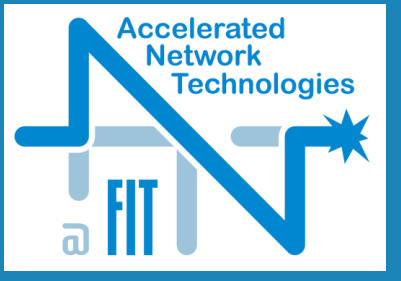
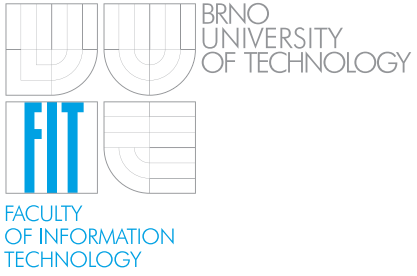
NFA redukce pomocí simulace



Vlastimil Košař

Brno University of Technology, Faculty of Information Technology
Božetěchova 2, 612 00 Brno, CZ
www.fit.vutbr.cz/~ikosar



INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Obsah

Motivace

- Potřeba redukovat velikost NFA vygenerovaného z množiny RV
- Minimalizace NFA nepřipadá v úvahu - NP-úplný problém
- Nutné použít metody redukce NFA mající polynomiální složitost
- Požadujeme, aby nebyly koncové stavy sloučeny redukcí do jednoho

Minimalizace NFA

Vlastnosti

- Určení tříd ekvivalence pro minimalizaci nedeterministického automatu je NP-úplný problém
- Pro daný NFA neexistuje jediný minimální NFA

Algoritmus minimalizace NFA

1. Začni s jedním stavem
2. Přidávej vhodné přechody a kontroluj ekvivalenci s NFA
3. Vyčerpaly-li se možnosti kombinací přechodů, přidej další stav a prováděj předchozí bod
4. Algoritmus končí, je-li nalezen ekvivalentní automat

Vlastnosti redukce

- Redukují velikost automatu
- Mají polynomiální složitost
- Založeny na různě definovaných relacích stavové ekvivalence

Trace equivalence

Vrchol v trace-dominates vrchol u pokud pro každou konečnou u-kořenovou cestu $\overline{u}$ existuje v-kořenová cesta $\overline{v}$ taková, že platí $\langle\langle\overline{u}\rangle\rangle = \langle\langle\overline{v}\rangle\rangle$. Vrcholy u a v jsou trace-equivalent ($u \approx^T v$) pokud v trace-dominates u a u trace-dominates v.

Bisimilarity

Binární relace $\cong \subseteq V^2$ je bisimulace pokud $u \cong v$ implikuje:

1. $\langle\langle u\rangle\rangle = \langle\langle v\rangle\rangle$
2. Pro všechny vrcholy $\acute{u} \in post(u)$ existuje vrchol $\acute{v} \in post(v)$ takový, že platí $\acute{u} \cong \acute{v}$
3. Pro všechny vrcholy $\acute{v} \in post(v)$ existuje vrchol $\acute{u} \in post(u)$ takový, že platí $\acute{u} \cong \acute{v}$

Vrcholy u a v jsou bisimilar ($u \approx^B v$), pokud existuje bisimulace $\cong$ taková, že platí $u \cong v$

Význačné druhy stavové ekvivalence 2

Similarity

Binární relace $\preceq \subseteq V^2$ je simulace pokud $u \preceq v$ implikuje:

- $\langle\langle u \rangle\rangle = \langle\langle v \rangle\rangle$
- Pro všechny vrcholy $\acute{u} \in post(u)$ existuje vrchol $\acute{v} \in post(v)$ takový, že platí $\acute{u} \cong \acute{v}$

Vrcholy u a v jsou similar ($u \approx^S v$), pokud platí $u \subseteq v$ a $v \subseteq u$

Vlastnosti

- Jednotlivé druhy stavové ekvivalence se liší tím, jaké vlastnosti zachovávají (např. z pohledu temporální logiky,...)
- Platí: $u \approx^B v \Rightarrow u \approx^S v \Rightarrow u \approx^T v$

Vysvětlivky

- Ohodnocený graf $G = \{V, E, A, \langle\langle . \rangle\rangle\}$
- $post(v) = \{u | (v, u) \in E\}$

Postup simulační redukce

Kroky

1. Vypočítej počáteční simulaci
2. Vypočítej relaci simulace
3. Urči relaci similar - výběr symetrické podrelace relace simulace
4. Vybereme ireflexivní podrelaci
5. Stavy, které jsou v relaci s jiným sloučíme

Počáteční simulace

- Umožnuje ovlivnit určité aspekty chování redukce - zachování informace o příslušnosti koncového stavu k RV, apod.
- V základní verzi každý stav simuluje každý stav - sloučeny všechny počáteční a koncové stavy
- Ve verzi se zachování informace o příslušnosti koncového stavu k RV zabráníme tomu, aby koncový stav simuloval nekoncový stav a aby se koncové stavy navzájem simulovaly pokud nemají stejné číslo RV
- Ve verzi se zachováním počátečních stavů postupujeme obdobně jako v předchozí verzi

Výpočet relace simulace

Cíl

- Iteračním výpočtem určit restrikcí počáteční simulace relaci simulace

Algoritmus

```
while refine == True:
    refine = False
    for src_state in self._automaton.states.keys():
        pair = False
        for simulation_pair in simulation[src_state]:
            lhs = True
            for lhs_transition in mapper[simulation_pair[0]]:
                rhs = False
                for rhs_transition in mapper[simulation_pair[1]]:
                    rhs = rhs or (lhs_transition[1] == rhs_transition[1] and ((lhs_transition[2],
rhs_transition[2]) in simulation[lhs_transition[2]]))
                lhs = lhs and rhs
            pair = pair or lhs
            if lhs == True:
                new_simulation[simulation_pair[0]].append(simulation_pair)
        refine = refine or pair
    if simulation == new_simulation:
        refine = False
```

Opakovaná obousměrná redukce

- Lepších výsledků redukce je možné dosáhnout opakovaným aplikování redukce v obou směrech
- Místo implementace redukce ve druhém směru je možné použít reverze automatu a redukce

Reverze automatu

Algoritmus

1. Počáteční stavy se stanou koncovými stavy reverzovaného automatu
2. Koncové stavy stavy se stanou počátečními stavy reverzovaného automatu
3. Záměnou zdrojových a cílových stavů v přechodové funkci zajistíme reverzaci přechodů v reverzovaném automatu

Opakovaná obousměrná redukce

Algoritmus

1. Proved simulační redukci NFA
2. Reverzuj automat
3. Opakuj kroky 1 a 2 dokud se velikost automatu zmenšuje
4. V případě potřeby reverzuj automat zpět

Ilustrace opakované obousměrné simulační redukce

Sada regulárních výrazů

- $/[ab]^*aba^*/$
- $/[ab]^*aa^*b(b|aa^*b)/$

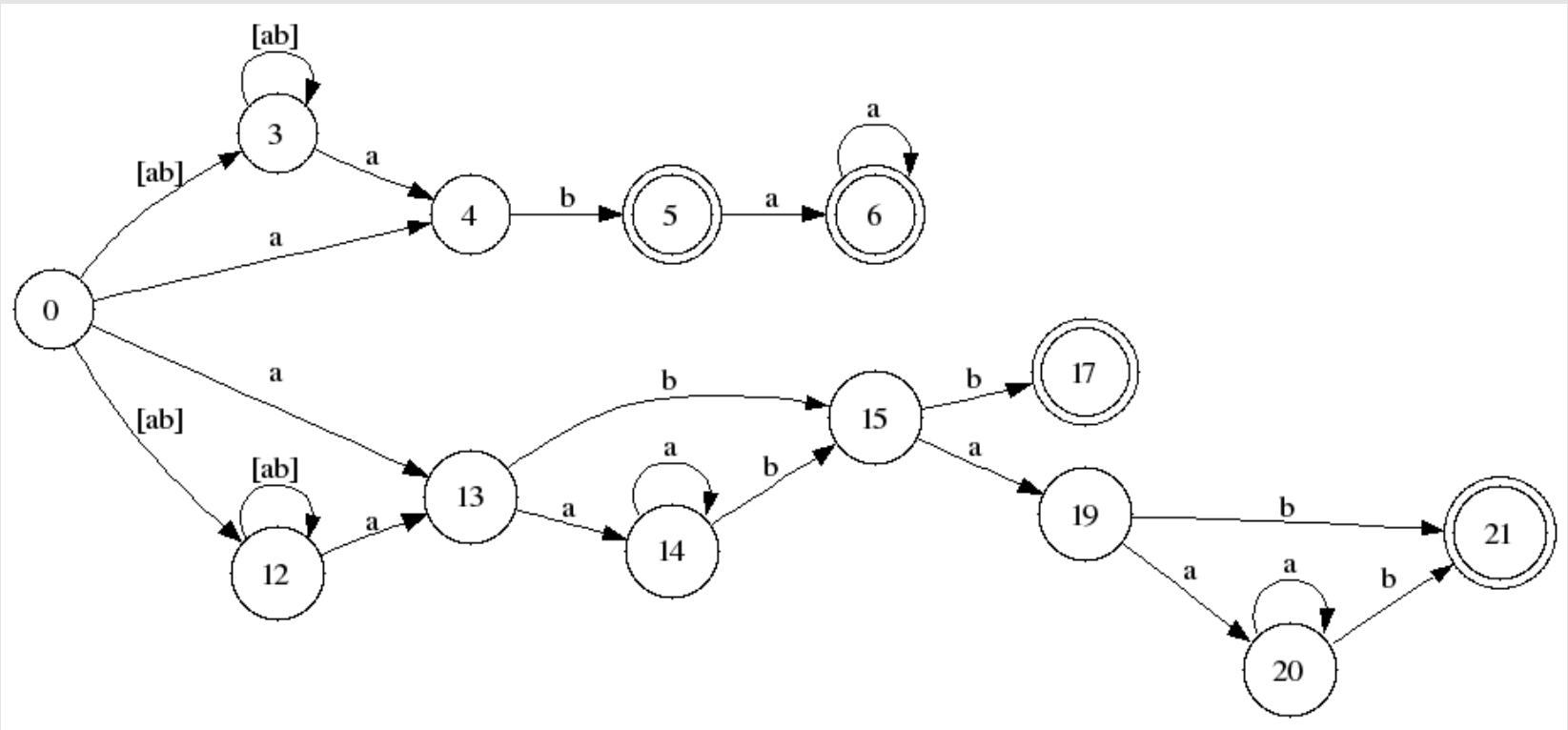
Počet stavů a přechodů automatu

- Stavů: 13
- Přechodů: 21

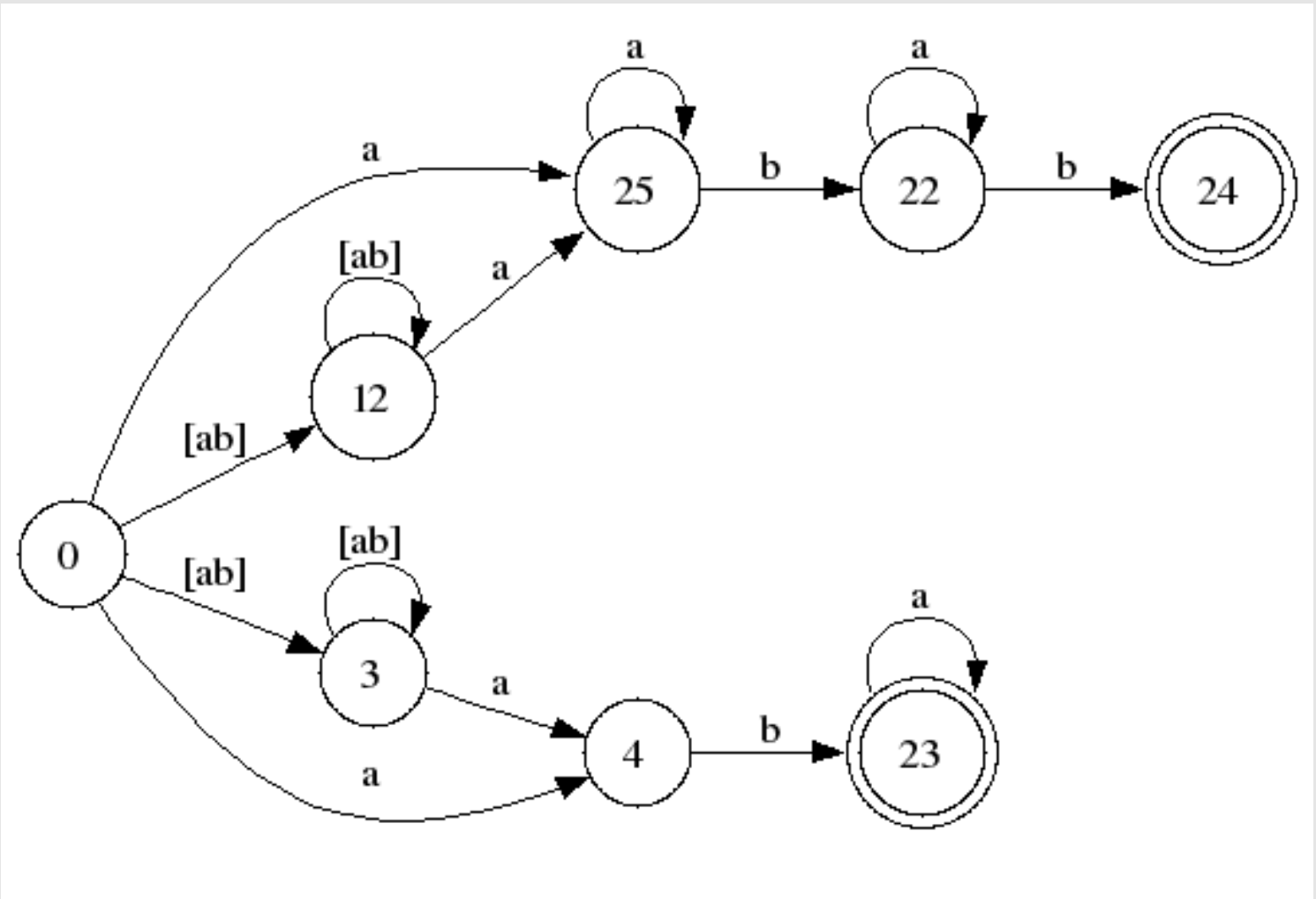
Počet stavů a přechodů redukovaného automatu

- Stavů: 5
- Přechodů: 8

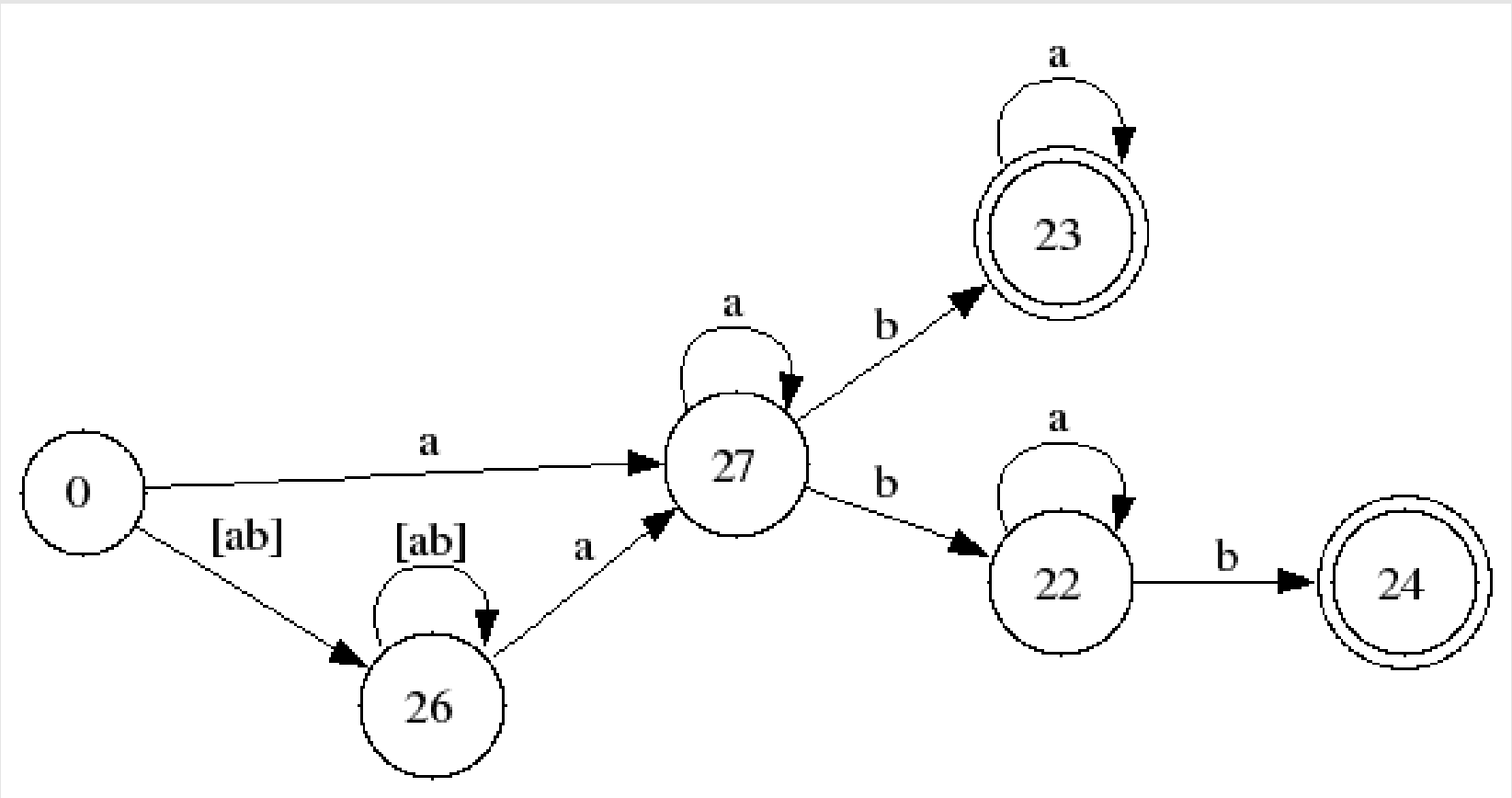
Originální NFA



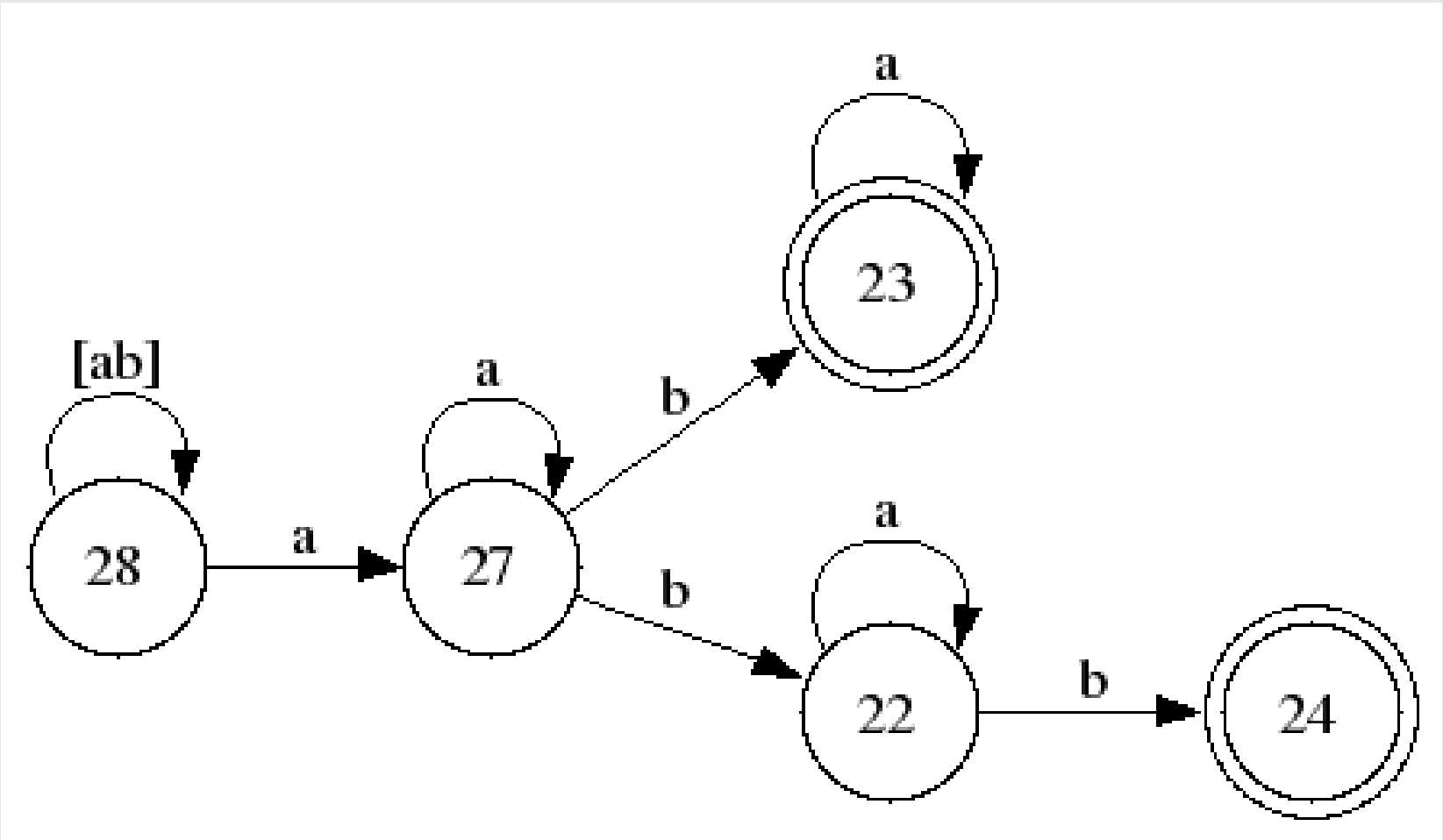
I. iterace opakované obousměrné simulační redukce



II. iterace opakované obousměrné simulační redukce



III. iterace opakované obousměrné simulační redukce



Simulační redukce

Pravidla	Originální NFA		Reduk. NFA		Úspora (%)	
	Stavy	Přech	Stavy	Přech	Stavy	Přech
chat	186	226	177	210	4.84	7.07
dpd	218	314	181	252	16.97	19.75
dos	182	227	159	183	12.64	19.38
web-php	322	368	269	325	16.46	11.68
virus	52	57	52	57	0	0
shellcode	95	117	81	98	14.74	16.24
l7	806	950	733	873	9.06	8.1
backdoor	3888	4431	3783	4271	2.7	3.6

Opakovaná obousměrná simulační redukce

Pravidla	Originální NFA		Reduk. NFA		Úspora (%)	
	Stavy	Přech	Stavy	Přech	Stavy	Přech
chat	186	226	120	135	35.48	40.27
dpd	218	314	115	150	47.25	52.23
dos	182	227	73	83	59.89	63.44
web-php	322	368	244	283	24.22	23.10
virus	52	57	19	19	63.46	66.67
shellcode	95	117	60	69	36.84	41.03
l7	806	950	613	751	23.95	20.95
backdoor	3888	4431	3229	3679	16.94	16.97

Závěr

Shrnutí

1. Je možné provést redukci velikosti automatu v polynomiálním čase
2. Při použití redukce úspora až 20% na reálných pravidlech
3. Při použití opakované obousměrné redukce úspora až 66% na reálných pravidlech
4. Doba běhu je závislá na tom, jak moc si jsou pravidla podobná, což vede zpravidla ke zvětšování počtu iterací
5. Doba běhu je závislá na tom, kolik se vyskytuje stavů s velkým počtem přechodů mezi nimi, což vede k prodlužování doby běhu iterace

Otázky