VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

PŘEŽITÍ SILNĚJŠÍHO V MINIMALIZACI NEDETERMINISTICKÝCH AUTOMATŮ

SURVIVAL OF THE STRONGER IN MINIMIZATION OF NONDETERMINISTIC AUTOMATONS

PROJEKTOVÁ PRAXE
PROJECT PRACTICE

AUTOR PRÁCE — **MICHAL ŠEDÝ**
AUTHOR

VEDOUCÍ PRÁCE — **Mgr. LUKÁŠ HOLÍK, Ph.D.**
SUPERVISOR

BRNO 2019

Abstrakt

Problém minimalizace nedeterministických konečných automatů je řešen pomocí výpočtů jazykových inkluzí. Tento přístup je ale výpočetně náročný, a proto se využívá aproximační metoda, simulace. Ta ale nemusí detekovat veškeré jazykové inkluze. Práce přináší nový přístup k minimalizaci dvou stavů automatu. Definuje případy, ve kterých se nevyužívá slučování, ale odstranění slabšího z nich. Dále jsou uvedeny podmínky, za kterých lze provést minimalizaci stavů pouze na základě jednostranné jazykové inkluze, která byla doposud ve výsledcích simulace opomíjena. Tímto přístupem se zefektivní využití získaných dat ze simulace a dosáhne minimálnější formy výsledného automatu.

Abstract

The problem of the minimization of nondeterministic finite automata is solved by the calculation of language inclusions. This approach has high time complexity, therefore is using the approximation method, simulation. This method does not have to detect all language inclusions. The paper presents a new approach to the minimization of two automaton's states. Defines cases in which the merging of states is not used, but the weaker is removed. The next part determines statements, in which is possible to do minimization of two states based on unilateral language inclusion. This method is very helpful because the simulation detects a lot of unilateral language inclusions, that have not yet been used. These approaches make simulation data more effective, which will lead to a more minimal form of the automaton.

Klíčová slova

konečný automat, nedeterministický automat, minimalizace, jazyková inkluze, relace simulace

Keywords

finite automata, nondeterministic automata, minimization, language inclusion, simulation relation

Citace

ŠEDÝ, Michal. *Přežití silnějšího v minimalizaci nedeterministických automatů*. Brno, 2019. Projektová praxe. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Mgr. Lukáš Holík, Ph.D.

Obsah

Kapitola 1

Úvod

Konečné automaty, deterministické i nedeterministické, hrají významnou úlohu v reprezentaci výpočetních modelů. Pomáhají popisovat a rozpozpoznávat regulární jazyky, s čímž také souvisí výstavba překladačů, ve které se uplatňují především při detekci lexémů daného jazyka. Dále své uplatnění nacházejí v hardwaru, v jeho návrhu, kontrole síťové komunikace, ve verifikaci programů, vykonávání úkolů strojového učení a v mnoha dalších oblastech. Nedeterministické konečné automaty (NKA) [6] skýtají na rozdíl od deterministických (DKA), výhodu minimalističtější formy. Pravidla přechodů automatu typu NKA umožňují více přechodů ze stávajícího stavu po přijetí stejného symbolu. Díky této klíčové vlastnosti mohou obsahovat méně stavů a přechodů než jejich funkčně ekvivalentní deterministické varianty.

U deterministických automatů existují v současnosti efektivní algoritmy, pro jejich minimalizaci, jak prezentovali Ilie, Navarro a Yu [2]. Pří výpočtu minimalizace NKA jsou časově složité výpočty jazykových inkluzí nahrazeny aproximační metodou, simulací. Přístup založený na simulaci vyhodnocuje relaci dvou stavů a zkoumá, zda stav p nemůže být simulován jiným stavem q. Nevýhodou simulace je skutečnost, že se jedná pouze o aproximaci jazykové inkluze [1]. Simulace sice implikuje inkluzi jazyků, ale opačné tvrzení neplatí. Existují případy, kdy simulace jazykovou inkluzi nenalezne.

Tato práce specifikuje případy, ve kterých lze provést minimalizaci pouhým odstraněním stavu. Dále přináší nové podmínky, díky nimž lze informace získané výpočtem simulace o jazykových inkluzích efektivněji využít, se zaměřením na jednostrannou jazykovou inkluzi, a tím zvýšit počet možných minimalizací i s případně chybějící informací, kterou simulace neodhalila.

Kapitola 2 obsahuje základní definice týkající se konečného nedeterministického automatu. Definice případů, ve kterých lze provést minimalizaci odstraněnín stavu, nalezne čtenář v kapitole 3. V kapitole 4 jsou uvedeny speciální podmínky, umožňující minimalizaci pouze na základě jednostranné jazykové inkluze.

Kapitola 2

Konečné automaty

Táto kapitola obsahuje základní definice a pojmy nedeterministického konečného automatu (NKA), které čtenáře seznámí s notací a tak jim umožní pochopit prezentované myšlenky. (Definice 2.1.–4. byly převzaty z [3, 4] a definice 2.8.–10. z [5].)

2.1 Prerekvizity

Definice 2.1 *Nechť je dána abeceda* Σ*, pak rekurzivní definice* ***řetězce nad abecedou*** *má následující tvar:*

1. ϵ *je řetězcem nad abecedou* Σ *a značí prázdný řetězec.*
2. *Pokud je* w *řetězec nad abecedou* Σ*, pak je i* aw*, kde* $a \in \Sigma$*, také řetězcem nad abecedou* Σ*.*

Definice 2.2 *Nechť* Σ^* *značí množinu všech řetězců nad* Σ*. Každá podmnožina* $L \subseteq \Sigma^*$ *je* ***jazyk*** *nad* Σ*.*

Jazyk představuje libovolnou množinu řetězců, které vznikly kombinací znaků dané abecedy. Přestože řetězce a abeceda nabývají konečných délek, jazyk může být nekonečný. Mějme kupříkladu abecedu $\{0, 1\}$, pak nad touto abecedou může existovat řetězec ve tvaru 01001. Jazyk může být definován třeba jako množina všech řetězců se sudým počtem znaku 1.

Definice 2.3 ***nedeterministický konečný automat*** je pětice $M = (Q, \Sigma, \delta, q_0, F)$

1. Q *je konečná neprázdná množiva stavů,*
2. Σ *je vstupní abeceda,*
3. $\delta : Q \times \Sigma_\epsilon \longrightarrow 2^Q$ *je přechodová funkce,*
4. $q_0 \in Q$ *je počáteční stav,*
5. $F \subseteq Q$ *je konečná neprázdná množina koncových stavů.*

Přechodová funkce δ specifikuje množinu pravidel přechodů R. Pro pravidlo r: $q \in \delta(p, a)$ budeme využívat zápis $pa \rightarrow q$, kde p, q, $\in Q$, $a \in \Sigma$ a $r \in R$. Tento přechod značí, že po přečtení vstupního znaku a ve stavu p dojde k přechodu do stavu q.

Definice 2.4 *Nechť je dán NKA* $M = (Q, \Sigma, \delta, q_0, F)$*, pak* ***konfigurace*** *je řetězec* $\chi \in Q\Sigma^*$

Konfigurace automatu znázorňuje informaci o aktuálním stavu, ve kterém se automat nachází, a zbytku vstupního řetězce čekajícího na zpracování. Například pokud se automat M nachází ve stavu q a na vstupu zbývá řetězec ab, pak nabývá konfigurace tvaru qab.

Definice 2.5 *Nechť* paw *a* qw *jsou dvě konfigurace NKA* M*, kde* $p, q \in Q$*,* $a \in \Sigma \cup \epsilon$ *a* $w \in \Sigma^*$*. Nechť* $r = pa \rightarrow q \in R$ *je pravidlo. Potom* M *může provést* ***přechod*** *z* paw *do* qw *za použití* r*, zapsáno* $paw \vdash qw$ $[r]$ *nebo zjednodušeně* $paw \vdash qw$

Nechť χ je konfigurace. M provede nula přechodů z χ do χ. Zapisujeme: $\chi \vdash^0 \chi$ $[\epsilon]$ nebo zjednodušeně $\chi \vdash^0 \chi$.

Nechť χ_0, χ_1, ...,χ_n, je sekvence konfigurací pro $n \leq 1$ a $\chi_{i-1} \vdash \chi_i$ $[r_i]$, $r_i \in R$ pro všechna $i = 1, ..., n$, což znamená: $\chi_0 \vdash \chi_1$ $[r_1] \vdash \chi_2$ $[r_2]$ ... $\vdash \chi_n$ $[r_n]$. Pak M provede *n-přechodů* z χ_0 do χ_n. Zapisujeme: $\chi_0 \vdash^n \chi_n$ $[r_1...r_n]$ nebo zjednodušeně $\chi_0 \vdash^n \chi_n$.

Pokud $\chi_0 \vdash^n \chi_n$ $[\rho]$ pro nějaké $n \geq 1$, pak jej značíme $\chi_0 \vdash^+ \chi_n$ $[\rho]$.

Pokud $\chi_0 \vdash^n \chi_n$ $[\rho]$ pro nějaké $n \geq 0$, pak jej značíme $\chi_0 \vdash^* + \chi_n$ $[\rho]$.

Definice 2.6 *Nechť je dán NKA* $M = (Q, \Sigma, \delta, q_0, F)$ *a řetězec* w *nad abecedou* Σ*. Pak říkáme, že* w *je* ***řetězcem akceptovaným*** *[6] automatem* M *pokud jej můžeme zapsat ve tvaru* $w = y_1y_2...y_n$*, kde* $y_i \in \Sigma_\epsilon$ *pro* $i = 1...n$ *a existuje taková sekvence stavů* $r_0, r_1, ..., r_n \in Q$*, pro kterou platí následující tři podmínky:*

1. $r_0 = q_0$
2. $r_{i+1} = \delta(r_i, y_{i+1})$ *pro* $i = 0, ..., n$-1
3. $r_n \subseteq F$

Podmínka 1. říká, že automat musí začít ve stavu q_0. Podmínka 2. vyžaduje, aby pro každý stav r_i existoval přechod do stavu r_{i+1} za přečtení znaku y_{i+1}. Poslední podmínka 3. říká, že řetězec w je akceptovatelný automatem M, pokud poslední stav r_n je zároveň koncovým stavem automatu.

Definice 2.7 *Nechť je dán NKA* $M = (Q, \Sigma, \delta, q_0, F)$*, pak* ***jazyk přijímaný*** *konečným automatem* M *je definován:* $L(M) = \{w |\ w \in \Sigma^*, q_0w \vdash^* f,$ kde $f \in F\}$

Přijímaným jazykem NKA je taková množina řetězců, pro které platí, že po zpracování všech znaků se z počátečního stavu přes odpovídající hrany přechodů dostane automat do stavu koncovémho. V případě nedeterministického přechodu, kdy se může pomocí jednoho znaku přejít do více stavů, dojde k rozdělení vyhodnocování. Řetězec je poté akceptován, pokud alespoň jedna cesta dospěje do koncového stavu.

Definice 2.8 *Nechť je dán NKA* $M = (Q, \Sigma, \delta, q_0, F)$*, pak stav* $q \in Q$ *je* ***dostupným stavem****, pokud existuje* $w \in \Sigma^*$*, pro který platí* $q_0w \vdash^* q$*. Jinak je nedostupný.*

Stavy automatu dělíme na dostupné a nedostupné v závislostí na existenci dopředné cesty mezi počátečním stavem a stavem zkoumaným. Pokud taková cesta neexistuje, pak se jedná o nedostupný stav. Nedostupné stavy nemají na funkci automatu žádný vliv, a proto

je možné tyto stavy odstranit. Přítomnost nedostupných stavů je pro minimalizaci výhodou. Metoda odstranění stavu, která je prezentována v kapitole 3 se snaží svým přístupem právě takové stavy vytvářet.

Definice 2.9 *Nechť je dán NKA* $M = (Q, \Sigma, \delta, q_0, F)$*, pak stav* $q \in Q$ *je* ***ukončujícím stavem****, pokud existuje* $w \in \Sigma^*$*, pro který platí* $qw \vdash^* f$*, kde* $f \in F$*. Jinak je neukončující.*

Stavy automatu je také možné členit do dvou kategorii podle existence dopředné cesty mezi zkoumaným stavem a stavem koncovým. Existence takové cesty přisuzuje stavu vlastnost ukončujícího stavu. Neukončující stavy nijak funkčně NKA neovlivňují, proto mohou být smazány. Obdobně jako u nedostupných stavů je i tato vlastnost využívána při aplikování minimalizační metody odstraněním.

Definice 2.10 *Nechť je dán NKA* $M = (Q, \Sigma, \delta, q_0, F)$*, pak* ***pravý jazyk stavu*** q*, kde* $q \in Q$*, konečného automatem* M *je definován:* $\overrightarrow{L}(q) = \{w_r |\ w_r \in \Sigma^*, qw_r \vdash^* f\}$

Pravý jazyk stavu q je množina všech řetězců, pomocí kterých lze dosáhnout z daného stavu do stavu koncového.

Definice 2.11 *Nechť je dán NKA* $M = (Q, \Sigma, \delta, q_0, F)$*, pak* ***levý jazyk stavu*** q*, kde* $q \in Q$*, konečného automatem* M *je definován:* $\overleftarrow{L}(q) = \{w_l |\ w_l \in \Sigma^*, q_0 w_l \vdash^* q\}$

Definice 2.12 *Automaty* M *a* N *jsou* ***ekvivalentní automaty****, pokud* $L(M) = L(N)$*.*

Dva automaty M a N jsou si ekvivalentní tehdy, pokud se rovnají jejich jazyky. Z tohoto tvrzení plyne, že všechny řetězce přijímané automatem M musí být zároveň přijímané automatem N a obráceně.

Definice 2.13 *Nechť je dán NKA* $M = (Q, \Sigma, \delta, q_0, F)$*, pak* ***simulace*** [1] *nad* M *je kvaziuspořádáním* $\preceq \subseteq Q \times Q$*, například* $p \preceq q$*, pro které platí*:

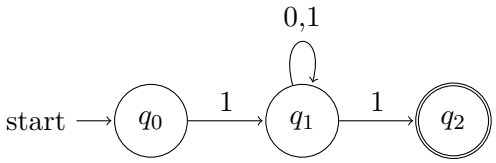
1. $q \in F \implies p \in F$ a zároveň
2. pro každé $pa \rightarrow p'$ existuje $qa \rightarrow q'$ a zárovň pletí $p' \preceq q'$.

Podmínka 1. vyjadřuje, že pokud je stav q koncovým, pak musí být koncový i stav p. Podmínka 2. říká, že pro každý přechod z p do p' pomocí znaku a musí existovat přechod z q do q' pomocí stejného znaku. Nad stavy p' a q' jsou pak dále rekurzivně znovu aplikována pravidla 1 a 2. Pokud jsou veškeré podmínky splněny, pak je stav p simulován stavem q.

Tvrzení 2.13 Existence simulace $r \preceq q$ implikuje $L(r) \subseteq L(q)$. Pokud existuje nad stavu p a q relace simulace, pak je jazyk stavu p podmnožinou jazyka stavu q. Opačné tvrzení ale neplatí. Případ, kdy je vypočítaná čistě jazyková inkluze, nemusí nutně znamenat, že zde existuje i relace simulace. Simulace může tedy některé jazykové inkluze "přehlédnout".

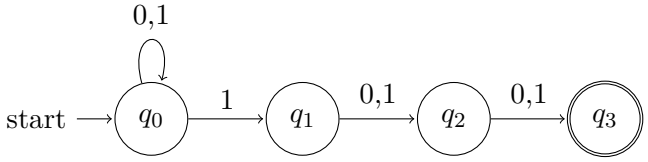
2.2 Příklady

Příklad 2.1 Nechť je dán NKA $M_1 = (Q, \Sigma, \delta, q_0, F)$ tak, že $\Sigma = \{0, 1\}$ a $L(M) = \{w|\ w$ je záporné liché číslo v doplňkovém kódu$\}$ (např. 11, 1001, 101, atd.), pak automat vypadá následovně:



Obrázek 2.1: automat M_1 z příkladu 2.1

Příklad 2.2 [6] Nechť je dán NKA $M_2 = (Q, \Sigma, \delta, q_0, F)$ tak, že $\Sigma = \{0, 1\}$ a $L(M) = \{w|w$ obsahuje na třetí pozici zprava '1'$\}$ (např. 0000100, 010100, ale neobsahuje 00011 atd.), pak automat rozpoznávající tento jazyk vypadá následovně:



Obrázek 2.2: automat M_2 z příkladu 2.2

Kapitola 3

Minimalizace odstraněním

Tako kapitola seznámí čtenáře s novými definicemi a pojmy potřebnými k provedení minimalizace automatu odstraněním stavu. Dále budou nastíněny případy, ve kterých je minimalizace tímto způsobem možná. Uvedená lemmata jsou hypotézami, které je potřeba dále dokázat.

3.1 Definice pro odstranění

Definice 3.1 *Nechť je dán NKA* $M = (Q, \Sigma, \delta, q_0, F)$*, pak* ***jazyk mezi dvěma stavy*** p *a* q*, kde* $p, q \in Q$ *je definován:* $L(p, q) = \{w_b |\ w_b \in \Sigma^*, pw_b \vdash^* q\}$

Stávající jazykové definice jsou obohaceny o jazyk mezi stavy p a q, který představuje množinu řetězců, pomocí kterých se dá s využitím příslušných hran přechodů v automatu přejít ze stavu p do stavu q.

Pro popis případů, ve kterých se dá minimalizace provést odstraněním, je nutné ustanovit pomocný pojem, je jim ryzí jazyk, a to jak levý, pravý, tak i mezi dvěma stavy. Ryzost rozšiřuje vlastnosti jazyka o podmínku vynechání určitého stavu automatu při tvoření jeho řetězců. Například levý jazyk stavu q, který nevyužívá stav r je taková množina řetězců, pomocí kterých je možné přejít ze stavu počátečního do stavu q, a to bez použití stavu r.

Definice 3.2 *Nechť je dán NKA* $M = (Q, \Sigma, \delta, q_0, F)$*, pak* ***ryze levý jazyk*** *stavu* $q \in Q$ *značený* $\overleftarrow{L}(q, \overline{p})$ *je definován jako množina všech řetězců* w *ve tvaru* $w = y_1y_2...y_n$*, kde* $y_i \in \Sigma$ *pro* $i = 1 .. n$ *a existuje taková sekvence stavů,* $r_0r_1r_2...r_n \in Q \backslash \{p\}$*, pro kterou platí následující tři podmínky:*

1. $r_0 = q_0$
2. $r_{i+1} = \delta(r_i, y_{i+1})$ *pro* $i = 0, ..., n\text{-}1$
3. $r_n = q$

Definice 3.3 *Nechť je dán NKA* $M = (Q, \Sigma, \delta, q_0, F)$*, pak* ***ryze pravý jazyk*** *stavu* $q \in Q$ *značený* $\overrightarrow{L}(q, \overline{p})$ *je definován jako množina všech řetězců* w *ve tvaru* $w = y_1y_2...y_n$*, kde* $y_i \in \Sigma$ *pro* $i = 1 .. n$ *a existuje taková sekvence stavů,* $r_0r_1r_2...r_n \in Q \backslash \{p\}$*, pro kterou platí následující tři podmínky:*

1. $r_0 = q$

2. $r_{i+1} = \delta(r_i, y_{i+1})$ *pro* $i = 0, ..., n\text{-}1$

3. $r_n \in F$

Definice 3.4 *Nechť je dán NKA* $M = (Q, \Sigma, \delta, q_0, F)$*, pak* ***ryzí jazyk mezi stavy*** p *a* q *bez použití* s*, kde* $p, q, s \in Q$ *značený* $L(p, q, \overline{s})$ *je definován jako množina všech řetězců* w *ve tvaru* $w = y_1y_2...y_n$*, kde* $y_i \in \Sigma$ *pro* $i = 1 .. n$ *a existuje taková sekvence stavů,* $r_0r_1r_2...r_n \in Q\backslash\{s\}$*, pro kterou platí následující tři podmínky:*

1. $r_0 = p$

2. $r_{i+1} = \delta(r_i, y_{i+1})$ *pro* $i = 0, ..., n\text{-}1$

3. $r_n = q$

Definice 3.5 *Nechť je dán NKA* $M = (Q, \Sigma, \delta, q_0, F)$ *a při výpočtu minimalizace je rozhodnuto, že minimalizace stavů* $p, q \in Q$ *bude provedena smazáním stavu* p*, pak výsledný* ***automat se smazaným stavem*** *je definován jako* $M' = (Q', \Sigma, \delta', q_0, F')$*, kde*:

1. $Q' = Q\backslash\{p\}$,

2. Σ *se nemění,*

3. $\delta'(a, q) = \delta(a, q) \cap Q'$*, pro* $q \in Q'$*,* $a \in \Sigma_\epsilon$,

4. $q_0 \in Q'$ *zůstává počátečním stavem,*

5. $F' = F \cap Q'$.

Automat s odstraněným stavem oproti původnímu automatu postrádá smazaný stav, a také veškeré přechody s tímto stavem spojené. Tento způsob minimalizace nabízí možnost vzniku nedosažitelných nebo neukončujících stavů, které se dají snadno detekovat a odstranit a tím přispět k větší minimalizaci. Vzhledem k tomu, že u automatu M vytvářela přechodová funkce δ množinu pravidel přechodů R, pak i nově vzniklá přechodová funkce δ' utváří množinu pravidel, nyní už ovšem značenou R'.

Lemma 3.1 Pokud platí podmínka $\overleftarrow{L}(p) \subseteq \overleftarrow{L}(q)$ a existuje w takové, že $qw\vdash^*p$, pak platí, že $\overleftarrow{L}(q) \supseteq \overleftarrow{L}(r, \overline{p})\cdot L(q, p)^*$, kde $r \in Q\backslash\{p\}$, pro které platí $rw\vdash^*r \wedge rx\vdash^*q$, $w \in L(r, r, \overline{p})$ a $x \in L(r, q, \overline{p})^* \wedge x \in \{w\}^*$.

Informace o levé jazykové inkluzi a cestě ze stavu q do stavu p podmiňuje existenci cyklu nad stavem r ke konci levého jazyka stavu q. Levý jazyk slabšího stavu p je sjednocen s konkatenací levého jazyka q s cestou mezi stavu, $\overleftarrow{L}(p) = \overleftarrow{L}(p, \overline{q}) \cup \overleftarrow{L}(q, \overline{p})\cdot L(q, p)$. Pro zachování podmínky jazykové inkluze musí tedy levý jazyk stavu q tuto konkatenaci kompenzovat a to jedině iterací jazyka, který je shodný s jazykem $L(q, p)$, tato iterace je uskutečněna smyčkou nad stavem r. Tato smyčka nesmí procházet přes stav p. Dále zbytek cesty mezi stavem r a stavem q musí být provedena pomocí n–té mocniny řetězce w, a to také bez průchodu přes stav p.

Lemma 3.2 Pokud platí podmínka $\overrightarrow{L}(p) \subseteq \overrightarrow{L}(q)$ a existuje w takové, že $pw\vdash^*q$, pak platí, že $\overrightarrow{L}(q) \supseteq L(p, q)^*\cdot\overrightarrow{L}(r, \overline{p})$, kde r $\in Q\backslash\{$p$\}$, pro které platí $rw\vdash^*r \wedge qx\vdash^*r$, $w \in L(r, r, \overline{p})$ a $x \in L(q, r, \overline{p})^* \wedge x \in \{w\}^*$.

Informace o pravé jazykové inkluzi a cestě ze stavu p do stavu q podmiňuje existenci cyklu nad stavem r na začátku pravého jazyka stavu q. Pravý jazyk slabšího stavu p je sjednocen s konkatenací cesty mezi stavy p a q a pravého jazyka q, $\overrightarrow{L}(p) = \overrightarrow{L}(p, \overline{q}) \cup L(p, q) \cdot \overrightarrow{L}(q, \overline{p})$. Pro zachování podmínky jazykové inkluze musí tedy pravý jazyk stavu q tuto konkatenaci kompenzovat, a to jedině iterací jazyka za pomocí smyčky nad stavem r, který je shodný s jazykem $L(p, q)$. Smyčka nad stavem r nesmí procházet přes stav p. Dále cesta mezi stavem q a r musí být provedena pomocí n–té mocniny řetězce w, a to také bez průchodu přez stav p.

Lemma 3.3 Nechť jsou stavy p a q určeny k minimalizaci a existuje takový řetězec w nad abecedou Σ^*, pro který platí, že $pw\vdash^* q$ a zároveň $qw\vdash^* p$. Aby byla možno provést odstranění slabšího stavu, musí existovat takové r a $r' \in Q \backslash \{$slabší stav, např. $p\}$, aby platilo, $rw\vdash^* r \wedge rx\vdash^* q \wedge r'w\vdash r' \wedge qy\vdash^* r'$, $w \in L(r, r, \overline{p}) \wedge w \in L(r', r', \overline{p})$, $x \in L(r, q, \overline{p})^* \wedge x \in \{w\}^*$ a $y \in L(q, r, \overline{p})^* \wedge y \in \{w\}^*$. V opačné případě není odstranění možné z důvodu změny jazyka automatu. Musí se použít pouze sloučení stavů.

3.2 Případy odstranění

Minimalizace automatu odstraněním určeného stavu je možná pouze v uvedených případech, které zaručují, že po odstranění stavu p automatu M bude platit $L(M) = L(M')$. Odstranění musí být prováděno s ohledem na lemmu 3.3.

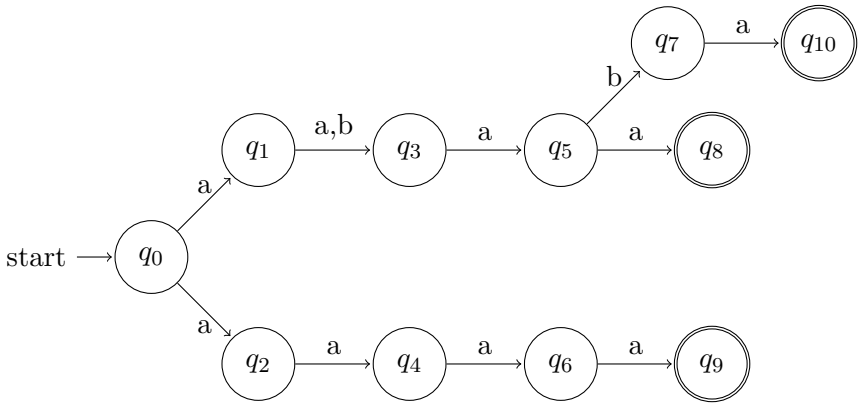
Pro případy ekvivalence levých, nebo pravých jazyků minimalizovaných stavů existuje spolehlivá metoda sloučení, která zohledňuje případné přechody mezi stavy. Metodu odstranění pouze při znalosti jednostranné ekvivalence nelze použít. Automat by tímto přístupem mohl přijít o velkou část svého jazyka. Tato skutečnost vytváří s odstraňování velmi strohou metodu plnou podmínek, i přesto by mohla vznést zefektivnění do minimalizace. Odstranění vytvoří nedostupné a neukončující stavy, tudíž pokud se mezi nimi bude nacházet stav nedetekovaný simulací, pak jej odstranění odstaví z cesty mezi počátečním a ukončujícím stavem, čímž nedostatek simulace napraví..

Odstranění 3.1 $\overleftarrow{L}(p) \subseteq \overleftarrow{L}(q) \wedge \overrightarrow{L}(p) \subseteq \overrightarrow{L}(q)$

Nechť je dán NKA $M = (Q, \Sigma, \delta, q_0, F)$, pak v případě oboustranné inkluze jazyka p pod q je minimalizace odstraněním těměř vždy možná a odstraní se stav p. Vyjímku tvoří případ, který je specifikován lemmou 3.3. Tedy existuje stejná cesta mezi stavy bez její simulace v ryzém levém a pravém jazyku silnějšího stavu. V takovém případě není provedeno odstanění, ale spojení.

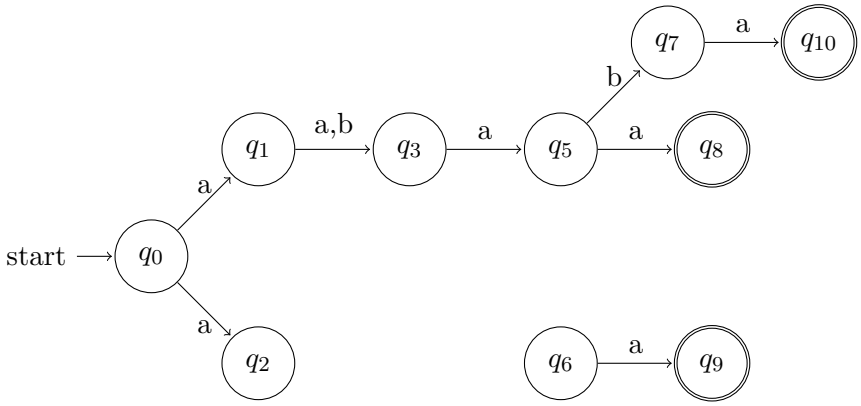
Povšimněte si, že u minimalizace existuje pouze jeden speciální případ, u kterého nelze stav odstranit.

Příklad 3.1 Nechť je dán NKA $M_3 = (Q, \Sigma, \delta, q_0, F)$ viz. obrázek 3.1



Obrázek 3.1: automat M_3

U automatu M_3 již bylo vyhodnoceno, že na stavy q_3 a q_4 se dá aplikovat odstranění. Je zrušen stav q_4. Výsledný automat s odstraněním stavem $M_3' = (Q', \Sigma, \delta', q_0, F')$ je na obrázku 3.2



Obrázek 3.2: automat s odstraněním M_3'

Z automatu M_3' je zřejmé, že odstranění stavu nabízí potenciální vznik nedosažitelných a neukončujících stavů, které je již jednoduché odstranit. Tato skutečnost by mohla pomoci urychlit minimalizaci tak, že bychom současně mazali i nově vzniklé nedosažitelné stavy.

Odstranění stavu při minimalizaci tedy nelze prakticky použít vždy, protože by mohlo dojít ke změně jazyka automatu.

Kapitola 4

Maximalizace minimalizace

Pro určení minimalizace dvou stavů automatu se využívají výpočty jazykových inkluzí, které mají exponenciální složitost. Pro urychlení je tato metoda nahrazena simulací, která inkluzi pouze aproximuje. To způsobuje, že některé inkluze mohou zůstat nedetekované. Současné minimalizační postupy se v informacích získaných simulací soustředí pouze na oboustranné inkluze a ekvivalence. Mnoho vypočtených inkluzí zůstává nevyužitých, což snižuje efektivitu provedeného výpočtu a úrovně minimalizace.

Tato poslední kapitola prezentuje speciální případy, ve kterých je možné provést minimalizaci dvou stavů přemapováním, nebo odstraněním, a to pouze se znalostí jednostranné jazykové inkluze. Tento postup umožňuje co možná nejefektivněji využít již vypočítané relace simulace a dosáhnout minimálnější formy automatu. Uvedená lemmata představují hypotézy, které je potřeba dále dokázat.

4.1 Pseudojazyková inkluze

Definice 4.1 *Nechť je dán NKA* $M = (Q, \Sigma, \delta, q_0, F)$ *a při výpočtu minimalizace je rozhodnuto o pravém přemapování stavu* $p \in Q$ *na množinu stavů* $B \subseteq Q'$. *Výsledný* ***automat s pravě přemapovaným stavem*** *je definován jako* $M' = (Q', \Sigma, \delta', q_0, F')$, kde

1. $Q' = Q \backslash \{p\}$,
2. Σ *se nemění,*
3. $\delta'(a, q') = \delta(a, q) \cap (Q' \cup B \Leftrightarrow q = p)$, pro $q' \in Q'$, $B \subseteq Q'$, $p, q \in Q$, $a \in \Sigma_\epsilon$
4. $q_0 \in Q'$ *zůstává počátečním stavem,*
5. $F' = F \cap Q'$.

Automat s pravě přemapovaným stavem vzniká v případech, kdy je odstraněn kupříkladu stav p a veškeré hrany přechodu vedoucí do p, kromě vlastní smyčky nad p jsou nasměrovány do stavů určených množinou B. Dochází tedy k substituci v pravé straně všech pravidel obsahujících p za každé $r \in B$.

Definice 4.1 *Nechť je dán NKA* $M = (Q, \Sigma, \delta, q_0, F)$ *a při výpočtu minimalizace je rozhodnuto o levém přemapování stavu* $p \in Q$ *na množinu stavů* $B \subseteq Q'$. *Výsledný* ***automat s levě přemapovaným stavem*** *je definován jako* $M' = (Q', \Sigma, \delta', q_0, F')$, kde

1. $Q' = Q \backslash \{p\}$,
2. Σ *se nemění,*
3. $\delta'(a, q') = \delta(a, q) \cap Q'$, pro každé $q' \in B \Leftrightarrow q = p$, jinak $q' \in Q'$, $B \subseteq Q'$, $p, q \in Q$, $a \in \Sigma_\epsilon$
4. $q_0 \in Q'$ *zůstává počátečním stavem,*
5. $F' = F \cap Q'$.

Automat s levě přemapovaným stavem vzniká v případech, kdy je odstraněn kupříkladu stav p a veškeré počátky hran přechodů vedoucích z p kromě vlastní smyčky nad p jsou nasměrovány do stavů určených množinou B. Dochází tedy k substituci v levé straně všech pravidel obsahujících p za každé $r \in B$.

Mimimalizace 4.1 $\overrightarrow{L}(p) \subseteq \overrightarrow{L}(q)$

Mějme automat $M = (Q, \Sigma, \delta, q_0, F)$, nechť p a $q \in Q$, je dvojice stavů označených k potenciální minimalizaci. Minimalizace stavu p může být provedena pouze při splnění následujících podmínek.

1. Pro každé w, kde $qw \vdash^* p$ musí existovat $qw \vdash^* r$, r $\in B = \{r \mid \overleftarrow{L}(r) \subseteq \overleftarrow{L}(p), r \in Q \backslash \{p\}\}$. Pokud existuje cesta mezi p a r, je kontrola w obohacena o podminku, že existuje pouze $qx \vdash^* r$ bez použití p, pro každé $x \in L(p, r)$, jinak řečeno $L(p, r) = L(q, r, \overline{p})$.
2. Po odstranění cest mezi stavem q a p musí již platit $\overleftarrow{L}(p) \subseteq \overleftarrow{L}(q)$.

Při minimalizaci je nejprve nalezena množina všech stavů pro které platí $\overleftarrow{L}(r) \subseteq \overleftarrow{L}(p)$, a pro které zároveň existuje cesta mezi stavem q a takovými stavy. Kontrolou jazykové inkluze r a p je zajištěno, že žádná jiná část automatu tyto stavy na cestě mezi q a r nevyužívá, a tudíž nebude vadit tuto cestu obohatit. Pro každou cestu z q do p musí existovat stav r z množiny B, který ji simuluje. Dále musí být zkontrolováno, že pokud existuje cesta mezi stavem p a r, je tato cesta totožná s jedinou cestou z q do r. Pokud by z q do r vedly i jiné cesty, byl by výsledný jazyk automatu změněn. Před minimalizací jsou odstraněny cesty mezi q a p. Dojde k nové kontrole levostranné jazykové inkluze, která musí potvrdit, že jazyk stavu q je nadmnožinou p. Pokud není tato podmínka splněna, nelze minimalizaci provést. V případě, že všechny stavy r jsou stavem q, pak může být stav p odstraněn spolu s jeho pravidly přechodů. Vznikne automat se smazaným stavem. V opačném případě je stav p odstraněn a všechny pravé strany pravidel obsahujících p budou zaměněny za každé r z množiny B. Vznikne automat s pravě přemapovaným stavem.

Minimalizace 4.2 $\overleftarrow{L}(p) \subseteq \overleftarrow{L}(q)$

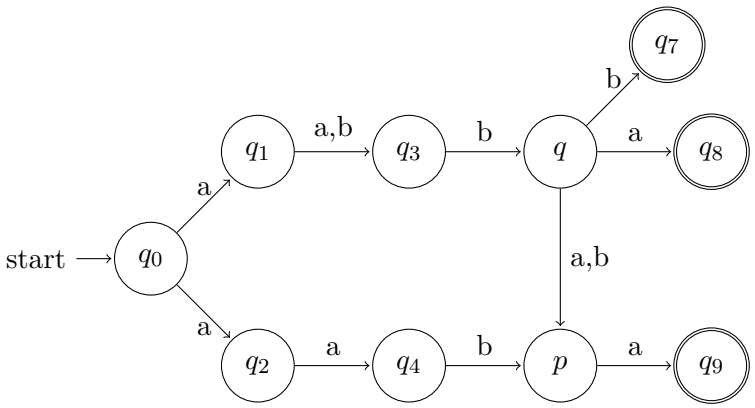
Mějme automat $M = (Q, \Sigma, \delta, q_0, F)$, nechť p a $q \in Q$, je dvojice stavů označených na základě jednostranné jazykové inkluze k potenciální minimalizaci. Minimalizace stavu p může být provedena pouze při splnění následujících podmínek.

1. Pro každé w, kde $pw \vdash^* q$ musí existovat $rw \vdash^* q$, r $\in B = \{r \mid \overrightarrow{L}(r) \subseteq \overrightarrow{L}(p), r \in Q \backslash \{p\}\}$. Pokud existuje cesta mezi p a r, je kontrola w obohacena o podmínku, že existuje pouze $rx \vdash^* q$ bez použití p, pro každé $x \in L(p, r)$, jinak řečeno $L(p, r) = L(r, q, \overline{p})$.
2. Po odstranění cest mezi stavem p a q musí již platit $\overrightarrow{L}(p) \subseteq \overrightarrow{L}(q)$.

Při minimalizaci je nejprve nalezena množina všech stavů r pro které platí $\overrightarrow{L}(r) \subseteq \overrightarrow{L}(p)$ a zároveň existuje cesta mezi takovými stavy a stavem q. Kontrolou jazykové inkluze r a p je zajištěno, že žádná jiná část automatu tyto stavy na cestě mezi r a q nevyužívá, a tudíž nebude vadit tuto cestu obohatit. Pro každou cestu z p do q musí existovat stav r z množiny B, který ji simuluje. Dále musí být zkontrolováno, že pokud existuje cesta mezi stavem p a r je tato cesta totožná s jedinou cestou z r do q. Pokud by z r do q vedly i jiné cesty, byl by výsledný jazyk automatu změněn. Před minimalizací jsou odstraněny cesty mezi p a q. Dojde k nové kontrole pravostranné jazykové inkluze, která musí potvrdit, že jazyk stavu q je nadmnožinou p. Pokud není tato podmínka splněna, nelze minimalizaci provést. V případě, že všechny stavy r jsou stavem q, pak může být stav p odstraněn spolu s jeho pravidly přechodů. Vznikne automat se smazaným stavem. V opačném případě je stav p odstraněn a všechny levé strany pravidel obsahujících p budou zaměněny za každé r z množiny B. Vznikne automat s levě přemapovaným stavem 4.1.

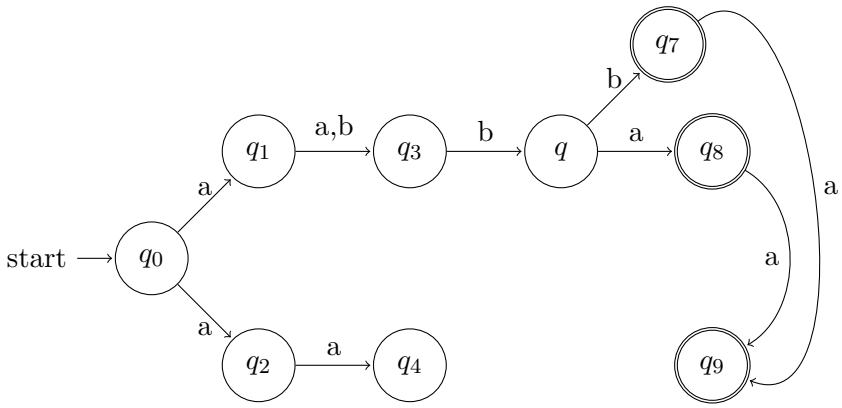
Množina stavů B, která byla využívana k minimalizaci pomocí přemapování, by měla typicky dosahovat minimálních rozměrů. V některých případech ale může obsahovat spoustu stavů, které simulací jen zlomek cesty mezi minimalizovanými stavy. Při implementaci bude nejvhodnější množinu B minimalizovat.

Příklad 4.1 Nechť je dán NKA $M_4 = (Q, \Sigma, \delta, q_0, F)$. Víme, že $\overrightarrow{L}(p) \subseteq \overrightarrow{L}(q)$, automat minimalizujeme za použití metody 4.1.



Obrázek 4.1: Automat M_4

Simulace určila, že levé jazyky stavů q_7 a q_8 jsou podmnožinou levého jazyka stavu p. Nyní zkontrolujeme, zda pro každou cestu mezi q a p existuje i taková cesta mezi simulací zvolenými stavy. Postavíme množinu $B = \{q_7, q_8\}$. Zkontrolujeme, zda neexistuje cesta mezi p a stavy r. Pokud není, přistoupíme ke kontrole levé jazykové inkluze $\overleftarrow{L}(p) \subseteq \overleftarrow{L}(q)$ po odstranění cesty mezi q a p. $\overleftarrow{L}(p) = \{aab\} \subseteq \overleftarrow{L}(q) = \{aab, abb\}$. Pokud jsou všechny podmínky splněny může být provedeno levé přemapování stavu p na množinu B. Vzniká automat s pravě přemapovaným stavem.



Obrázek 4.2: Automat M_4' s levě přemapovaným stavem.

Z minimalizovaného automatu M_4' lze vidět, že se jedná o modifikaci odstraňování stavů. I za použití metody přemapování je možno dosáhnout neukončujících nebo nedosažitelných stavů.

4.2 Ukrytá ekvivalence jazyků

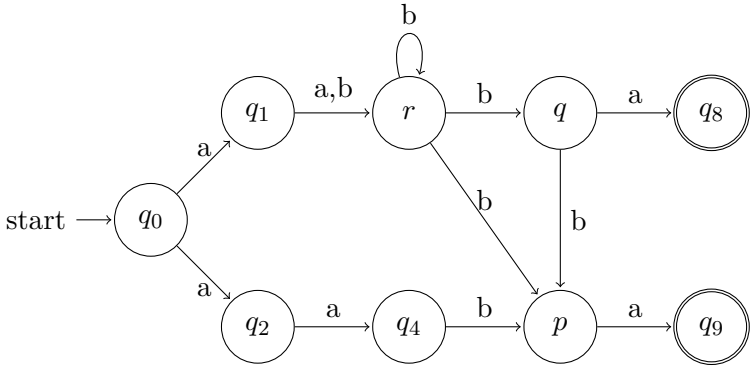
Jak již bylo na začátku kapitoly uvedeno, existují případy jazykových inkluzí, které simulace neidentifikuje. Proto mohou být některé jazykové ekvivalence přehlédnuty a poskytnuta pouze informace o jednostranné jazykové inkluzi. S pomocí získané neúplné informace a nalezení specifických hran přechodů mezi stavy je možno potvrdit přítomnost ekvivalence jazyků těchto stavů.

Lemma 4.1 $\overleftarrow{L}(p) \supseteq \overleftarrow{L}(q)$ a zároveň pro každé w, pro které platí $qw\vdash^* p$ existuje $r \in Q\backslash\{p\}$ takové, že $rw\vdash^* r \wedge rx\vdash^* q \wedge rx\vdash^* p$, kde $w \in L(r, r, \overline{p})$ a $x \in L(r, q, \overline{p})^* \wedge x \in \{w\}^*$, pak platí $\overleftarrow{L}(p) \subseteq \overleftarrow{L}(q)$.

Z pouhé znalosti jednostranné jazykové inkluze, například $\overleftarrow{L}(p) \supseteq \overleftarrow{L}(q)$, a existence cesty mezi q a p, je možné při splnění podmínek implikovat jejich ekvivalenci. Musí ale existovat takové stavy na konci levého jazyka stavu q, které budou obsahovat smyčku simulující cesty mezi q a p. Vzdálenost stavu p od této smyčky musí být stejná jako vzdálenost stavu q.

Lemma 4.2 $\overrightarrow{L}(p) \supseteq \overrightarrow{L}(q)$ a zároveň pro každé w, pro které platí $pw\vdash^* q$ existuje $r \in Q\backslash\{p\}$ takové, že $rw\vdash^* r \wedge qx\vdash^* r \wedge px\vdash^* r$, kde $w \in L(r, r, \overline{p})$ a $x \in L(q, r, \overline{p})^* \wedge x \in \{w\}^*$, pak platí $\overrightarrow{L}(p) \subseteq \overrightarrow{L}(q)$.

Příklad 4.2 Nechť je dán NKA $M_5 = (Q, \Sigma, \delta, q_0, F)$. Protože z výsledků simulace pouze víme, že $\overleftarrow{L}(q) \subseteq \overleftarrow{L}(p)$, snažíme se na základě lemma 4.1 najít ekvivalenci nad levými jazyky stavů q a p.



Obrázek 4.3: Automat M_5

Na základě podmínek z lemmy 4.1 hledáme cestu mezi stavy q a p. Taková cesta existuje, a uskutečňuje se pomocí řetězce $w = b$. Nyní musíme najít na levé straně q takový stav r, nad kterým je smyčka simulující cestu mezi q a p. Tím stavem je r. Je potřeba ověřit, zda cesta z r do q se dá provést na základě n–té mocniny řetězce a. Pokud ano, přistoupíme ke kontrole vzdálenosti r od q pomocí řetězce a, která musí být stejná jako vzdálenost r od p za pomocí stejného řetězce. Byly splněny veškeré podmínky udávané lemmou 4.1, byla nalezena ekvivalence jazyků a platí tedy také $\overleftarrow{L}(p) \subseteq \overleftarrow{L}(q)$. Nyní je možné provést minimalizaci stavů q a p pomocí sloučení. Metodu odstraňování nelze použít.

Kapitola 5

Závěr

Tato práce přinesla nový způsob minimalizace automatu, a to pomocí odstranění jeho stavu. Zdůrazňuje ty případy, kdy je nutno ověřit složitější podmínky umožňující tento způsob minimalizace. Poslední kapitola poskytuje podmínky, díky nimž lze stavy minimalizovat, a to při pouhé znalosti jednostranné jazykové inkluze dvou stavů a cesty mezi nimi. Tento klíčový poznatek umožňuje efektivněji využívat data získaná ze simulace.

Uvedené metody a tvrzení bych chtěl v další práci detailně prozkoumat, dokázat jejich platnost a dále je implementovat. Porovnat jejich časovou a paměťovou náročnost s dosavadními postupy při minimalizaci rozsáhlých nedeterministických automatů.

Literatura

[1] Holík, L., Vojnar, T., Abdulla, P., Chen, Y.-F. a Mayr, R. When Simulation Meets Antichains (On Checking Language Inclusion of Nondeterministic Finite (Tree) Automata). In: *Tools and Algorithms for the Construction and Analysis of Systems* [print]. LNCS 6015. Springer Verlag: Springer Verlag, March 2010, kap. 34731, s. 158–174.

[2] Ilie, L., Navarro, G. a Yu, S. On NFA Reductions. *Theory Is Forever Lecture Notes in Computer Science*. Springer, Berlin, Heidelberg. 2004, s. 112–124.

[3] Meduna, A. a Lukáš, R. *Formální jazyky a překladače: Abecedy, řetězce a znaky.* Božetěchova 1/2, 612 00 Brno-Královo Pole: FIT VUT v Brně, 2017.

[4] Meduna, A. a Lukáš, R. *Formální jazyky a překladače: Modely pro regulární jazyky.* Božetěchova 1/2, 612 00 Brno-Královo Pole: FIT VUT v Brně, 2017.

[5] Meduna, A. a Lukáš, R. *Formální jazyky a překladače: Speciální typy konečných automatů.* Božetěchova 1/2, 612 00 Brno-Královo Pole: FIT VUT v Brně, 2017.

[6] Sipser, M. Nondeterminism. In: *Introduction to the Theary of Computation*. 2. vyd. Boston: Thomson Course Technology, 2006. ISBN 0-534-95097-3.