Ročníkový projekt: Knihovnda pro testování restartujících automatů a webové prostředí

Jan Urban

October 2019

Obsah

Úvod

Tato práce pojednává o tvorbě knihovny pro práci s restartujícími automaty. A vizualizaci restartujících automatů ve webovém prostředí. V první kapitole zadefinuji třídy restartujících automatů a vlastnosti, které podporuje tato knihovna. Druhá kapitola se zaměřuje na webové rozhraní. Ve třetí je podrobněji rozepsáno jak s knihovnou pracovat. A ve čtvrté je jak knihovna funguje.

0.1 Redukční analýza

Redukční analýza je metoda, která pomocí odstraňování a přesouvání slov ve větě, redukuje větu na kratší větu a zachová správnost věty.

0.2 Příklad použití restartujících automatů

Příklad od lingvistů, kde restartující automat mohou použít na redukční analýzu věty:

```
"Trosky zaměstnance, který tam plnil své úkoly, zcela zavalily"
"Trosky zaměstnance, který plnil své úkoly, zcela zavalily"
"Trosky zaměstnance, který plnil úkoly, zcela zavalily"
"Trosky zaměstnance zcela zavalily"
"Trosky zaměstnance zavalily"
```

Což je korektní věta a z toho vyplývá, že počáteční věta je syntakticky správná. Na rozdíl od "Trosky zavalily"která je věta neúplná. Stejně nekorektní, tedy neúplná, je následující větev úprav:

```
"Trosky zaměstnance zcela zavalily"
"Trosky zcela zavalily"
```

Z čehož je už možné zjistit že věta je špatně, nebo po další redukci na "Trosky zavalily"

1 Typy automatů které program rozpozná

Zde jsou zadefinované třídy restartujících automatů a vypsané vlastnosti, které knihovna rozpoznává.

Definice 1 (RLWW) *Nechť M je RLWW automat pak platí $M = (Q, \Sigma, \Gamma, \#, \$, q_0, k, \delta)$.*

- *Q je konečná množina stavů.*
- *Σ je konečná vstupní abeceda.*
- *Γ je konečná pracovní abeceda která obsahuje Σ, , ,*
- *$\#, \$ \notin Q$ jsou symboly které označují levou zarážku a pravou zarážku, neboli začátek a konec vstupního slova.*
- *$q_0 \in Q$ je počáteční stav.*
- *$k \in N_+$ je nezáporné celé číslo udávající velikost čtecího/zapisovacího okna.*
- *$\delta : Q \times PC^{(k)} \to 2^{((Q \times (\{MVR, MVL\} \cup PC^{\leq (k-1)})) \cup \{Restart, Accept\})}$ je tranzitivní relací. Kde $PC^{(k)}$ je množina možných obsahů čtecího/zapisovacího okna M.*

$PC^{(k)} := (\# \cdot \Gamma^{i-1}) \cup \Gamma \cup (\Gamma^{\leq i-1} \cdot \$) \cup (\# \cdot \Gamma^{\leq i-2} \cdot \$)(i \geq 0)$

$\Gamma^{\leq n} := \bigcup_{i=0}^{n} \Gamma^i$

$PC^{\leq (k-1)} := \bigcup_{i=0}^{k-1} PC^i$

1.0.1 Něco více o pravidlech

1.

2. $(q, u) \to (q', v)$ – přepíše u na v, kde v musí být kratší než u. To se docílí smazáním alespoň jednoho symbolu a nahrazením dalších symbolů z u.

3. *Restart* změní stav na počáteční a přesune hlavu na začátek pásky, tak že první znak v okně bude levá zarážka (#).

Definice 2 (Konfigurace) *Konfigurací automatu A máme na mysli dvojici skládající se z současného stavu automatu A a ze vstupu z okna automatu A. Tato konfigurace je zapsána dvojicí* $(q \in Q, s \subseteq (\Sigma \cup \Gamma)^*)$. *Kde q je stavem automatu A a s je vstup z pásky.*

Definice 3 (Krok) *Krokem automatu A rozumíme provedení jednoho pravidla. Pravidla mají zápis* $(q_0, u) \to (q_1, pravidlo)$. *Kde* $q_0, q_1 \in Q$, $u \subseteq (\Sigma \cup \Gamma)^*$ *a pravidlo náleží jednomu ze 3. možností:*

- *MVL – posun hlavy automatu doleva*
- *MVR – posun hlavy automatu doprava*
- $v \subseteq (\Sigma \cup \Gamma)^*$ *– přepsání znaků pod hlavou automatu na v (vždy menší nebo stejné délky jako u)*
- *Restart – Automat přejde do počátečního stavu* q_0

Definice 4 (Přijmutí slova) *Slovo je přijato pokud existuje posloupnost pravidel, které se na daném slovu dostanou z počátečního stavu do přijímajícího.*

Definice 5 (Determinismus) *Pokud má automat pro každou dvojici* (q, u), *kde* $q \in Q$, $u \subseteq (\Sigma \cup \Gamma)^*$ *a platí alespoň jedno z následujících tří pravidel:* $|u| = k$, $|u| < k$ *a končí na \$, nebo* $|u| < k$ *začíná na # a končí na \$. nejvýše jedno pravidlo s levou stranou* (q, u), *tak říkáme, že automat je deterministický.*

1.1 monotonie

1.2 RRWW

Vychází z RLWW-automatu ale nepoužívá MVL – posun doleva

1.3 RRW (RLW)

Vychází z RRWW (RLWW)-automatu ale má totožnou pracovní abecedu s vstupní abecedou

1.4 RR (RL)

Vychází z RRW (RLW)-automatu, ale zakazuje přepisovat, tedy může jen mazat

1.5 R

Vychází z RR automatu, ale restartuje se ihned po přepsání (smazání)

1.6 RWW

Vychází z RRWW automatu, ale po přepsání se ihned restartuje

2 Webové prostředí

- Možnost zadání automatu buď v textové formě nebo nahráním souboru.
- Vypsat přijímané slova do nějaké velikosti zadané uživatelem.
- Otestovat zda dané slovo je přijímáno a jak bylo přijato (přes jaké stavy ...)
- nějaká možnost si zapsat program instrukcemi podporované pythonem a mojí knihovničkou, nejspíše nějakou webovou konzolí
- ptát se co to je za automat
- vybírat možnosti výstupu (vypsat celou cestu graficky, vypsat jen stavy)

3 Jak funguje tato knihovna

Tento automat je nedeterministický, z důvodů že musí umět přijímat i nedeterministické automaty a je zásobníkového typu. Tedy rozhoduje se zda přijme, nebo nepřijme dané slovo na základě stavu automatu.

3.1 Instrukce automatu

Tento automat implementuje několik instrukcí:

1. MVR/MVL (move right/left, posun doprava/doleva)
2. Restart
3. Accept (přijmi, vlastně restart)
4. "['x']" (přepiš svojí pozici na "x")

3.2 Jak inicializovat automat

Inicializovat automat můžeme dvěma způsoby, ze souboru a nebo vytvořením nového automatu. Soubor do kterého se soubor ukládá je typu json, tedy nějaký slovník klíčů a nějakých dat.

Listing 1: Inicializace automatu

```
from automata import automaton
a = automaton()
a.definition["size_of_window"] = 3
a.definition["s0"] = ["q0", 0]
a.add_to_alphabet("#", "$", "a", "b", "c", "d")
a.add_accepting_state("Accept")
```

3.3 Jak přidat jednotlivé pravidla

Pokud chceme přidat nějaké pravidlo, například ze stavu q_0 a s viditelnou páskou odpovídající #ab chceme zůstat ve stavu q_0 a posunout se doprava tedy: $(q_0, \#ab) -> (q_0, \text{MVR})$. Stačí napsat:

```
a.add_instr("q0", "#ab", "q0", "MVR")
```

Kvůli tomu že dovoluji, že znak abecedy může být i nějaká n-tice symbolů, tedy klidně i "VíceSymbolů"braný jako znak. Pokud by jsme chtěli použít nějaký nepísmenný znak u automatu s velikostí okna 3, tak použijeme tento zápis:

```
a.add_instr("q0", "[ '#', 'Tohle je jeden znak', 'b']",
  "q0", "MVR", value_as_list=True)
```

Můžeme zapsat pravidlo i v regulárních výrazech (Regular Expressions). Neboli * pro cokoliv, ? pro jeden znak ... (ještě není implementováno, bude doplněno) Jen je potřeba psát to "listovým"zápisem (viz výše).

```
a.add_instr("q0", "[ '#', '*', '?a*']",
  "q0", "MVR", value_as_list=True)
```

3.4 Otestování slova, zda bude přijata automatem

Otestování na výstup buď zamítne (napsáním False), nebo přijme (True) a když dané slovo přijme, tak vypíše na konzoli jak automat postupoval. Páska by měla být ohraničená zarážkami # a $. Zapsat můžeme tímto způsobem:

```
a.iterate_text("#aaabbbc$")
```

Více symbolové znaky se na pásku dají zapsat pomocí ohraničení do hranatých závorek. Berou se jen vnější závorky a vše co je uvnitř se chápe jako jeden znak. Ale pokud se uvnitř použijí hranaté závorky musí být vždy v páru.

```
a.iterate_text("#aa[Znak]bbc$")
a.iterate_text("#aa[[a]cb]bbc$")
```