT, b	→	MEM_B, b, R
MEM_A, a	→	MEM_A, a, R
MEM_A, b	→	MEM_A, b, R
MEM_A, λ	→	CHECK_A, λ, L
MEM_B, a	→	MEM_B, a, R
MEM_B, b	→	MEM_B, b, R
MEM_B, λ	→	CHECK_B, λ, L
CHECK_A, a	→	BACK, λ, L
CHECK_B, b	→	BACK, λ, L
BACK, a	→	BACK, a, L
BACK, b	→	BACK, b, L
BACK, λ	→	ERASE, λ, R
ERASE, a	→	INIT, λ, R
ERASE, b	→	INIT, λ, R
ERASE, λ	→	ACCEPT, λ, N

Simulace M se vstupem ab



$$
\begin{array}{rcl}
\texttt{INIT},\lambda & \to & \texttt{ACCEPT},\lambda,N \\
\texttt{INIT},\texttt{a} & \to & \texttt{MEM\_A},\texttt{a},R \\
\texttt{INIT},\texttt{b} & \to & \texttt{MEM\_B},\texttt{b},R \\
\texttt{MEM\_A},\texttt{a} & \to & \texttt{MEM\_A},\texttt{a},R \\
\texttt{MEM\_A},\texttt{b} & \to & \texttt{MEM\_A},\texttt{b},R \\
\texttt{MEM\_A},\lambda & \to & \texttt{CHECK\_A},\lambda,L \\
\texttt{MEM\_B},\texttt{a} & \to & \texttt{MEM\_B},\texttt{a},R \\
\texttt{MEM\_B},\texttt{b} & \to & \texttt{MEM\_B},\texttt{b},R \\
\texttt{MEM\_B},\lambda & \to & \texttt{CHECK\_B},\lambda,L \\
\texttt{CHECK\_A},\texttt{a} & \to & \texttt{BACK},\lambda,L \\
\texttt{CHECK\_B},\texttt{b} & \to & \texttt{BACK},\lambda,L \\
\texttt{BACK},\texttt{a} & \to & \texttt{BACK},\texttt{a},L \\
\texttt{BACK},\texttt{b} & \to & \texttt{BACK},\texttt{b},L \\
\texttt{BACK},\lambda & \to & \texttt{ERASE},\lambda,R \\
\texttt{ERASE},\texttt{a} & \to & \texttt{INIT},\lambda,R \\
\texttt{ERASE},\texttt{b} & \to & \texttt{INIT},\lambda,R \\
\texttt{ERASE},\lambda & \to & \texttt{ACCEPT},\lambda,N
\end{array}
$$

Výpočet skončil, vstup byl zamítnut

Problém a jazyk

- V rozhodovacím problému se ptáme, zda daná instance x splňuje danou podmínku
- Jazyk L je množina slov (řetězců) nad abecedou Σ
- Rozhodovací problém odpovídá jazyku jeho kladných instancí

HELLOWORLD

Instance: Zdrojový kód programu P v jazyce C a vstup I.

Otázka: Je pravda, že prvních 12 znaků, které daný program vypíše, je `Hello, world`? (Nevyžadujeme zastavení.)

$$\mathrm{HW} = \{\langle P, I\rangle \mid \text{prvních } 12 \text{ znaků, které vypíše program } P \text{ se vstupem } I, \text{ jsou } \texttt{Hello, world}\}$$

Turingovsky rozhodnutelné jazyky

TS M přijímá slovo w výpočet M se vstupem w přijímající
- Skončí v přijímajícím stavu

TS M odmítá slovo w výpočet M se vstupem w zamítající
- Skončí ve stavu, který není přijímající

$L(M)$ jazyk slov přijímaných TS M

$M(w)\downarrow$ výpočet TS M nad vstupem w skončí

$M(w)\uparrow$ výpočet TS M nad vstupem w neskončí

Definice

- Jazyk L je částečně (Turingovsky) rozhodnutelný (též rekurzivně spočetný), je-li přijímán nějakým Turingovým strojem M (tj. $L = L(M)$)
- Jazyk L je (Turingovsky) rozhodnutelný (též rekurzivní), je-li přijímán nějakým Turingovým strojem M (tj. $L = L(M)$), který se s každým vstupem zastaví

Turingovsky vyčíslitelné funkce

- Turingův stroj M s páskovou abecedou Σ počítá nějakou částečnou funkci $f_M : \Sigma^* \to \Sigma^*$
- $M(w)\downarrow \implies$
  - $f_M(w)\downarrow$ (definovaná)
  - $f_M(w) =$ slovo na (výstupní) pásce $M(w)$ po ukončení výpočtu
- $M(w)\uparrow \implies f_M(w)\uparrow$ (nedefinovaná)

Definice

Funkce $f : \Sigma^* \to \Sigma^*$ je turingovsky vyčíslitelná, pokud existuje Turingův stroj M, který ji počítá.

Každá turingovsky vyčíslitelná funkce má nekonečně mnoho různých Turingových strojů, které ji počítají!

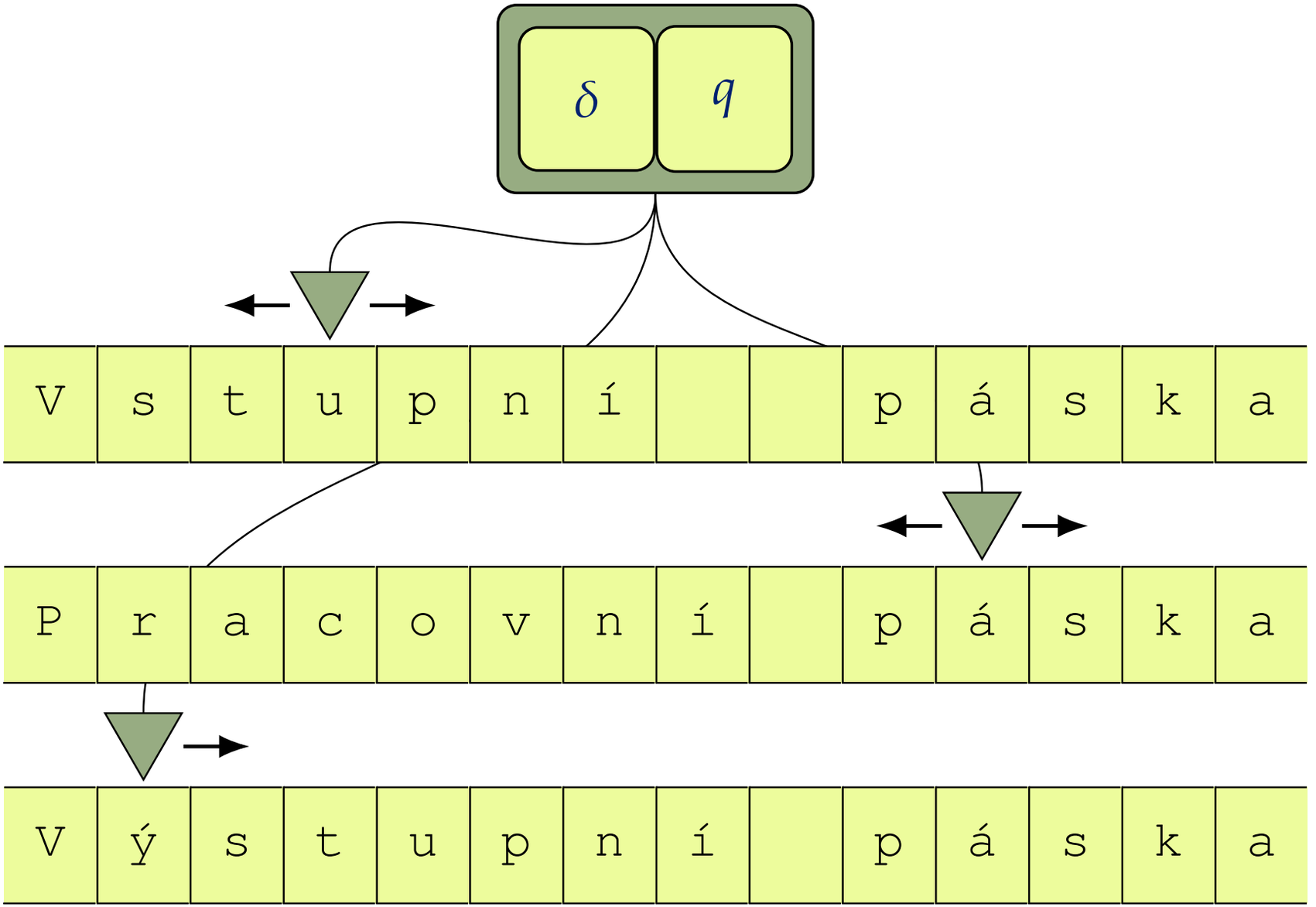
Varianty Turingových strojů

- TS s jednosměrně neomezenou páskou.
- TS s více páskami (vstupní/výstupní/pracovní).
- TS s více hlavami na páskách,
- TS s pouze binární abecedou,
- nedeterministické TS.

Zmíněné varianty jsou ekvivalentní „našemu“ modelu.

- všechny přijímají touž třídu jazyků
- všechny vyčíslují touž třídu funkcí

Struktura 3-páskového Turingova stroje



k-páskový Turingův stroj

- Má k pásek, na každé je zvláštní hlava

  Vstupní páska na počátku obsahuje vstupní řetězec
  - *Často jen pro čtení*

  Pracovní pásky jsou určeny pro čtení i zápis

  Výstupní páska na konci obsahuje výstupní řetězec
  - *Často jen pro zápis s pohybem hlavy jen vpravo*
- Hlavy na páskách se pohybují nezávisle na sobě
- Přechodová funkce je typu

$$\delta : Q \times \Sigma^k \to Q \times \Sigma^k \times \{R, N, L\}^k \cup \{\bot\}$$

Palindromy — algoritmus 2-páskového stroje

- Popíšeme Turingův stroj M se dvěma páskami
  - vstupní páskou a
  - pracovní páskou

Výpočet M se vstupem $w = w_1 \dots w_n$

---

1. Přesuň hlavu na vstupní pásce nad poslední znak vstupu
2. Přesuň hlavu na vstupní pásce zpět na první znak vstupu, současně kopíruj znaky na pracovní pásku

   `// Výsledkem je` w^R `na pracovní pásce`
3. Přesuň hlavu na pracovní pásce nad první znak w^R
4. Pohybem hlav na obou páskách současně zkontroluj, zda obsahují totéž slovo (tedy zda $w = w^R$)

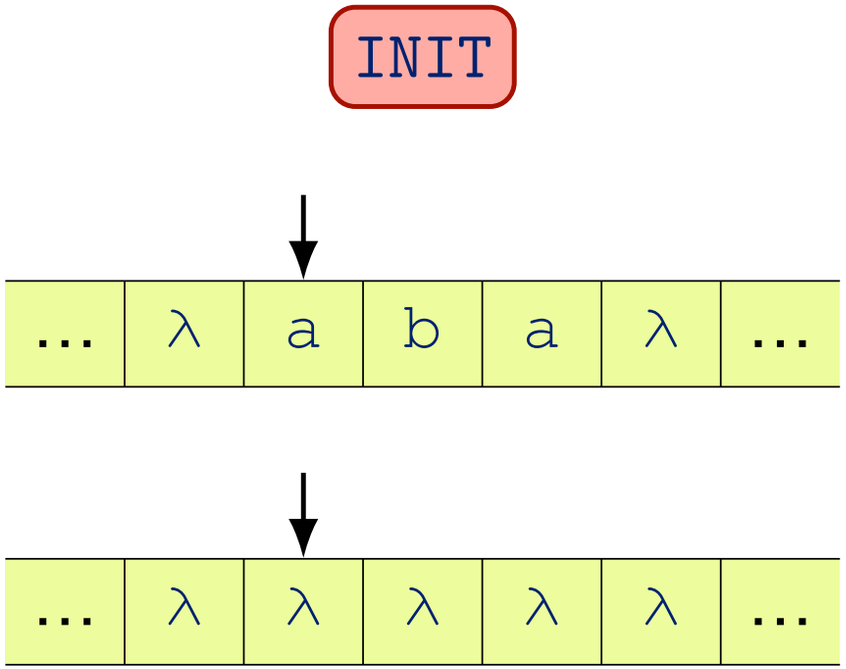
Rychlejší a jednodušší než s jednou páskou.

Palindromy — 2-páskový stroj

- $Q = \{\texttt{INIT}, \texttt{COPY}, \texttt{WBACK}, \texttt{CHECK}, \texttt{ACCEPT}\}$
- $\Sigma = \{\lambda, \texttt{a}, \texttt{b}\}$
- Počáteční stav $\texttt{INIT}$
- Přijímající stav $\texttt{ACCEPT}$

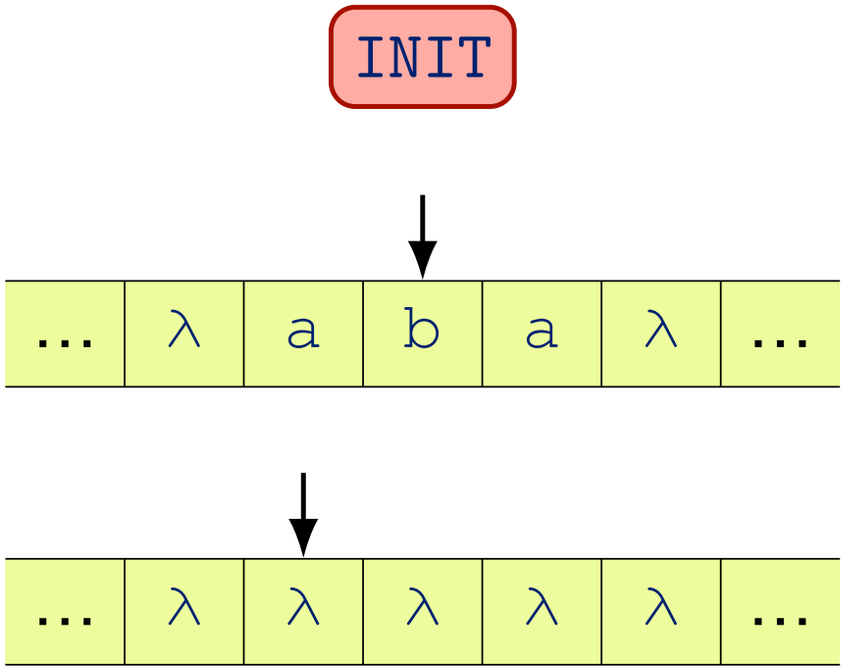
q, c_1, c_2	$\rightarrow$	q', c_1', c_2', Z_1, Z_2
$\texttt{INIT}, \texttt{a}, \lambda$	$\rightarrow$	$\texttt{INIT}, \texttt{a}, \lambda, R, N$
$\texttt{INIT}, \texttt{b}, \lambda$	$\rightarrow$	$\texttt{INIT}, \texttt{b}, \lambda, R, N$
$\texttt{INIT}, \lambda, \lambda$	$\rightarrow$	$\texttt{COPY}, \lambda, \lambda, L, N$
$\texttt{COPY}, \texttt{a}, \lambda$	$\rightarrow$	$\texttt{COPY}, \texttt{a}, \texttt{a}, L, R$
$\texttt{COPY}, \texttt{b}, \lambda$	$\rightarrow$	$\texttt{COPY}, \texttt{b}, \texttt{b}, L, R$
$\texttt{COPY}, \lambda, \lambda$	$\rightarrow$	$\texttt{WBACK}, \lambda, \lambda, N, L$
$\texttt{WBACK}, \lambda, \texttt{a}$	$\rightarrow$	$\texttt{WBACK}, \lambda, \texttt{a}, N, L$
$\texttt{WBACK}, \lambda, \texttt{b}$	$\rightarrow$	$\texttt{WBACK}, \lambda, \texttt{b}, N, L$
$\texttt{WBACK}, \lambda, \lambda$	$\rightarrow$	$\texttt{CHECK}, \lambda, \lambda, R, R$
$\texttt{CHECK}, \texttt{a}, \texttt{a}$	$\rightarrow$	$\texttt{CHECK}, \texttt{a}, \texttt{a}, R, R$
$\texttt{CHECK}, \texttt{b}, \texttt{b}$	$\rightarrow$	$\texttt{CHECK}, \texttt{b}, \texttt{b}, R, R$
$\texttt{CHECK}, \lambda, \lambda$	$\rightarrow$	$\texttt{ACCEPT}, \lambda, \lambda, N, N$

Simulace M se vstupem aba



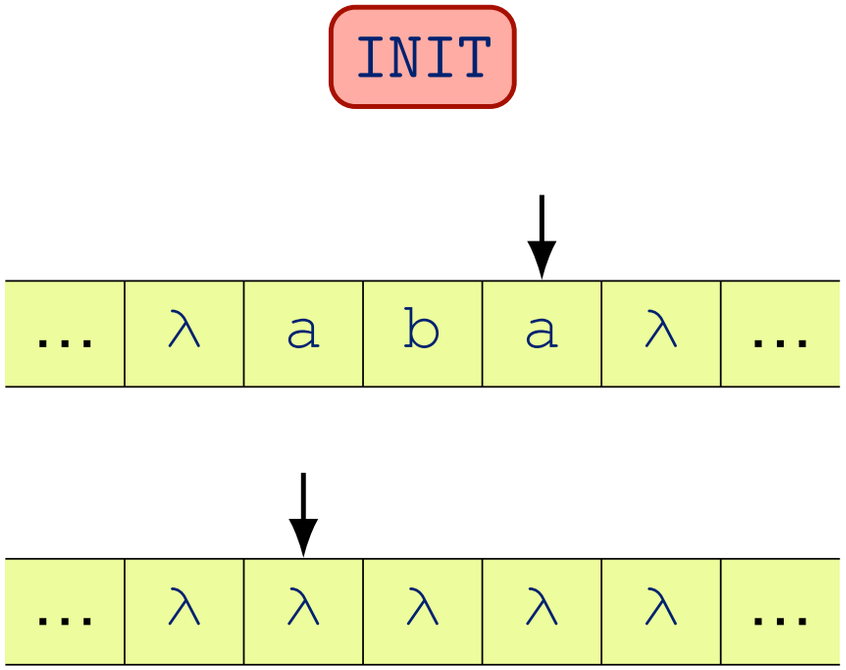
INIT, a, λ	→	INIT, a, λ, R, N
INIT, b, λ	→	INIT, b, λ, R, N
INIT, λ, λ	→	COPY, λ, λ, L, N
COPY, a, λ	→	COPY, a, a, L, R
COPY, b, λ	→	COPY, b, b, L, R
COPY, λ, λ	→	WBACK, λ, λ, N, L
WBACK, λ, a	→	WBACK, λ, a, N, L
WBACK, λ, b	→	WBACK, λ, b, N, L
WBACK, λ, λ	→	CHECK, λ, λ, R, R
CHECK, a, a	→	CHECK, a, a, R, R
CHECK, b, b	→	CHECK, b, b, R, R
CHECK, λ, λ	→	ACCEPT, λ, λ, N, N

Simulace M se vstupem aba



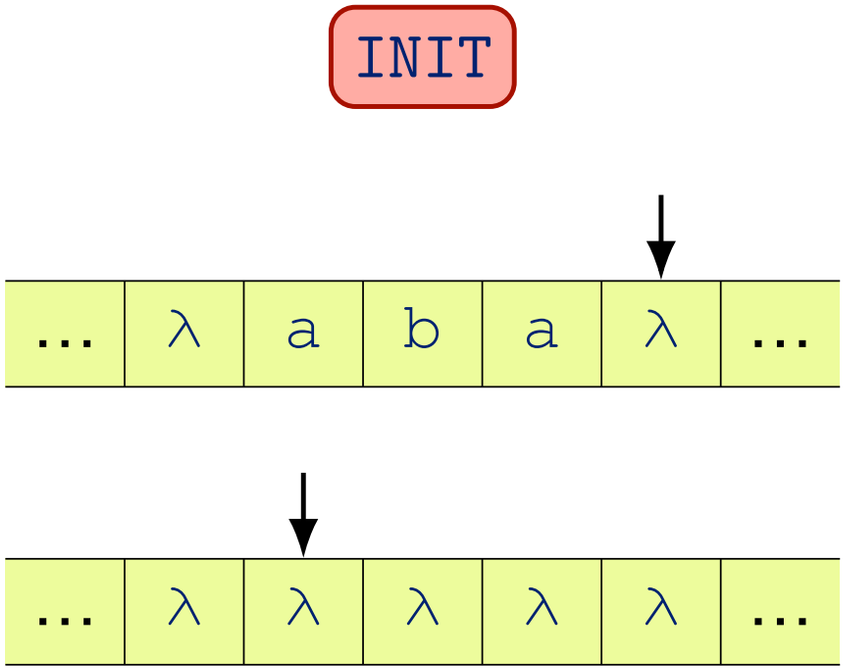
INIT, a, λ	→	INIT, a, λ, R, N
INIT, b, λ	→	INIT, b, λ, R, N
INIT, λ, λ	→	COPY, λ, λ, L, N
COPY, a, λ	→	COPY, a, a, L, R
COPY, b, λ	→	COPY, b, b, L, R
COPY, λ, λ	→	WBACK, λ, λ, N, L
WBACK, λ, a	→	WBACK, λ, a, N, L
WBACK, λ, b	→	WBACK, λ, b, N, L
WBACK, λ, λ	→	CHECK, λ, λ, R, R
CHECK, a, a	→	CHECK, a, a, R, R
CHECK, b, b	→	CHECK, b, b, R, R
CHECK, λ, λ	→	ACCEPT, λ, λ, N, N

Simulace M se vstupem aba



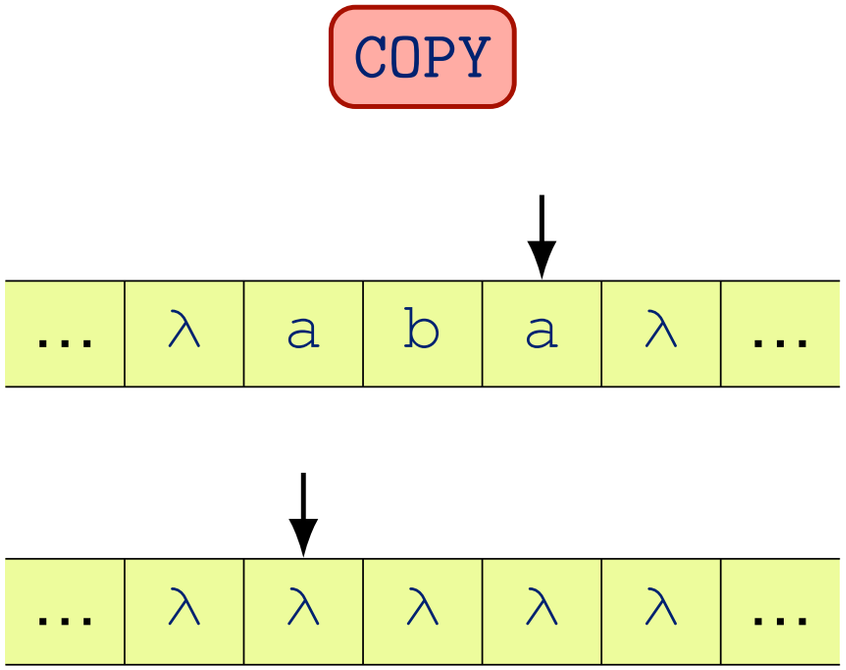
INIT, a, λ	→	INIT, a, λ, R, N
INIT, b, λ	→	INIT, b, λ, R, N
INIT, λ, λ	→	COPY, λ, λ, L, N
COPY, a, λ	→	COPY, a, a, L, R
COPY, b, λ	→	COPY, b, b, L, R
COPY, λ, λ	→	WBACK, λ, λ, N, L
WBACK, λ, a	→	WBACK, λ, a, N, L
WBACK, λ, b	→	WBACK, λ, b, N, L
WBACK, λ, λ	→	CHECK, λ, λ, R, R
CHECK, a, a	→	CHECK, a, a, R, R
CHECK, b, b	→	CHECK, b, b, R, R
CHECK, λ, λ	→	ACCEPT, λ, λ, N, N

Simulace M se vstupem aba



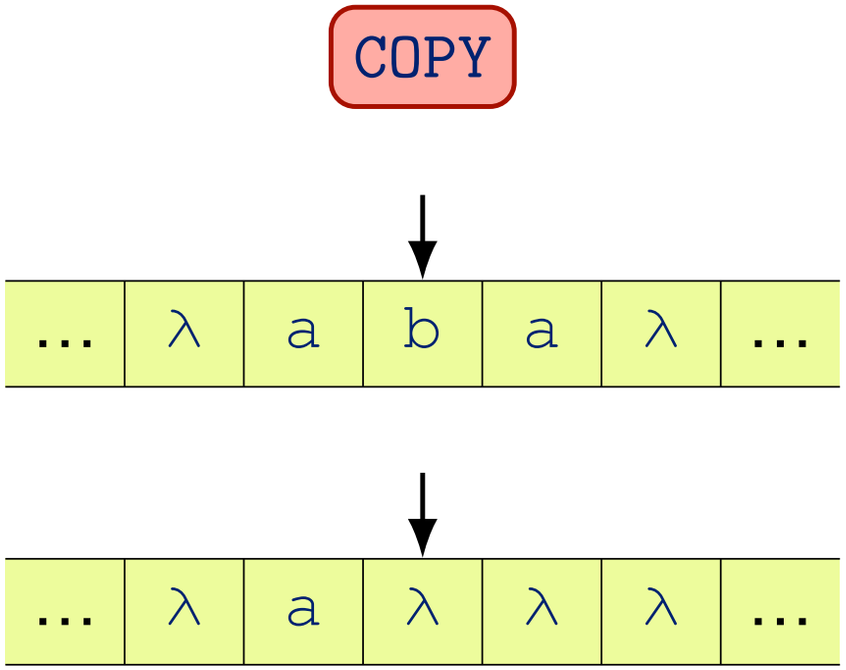
INIT, a, λ	→	INIT, a, λ, R, N
INIT, b, λ	→	INIT, b, λ, R, N
INIT, λ, λ	→	COPY, λ, λ, L, N
COPY, a, λ	→	COPY, a, a, L, R
COPY, b, λ	→	COPY, b, b, L, R
COPY, λ, λ	→	WBACK, λ, λ, N, L
WBACK, λ, a	→	WBACK, λ, a, N, L
WBACK, λ, b	→	WBACK, λ, b, N, L
WBACK, λ, λ	→	CHECK, λ, λ, R, R
CHECK, a, a	→	CHECK, a, a, R, R
CHECK, b, b	→	CHECK, b, b, R, R
CHECK, λ, λ	→	ACCEPT, λ, λ, N, N

Simulace M se vstupem aba



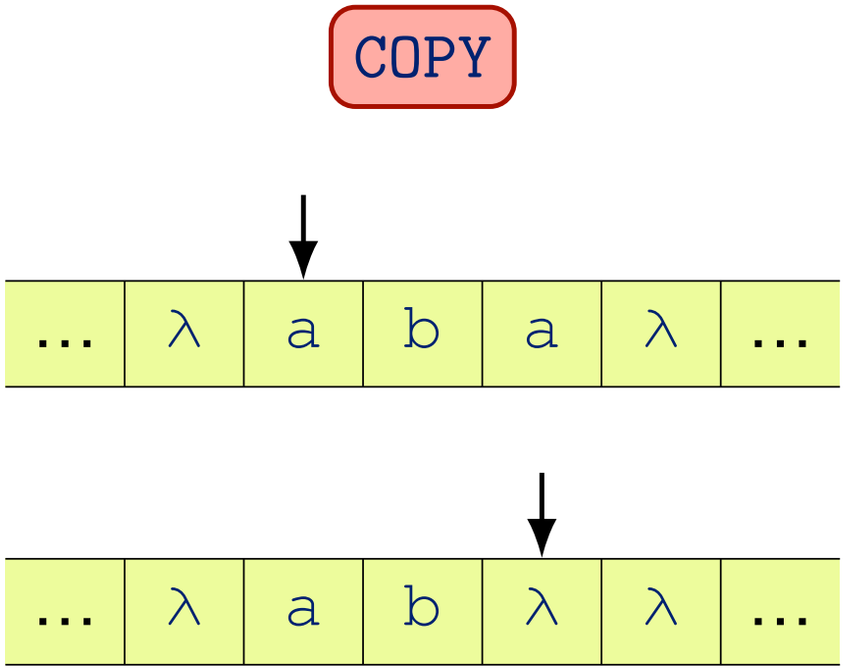
INIT, a, λ	→	INIT, a, λ, R, N
INIT, b, λ	→	INIT, b, λ, R, N
INIT, λ, λ	→	COPY, λ, λ, L, N
COPY, a, λ	→	COPY, a, a, L, R
COPY, b, λ	→	COPY, b, b, L, R
COPY, λ, λ	→	WBACK, λ, λ, N, L
WBACK, λ, a	→	WBACK, λ, a, N, L
WBACK, λ, b	→	WBACK, λ, b, N, L
WBACK, λ, λ	→	CHECK, λ, λ, R, R
CHECK, a, a	→	CHECK, a, a, R, R
CHECK, b, b	→	CHECK, b, b, R, R
CHECK, λ, λ	→	ACCEPT, λ, λ, N, N

Simulace M se vstupem aba



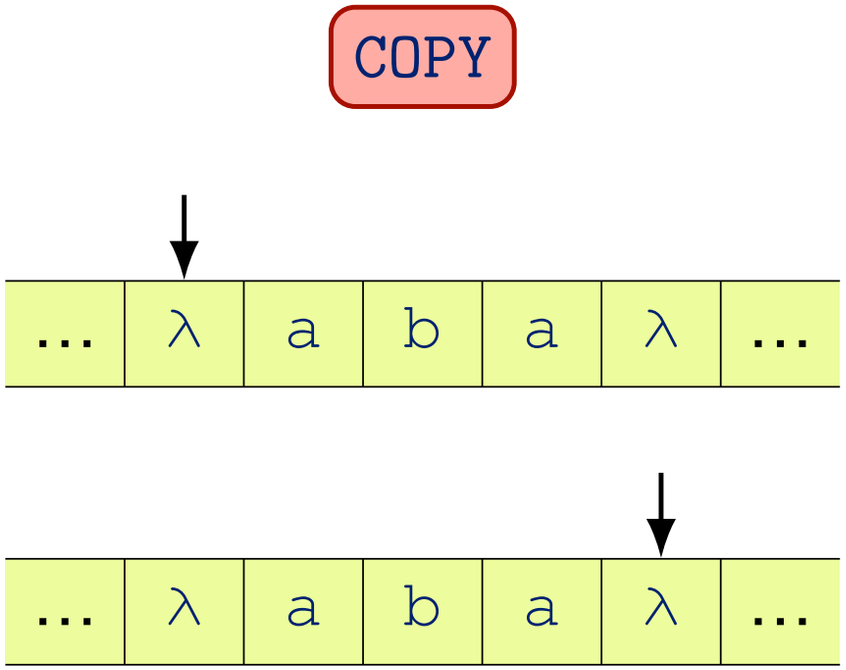
INIT, a, λ	→	INIT, a, λ, R, N
INIT, b, λ	→	INIT, b, λ, R, N
INIT, λ, λ	→	COPY, λ, λ, L, N
COPY, a, λ	→	COPY, a, a, L, R
COPY, b, λ	→	COPY, b, b, L, R
COPY, λ, λ	→	WBACK, λ, λ, N, L
WBACK, λ, a	→	WBACK, λ, a, N, L
WBACK, λ, b	→	WBACK, λ, b, N, L
WBACK, λ, λ	→	CHECK, λ, λ, R, R
CHECK, a, a	→	CHECK, a, a, R, R
CHECK, b, b	→	CHECK, b, b, R, R
CHECK, λ, λ	→	ACCEPT, λ, λ, N, N

Simulace M se vstupem aba



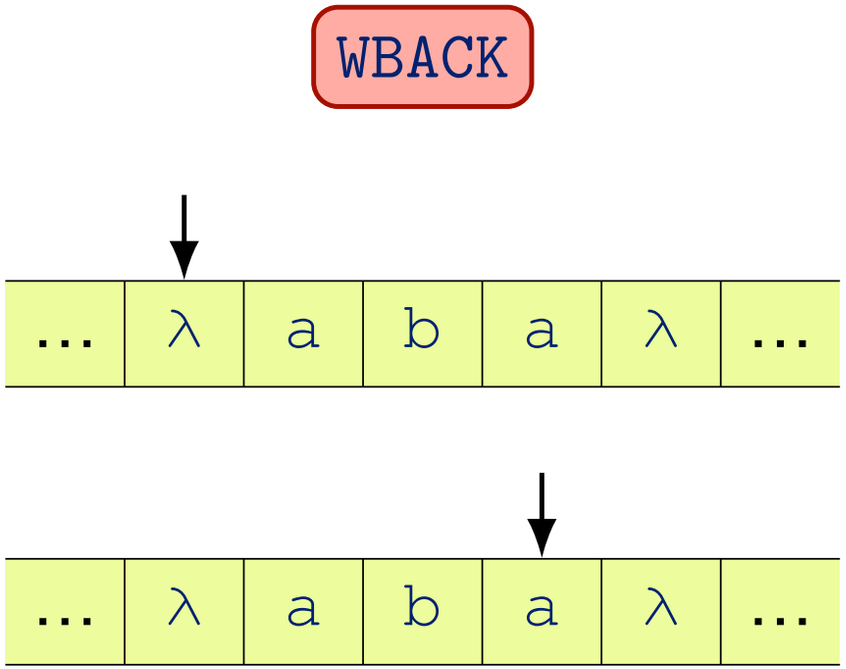
INIT, a, λ	→	INIT, a, λ, R, N
INIT, b, λ	→	INIT, b, λ, R, N
INIT, λ, λ	→	COPY, λ, λ, L, N
COPY, a, λ	→	COPY, a, a, L, R
COPY, b, λ	→	COPY, b, b, L, R
COPY, λ, λ	→	WBACK, λ, λ, N, L
WBACK, λ, a	→	WBACK, λ, a, N, L
WBACK, λ, b	→	WBACK, λ, b, N, L
WBACK, λ, λ	→	CHECK, λ, λ, R, R
CHECK, a, a	→	CHECK, a, a, R, R
CHECK, b, b	→	CHECK, b, b, R, R
CHECK, λ, λ	→	ACCEPT, λ, λ, N, N

Simulace M se vstupem aba



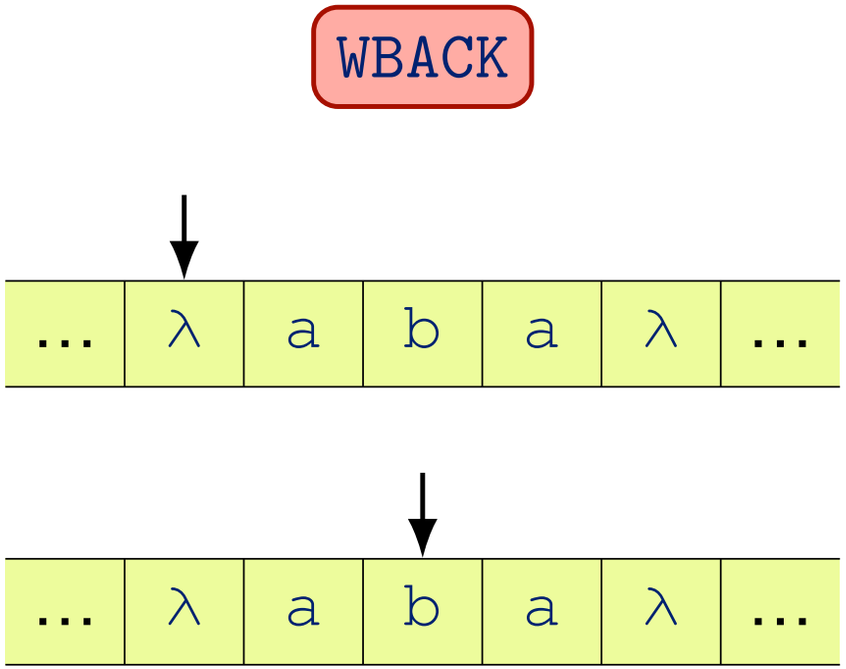
INIT, a, λ	→	INIT, a, λ, R, N
INIT, b, λ	→	INIT, b, λ, R, N
INIT, λ, λ	→	COPY, λ, λ, L, N
COPY, a, λ	→	COPY, a, a, L, R
COPY, b, λ	→	COPY, b, b, L, R
COPY, λ, λ	→	WBACK, λ, λ, N, L
WBACK, λ, a	→	WBACK, λ, a, N, L
WBACK, λ, b	→	WBACK, λ, b, N, L
WBACK, λ, λ	→	CHECK, λ, λ, R, R
CHECK, a, a	→	CHECK, a, a, R, R
CHECK, b, b	→	CHECK, b, b, R, R
CHECK, λ, λ	→	ACCEPT, λ, λ, N, N

Simulace M se vstupem aba



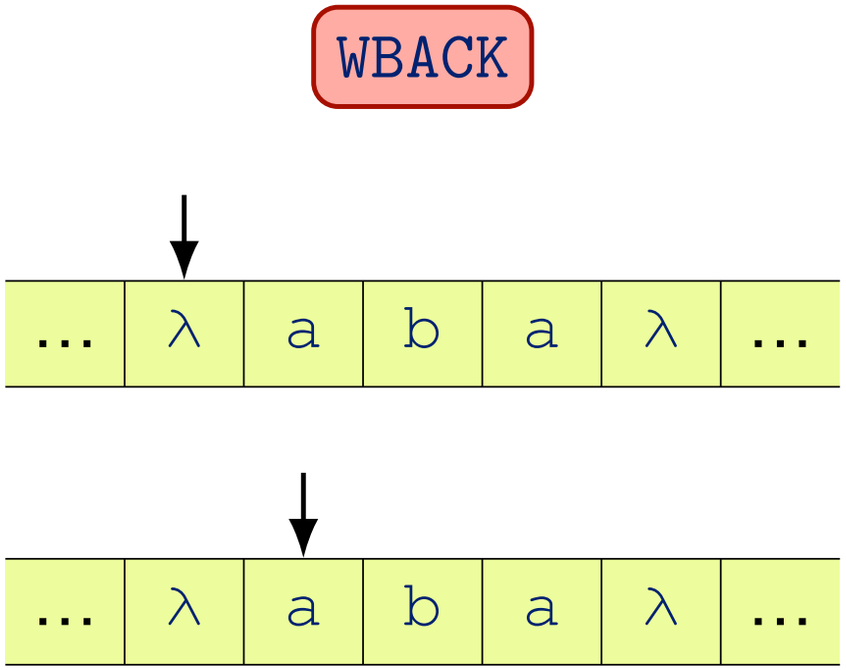
INIT, a, λ	→	INIT, a, λ, R, N
INIT, b, λ	→	INIT, b, λ, R, N
INIT, λ, λ	→	COPY, λ, λ, L, N
COPY, a, λ	→	COPY, a, a, L, R
COPY, b, λ	→	COPY, b, b, L, R
COPY, λ, λ	→	WBACK, λ, λ, N, L
WBACK, λ, a	→	WBACK, λ, a, N, L
WBACK, λ, b	→	WBACK, λ, b, N, L
WBACK, λ, λ	→	CHECK, λ, λ, R, R
CHECK, a, a	→	CHECK, a, a, R, R
CHECK, b, b	→	CHECK, b, b, R, R
CHECK, λ, λ	→	ACCEPT, λ, λ, N, N

Simulace M se vstupem aba



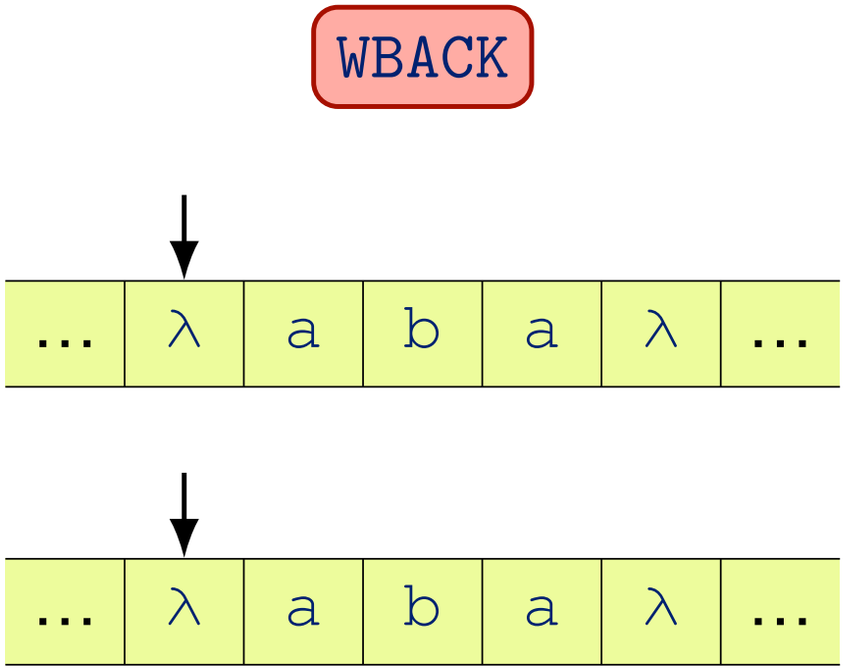
INIT, a, λ	→	INIT, a, λ, R, N
INIT, b, λ	→	INIT, b, λ, R, N
INIT, λ, λ	→	COPY, λ, λ, L, N
COPY, a, λ	→	COPY, a, a, L, R
COPY, b, λ	→	COPY, b, b, L, R
COPY, λ, λ	→	WBACK, λ, λ, N, L
WBACK, λ, a	→	WBACK, λ, a, N, L
WBACK, λ, b	→	WBACK, λ, b, N, L
WBACK, λ, λ	→	CHECK, λ, λ, R, R
CHECK, a, a	→	CHECK, a, a, R, R
CHECK, b, b	→	CHECK, b, b, R, R
CHECK, λ, λ	→	ACCEPT, λ, λ, N, N

Simulace M se vstupem aba



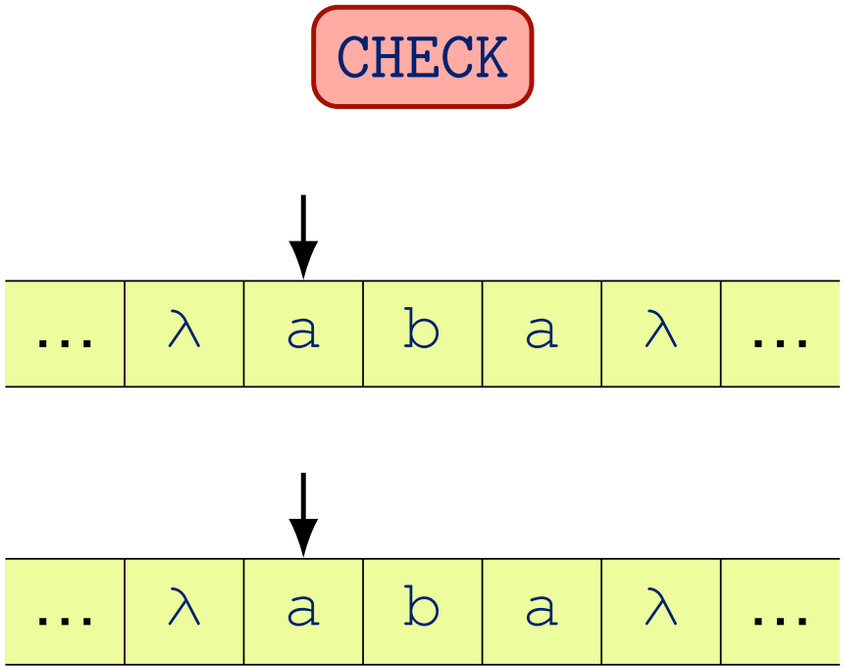
INIT, a, λ	→	INIT, a, λ, R, N
INIT, b, λ	→	INIT, b, λ, R, N
INIT, λ, λ	→	COPY, λ, λ, L, N
COPY, a, λ	→	COPY, a, a, L, R
COPY, b, λ	→	COPY, b, b, L, R
COPY, λ, λ	→	WBACK, λ, λ, N, L
WBACK, λ, a	→	WBACK, λ, a, N, L
WBACK, λ, b	→	WBACK, λ, b, N, L
WBACK, λ, λ	→	CHECK, λ, λ, R, R
CHECK, a, a	→	CHECK, a, a, R, R
CHECK, b, b	→	CHECK, b, b, R, R
CHECK, λ, λ	→	ACCEPT, λ, λ, N, N

Simulace M se vstupem aba



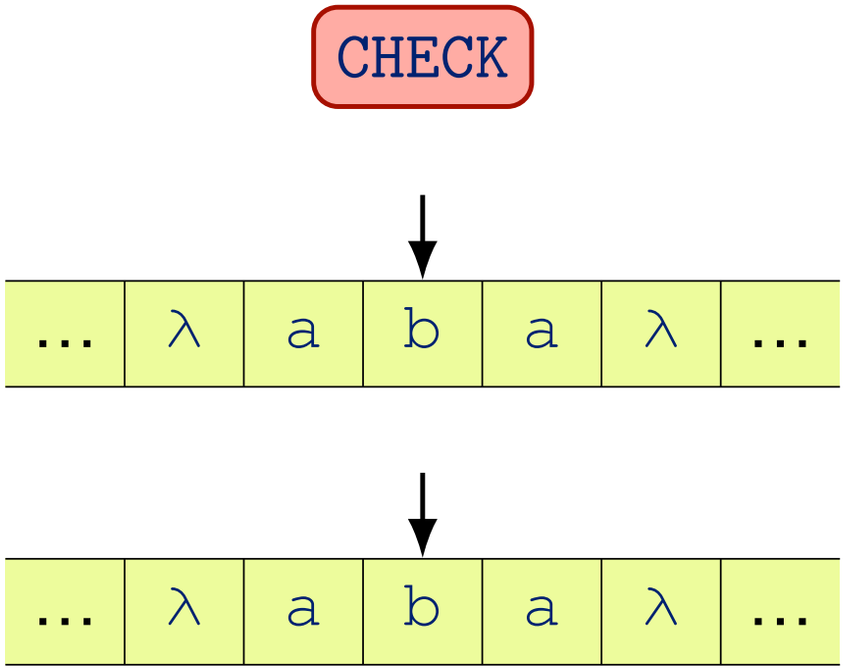
INIT, a, λ	→	INIT, a, λ, R, N
INIT, b, λ	→	INIT, b, λ, R, N
INIT, λ, λ	→	COPY, λ, λ, L, N
COPY, a, λ	→	COPY, a, a, L, R
COPY, b, λ	→	COPY, b, b, L, R
COPY, λ, λ	→	WBACK, λ, λ, N, L
WBACK, λ, a	→	WBACK, λ, a, N, L
WBACK, λ, b	→	WBACK, λ, b, N, L
WBACK, λ, λ	→	CHECK, λ, λ, R, R
CHECK, a, a	→	CHECK, a, a, R, R
CHECK, b, b	→	CHECK, b, b, R, R
CHECK, λ, λ	→	ACCEPT, λ, λ, N, N

Simulace M se vstupem aba



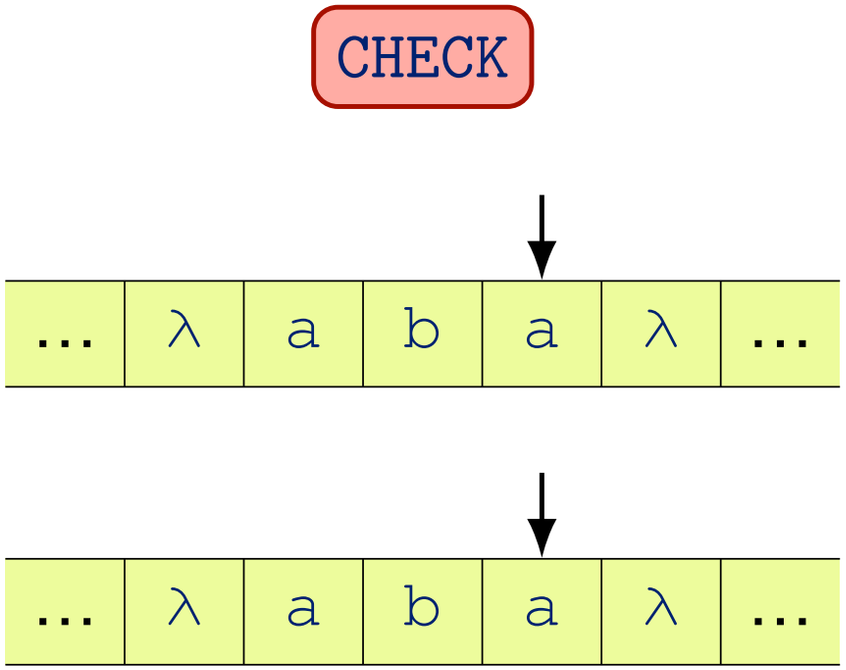
INIT, a, λ	→	INIT, a, λ, R, N
INIT, b, λ	→	INIT, b, λ, R, N
INIT, λ, λ	→	COPY, λ, λ, L, N
COPY, a, λ	→	COPY, a, a, L, R
COPY, b, λ	→	COPY, b, b, L, R
COPY, λ, λ	→	WBACK, λ, λ, N, L
WBACK, λ, a	→	WBACK, λ, a, N, L
WBACK, λ, b	→	WBACK, λ, b, N, L
WBACK, λ, λ	→	CHECK, λ, λ, R, R
CHECK, a, a	→	CHECK, a, a, R, R
CHECK, b, b	→	CHECK, b, b, R, R
CHECK, λ, λ	→	ACCEPT, λ, λ, N, N

Simulace M se vstupem aba



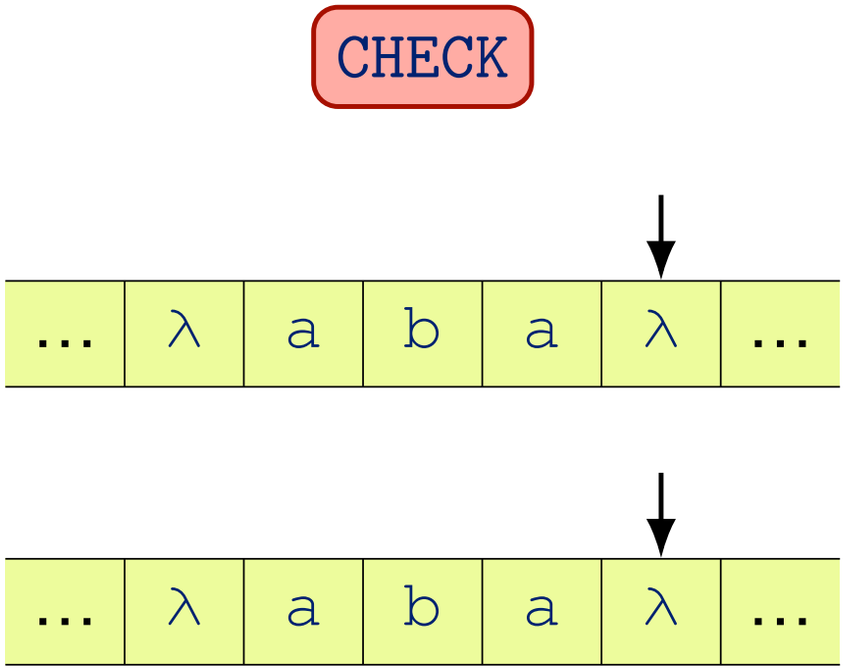
INIT, a, λ	→	INIT, a, λ, R, N
INIT, b, λ	→	INIT, b, λ, R, N
INIT, λ, λ	→	COPY, λ, λ, L, N
COPY, a, λ	→	COPY, a, a, L, R
COPY, b, λ	→	COPY, b, b, L, R
COPY, λ, λ	→	WBACK, λ, λ, N, L
WBACK, λ, a	→	WBACK, λ, a, N, L
WBACK, λ, b	→	WBACK, λ, b, N, L
WBACK, λ, λ	→	CHECK, λ, λ, R, R
CHECK, a, a	→	CHECK, a, a, R, R
CHECK, b, b	→	CHECK, b, b, R, R
CHECK, λ, λ	→	ACCEPT, λ, λ, N, N

Simulace M se vstupem aba



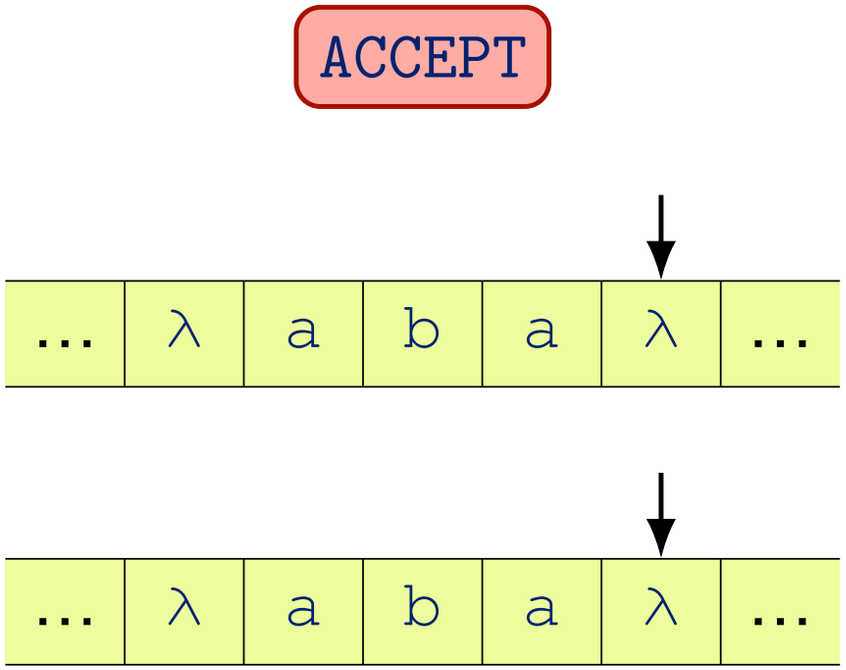
INIT, a, λ	→	INIT, a, λ, R, N
INIT, b, λ	→	INIT, b, λ, R, N
INIT, λ, λ	→	COPY, λ, λ, L, N
COPY, a, λ	→	COPY, a, a, L, R
COPY, b, λ	→	COPY, b, b, L, R
COPY, λ, λ	→	WBACK, λ, λ, N, L
WBACK, λ, a	→	WBACK, λ, a, N, L
WBACK, λ, b	→	WBACK, λ, b, N, L
WBACK, λ, λ	→	CHECK, λ, λ, R, R
CHECK, a, a	→	CHECK, a, a, R, R
CHECK, b, b	→	CHECK, b, b, R, R
CHECK, λ, λ	→	ACCEPT, λ, λ, N, N

Simulace M se vstupem aba



INIT, a, λ	→	INIT, a, λ, R, N
INIT, b, λ	→	INIT, b, λ, R, N
INIT, λ, λ	→	COPY, λ, λ, L, N
COPY, a, λ	→	COPY, a, a, L, R
COPY, b, λ	→	COPY, b, b, L, R
COPY, λ, λ	→	WBACK, λ, λ, N, L
WBACK, λ, a	→	WBACK, λ, a, N, L
WBACK, λ, b	→	WBACK, λ, b, N, L
WBACK, λ, λ	→	CHECK, λ, λ, R, R
CHECK, a, a	→	CHECK, a, a, R, R
CHECK, b, b	→	CHECK, b, b, R, R
CHECK, λ, λ	→	ACCEPT, λ, λ, N, N

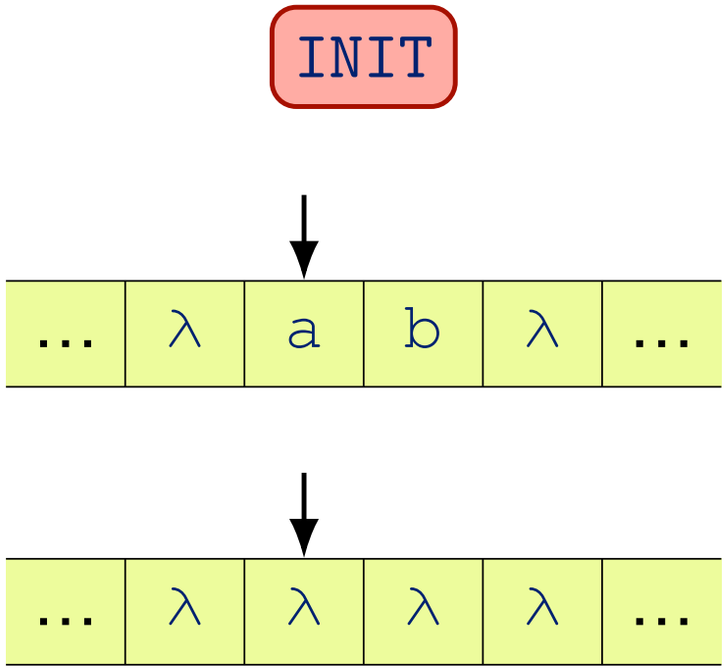
Simulace M se vstupem aba



$$
\begin{array}{rcl}
\texttt{INIT},\texttt{a},\lambda & \to & \texttt{INIT},\texttt{a},\lambda,R,N \\
\texttt{INIT},\texttt{b},\lambda & \to & \texttt{INIT},\texttt{b},\lambda,R,N \\
\texttt{INIT},\lambda,\lambda & \to & \texttt{COPY},\lambda,\lambda,L,N \\
\texttt{COPY},\texttt{a},\lambda & \to & \texttt{COPY},\texttt{a},\texttt{a},L,R \\
\texttt{COPY},\texttt{b},\lambda & \to & \texttt{COPY},\texttt{b},\texttt{b},L,R \\
\texttt{COPY},\lambda,\lambda & \to & \texttt{WBACK},\lambda,\lambda,N,L \\
\texttt{WBACK},\lambda,\texttt{a} & \to & \texttt{WBACK},\lambda,\texttt{a},N,L \\
\texttt{WBACK},\lambda,\texttt{b} & \to & \texttt{WBACK},\lambda,\texttt{b},N,L \\
\texttt{WBACK},\lambda,\lambda & \to & \texttt{CHECK},\lambda,\lambda,R,R \\
\texttt{CHECK},\texttt{a},\texttt{a} & \to & \texttt{CHECK},\texttt{a},\texttt{a},R,R \\
\texttt{CHECK},\texttt{b},\texttt{b} & \to & \texttt{CHECK},\texttt{b},\texttt{b},R,R \\
\texttt{CHECK},\lambda,\lambda & \to & \texttt{ACCEPT},\lambda,\lambda,N,N
\end{array}
$$

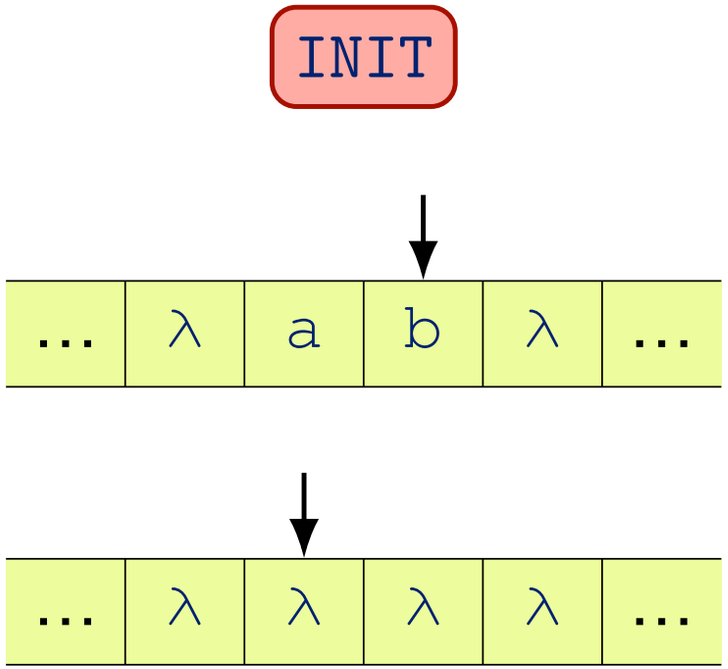
Výpočet skončil, vstup byl přijat

Simulace M se vstupem ab



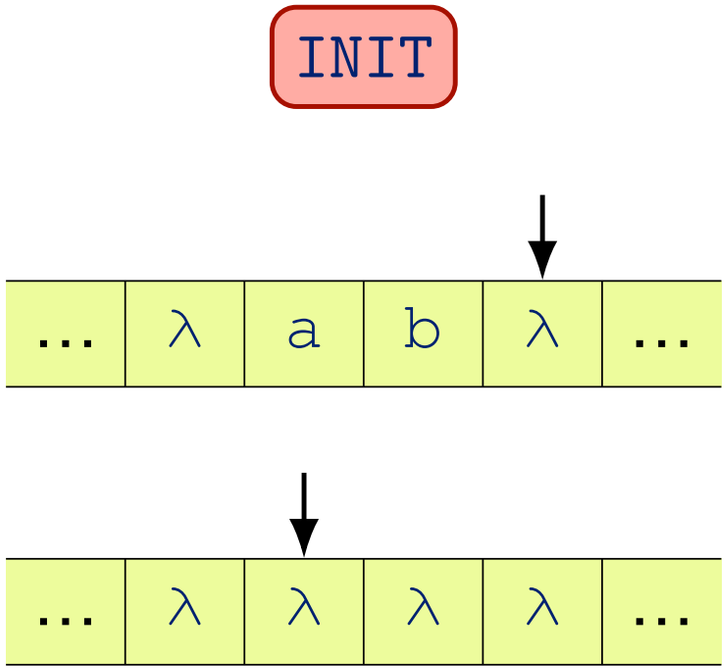
INIT, a, λ	→	INIT, a, λ, R, N
INIT, b, λ	→	INIT, b, λ, R, N
INIT, λ, λ	→	COPY, λ, λ, L, N
COPY, a, λ	→	COPY, a, a, L, R
COPY, b, λ	→	COPY, b, b, L, R
COPY, λ, λ	→	WBACK, λ, λ, N, L
WBACK, λ, a	→	WBACK, λ, a, N, L
WBACK, λ, b	→	WBACK, λ, b, N, L
WBACK, λ, λ	→	CHECK, λ, λ, R, R
CHECK, a, a	→	CHECK, a, a, R, R
CHECK, b, b	→	CHECK, b, b, R, R
CHECK, λ, λ	→	ACCEPT, λ, λ, N, N

Simulace M se vstupem ab



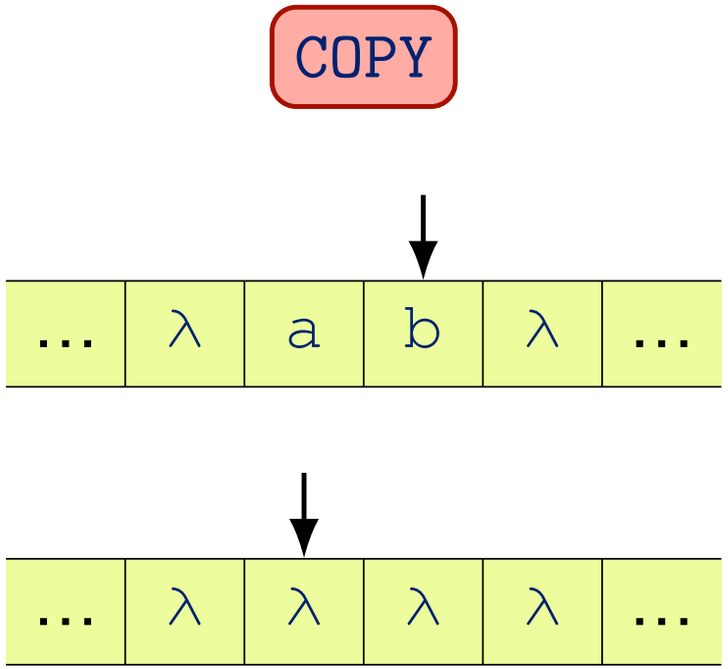
INIT, a, λ	→	INIT, a, λ, R, N
INIT, b, λ	→	INIT, b, λ, R, N
INIT, λ, λ	→	COPY, λ, λ, L, N
COPY, a, λ	→	COPY, a, a, L, R
COPY, b, λ	→	COPY, b, b, L, R
COPY, λ, λ	→	WBACK, λ, λ, N, L
WBACK, λ, a	→	WBACK, λ, a, N, L
WBACK, λ, b	→	WBACK, λ, b, N, L
WBACK, λ, λ	→	CHECK, λ, λ, R, R
CHECK, a, a	→	CHECK, a, a, R, R
CHECK, b, b	→	CHECK, b, b, R, R
CHECK, λ, λ	→	ACCEPT, λ, λ, N, N

Simulace M se vstupem ab



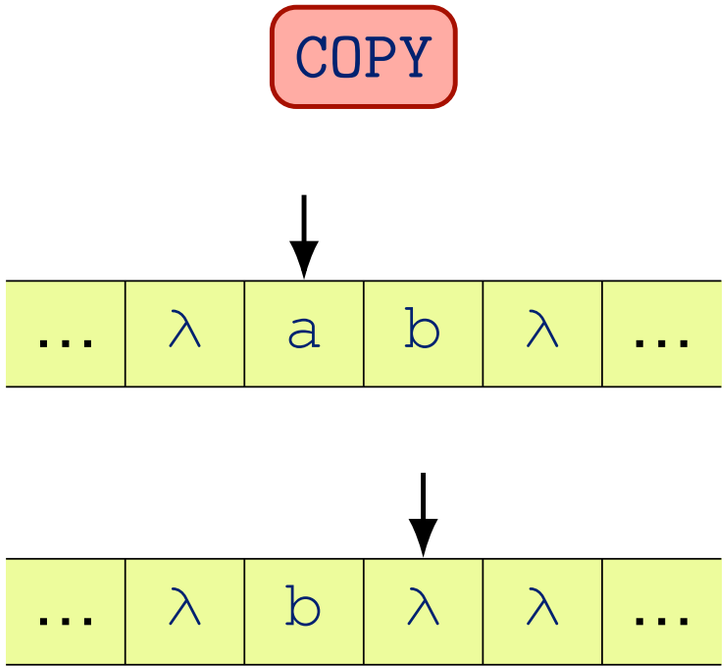
INIT, a, λ	→	INIT, a, λ, R, N
INIT, b, λ	→	INIT, b, λ, R, N
INIT, λ, λ	→	COPY, λ, λ, L, N
COPY, a, λ	→	COPY, a, a, L, R
COPY, b, λ	→	COPY, b, b, L, R
COPY, λ, λ	→	WBACK, λ, λ, N, L
WBACK, λ, a	→	WBACK, λ, a, N, L
WBACK, λ, b	→	WBACK, λ, b, N, L
WBACK, λ, λ	→	CHECK, λ, λ, R, R
CHECK, a, a	→	CHECK, a, a, R, R
CHECK, b, b	→	CHECK, b, b, R, R
CHECK, λ, λ	→	ACCEPT, λ, λ, N, N

Simulace M se vstupem ab



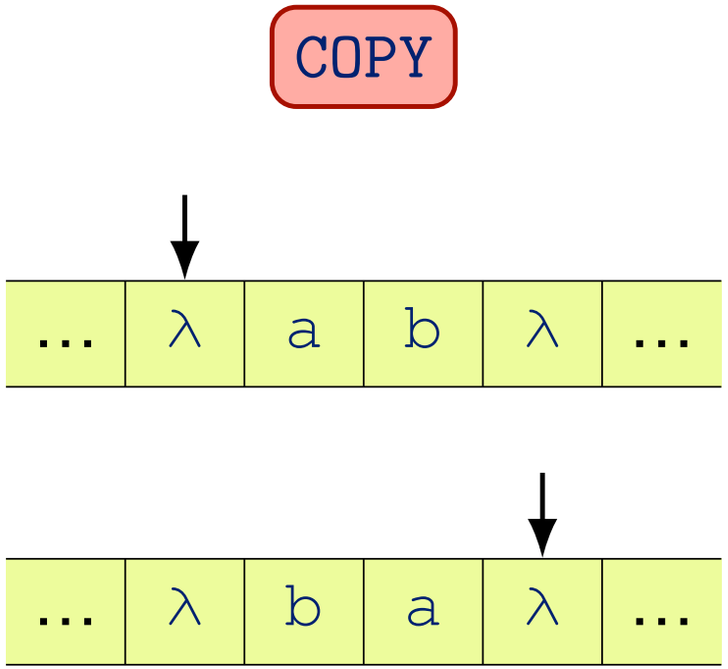
INIT, a, λ	→	INIT, a, λ, R, N
INIT, b, λ	→	INIT, b, λ, R, N
INIT, λ, λ	→	COPY, λ, λ, L, N
COPY, a, λ	→	COPY, a, a, L, R
COPY, b, λ	→	COPY, b, b, L, R
COPY, λ, λ	→	WBACK, λ, λ, N, L
WBACK, λ, a	→	WBACK, λ, a, N, L
WBACK, λ, b	→	WBACK, λ, b, N, L
WBACK, λ, λ	→	CHECK, λ, λ, R, R
CHECK, a, a	→	CHECK, a, a, R, R
CHECK, b, b	→	CHECK, b, b, R, R
CHECK, λ, λ	→	ACCEPT, λ, λ, N, N

Simulace M se vstupem ab



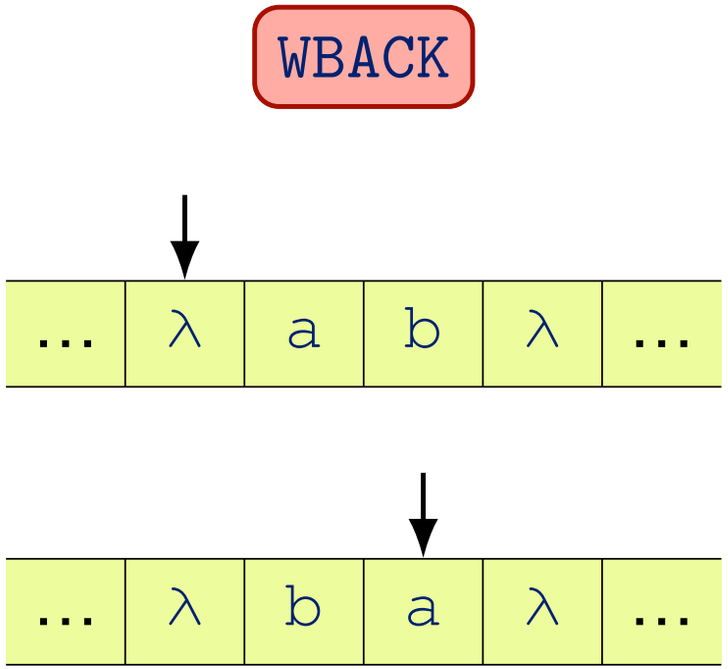
INIT, a, λ	→	INIT, a, λ, R, N
INIT, b, λ	→	INIT, b, λ, R, N
INIT, λ, λ	→	COPY, λ, λ, L, N
COPY, a, λ	→	COPY, a, a, L, R
COPY, b, λ	→	COPY, b, b, L, R
COPY, λ, λ	→	WBACK, λ, λ, N, L
WBACK, λ, a	→	WBACK, λ, a, N, L
WBACK, λ, b	→	WBACK, λ, b, N, L
WBACK, λ, λ	→	CHECK, λ, λ, R, R
CHECK, a, a	→	CHECK, a, a, R, R
CHECK, b, b	→	CHECK, b, b, R, R
CHECK, λ, λ	→	ACCEPT, λ, λ, N, N

Simulace M se vstupem ab



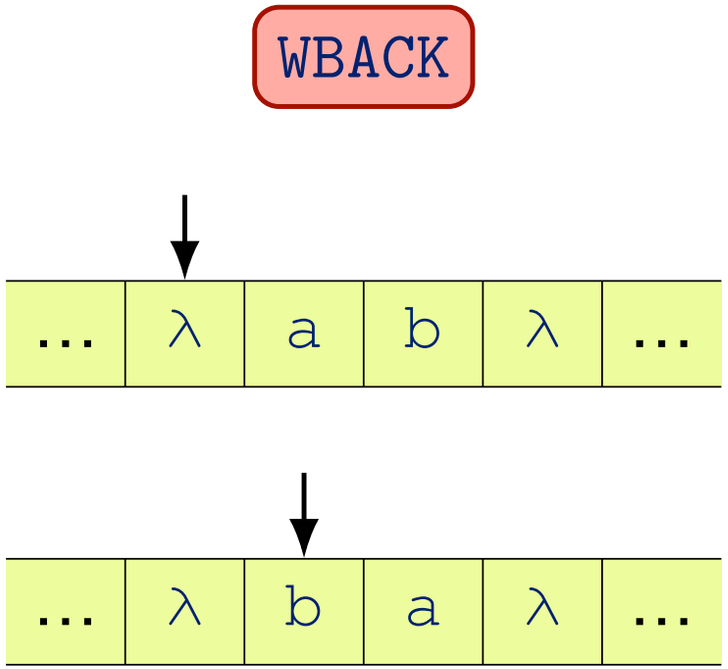
INIT, a, λ	→	INIT, a, λ, R, N
INIT, b, λ	→	INIT, b, λ, R, N
INIT, λ, λ	→	COPY, λ, λ, L, N
COPY, a, λ	→	COPY, a, a, L, R
COPY, b, λ	→	COPY, b, b, L, R
COPY, λ, λ	→	WBACK, λ, λ, N, L
WBACK, λ, a	→	WBACK, λ, a, N, L
WBACK, λ, b	→	WBACK, λ, b, N, L
WBACK, λ, λ	→	CHECK, λ, λ, R, R
CHECK, a, a	→	CHECK, a, a, R, R
CHECK, b, b	→	CHECK, b, b, R, R
CHECK, λ, λ	→	ACCEPT, λ, λ, N, N

Simulace M se vstupem ab



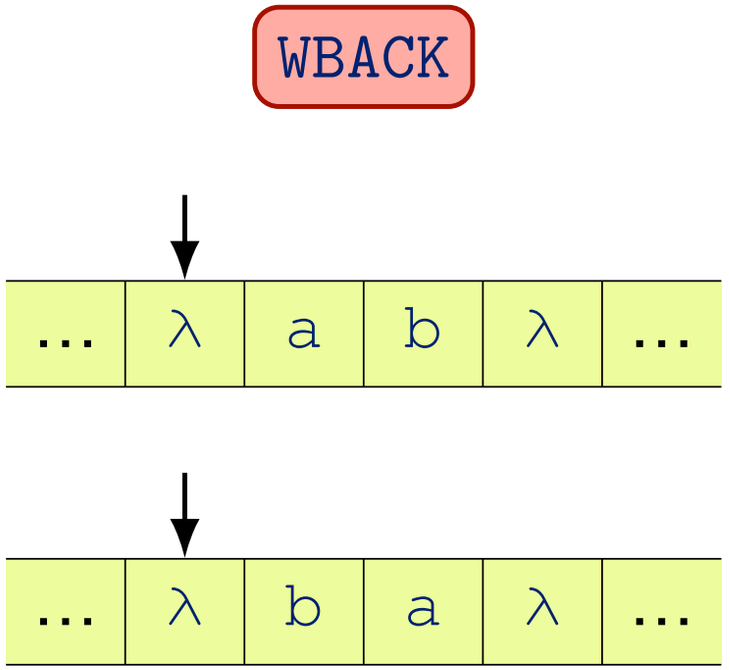
INIT, a, λ	→	INIT, a, λ, R, N
INIT, b, λ	→	INIT, b, λ, R, N
INIT, λ, λ	→	COPY, λ, λ, L, N
COPY, a, λ	→	COPY, a, a, L, R
COPY, b, λ	→	COPY, b, b, L, R
COPY, λ, λ	→	WBACK, λ, λ, N, L
WBACK, λ, a	→	WBACK, λ, a, N, L
WBACK, λ, b	→	WBACK, λ, b, N, L
WBACK, λ, λ	→	CHECK, λ, λ, R, R
CHECK, a, a	→	CHECK, a, a, R, R
CHECK, b, b	→	CHECK, b, b, R, R
CHECK, λ, λ	→	ACCEPT, λ, λ, N, N

Simulace M se vstupem ab



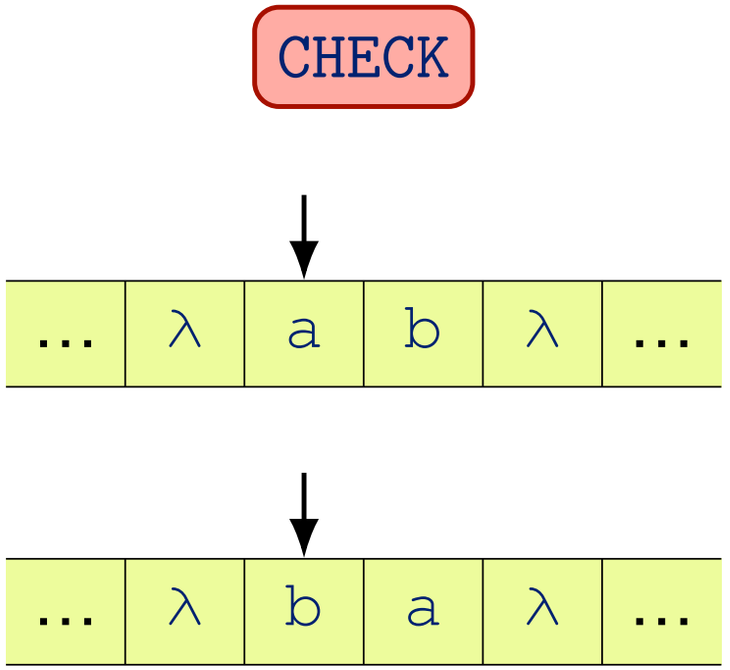
INIT, a, λ	→	INIT, a, λ, R, N
INIT, b, λ	→	INIT, b, λ, R, N
INIT, λ, λ	→	COPY, λ, λ, L, N
COPY, a, λ	→	COPY, a, a, L, R
COPY, b, λ	→	COPY, b, b, L, R
COPY, λ, λ	→	WBACK, λ, λ, N, L
WBACK, λ, a	→	WBACK, λ, a, N, L
WBACK, λ, b	→	WBACK, λ, b, N, L
WBACK, λ, λ	→	CHECK, λ, λ, R, R
CHECK, a, a	→	CHECK, a, a, R, R
CHECK, b, b	→	CHECK, b, b, R, R
CHECK, λ, λ	→	ACCEPT, λ, λ, N, N

Simulace M se vstupem ab



INIT, a, λ	→	INIT, a, λ, R, N
INIT, b, λ	→	INIT, b, λ, R, N
INIT, λ, λ	→	COPY, λ, λ, L, N
COPY, a, λ	→	COPY, a, a, L, R
COPY, b, λ	→	COPY, b, b, L, R
COPY, λ, λ	→	WBACK, λ, λ, N, L
WBACK, λ, a	→	WBACK, λ, a, N, L
WBACK, λ, b	→	WBACK, λ, b, N, L
WBACK, λ, λ	→	CHECK, λ, λ, R, R
CHECK, a, a	→	CHECK, a, a, R, R
CHECK, b, b	→	CHECK, b, b, R, R
CHECK, λ, λ	→	ACCEPT, λ, λ, N, N

Simulace M se vstupem ab



INIT, a, λ	→	INIT, a, λ, R, N
INIT, b, λ	→	INIT, b, λ, R, N
INIT, λ, λ	→	COPY, λ, λ, L, N
COPY, a, λ	→	COPY, a, a, L, R
COPY, b, λ	→	COPY, b, b, L, R
COPY, λ, λ	→	WBACK, λ, λ, N, L
WBACK, λ, a	→	WBACK, λ, a, N, L
WBACK, λ, b	→	WBACK, λ, b, N, L
WBACK, λ, λ	→	CHECK, λ, λ, R, R
CHECK, a, a	→	CHECK, a, a, R, R
CHECK, b, b	→	CHECK, b, b, R, R
CHECK, λ, λ	→	ACCEPT, λ, λ, N, N

Výpočet skončil, vstup byl zamítnut

Redukce počtu pásek

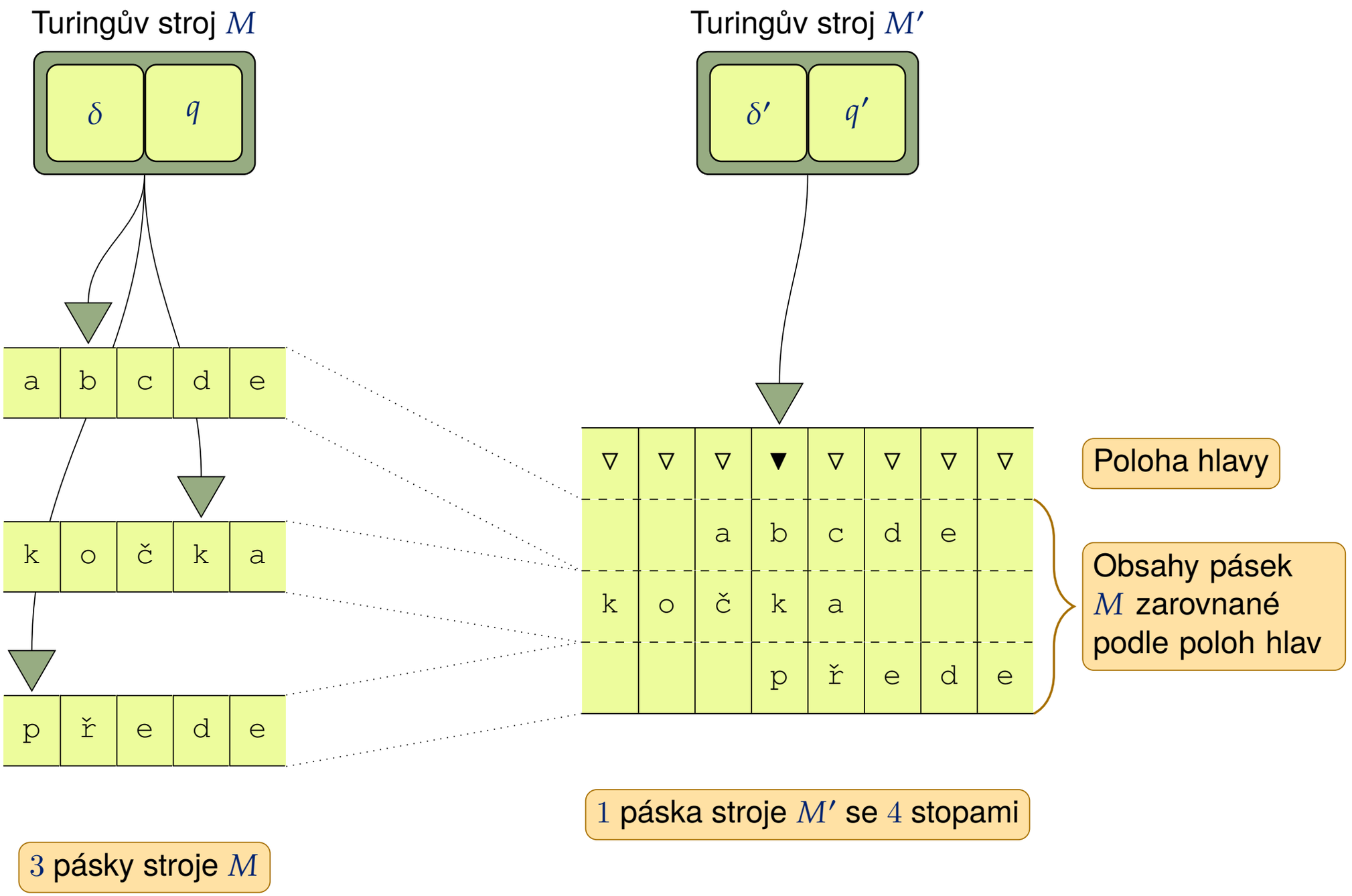
Věta

Ke každému k-páskovému Turingovu stroji M existuje jednopáskový Turingův stroj M', který simuluje práci M.

Přesněji

- M' přijímá týž jazyk jako M
- M' počítá touž funkci jako M

Reprezentace k pásek na jedné pásce



Konstrukce jednopáskového stroje M'

- Předpokládejme, že $M' = (Q, \Sigma, \delta, q_0, F)$ je 3-páskový stroj
- Tři pásky M reprezentujeme na třech stopách jedné pásky M'.
- Obsahy pásek jsou zarovnané podle poloh hlav
- Navíc páska M' obsahuje stopu se značkami vyznačujícími polohu hlav M
- Abeceda stroje M' je tedy

$$\Sigma' = \Sigma \cup \{\blacktriangledown, \triangledown\} \times \Sigma \times \Sigma \times \Sigma$$

- Abeceda Σ je součástí Σ', protože M' čte vstup v abecedě Σ
- Prvním krokem M' je přeformátování vstupu do 4 stop
- Podobně může být nutné zpět přeformátovat i výstup

Simulace kroku k-páskového stroje

Instrukci stroje M

$$\delta(\overbrace{q}^{\text{stav z } Q}, \overbrace{a_1, a_2, a_3}^{\text{znaky z } \Sigma}) = (\overbrace{q'}^{\text{stav z } Q}, \overbrace{b_1, b_2, b_3}^{\text{znaky z } \Sigma}, \overbrace{Z_1, Z_2, Z_3}^{\text{pohyby } L, N, R})$$

odpovídá posloupnost instrukci stroje M', která provede

- Změnu stavu jako u k-páskového stroje
  - Stav M je uložen jako část stavu M'
- Přepis obsahu políček v jednotlivých stopách
  - Přepíšou se všechna najednou,
  - v M' jde o jeden znak
- Posun obsahu pásek ve stopách proti směru pohybu tak, aby hlavy byly nakonec opět zarovnané