Stavové automaty

Přednáška 6

Semafor

- Úloha: Navrhněte stavový automat pro semafor
- Stavy
  - Červená
  - Oranžová
  - Zelená
  - Oranžovo-Červená
- Co tyto stavy reprezentují?
  - Jednoznačnou situaci
  - Minulost i budoucnost
- Co reprezentují přechody
  - Změna stavu
  - Akce

Semafor

- Stavy
  - Červená
  - Oranžová
  - Zelená
  - Oranžovo-Červená
- Co spustí přechod
  - Vstupní signál
  - Časování
- (A)synchronnost

Semafor

- Stavy
  - Červená
  - Oranžová
  - Zelená
  - Oranžovo-Červená
- Speciální stavy
  - Nečinnost
  - OranžováBliká
  - ČervenáStále
  - ČervenáChyba

Semafor

- Stavy
  - Červená
  - Oranžová
  - Zelená
  - Oranžovo-Červená
  - Nečinnost
  - OranžováBliká
  - ČervenáStále
  - ČervenáChyba
- Počáteční stav
- Konečný stav
- Chybový stav

Semafor

- Stavy
  - Červená
  - Oranžová
  - Zelená
  - Oranžovo-Červená
  - Nečinnost
  - OranžováBliká
  - ČervenáStále
  - ČervenáChyba
- Jak řešit více přechodů z jednoho stavu?
  - Vstupní signál

Semafor

- Výhody řešení pomocí stavových automatů
  - Rozšiřitelnost
    - o podmínky
    - o požadavky
  - Dobrá grafická reprezentace
  - Udržitelnost kódu

Stavový automat

- Abstraktní model
- Základní typy
  - konečné automaty deterministické (default)
  - nedeterministické
- Různé definice
  - na automaty
  - terminologii
- Detailně v jiných předmětech – zde seznámení

Stavový automat - obsah

- Přechodová funkce
- Stavy
  - Počáteční
  - Přechodové
  - Koncové
  - Chybové
  - Aktuální
- Zdroj dat/zdrojů (i HW)
- Abeceda – množina vstupních znaků (i prázdná množina)
- Reakce
  - Akce
  - Přechod do jiného stavu
  - Reset

Přechodová funkce

- Skládá se z přechodů
  - každý má
    - definiční obor
    - obor hodnot
- Přechod je dán
  - stavem odkud
  - akcí
  - stavem kam
  - a případně řeší minulost

Přechodová funkce

- Lambda přechod
  - přechod, který nereaguje na vstupní data
  - příklad:
    - Zpracování sekvence dat
    - Sekvence „1 2“ značí začátek zprávy
    - Sekvence „242“ značí konec zprávy
    - Uprostřed konečný počet opakování jiné sekvence

    - Lambda přechod zajistí opakování

    - Zpracování sekvence
      - 1 2 2 4 1 2 4 1 2 4 1 2 4 2

Abeceda

- Množina vstupních znaků
- Konečná množina
- Gramatika
  - Terminální znaky
  - Neterminální znaky
  - Pravidla
  - Počáteční terminál (kořen gramatiky)

Příklad - Vyhledávací automat

- Hledáme podřetězec (i více) v podřetězci

- Nedeterministický
  - Automat má více větví se stejným vstupem
- Deterministický
  - Většinou lze z nedeterministického
  - To vede k větší komplexnosti

Příklad - Vyhledávací automat

- Hledáme „ahoj“
  - Nedeterministické řešení
    - v prvním stavu čekáme na příchod znaku „a“
    - po přechodu na další stavy zároveň čekáme na další příchod znaku „a“
    - koncový stav zastaví všechny ostatní stavy/započaté vyhledávání
- Hledáme slovo „babka“
  - Opakuje se začátek
  - Složitější automat
- Hledáme zároveň slovo „jeden“ a „den“
  - Spojení větví
  - V konečném stavu nevíme, které slovo jsme našli
  - Podobně „end“ a „endif“ – u preprocesoru

Konečný automat

- matematický model
- výpočetní stroj s konečnou pamětí
- pokud vstupní slovo je akceptováno
  - dostáváme se do (některého) konečného stavu
- nemusí existovat jediné řešení
- dostaneme-li se do konečného stavu
  - sekvence patří mezi akceptovaná slova
- známe cestu
  - syntaktická analýza
  - víme přesně „co“ přišlo

Turingův stroj

- obecný model výpočtu (algoritmu)
- snaha zjistit co se dá spočítat
  - je to algoritmus?
  - má konečném počtu kroků?
  - je krok realizovatelný?
  - má končený počet stavů?
- v závislosti na načteném znaku a stavu:
  - (ne)změní stav
  - zapíše znak (zde na místo znaku čteného)
  - posune se pro další čtení (zde dopředu nebo dozadu)

Deterministické konečné automaty

- Z každého stavu při daném vstupu vždy jen jeden možný přechod
- Základní představitelé
  - Moore
  - Mealy

  - Možné převádět mezi nimi

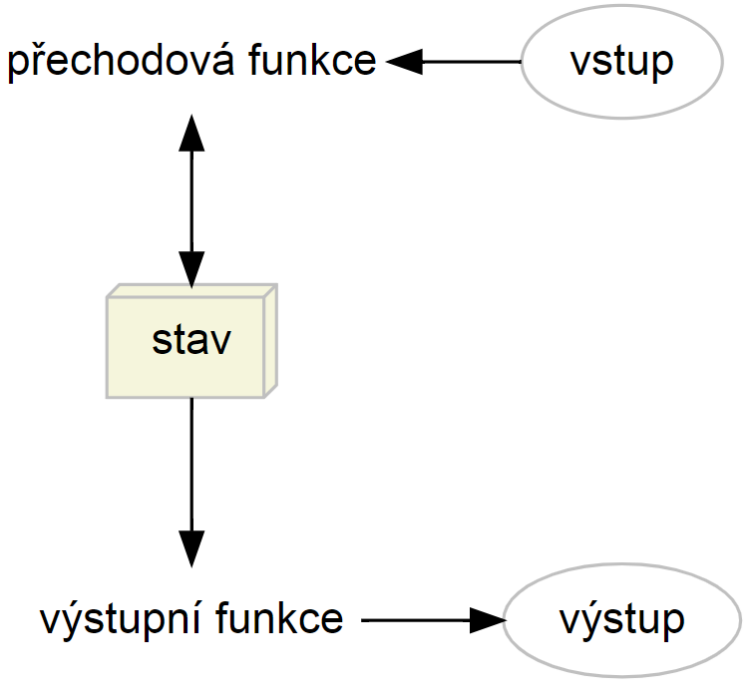
Moore

- Konečný počet vnitřních stavů
- Každý stav definuje právě jeden výstup
- Přechody na základě vstupů
- Musí mít výchozí vnitřní stav
- **Výstup určen jen na základě vnitřního stavu**

```
u - vstupy
x - stavy
y - výstupy
```

$$x = f(x,u)$$

$$y = f(x)$$

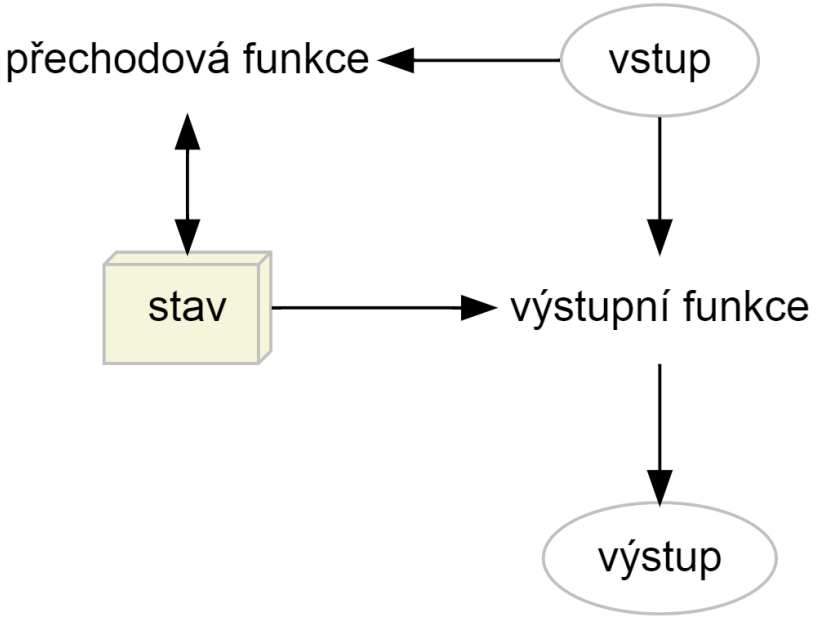


Mealy

- Konečný počet vnitřních stavů
- Méně stavů než srovnatelný Moore
- Každý stav definuje právě jeden výstup
- Přechody na základě vstupů
- Musí mít výchozí vnitřní stav
- **Výstup určen na základě vnitřního stavu a vstupu**

```
u – vstupy
x – stavy
y – výstupy
```

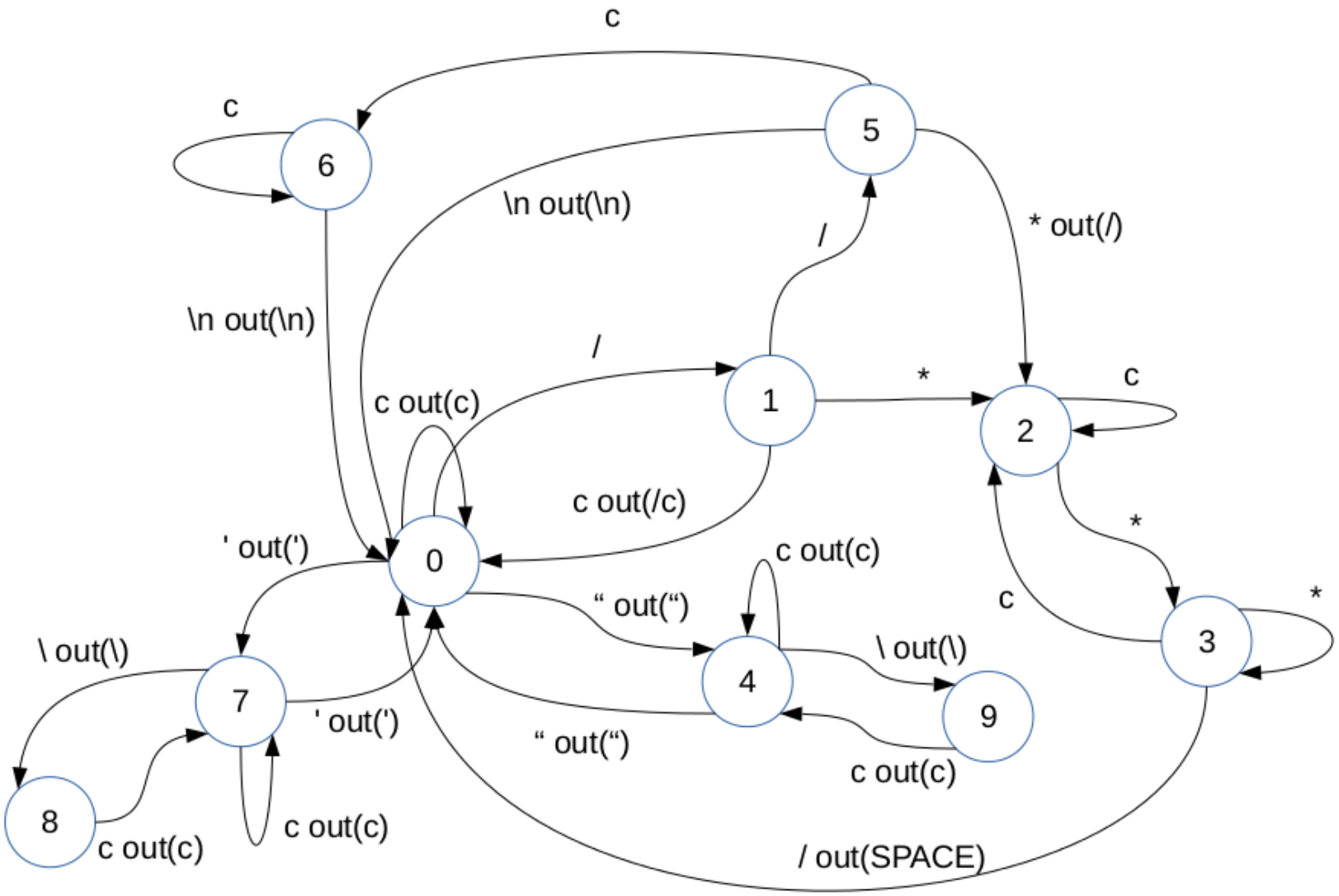
$$x = f(x,u)$$
$$y = f(x,u)$$



Příklad: Odstranění komentářů z kódu

- Napište program, který odstraní z kódu komentáře
  - jednořádkové `//komentář`
  - víceřádkové `/*dlouhý komentář*/`
  - v řetězci nic nedělej
  - Víceřádkový komentář nahraďte mezerou
- Stavy
  - TEXT
  - LOMITKO
  - KOMENTAR1
  - KOMENTAR2_ZACATEK
  - KOMENTAR2_KONEC

Příklad: Odstranění komentářů z kódu



https://www.itnetwork.cz/cplusplus/cecko/linux/cecko-linux-zdrojove-kody/program-c-linux-delcom/

Realizace stavových automatů

- Podmíněné rozhodování
  - If () … else …
  - Switch
  - Stavy uložené v Enum

- Ukazatele na funkce
  - Vysvětlení ukazatelů na funkce
  - Stavový automat s ukazateli na funkce

Podmíněné rozhodování

- Uložení stavů

```
enum stavy { TEXT,LOMITKO,…}  //definice typu
enum stavy stav = TEXT;       //uložení stavu
```

- Rozhodování

```
while(1) {
      if(stav == TEXT)
            akce1();
      else if (stav == LOMITKO)
            akce2();
      else
            akce3();
      čekej(čas)
}
```

Podmíněné rozhodování

- Uložení stavů

```
enum stavy { TEXT,LOMITKO,…}  //definice typu
enum stavy stav = TEXT;       //uložení stavu
```

- Rozhodování

```
while(1) {
      switch(stav) {
      case TEXT:
            akce1();
            break;
      case LOMITKO:
            akce2();
            break;
      default:
            akce3();
      }
      čekej(čas)
}
```

Ukazatele na funkce

- Uložení stavů

```
void text();                    //definice stavu
void lomitko();
void (*stav)() = text;          //uložení stavu
```

- Rozhodování

```
void text() { akce1(); stav = lomitko; }
void lomitko(){ akce2(); stav = text; }

int main (){
      while(1) {
            (*stav)();
            čekej(čas)
      }
}
```

Ukazatele na funkce

- Uložení stavů

```
typedef void *(*StavPtr)();
      //definice ukazatele na další funkci
void *text();                    //definice stavu
void *lomitko();
```

- Rozhodování

```
void *text() { akce1(); return lomitko; }
void *lomitko(){ akce2(); return text; }

int main (){
      StavPtr stav = text;
      while(1) {
            stav = (StavPtr)(*stav)();
            čekej(čas)
      }
}
```

Děkuji za pozornost