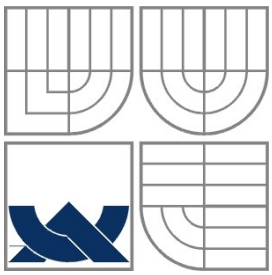
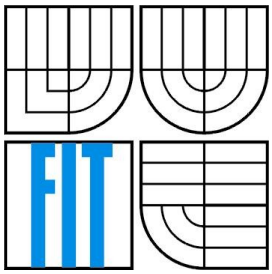


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SYSTÉMY PŘEVODNÍKŮ

TRANSDUCER SYSTEMS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JIŘÍ SKÁCEL

VEDOUCÍ PRÁCE
SUPERVISOR

Prof. RNDr. ALEXANDER MEDUNA, CSc.

BRNO 2016

Zadání diplomové práce/18153/2015/xskace10

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2015/2016

Zadání diplomové práce

Řešitel: **Skácel Jiří, Bc.**
Obor: Informační systémy
Téma: **Systémy převodníků**
Transducer Systems
Kategorie: Teoretická informatika

Pokyny:

1. Dle pokynů školitele se seznamte s vybranými systémy automatů a gramatik.
2. Zaveďte systémy převodníků dle pokynů školitele.
3. Studujte vyjadřující sílu systémů z bodu 2.
4. Studujte aplikaci těchto systémů (např. syntaktická analýza). Navrhněte a implementujte vhodnou aplikaci na nich založenou. Testujte ji.
5. Zhodnoťte dosažené výsledky a diskutujte další možný vývoj projektu.

Literatura:

- Meduna, A.: Automata and Languages, Springer, London, 200

Při obhajobě semestrální části projektu je požadováno:

- Body 1 a 2.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Meduna Alexander, prof. RNDr., CSc.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 25. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Tato práce zavádí systémy zásobníkových převodníků. Základní myšlenka vychází z kooperujících distribuovaných gramatických systémů, které umožňují práci více gramatik na jednom řetězci. Převodníky spolupracují předáváním výsledků svých překladů jako vstupu pro další komponentu. Dále tato práce zkoumá jejich popisnou sílu a ekvivalenci systémů s různými počty komponent. Hlavním závěrem je pak porovnání popisné síly s Turingovými stroji a to s ohledem na jimi definovaný překlad i přijímané jazyky.

Abstract

This document defines systems of pushdown transducers. The idea of cooperating distributed grammar systems for components working on one word is adjusted for use of transducers instead of grammars. The transducers cooperate by passing output of one to input of another component. It discusses their descriptive power and equivalency between systems with arbitrary numbers of components. The main conclusion is then comparison of their descriptive power with Turing machines with regard to their translation and accepted languages.

Klíčová slova

Převodníky, zásobníkové převodníky, CD GS, gramatické systémy, Turingovy stroje, překlady

Keywords

Transducers, pushdown transducers, CD GS, grammar systems, Turing machines, translations

Citace

SKÁCEL, Jiří. *Systémy převodníků.* Brno, 2016. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Meduna Alexander.

Systémy převodníků

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana profesora Alexandera Meduny. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jiří Skácel
24. května 2016

Poděkování

Zde bych rád poděkoval vedoucímu mé diplomové práce, profesoru Alexanderu Medunovi, za jeho vedení a shovívavost.

Obsah

1 Úvod

Základem informačních technologií jsou matematické modely pracující s řetězci symbolů, z nichž definují jazyky – množiny řetězců, které vyhovují danému modelu. Tyto jazyky pak využíváme pro řešení matematických otázek, např. máme-li model jehož jazyk popisuje všechna prvočísla, patří dané číslo zapsané vhodným způsobem do tohoto jazyka? Různé modely se pak liší svou schopností popsat různě složité jazyky a také svou schopností rozhodovat otázky ohledně svých jazyků. Tyto síly jdou vždy proti sobě, takže u slabých modelů jsme schopni řešit jen malé množství problémů, ale umíme odpovídat na více otázek o těchto problémech.

Tato práce definuje nový model založený na starších *převodnících*, jež jsou obohaceny využitím principu relativně nových *gramatických systémů*. *Zásobníkové převodníky* jsou definovány v [1] z roku 1972, ale stroje s podobnou myšlenkou lze najít už o pár let dříve. Využívají zásobníkový automat pro rozhodování o přijetí vstupního řetězce a zároveň během práce generují vlastní výstup. Takto mohou popisovat nejen vstupní (přijímající) jazyk, ale i překlad – relaci mezi vstupními a výstupními řetězci. Překladů a převodníků bylo využíváno jako základu kompilátorů programovacích jazyků v sedmdesátých letech. Dnes se však při konstrukci překladačů (např. [2]) využívají pouze automaty pro definici vstupního jazyka jednotlivých fází překladu a výstup je určen akcemi mimo formální model. Převodníky tedy ztratily svou přirozenou doménu, ale jejich dnešní varianty – *vážené převodníky*, např. [3] – jsou využívány pro zpracování obrazu nebo řeči s nejistými nebo pravděpodobnostními vstupními daty.

Druhým, modernějším základním kamenem pro tento nový model jsou *kooperující distribuované gramatické systémy* (CD GS). Tyto systémy podle [4] začaly být intenzivněji studovány v devadesátých letech a přinesly myšlenky přebírané dnes i do jiných modelů. Hlavní novinkou je rozdělení práce na jednom řetězci mezi více komponent, které spolupracují a předávají si generovaný řetězec. Právě různá omezení, kdy je možné předat řízení jiné komponentě, zvyšují popisnou sílu.

Zde definované *systémy převodníků* využívají právě principu spolupráce CD gramatických systémů s využitím zásobníkových převodníků jako komponent. Celý systém definuje překlad takto: vstupní řetězec je předán některé komponentě, která jej přeloží a svůj výsledek pošle na vstup jiné komponenty a tak dále, dokud poslední komponenta nepředá výsledek systému, který jej prohlásí za výstup překladu daného vstupu.

Tato práce je organizována následovně. V druhé kapitole předkládám definici zásobníkových automatů převzatou z [1] spolu s definicemi dalších pojmů z [5]. Následuje povinný příklad překladu prefixové notace výrazů na postfixovou notaci.

Ve třetí kapitole definuji systémy zásobníkových převodníků podle vzoru předchozí kapitoly a dva příklady pro ilustraci jejich síly a práce. Důležitým bodem této kapitoly je pak věta o ekvivalenci síly systémů s různými počty komponent spolu s důkazem.

Čtvrtá kapitola je zaměřena na porovnání síly s *Turingovými stroji*. Proto nejprve definuji Turingovy stroje. Poté předložím dvě věty spolu s důkazy: větu o ekvivalenci síly s ohledem na přijímané jazyky a větu o větší síle Turingových strojů s ohledem na definované překlady s příkladem Turingova stroje.

Poslední pátá kapitola představuje přiloženou aplikaci, která implementuje systémy převodníků, Turingovy stroje a uvedené algoritmy pro důkazy ekvivalencí.

1.1 Základní pojmy a definice

V této práci předpokládám základní znalosti z oblasti formálních jazyků, zejména zásobníkových automatů a Turingových strojů. Vhodným úvodem do této problematiky je například [5] nebo méně formální [6]. Využívám však trochu jinou notaci, získanou při studiu [7]. Nezbytnými znalostmi jsou operace nad množinami, ostatní pojmy se pokusím alespoň shrnout v této sekci spolu s použitými konvencemi značení.

Dále platí, že všechna čísla, značená typicky písmeny i až m, jsou *přirozená*, tedy nezáporná.

Abeceda je konečná množina symbolů. Na rozdíl od častých definic, v této práci povoluji a využívám i prázdných abeced. Značím velkými řeckými písmeny, typicky Σ, Γ, Δ a K (kappa).

Symbolem nebo znakem rozumím prvek abecedy. Značím malým písmenem latinky, typicky od znaku a dále.

Řetězec nebo *slovo* nad abecedou Σ je konečná posloupnost symbolů z této abecedy. Například pro $\Sigma=\{a,b\}$, existuje řetězec $w=aaabba$. Značím malým písmenem latinky, typicky u a dále, nebo malým písmenem řečtiny, typicky α a dále. Pro řetězce různý abeced využívám různá písmena: pro Σ je to typicky $x, y, \ldots$, pro Γ $\alpha, \beta, \ldots$, pro Δ $u, v, \ldots$.

Délku řetězce x značím $|x|$ a je to počet symbolů, ze kterých se x skládá.

Existuje *prázdný řetězec* značený ε, pro který platí $|\varepsilon|=0$.

Konkatenací dvou řetězců x a y vznikne řetězec xy obsahující symboly x následované symboly y.

Mocnina řetězce x^n je mocnina konkatenace. Platí $x^0=\varepsilon$ a $x^n=x x^{n-1}$ pro libovolný řetězec x.

Prefix řetězce x je řetězec y, pro který jsme schopni nalézt řetězec z, aby platilo $x=yz$.

Sufix řetězce x je naopak řetězec z, pro který jsme schopni nalézt řetězec y, aby platilo $x=yz$. Prázdný řetězec je prefixem i sufixem každého řetězce.

Množinu všech řetězců nad abecedou Σ značíme jako Σ^*. Pro prázdnou abecedu platí $\emptyset^*=\{\varepsilon\}$.

Množinu všech neprázdných řetězců nad abecedou Σ značíme jako Σ^+.

Množinu všech nekonečných řetězců nad abecedou Σ značíme jako Σ^ω. Na takové řetězce narazíme pouze v případě sufixů pásky Turingova stroje.

Jazyk je libovolná podmnožina množiny Σ^*.

Překlad je libovolná relace mezi dvěma jazyky označovanými jako vstupní a výstupní. Tedy $T\subseteq L_I\times L_O$, kde L_I je jazyk nad vstupní abecedou, typicky Σ, a L_O je jazyk nad výstupní abecedou označovanou Δ.

Operaci konkatenace a umocňování lze jednoduše rozšířit na jazyky:

- konkatenace jazyků L_1 a L_2, $L_1L_2=\{xy\,|\,x\in L_1, y\in L_2\}$,
- mocnina jazyka L, $L^0=\{\varepsilon\}$ a $L^n=L\,L^{n-1}$,
- iterace jazyka L, $L^*=L^0\cup L^1\cup\cdots$ a
- pozitivní iterace jazyka L, $L^+=L^1\cup L^2\cup\cdots$.

Dále je vhodné znát následující konvence.

Symboly zásobníkové abecedy jsou značeny velkým písmenem Z. Řetězce pak písmeny řecké abecedy α a dále.

Stavy jsou značeny malými písmeny q a p.

2 Zásobníkové převodníky

Zásobníkový převodník, tak jak je definován v [1], případně [5], vychází ze *zásobníkového automatu* nebo alternativně z *konečného převodníku*. Od zásobníkového automatu se liší výstupní páskou, na kterou během zpracovávání vstupu vypisuje postupně svůj výstup. Od konečného převodníku se liší podobně jako zásobníkový automat od konečného automatu, tedy pomocným zásobníkem, na který může zapisovat symboly a z jehož vrcholu lze číst. Zásobníkové převodníky se tedy skládají ze:

- *vstupní pásky* a *čtecí hlavy,*
- *výstupní pásky* a *zápisové hlavy,*
- *zásobníkové pásky* a *zásobníkové hlavy* a
- *konečného řízení*.

Vstupní páska obsahuje na počátku zpracování zadaný vstupní řetězec symbolů *vstupní abecedy* a čtecí hlava se nachází nad prvním symbolem tohoto řetězce. V průběhu výpočtu se hlava pohybuje pouze jedním směrem zleva doprava. Zápisová hlava dovoluje zápis symbolů *výstupní abecedy* v průběhu zpracování na konec výstupní pásky. Ta je na počátku prázdná. Zásobníková páska slouží jako pracovní prostor pro symboly *zásobníkové abecedy* s pevně daným počátečním symbolem. V průběhu výpočtu je zásobníková hlava vždy nad posledním symbolem této pásky a v každém kroku jej přečte a nahradí libovolným řetězcem symbolů zásobníkové abecedy. Tyto tři hlavy jsou řízeny na základě konečného počtu stavů a přechodové relace, která v rámci jednoho *kroku* podle aktuálního stavu a symbolu pod čtecí a zásobníkovou hlavou určuje následný stav a řetězce pro zapsání na zásobníkovou a výstupní pásku. Zpracování vstupního řetězce v konečném počtu kroků na výstupní řetězec pak nazýváme *překladem*.

2.1 Základní definice

Definice 2.1

Zásobníkový převodník P je osmice $(Q, \Sigma, \Gamma, \Delta, \delta, q_0, Z_0, F)$, kde

Q je konečná neprázdná *množina stavů*,
Σ je *vstupní abeceda*,
Γ je *zásobníková abeceda*,
Δ je *výstupní abeceda*,
δ je *přechodová relace* z $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma$ do $Q \times \Gamma^* \times \Delta^*$,
$q_0 \in Q$ je *počáteční stav*,
$Z_0 \in \Gamma$ je *počáteční symbol na zásobníku* a
$F \subseteq Q$ je *množina koncových stavů*.

Přechodová relace je množina šestic, kde první je aktuální stav, druhý čtený symbol, třetí vrchol zásobníku, čtvrtý následný stav, pátý je řetězec, který nahradí vrchol zásobníku a poslední šestý je řetězec zapsaný na výstup při využití tohoto přechodu.

Definice 2.2

Mějme zásobníkový automat $P = (Q, \Sigma, \Gamma, \Delta, \delta, q_{0,} Z_{0,} F)$. Pak *vstupní zásobníkový automat* M je definován jako

$$M = (Q, \Sigma, \Gamma, \delta', q_{0,} Z_{0,} F),$$

kde $\delta' = \{(q, a, Z, p, \gamma) \mid \exists u \in \Delta^* : (q, a, Z, p, \gamma, u) \in \delta\}$.

Definice 2.3

Konfigurace převodníku P je čtveřice (q, x, α, u), kde

$q \in Q$ je *aktuální stav*,
$x \in \Sigma^*$ je dosud nepřečtená část vstupního řetězce,
$\alpha \in \Gamma^*$ je řetězec na *zásobníku*, přičemž jako vrchol chápeme jeho nejlevější symbol a
$u \in \Delta^*$ je dosud vyprodukovaná část výstupního řetězce.

Takto definovaná konfigurace jednoznačně popisuje stav převodníku. Ze vstupní pásky postačuje zaznamenat pouze nezpracovaný vstup, protože dřívější symboly už nemohou ovlivnit další kroky. Zásobníková a výstupní páska jsou pak zaznamenány celé.

Definice 2.4

Mějme dvě konfigurace $C_1 = (q, x, \alpha, u)$ a $C_2 = (p, y, \beta, v)$ zásobníkového převodníku P, kde $q, p \in Q$, $x, y \in \Sigma^*$, $\alpha, \beta \in \Gamma^*$, $u, v \in \Delta^*$.

Převodník P provádí *krok* z konfigurace C_1 do konfigurace C_2 právě tehdy, když platí jedna z následujících podmínek:

- jestliže $x = ay$, $\alpha = Z\gamma$, $\beta = \beta'\gamma$, $v = uv'$ a (q, a, Z, p, β', v') je obsaženo v δ nebo
- jestliže $x = y$, $\alpha = Z\gamma$, $\beta = \beta'\gamma$, $v = uv'$ a $(q, \varepsilon, Z, p, \beta', v')$ je obsaženo v δ.

Zapisujeme

$$(q, x, \alpha, u) \vdash (p, y, \beta, v)$$

nebo zjednodušeně

- $(q, ay, Z\gamma, u) \vdash (p, y, \beta'\gamma, uv')$, jestliže $(q, a, Z, p, \beta, v') \in \delta$ a
- $(q, x, Z\gamma, u) \vdash (p, x, \beta'\gamma, uv')$, jestliže $(q, \varepsilon, Z, p, \beta', v') \in \delta$.

První možnost je krok při přečtení vstupního symbolu a posunutí čtecí hlavy, druhá možnost vstupní pásku nečte a pouze pracuje se stavem řízení, zásobníkem a výstupní páskou.

Pro zdůraznění náležitosti relace $\vdash$ převodníku P můžeme psát $\vdash_P$.

Sekvenci kroků chápeme jako skládání kroků ve smyslu skládání relací. Tedy sekvence n kroků je relace $\vdash$ na n-tou, zapisujeme

$$(q, x, \alpha, u) \vdash^n (p, y, \beta, v).$$

Důležité jsou pak ještě *tranzitivní uzávěr* a *reflexivní a tranzitivní uzávěr* této relace značené

$$(q, x, \alpha, u) \vdash^+ (p, y, \beta, v), \text{ resp. } (q, x, \alpha, u) \vdash^* (p, y, \beta, v).$$

Definice 2.5

Výstup zásobníkového převodníku $P=(Q,\Sigma,\Gamma,\Delta,\delta,q_{0,}Z_{0,}F)$ pro *vstup* $x\in\Sigma^*$ je $u\in\Delta^*$ právě tehdy, když $(q_{0,}x,Z_{0,}\varepsilon)\vdash^*(q,\varepsilon,\alpha,u)$ pro nějaké $q\in F$ a $\alpha\in\Gamma^*$.

Překlad definovaný zásobníkovým převodníkem P, značený $T(P)$ je definován jako

$$T(P)=\{(x,u)\mid x\in\Sigma^*,u\in\Delta^*,\exists q\in F,\alpha\in\Gamma^*:(q_{0,}x,Z_{0,}\varepsilon)\vdash^*(q,\varepsilon,\alpha,u),\}.$$

Vstupní jazyk zásobníkového převodníku P značený $L_I(P)$ definujeme jako

$$L_I(P)=\{x\mid\exists u\in\Delta^*:(x,u)\in T(P)\}.$$

Výstupní jazyk zásobníkového převodníku P značený $L_O(P)$ definujeme jako

$$L_O(P)=\{u\mid\exists x\in\Sigma^*:(x,u)\in T(P)\}.$$

2.2 Příklad zásobníkových převodníků

Mějme zásobníkový převodník $P=(Q,\Sigma,\Gamma,\Delta,\delta,s,\#,\{f\})$, kde

$Q=\{s,q,f\}$,
$\Sigma=\{i,+,*\}$,
$\Delta=\{i,+,*\}$,
$\Gamma=\{\#,E,+,*\}$,
$\delta=\{$
$(\ s,\ \varepsilon,\ \#,\ q,\ E\#,\ \varepsilon\ ),$
$(\ q,\ i,\ E,\ q,\ \varepsilon,\ i\ ),$
$(\ q,\ +,\ E,\ q,\ EE+,\ \varepsilon\ ),$
$(\ q,\ *,\ E,\ q,\ EE*,\ \varepsilon\ ),$
$(\ q,\ \varepsilon,\ +,\ q,\ \varepsilon,\ +\ ),$
$(\ q,\ \varepsilon,\ *,\ q,\ \varepsilon,\ *\ ),$
$(\ q,\ \varepsilon,\ \#,\ f,\ \varepsilon,\ \varepsilon\ )$
$\}$.

Z prvotní konfigurace $(s,...,\#,\varepsilon)$ můžeme provést pouze krok do konfigurace $(q,...,E\#,\varepsilon)$. Jak lze vidět z relace δ, symboly E na zásobníku se pokrývají symboly i ze vstupu za výpisu i na výstup nebo se při přečtení symbolu operace přepisují na $EE+$ resp. $EE*$. Symboly operací na zásobníku se mažou bez dalšího čtení ze vstupu se vypisují na výstup. Lze vidět, že za každým symbolem operace na vstupu následují právě dva výrazy – symbol i nebo další výraz operace s dalšími operandy. Na vstupu tedy musí být výraz v prefixové notaci. Protože každý symbol i je ihned přenesen na výstup, ale symbol operace je odložen až po zpracování obou operandů, výstupem je tentýž výraz v postfixové notaci. Ilustrujme to na výrazu $+*ii*ii$ (v infixové notaci $i*i+i*i$.

$$
\begin{array}{llllcllll}
(s, & +*ii*ii, & \#, & \varepsilon &) \vdash & (q, & +*ii*ii, & E\#, & \varepsilon &) \vdash \\
(q, & *ii*ii, & EE+\#, & \varepsilon &) \vdash & (q, & ii*ii, & EE*E+\#, & \varepsilon &) \vdash \\
(q, & i*ii, & E*E+\#, & i &) \vdash & (q, & *ii, & *E+\#, & ii &) \vdash \\
(q, & *ii, & E+\#, & ii* &) \vdash & (q, & ii, & EE*+\#, & ii* &) \vdash \\
(q, & i, & E*+\#, & ii*i &) \vdash & (q, & \varepsilon, & *+\#, & ii*ii &) \vdash \\
(q, & \varepsilon, & +\#, & ii*ii* &) \vdash & (q, & \varepsilon, & \#, & ii*ii*+ &) \vdash \\
(f, & \varepsilon, & \varepsilon, & ii*ii*+ &) & & & & &
\end{array}
$$

Výstupem překladu je $ii*ii*+$, tedy původní výraz v postfixové notaci.

3 Systémy zásobníkových převodníků

Podobně jako byly definovány *gramatické systémy* (konkrétně *CD GS – kooperující distribuované gramatické systémy*) na základě bezkontextových gramatik, můžeme taktéž zásobníkové převodníky seskupit do systémů a studovat jejich vlastnosti. Na rozdíl od gramatických systémů, které se přepínají po jednom či více využití přepisovacích derivačních pravidel, vytváříme systémy nad stroji a přepínaní komponent například po jednom kroku se nejeví jako vhodné už jen z důvodu otázky řízení. Intuitivnější je přepínat komponentu až po jednom celém překladu.

Mějme několik zásobníkových převodníků, které mají kompatibilní vstupní a výstupní abecedy. Z hlediska systému si můžeme představit, že vytváříme neomezené konečné sekvence převodníků, kde výstupní páska předchozího je ztotožněna se vstupní páskou následujícího převodníku v řadě. Vstupem prvního převodníku je pak vstupní řetězec a výstupem posledního převodníku je výstupní řetězec. V takovéto řadě pracuje vždy právě jeden převodník a další začíná, až když předchozí dokončí svůj překlad. Paralelismus zřetězeným zpracováním, kdy převodník začíná pracovat už v okamžiku, kdy se na jeho vstupu objeví první symbol je možný, ale díky neomezenosti předávaných řetězců, které fungují jako fronty mezi zpracovatelskými uzly, je výsledný překlad ekvivalentní.

Systém převodníků pokrývá všechny možné takovéto sekvence svých komponent, jejichž výskyt není nijak omezován. Každá komponenta definuje svůj vlastní překlad, na který může v nějaké sekvenci navázat další komponenta se svým překladem a tak dále. Lze nahlédnout, že celkový překlad definovaný takovýmto systémem je tranzitivním uzávěrem sjednocení překladů všech jeho komponent.

Stejně jako se popisná síla nejen gramatických systémů zvyšuje přidáváním omezení, bylo by vhodné přidat omezení například na tvary sekvencí komponent pomocí regulárního jazyka popisujícího přijatelné sekvence (po vzoru *regulovaných zásobníků* nebo *regulovaných gramatik* [8]) nebo omezení na vstup a výstup (podobně jako v gramatikách existují neterminály a generování není ukončeno, dokud řetězec neobsahuje pouze terminály). Omezení vstupu a výstupu je jednodušší, postačuje explicitně definovat *abecedu komunikačních symbolů* pro daný systém, přičemž komponenty mezi sebou používají sjednocení vstupní a této abecedy a pro vstup a výstup využívají pouze vstupní resp. výstupní abecedu. Jako základní vezmeme takto omezené systémy a ukážeme, že jejich síla je rovna síle Turingových strojů s ohledem na vstupní jazyk. Na toto omezení můžeme nahlížet i jako na dodatečné zúžení všech prvků překladu pomocí průniku s množinou $\{(x,u) \mid x \in \Sigma^*, u \in \Delta^*\}$, kde Σ je vstupní a Δ je výstupní abeceda.

3.1 Základní definice

Definice 3.1

Systém zásobníkových převodníků Π je $(n+4)$-tice $(\Sigma, \Gamma, \mathrm{K}, \Delta, P_1, P_2, \ldots P_n)$, kde

Σ je *vstupní abeceda*,
Γ je *zásobníková abeceda*,
K je *komunikační abeceda*,
Δ je *výstupní abeceda* a
P_i pro $1 \le i \le n$ je zásobníkový převodník.

$P_i = (Q_i, \Sigma \cup \mathrm{K}, \Gamma, \Delta \cup \mathrm{K}, \delta_i, q_{0,i}, Z_{0,i}, F_i)$ se nazývá i-tou komponentou systému Π. Výstupní abeceda definuje, které jediné symboly jsou povoleny na výstupu, a omezuje tak výsledný překlad.

Definice 3.2

Konfigurace převodníkového systému Π je pětice (i, q, x, α, u), kde

$1 \le i \le n$ je číslo *aktuální komponenty*,
$q \in Q_i$ je *aktuální stav* aktuální (i-té) komponenty,
$x \in (\Sigma \cup \mathrm{K})^*$ je řetězec dosud nepřečteného vstupu aktuální komponenty,
$\alpha \in \Gamma^*$ je řetězec na *zásobníku*, přičemž jako vrchol chápeme jeho nejlevější symbol a
$u \in (\Delta \cup \mathrm{K})^*$ je řetězec dosud vyprodukovaného aktuálního výstupu.

Můžeme vidět, že konfigurace systému je oproti konfiguraci jednoho převodníku rozšířena pouze o identifikaci komponenty, která aktuálně pracuje.

Definice 3.3

Mějme dvě konfigurace $C_1 = (i, q, x, \alpha, u)$ a $C_2 = (j, p, y, \beta, v)$ systému zásobníkových převodníků Π definovaného výše, kde $1 \le i, j \le n$, $q \in Q_i$, $p \in Q_j$, $x, y \in (\Sigma \cup \mathrm{K})^*$, $\alpha, \beta \in \Gamma^*$, $u, v \in (\Delta \cup \mathrm{K})^*$.

Systém Π provádí *krok* z konfigurace C_1 do konfigurace C_2 právě tehdy, když platí jedna z následujících podmínek:

- jestliže $i = j$, $x = ay$, $\alpha = Z\gamma$, $\beta = \beta'\gamma$, $v = uv'$ a (q, a, Z, p, β', v') je obsaženo v δ_i nebo
- jestliže $i = j$, $x = y$, $\alpha = Z\gamma$, $\beta = \beta'\gamma$, $v = uv'$ a $(q, \varepsilon, Z, p, \beta', v')$ je obsaženo v δ_i nebo
- jestliže $x = \varepsilon$, $u = y$, $p = q_{0,j}$, $\beta = Z_{0,j}$, $v = \varepsilon$ a $q \in F_i$.

Zapisujeme

$$(i, q, x, \alpha, u) \vdash (j, p, y, \beta, v)$$

nebo zjednodušeně

- $(i, q, ay, Z\gamma, u) \vdash (i, p, y, \beta'\gamma, uv')$, jestliže $(q, a, Z, p, \beta, v') \in \delta_i$,
- $(i, q, x, Z\gamma, u) \vdash (i, p, x, \beta'\gamma, uv')$, jestliže $(q, \varepsilon, Z, p, \beta', v') \in \delta_i$ a
- $(i, q, \varepsilon, \alpha, u) \vdash (j, q_{0,j}, u, Z_{0,j}, \varepsilon)$, jestliže $q \in F_i$.

První dvě možnosti jsou provedení kroku v rámci jedné komponenty tak, jak je definován krok zásobníkového převodníku. Poslední možnost je přepnutí komponenty, která skončila a je tedy v koncovém stavu a má prázdný vstup, na další komponentu, která začíná v počátečním stavu se svým počátečním zásobníkem a prázdným výstupem.

Pro zdůraznění náležitosti relace $\vdash$ systému Π můžeme psát $\vdash_\Pi$.

Sekvenci kroků chápeme jako skládání kroků ve smyslu skládání relačního. Tedy sekvence n kroků je n-tá mocnina relace $\vdash$, zapisujeme

$$(i, q, x, \alpha, u) \vdash^n (j, p, y, \beta, v).$$

Důležité jsou pak ještě *tranzitivní uzávěr* a *reflexivní a tranzitivní uzávěr* této relace značené

$$(i,q,x,\alpha,u) \vdash^{+} (j,p,y,\beta,v) \text{, resp. } (i,q,x,\alpha,u) \vdash^{*} (j,p,y,\beta,v) .$$

Navíc oproti krokům jednoho převodníku definujeme *průběh komponenty*, jako sekvenci kroků z počáteční konfigurace komponenty i do konfigurace, ve které se může přepnout komponenta

$$(i,q_{0,i},x,Z_{0,i},\varepsilon) \vdash^{P_i} (i,q,\varepsilon,\alpha,u),$$

jestliže $(i,q_{0,i},x,Z_{0,i},\varepsilon) \vdash^{n} (i,q,\varepsilon,\alpha,u)$ pro nějaké $0 \leq n$ a $q \in F_i$.

Druhý speciální krok je *přepnutí komponenty*, tedy třetí možnost z definice jednoho kroku. Budeme jej značit

$$(i,q,\varepsilon,\alpha,u) \vdash^{S} (j,q_{0,j},u,Z_{0,j},\varepsilon),$$

jestliže $(i,q,\varepsilon,\alpha,u) \vdash (j,q_{0,j},u,Z_{0,j},\varepsilon)$.

Jak lze vidět, každou sekvenci kroků můžeme vyjádřit jako sekvenci střídající průběh komponenty a přepnutí komponenty. Pomocí skládání relací můžeme psát

$$\vdash^{*} = \vdash^{P_{i_m}} \circ \vdash^{S} \circ \cdots \circ \vdash^{S} \circ \vdash^{P_{i_2}} \circ \vdash^{S} \circ \vdash^{P_{i_1}},$$

kde P_{i_j} pro $1 \leq i_j \leq n$ a $1 \leq j \leq m$ značí komponentu použitou na j-tém místě.

Definice 3.4

Výstup systému zásobníkových převodníků Π pro *vstup* $x \in \Sigma^{*}$ je $u \in \Delta^{*}$ právě tehdy, když $(i,q_{0,i},x,Z_{0,i},\varepsilon) \vdash^{*} (j,q,\varepsilon,\alpha,u)$ pro nějaké $1 \leq i,j \leq n$, $q \in F_j$ a $\alpha \in \Gamma^{*}$. Můžeme určit *sekvenci komponent* využitelnou pro překlad (x,u), pokud relaci $\vdash^{*}$ rozepíšeme na relace $\vdash^{S}$ a $\vdash^{P_i}$, jak je definováno výše. Získáme jednu z možných sekvencí komponent $P_{i_1}P_{i_2}\ldots P_{i_m}$.

Překlad definovaný systémem zásobníkových převodníků Π, značený $T(\Pi)$ je definován takto

$$T(\Pi) = \{(x,u) \mid x \in \Sigma^{*}, u \in \Delta^{*}, \exists 1 \leq i,j \leq n, q \in F_j, \alpha \in \Gamma^{*} : (i,q_{0,i},x,Z_{0,i},\varepsilon) \vdash^{*} (j,q,\varepsilon,\alpha,u),\}.$$

Vstupní jazyk systému zásobníkových převodníků Π značený $L_I(\Pi)$ definujeme jako

$$L_I(\Pi) = \{x \mid \exists u \in \Delta^{*} : (x,u) \in T(\Pi)\}.$$

Výstupní jazyk systému zásobníkových převodníků Π značený $L_O(\Pi)$ definujeme jako

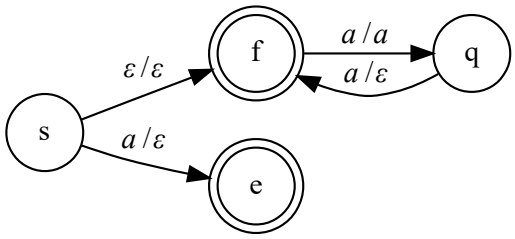
$$L_O(\Pi) = \{u \mid \exists x \in \Sigma^{*} : (x,u) \in T(\Pi)\}.$$

3.2 Příklady systémů převodníků

Příklad 3.1

Mějme systém zásobníkových převodníků $\Pi=(\Sigma,\Gamma,\mathrm{K},\Delta,P)$, kde

$\Sigma=\mathrm{K}=\{a\}$, $\Gamma=\{\#\}$,
$\Delta=\emptyset$,
$P=(\{s,q,f,e\},\Sigma\cup\mathrm{K},\Gamma,\Delta\cup\mathrm{K},\delta,s,\#,\{e,f\})$
$\delta=\{$
$(\ s,\ \varepsilon,\ \#,\ f,\ \#,\ \varepsilon\ ),$
$(\ s,\ a,\ \#,\ e,\ \#,\ \varepsilon\ ),$
$(\ f,\ a,\ \#,\ q,\ \#,\ a\ ),$
$(\ q,\ a,\ \#,\ f,\ \#,\ \varepsilon\ )$
$\}.$



Ilustrace 3.1: Graf přechodů převodníku P

Tento minimalistický systém s jednou komponentou a čtyřmi stavy definuje vstupní jazyk $\{a^{2^n}|0\le n\}\cup\{\varepsilon\}$ a překlad $\{(x,\varepsilon)|x\in L_I(\Pi)\}$. Začněme analýzou od konce. Výsledek překladu musí být prázdný řetězec, protože výstupní abeceda je prázdná. Pokud jsme skončili ve stavu e, museli jsme začít s řetězcem a a vyprodukujeme prázdný řetězec. Pokud překlad ukončíme ve stavu f s prázdným výstupem, museli jsme mít na vstupu prázdný řetězec. Základní dva prvky vstupního jazyka jsou tedy prázdný řetězec a řetězec a.

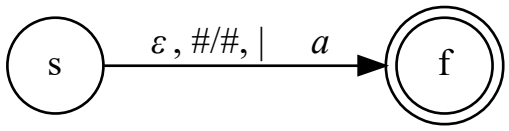
Zkoumejme dál, kdy jsme mohli vyprodukovat řetězec a. Ze stavu e to být nemohlo, jediná cesta z počátečního stavu má prázdný výstup. Pro stav f můžeme vygenerovat jednoprvkový výstup pokud projdeme sekvenci stavů s,f,q,f. Vstupem tedy musel být dvouprvkový řetězec. Pro dvouprvkový řetězec můžeme s analýzou pokračovat obdobně a každý průběh komponenty zdvojnásobí délku řetězce. Výsledný jazyk tedy obsahuje všechny řetězce délky mocniny dvou. Ukažme si rozbor řetězce $aaaa$:

$$
\begin{aligned}
&(s,\ aaaa,\ \#,\ \varepsilon\ ) \vdash (f,\ aaaa,\ \#,\ \varepsilon\ ) \vdash\\
&(q,\ aaa,\ \#,\ a\ ) \vdash (f,\ aa,\ \#,\ a\ ) \vdash\\
&(q,\ a,\ \#,\ aa\ ) \vdash (f,\ \varepsilon,\ \#,\ aa\ ) \vdash\\
&(s,\ aa,\ \#,\ \varepsilon*\ ) \vdash (f,\ aa,\ \#,\ \varepsilon*\ ) \vdash\\
&(q,\ a,\ \#,\ a\ ) \vdash (f,\ \varepsilon,\ \#,\ a\ ) \vdash\\
&(s,\ a,\ \#,\ \varepsilon\ ) \vdash (e,\ \varepsilon,\ \#,\ \varepsilon\ ).
\end{aligned}
$$

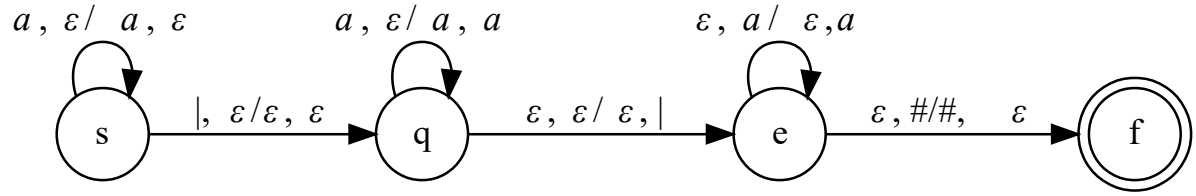
Příklad 3.2

Mějme systém zásobníkových převodníků $\Pi=(\Sigma,\Gamma,\mathrm{K},\Delta,P_1,P_{2,}P_3)$, kde

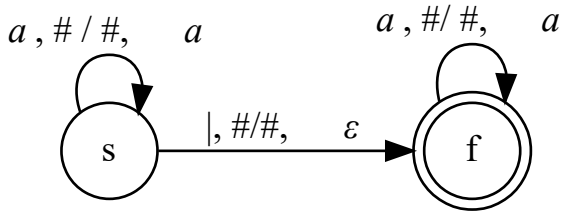
$\Sigma=\emptyset$,
$\Gamma=\{\#,a\}$,
$\mathrm{K}=\{a,|\}$
$\Delta=\{a\}$,
$P_1=(\{s,f\},\Sigma\cup\mathrm{K},\Gamma,\Delta\cup\mathrm{K},\delta_1,s,\#,\{f\})$,
$P_2=(\{s,q,e,f\},\Sigma\cup\mathrm{K},\Gamma,\Delta\cup\mathrm{K},\delta_2,s,\#,\{f\})$,
$P_3=(\{s\},\Sigma\cup\mathrm{K},\Gamma,\Delta\cup\mathrm{K},\delta_3,s,\#,\{s\})$.



Ilustrace 3.2: δ_1 pro P_1



Ilustrace 3.3: δ_2 pro P_2



Ilustrace 3.4: δ_3 pro P_3

Poznámka pro čtení grafů: přechody jsou značeny ve tvaru `<čtený symbol>, <vrchol zásobníku> / <nový zásobník>, <zapsaný řetězec>.`

Tento zajímavý systém je zaměřený naopak na výstupní jazyk. Vstupní abeceda je prázdná, takže každý překlad začíná prázdným řetězcem. Lze tedy říct, že z překladače je vyroben generátor.

První komponenta generuje pouze z prázdného řetězce řetězec $|a$.

Třetí komponenta kopíruje na výstup symboly a vlevo od oddělovače $|$ a pak i symboly vpravo od oddělovače. Překládá tedy $a^i|a^j$ na a^{i+j}.

Druhá komponenta nejprve naskládá symboly a vlevo od oddělovače na zásobník a pak přečte oddělovač, zkopíruje symboly a vpravo od oddělovače na výstup i zásobník. V následném stavu vyprázdní zásobník až ke dnu $\#$ na výstup. Realizuje tedy překlad $a^i|a^j$ na řetězec $a^j|a^{i+j}$.

Díky vhodným abecedám a formátu je zajištěna posloupnost komponent začínající první, následovaná nějakým počtem druhých komponent a ukončena třetí komponentou. Lze ověřit, že

výsledný výstupní jazyk tvoří řetězce délek 1, 2, 3, 5, 8, 13…, tedy Fibbonaciho posloupnosti s počátkem 0, 1. Zkusme generovat řetězec a^5.

3.3 Ekvivalence systémů zásobníkových převodníků

Převodníkové systémy můžeme dělit podle počtu komponent do tříd a tyto pak porovnávat vzhledem k jejich popisné síle. Je zřejmé, že systémy s více komponentami jsou alespoň stejně silné jako systémy s méně komponentami už jen proto, že přidaná komponenta se nemusí vůbec zapojit do překladu. Otázkou je, jestli přidáním komponenty zvýšíme sílu či jestli tato zůstane stejná. Odpovědí je algoritmus, který ke každému systému vytvoří nový jednokomponentní systém, který definuje stejný překlad. Pak lze překlad libovolného systému definovat systémem s jednou komponentou. S využitím těchto dvou poznatků lze překlad definovaný systémem s libovolným počtem komponent definovat také i systémem s libovolným jiným počtem komponent. Třídy systémů podle počtu komponent tedy jsou vzhledem k popisné síle ekvivalentní.

Algoritmus pro redukci počtu komponent vychází z algoritmu pro sjednocení jazyků dvou zásobníkových automatů. Nejprve z počátečního stavu nedeterministicky rozhodneme, která komponenta bude simulována první, a ε-přechodem přejdeme do počátečního stavu této komponenty. Při zajištění disjunktnosti stavů komponent nelze dostat řízení z jedné komponenty do druhé jinak než dokročením na konec překladu jedné komponenty a jejím přepnutím. Během průchodu komponentou se generuje výstup, který je v nezměněné podobě předán na vstup dalšího běhu simulační komponenty.

Algoritmus 3.5

Mějme systém zásobníkových převodníků $\Pi = (\Sigma, \Gamma, \mathrm{K}, \Delta, P_1, P_2, \ldots, P_n)$. Systém $\Pi^1 = (\Sigma, \Gamma \cup \{\#\}, \mathrm{K}, \Delta, R_1)$ definující stejný překlad vytvoříme takto:

$R_1 = (Q_R, \Sigma \cup \mathrm{K}, \Gamma \cup \{\#\}, \Delta \cup \mathrm{K}, \delta_R, S, \#, F_R)$, kde

$\# \notin \Gamma$,

$S \notin Q_i$ pro $1 \le i \le n$,

$Q_R = \{S\} \cup \{(q,i) \mid q \in Q_i, 1 \le i \le n\}$,

$F_R = \{(q,i) \mid q \in F_i, 1 \le i \le n\}$,

$\delta_R = \{(S, \varepsilon, \#, (q_{0,i}, i), Z_{0,i}, \varepsilon) \mid 1 \le i \le n\} \cup$

$\{((q,i), x, Z, (p,i), \alpha, u) \mid (q, x, Z, p, \alpha, u) \in \delta_i, 1 \le i \le n\}$.

Tento systém obsahuje ve své jediné komponentě všechny stavy a přechody ze zadaného systému. Navíc obsahuje nový vrchol zásobníku a počáteční stav, ze kterého lze provést krok ε-přechodem do počáteční konfigurace libovolné komponenty v původním systému.

Tvrzení 3.6

Algoritmus 3.5 je správný neboli Systém Π a Π^1 generují stejné překlady.

Důkaz

Potřebujeme dokázat, že každý krok v původním systému lze simulovat v jedno-komponentním systému. Problém rozdělíme na dokázání správnosti průběhu komponenty a přepnutí komponenty. Definujeme zobrazení Φ, které každé konfiguraci (i, q, x, α, u) původního systému přiřadí

konfiguraci $(1,(q,i),x,\alpha,u)$ nového systému a které zachovává relaci kroku $\vdash$, přesněji pro všechny konfigurace C_1, C_2 původního systému platí

$$C_1 \vdash_{\Pi} C_2 \Leftrightarrow \Phi(C_1) \vdash_{\Pi^1}^{+} \Phi(C_2).$$

Průběh komponenty i $\vdash_{\Pi}^{C_i}$ se skládá z konečného počtu kroků $\vdash_{\Pi}$. Krok bez využití ε-pravidla zapisujeme jako

$$(i,q,ax,Z\gamma,u) \vdash_{\Pi} (i,p,x,\beta'\gamma,uv'),$$

jestliže $(q,a,Z,p,\beta',v') \in \delta_i$. Protože δ_R obsahuje $((q,i),a,Z,(p,i),\beta',z')$, v jednokomponentním systému můžeme psát

$$\Phi((i,q,ax,Z\gamma,u)) = (1,(q,i),ax,Z\gamma,u) \vdash_{\Pi^1} (1,(p,i),x,\beta'\gamma,uv') = \Phi((i,q,x,\beta'\gamma,uv')).$$

Zobrazení Φ tedy zachovává krok bez využití ε-pravidla.

Druhou možností je využití ε-pravidla. Postupujeme obdobně jako v předchozím případě. Krok zapisujeme jako

$$(i,q,x,Z\gamma,u) \vdash_{\Pi} (i,p,x,\beta'\gamma,uv'),$$

jestliže $(q,\varepsilon,Z,p,\beta',v') \in \delta_i$. Protože δ_R obsahuje $((q,i)\varepsilon,Z,(p,i),\beta',v')$, v jednokomponentním systému můžeme psát

$$\Phi((i,q,x,Z\gamma,u)) = (1,(q,i),x,Z\gamma,u) \vdash_{\Pi^1} (1,(p,i),x,\beta'\gamma,uv') \Phi((i,p,x,\beta\gamma,uv')).$$

Zobrazení Φ tedy zachovává krok s využitím ε-pravidla.

Poslední možností je přepnutí komponenty $\vdash_{\Pi}^{S}$. Podle definice se jedná o krok tvaru

$$(i,q,\varepsilon,\alpha,u) \vdash_{\Pi} (j,q_{0,j},u,Z_{0,j},\varepsilon),$$

jestliže $q \in F_i$. Protože F_R obsahuje (q,i) v jednokomponentním systému můžeme psát

$$\Phi((i,q,\varepsilon,\alpha,u)) = (1,(q,i),\varepsilon,\alpha,u) \vdash_{\Pi^1} (1,S,u,\#,\varepsilon) \vdash_{\Pi^1} (1,(q_{0,j},j),u,Z_{0,j},\varepsilon) = \Phi((j,q_{0,j},u,Z_{0,j},\varepsilon)).$$

Zobrazení Φ tedy zachovává i přepnutí komponenty.

Nyní postupujme induktivně podle počtu kroků při překladu řetězce $x \in \Sigma^*$. Indukční hypotéza zní

$$(i,q_{0,i},x,Z_{0,i},\varepsilon) \vdash_{\Pi}^{k} (j,q,y,\alpha,u) \Leftrightarrow (1,S,x,\#,\varepsilon) \vdash_{\Pi^1}^{k+s+1} (1,(q,j),y,\alpha,u),$$

kde k je počet kroků, a s je počet přepnutí komponent. Pro $k=0$ dostáváme $s=0$ a

$$(i,q_{0,i},x,Z_{0,i},\varepsilon)\vdash_{\Pi}^{0}(i,q_{0,i},x,Z_{0,i},\varepsilon)\Leftrightarrow$$
$$(1,S,x,\#,\varepsilon)\vdash_{\Pi^1}^{1}(1,(q_{0,i},i),x,Z_{0,i},\varepsilon).$$

Nyní předpokládejme, že indukční hypotéza platí pro $l>0$ a dokažme, že platí i pro $l+1$:

$$(i,q_{0,i},x,Z_{0,i},\varepsilon)\vdash_{\Pi}^{l}(j,q,y,\alpha,u)\Leftrightarrow$$
$$(i,q_{0,i},x,Z_{0,i},\varepsilon)\vdash_{\Pi^1}^{l-1}(i',q',y',\alpha',u')\vdash^{1}(j,q,y,\alpha,u).$$

Výše jsme dokázali, že $C_1\vdash_{\Pi}C_2\Leftrightarrow\Phi(C_1)\vdash_{\Pi^1}^{+}\Phi(C_2)$ pro C_1,C_2 konfigurace původního systému. Platí tedy $(1,(q',1),y',\alpha',u')\vdash_{\Pi^1}^{m}(1,(q,1),y,\alpha,u)$ a $m=1$, jestliže se nepřepnula komponenta, jinak $m=2$. Díky indukční hypotéze tedy platí

$$(1,S,x,\#,\varepsilon)\vdash_{\Pi^1}^{l+s+1}(1,(q',i'),y',\alpha',u')\vdash_{\Pi^1}^{m}(1,(q,j)y,\alpha,u)\Leftrightarrow$$
$$(1,S,x,\#,\varepsilon)\vdash_{\Pi^1}^{l+s+m+1}(1,(q,j),y,\alpha,u),$$

kde $m=1$, jestliže se nepřepnula komponenta, jinak $m=2$. Dokázali jsme tedy, že kroky při překladu řetězce jsou v obou systémech ekvivalentní.

Pokud nyní volíme $q\in F_j$ a $y=\varepsilon$, pak $(q,j)\in F_R$ a tedy

$$(x,u)\in T(\Pi)\Leftrightarrow(x,u)\in T(\Pi^1).$$

Věta 3.7

Systémy zásobníkových automatů jsou s různými počty komponent mají stejnou popisnou sílu.

Důkaz

Plyne z věty 3.6 a zřejmé inkluze opačným směrem, tedy že méně komponentní systémy nejsou silnější než více komponentní systémy.

4 Turingovy stroje

Při studiu síly systémů zásobníkových převodníků můžeme porovnávat rodiny překladů, které definují, ale – protože každý překlad má svůj vstupní a výstupní jazyk – můžeme převodníky také chápat jako přijímací stroje. Tento pohled nám poskytuje možnost porovnat sílu převodníků se zavedenějšími přijímajícími stroji jako jsou regulární a zásobníkové automaty a především *Turingovy stroje*. Jednou z nejdůležitějších tézí teoretické informatiky je tzv. *Church-Turingova téze*, která říká (v různých formách v [4], [5], zde z [6]), že neexistují stroje (abstraktní či reálné), které jsou výpočetně silnější než Turingovy stroje. Proto je vhodné se pokusit porovnat sílu systému převodníků právě s Turingovými stroji.

Turingovy stroje byly definovány *Alanem Turingem* už v roce 1936 (podle [6]). Jsou to *univerzální matematické stroje*, které jsou schopné implementovat libovolný algoritmus, nebo možná naopak, správně definovaný *algoritmus* je vyjádřitelný Turingovým strojem.

Vyjdeme-li ze zásobníkových automatů, které se skládají ze vstupní pásky, čtecí hlavy, zásobníkové pásky, zásobníkové hlavy a konečného řízení, stačí nám uvolnit některá omezení, abychom získali Turingův stroj. Nejdůležitějším rozdílem je existence pouze *jediné pásky* a jediné hlavy, která je schopna číst, psát a především se *volně pohybovat* po pásce. Páska samotná je *jednostranně neomezená*, má tedy svůj začátek vlevo, před který se nemůže hlava posunout, ale směrem doprava je pohyb bez omezení. Stroj pak na této pásce přepisuje symboly a pohybuje hlavou podle stavu řízení a symbolem pod hlavou. Výpočet skončí, když se řízení dostane do *koncového stavu*, nemůže pokračovat dále nebo se hlava pokusí přejít před začátek pásky. Pokud tedy chceme rozhodnout zda nějaké slovo zařadíme do jazyka přijímaného Turingovým strojem, předložíme mu ho na začátku výpočtu na pásce rozběhneme výpočet z počátečního stavu a určíme jakým způsobem stroj skončil. Pokud skončil přechodem do koncového stavu, slovo do jazyka zařadíme, jinak (nebo také jestliže stroj nikdy neskončí) slovo nepřijmeme. Jak lze vidět, na konci výpočtu se na pásce nachází nějaký řetězec, který ale pro rozhodování není důležitý.

Pokud se budeme zajímat i o obsah pásky po skončení výpočtu, můžeme Turingovy stroje chápat také jako převodníky. Dále v této kapitole takto definujeme překlad pro daný Turingův stroj. Nejprve ovšem ukážeme, že systémy převodníků lze využít pro přijímání jazyků a jejich síla je ekvivalentní síle Turingových strojů.

4.1 Základní definice

Definice 4.1

Turingův stroj M je šestice $(Q, \Sigma, \Gamma, \delta, q_0, q_F)$, kde

Q je konečná neprázdná *množina stavů*,
Σ je *vstupní abeceda* a $\Delta \notin \Sigma$,
Γ je *pásková abeceda* a $\Sigma \subset \Gamma$, $\Delta \in \Gamma$,
$\delta : (Q \setminus \{q_F\}) \times \Gamma \to Q \times (\Gamma \cup \{L, R\})$ je parciální *přechodová funkce* a $L, R \notin \Gamma$,
$q_0 \in Q$ je *počáteční stav* a
$q_F \in Q$ je *koncový stav.*

Symbol Δ je tzv. *prázdný symbol* a vykytuje na začátku výpočtu na všech jinak nevyužitých polích pásky. Všimněme si, že δ je funkce – každé kombinaci stavu a symbolu pod hlavou přiřazuje nejvýše jednu dvojici následujícího stavu a jedné z akcí: přesun doprava, přesun doleva a přepis na symbol. Turingovy stroje jsou tedy deterministické na rozdíl od dosud zavedených zásobníkových převodníků a jejich systémů.

Definice 4.1

Konfigurace Turingova stroje M je trojce (q, γ, n), kde

$q \in Q$ je *aktuální stav*,

$\gamma \in \Gamma^*$ je nejdelší *prefix pásky*, který obsahuje všechny neprázdné symboly na pásce (celá páska se tedy dá zapsat jako $\gamma \Delta^\omega$) a

$1 \leq n$ je *pozice hlavy* na pásce, konkrétně počet symbolů vlevo od symbolu pod hlavou.

Pásku můžeme rozdělit na tři důležité části: řetězec vlevo od hlavy, symbol pod hlavou a řetězec vpravo od pásky. Jestliže označíme n-tý symbol na pásce $\gamma \Delta^\omega$ jako γ_n, můžeme celou pásku zapsat jako

$$\gamma \Delta^\omega = \gamma_1 \gamma_2 \dots \gamma_{n-1} \gamma_n \gamma_{n+1} \dots \gamma_m \Delta^\omega = \overleftarrow{\gamma}^n \gamma_n \overrightarrow{\gamma}^n \Delta^\omega,$$

kde $\gamma_m \neq \Delta$ nebo $m = n$ a $\gamma_m = \Delta$.

Definice 4.2

Mějme dvě konfigurace $C_1 = (q, \gamma_{1,} n_1)$ a $C_2 = (p, \gamma_{2,} n_2)$ Turingova stroje M, kde $q, p \in Q$, $\gamma_{1,} \gamma_2 \in \Gamma^*$, $1 \leq n_1, n_2$.

Turingův stroj M provádí *krok* z konfigurace C_1 do konfigurace C_2 právě tehdy, když platí jedna z následujících podmínek:

- $\gamma_1 = \gamma_2$, $n_1 + 1 = n_2$ a $\delta(q, \gamma_{1,n_1}) = (p, R)$ nebo
- $\gamma_1 = \gamma_2$, $n_1 - 1 = n_2$, $n > 0$ a $\delta(q, \gamma_{1,n_1}) = (p, L)$ nebo
- $\overleftarrow{\gamma}_1^{n_1} = \overleftarrow{\gamma}_2^{n_2}$, $\overrightarrow{\gamma}_1^{n_1} = \overrightarrow{\gamma}_2^{n_2}$, $n_1 = n_2$ a $\delta(q, \gamma_{1,n_1}) = (p, \gamma_{2,n_2})$.

Zapisujeme

$$(q, \gamma_{1,} n_1) \vdash (p, \gamma_{2,} n_2)$$

nebo zjednodušeně

- $(q, \gamma, n) \vdash (p, \gamma, n+1)$, jestliže $\delta(q, \gamma_n) = (p, R)$,
- $(q, \gamma, n) \vdash (p, \gamma, n-1)$, jestliže $\delta(q, \gamma_n) = (p, L)$ a $n > 0$,
- $(q, \overleftarrow{\gamma}^n \gamma_n \overrightarrow{\gamma}^n, n) \vdash (p, \overleftarrow{\gamma}^n b \overrightarrow{\gamma}^n, n)$, jestliže $\delta(q, \gamma_n) = (p, b)$ pro $b \in \Gamma$.

První dvě možnosti pouze pohnou hlavou doprava, resp. doleva o jedno pole. Páska se v těchto případech nemění. Třetí možnost naopak nehýbá hlavou, ale přepisuje symbol pod ní podle přechodové funkce.

Výpočet se skládá ze *sekvence kroků*, tedy relačního složení n kroků. Zapisujeme

$$(q, \gamma_{1,} n_1) \vdash^n (p, \gamma_{2,} n_2).$$

Opět důležité jsou *tranzitivní uzávěr* a *tranzitivní a reflexivní uzávěr* relace jednoho kroku

$$(q, \gamma_{1,} n_1) \vdash^{+} (p, \gamma_{2,} n_2) \ \ resp. \ \ (q, \gamma_{1,} n_1) \vdash^{*} (p, \gamma_{2,} n_2).$$

Definice 4.3

Jazyk přijímaný Turingovým strojem M, značený $L(M)$, definujeme jako

$$L(M) = \{w \mid w \in \Sigma^{*}, \exists \gamma \in \Gamma^{*}, 0 \leq n : (q_{0,} \Delta w, 0) \vdash^{*} (q_F, \gamma, n)\}.$$

Přijímané slovo je tedy zapsáno po jednom prázdném symbolu na pásku Turingova stroje a výpočet začne z počátečního stavu s hlavou nad tímto prázdným symbolem. Pokud stroj doběhne do koncového stavu s libovolným obsahem pásky a libovolnou pozicí hlavy, je zadané slovo přijato.

4.2 Ekvivalence Turingových strojů a systémů převodníků

Nyní dokážeme, že každý jazyk přijímaný Turingovým strojem je vstupním jazykem nějakého systémů převodníků. Nebude nás tedy zajímat samotný překlad definovaný systémem, ale pouze slova, která dokáže přeložit. Opačné tvrzení, tedy že vstupní jazyk každého systému převodníků je jazykem nějakého Turingova stroje se může opřít o *Church-Turingovu* tézi. Žádný stroj nemůže být výpočetně silnější než Turingovy stroje, a proto musí existovat Turingův stroj, který přijme vstupní jazyk daného systému převodníků.

Převodníkové systémy střídají průběh komponenty a přepnutí komponenty. Hlavní myšlenkou konstrukce systému pro daný Turingův stroj je využít právě průběhy komponent k simulování jednotlivých kroků Turingova stroje. Tyto kroky buď mění obsah pásky na jednom místě nebo posouvají hlavu o jedno pole. Vytvoříme celkem *pět komponent*: první přidá ke vstupnímu řetězci zarážku a informaci o počátečním stavu simulovaného stroje, další tři komponenty budou mít za úkol překládat řetězce pásky podle simulovaného kroku a poslední komponenta pouze zkontroluje, zda je na pásce zakódován koncový stav, nevyprodukuje žádný výstup svým způsobem ukončení rozhodne přijetí či nepřijetí vstupního řetězce. Každá z těchto komponent může být přepnuta kdykoli, takže je nutné zajistit, aby špatně zvolená komponenta nemohla vyprodukovat žádný výstup. Také je nutné nepřijímat mezivýsledky předávané při přepínání komponent – pokud by platný výstupní řetězec generovala i jiná komponenta než pátá, která zároveň kontroluje koncový stav simulovaného Turingova stroje, byl by přijat i nevhodný řetězec.

V předávaných řetězcích potřebujeme kódovat celou konfiguraci Turingova stroje, tedy obsah pásky, aktuální stav a pozici hlavy. Jako vhodné řešení se nabízí obohatit obsah pásky o symbol stavu, který zapíšeme těsně před symbol pod hlavou.

Definice 4.4

Zobrazení Φ z konfigurací Turingových strojů na řetězec je definováno jako

$$\Phi : Q \times \Gamma^{*} \times \mathbb{N} \to \{\#\} \Gamma^{*} Q \Gamma^{+}$$
$$\Phi(q, \gamma, n) = \# \overset{n}{\vec{\gamma}} q \gamma_n \overset{n}{\vec{\gamma}}.$$

Symbol $\# \notin \Delta_{\Pi}$ slouží jako *pojistka* proti přijetí řetězce – dokud bude řetězec předávaný mezi komponentami obsahovat alespoň jeden symbol mimo výstupní abecedu, nelze tento „mezivýsledek“ brát jako výstup. Všechny komponenty kromě poslední musí nejprve vygenerovat právě tento symbol

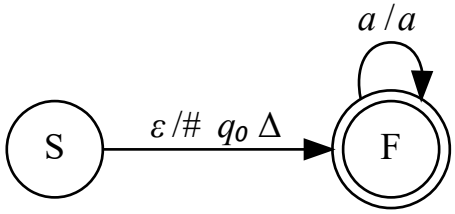
a všechny kromě první naopak musí zkontrolovat, zda se tento symbol nalézá na začátku jejich vstupu.

Dále předpokládáme, že $Q \cap \Gamma = \emptyset$ a $\# \notin \Gamma$. Definujme postupně tyto komponenty pro zadaný Turingův stroj $M = (Q, \Sigma, \Gamma, \delta, q_0, q_F)$:

Algoritmus 4.5

První komponentu pro úpravu vstupu P_1^M vytvoříme takto:

$P_1^M = (Q_1, \Gamma \cup \{\#\}, \{\#\}, \Gamma \cup \{\#\}, \delta_1, S, \#, \{F\})$, kde
$Q_1 = \{S, F\}$ a
$\delta_1 = \{(S, \varepsilon, \#, F, \#, \# q_0 \Delta)\} \cup \{(F, a, \#, F, \#, a) | a \in \Sigma\}$.



Ilustrace 4.1: δ_1 pro různá $a \in \Sigma$

Ke každému vstupnímu řetězci je zleva připojena *pojistka*, označení počátečního stavu Turingova stroje a prázdný znak. Takto výstup komponenty odpovídá výsledku funkce Φ aplikované na počáteční konfiguraci Turingova stroje.

Vstupní jazyk tohoto převodníku může obsahovat pouze symboly $a \in \Sigma$, protože obsahuje pouze ε-přechody a přechody, které čtou a. Tyto symboly mohou být v libovolném pořadí a počtu, takže vstupní jazyk odpovídá množině všech řetězců vstupní abecedy: $L_I(P_1^M) = \Sigma^*$. Dále pro každé slovo $w \in \Sigma^*$ o délce $|w| = n$ tato komponenta definuje překlad těmito kroky:

$$\begin{aligned} (S, w, \#, \varepsilon) &\vdash (F, w_1 w_2 \ldots w_n, \#, \# q_0 \Delta) \\ &\vdash (F, w_2 \ldots w_n, \#, \# q_0 \Delta w_1) \\ &\vdash^{n-1} (F, \varepsilon, \#, \# q_0 \Delta w_1 w_2 \ldots w_n) \\ &= (F, \varepsilon, \#, \# q_0 \Delta w). \end{aligned}$$

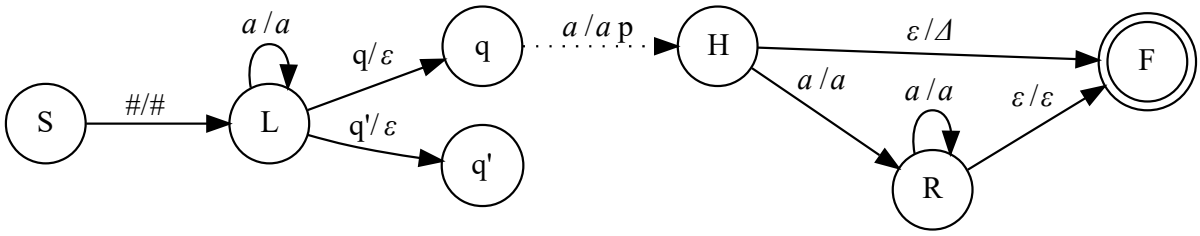
Výsledný překlad tedy je $T(P_1^M) = \{(w, \# q_0 \Delta w) | w \in \Sigma^*\}$. Všimněme si, že výstup je prvkem jazyka $\{\#\} \Gamma^* Q \Gamma^+$.

Algoritmus 4.6

Druhou komponentu pro simulaci posunu hlavy doprava P_R^M vytvoříme takto:

$P_R^M = (Q_R, \Gamma \cup \{\#\}, \{\#\}, \Gamma \cup \{\#\}, \delta_R, S, \#, \{F\})$, kde
$Q_R = \{S, L, H, R, F\} \cup Q$, $S, L, H, R, F \notin Q$ a

$$\delta_R = \{(S,\#,\#,L,\#,\#)\} \cup$$
$$\{(L,a,\#,L,\#,a) | a\in\Gamma\} \cup \{(L,q,\#,q,\#,\varepsilon) | q\in Q\} \cup$$
$$\{(q,a,\#,H,\#,ap) | a\in\Gamma, q\in Q, \delta(q,a)=(p,R)\} \cup$$
$$\{(H,a,\#,R,\#,a) | a\in\Gamma\} \cup \{(H,\varepsilon,\#,F,\#,\Delta)\} \cup$$
$$\{(R,a,\#,R,\#,a) | a\in\Gamma\} \cup \{(R,\varepsilon,\#,F,\#,\varepsilon)\}.$$



Ilustrace 4.2: δ_R *pro různé* $q\in Q$ *a* $a\in\Gamma$

q zde značí jeden z mnoha stavů z Q, pro něhož a symbol $a\in\Gamma$ je definována přechodová funkce $\delta(q,a)=(p,R)$. Naopak pro q' zde přechodová funkce takto definována není.

Druhá komponenta nejprve zkontroluje *pojistku* a dále překopíruje na výstup všechny symboly nalevo od hlavy. Po dosažení označení stavu jsme přesně před symbolem pod hlavou a úkolem této komponenty je hlavu posunout o jedno pole doprava. Zapamatuje si tedy přečtené označení stavu v rámci svého stavu a zkopíruje symbol pod hlavou na výstup. Zde také kontroluje, zda byla správně vybrána komponenta, protože při nevhodné kombinaci přečteného stavu a symbolu pod hlavou není definován přechod z vnitřního stavu q a převodník se tak zastaví v nekoncovém stavu. Naopak pokud byla správně vybrána komponenta, zapíše se za symbol pod hlavou nový stav p, čímž se efektivně posune hlava. Dále je nutné ověřit, jestli jsme na konci neprázdné části pásky. Pokud ano, přidá se na konec nový prázdný symbol, jinak se pouze zkopírují symboly napravo od původní hlavy.

Předpokládejme, že na vstupu je libovolné slovo z Σ^*. Převodník toto slovo nepřijme, protože vyžaduje jako první symbol *pojistku* $\#$, která není v Σ obsažena. Naopak zkusme slova $w\in\{\#\}\Gamma^* Q \Gamma^+$:

$$\begin{aligned}
(S,w,\#,\varepsilon)=(S,\#\vec{\gamma}^{n} q\gamma_n\vec{\gamma}^{n},\#,\varepsilon) \quad &\vdash \quad (L,\vec{\gamma}^{n} q\gamma_n\vec{\gamma}^{n},\#,\#)\\
&\vdash^{n-1} (L,q\gamma_n\vec{\gamma}^{n},\#,\#\overleftarrow{\gamma}^{n})\\
&\vdash \quad (q,\gamma_n\vec{\gamma}^{n},\#,\#\overleftarrow{\gamma}^{n})\\
&\vdash \quad (H,\vec{\gamma}^{n},\#,\#\overleftarrow{\gamma}^{n}\gamma_n p) \text{ právě tehdy, když } \delta(q,\gamma_n)=(p,R)\\
&\vdash \quad (R,\vec{\gamma}^{n+1},\#,\#\overleftarrow{\gamma}^{n+1} p\gamma_{n+1})\\
&\vdash^{|\vec{\gamma}^{n+1}|} (R,\varepsilon,\#,\#\overleftarrow{\gamma}^{n+1} p\gamma_{n+1}\vec{\gamma}^{n+1})\\
&\vdash \quad (F,\varepsilon,\#,\#\overleftarrow{\gamma}^{n+1} p\gamma_{n+1}\vec{\gamma}^{n+1}),
\end{aligned}$$

a v případě $\vec{\gamma}^{n}=\varepsilon$

$$
\begin{aligned}
(S,w,\#,\varepsilon)=(S,\#\overrightarrow{\gamma}^{n}q\gamma_n,\#,\varepsilon) \quad &\vdash \quad (L,\overrightarrow{\gamma}^{n}q\gamma_n,\#,\#)\\
&\vdash^{n-1} (L,q\gamma_n,\#,\#\overleftarrow{\gamma}^{n})\\
&\vdash \quad (q,\gamma_n,\#,\#\overleftarrow{\gamma}^{n})\\
&\vdash \quad (H,\varepsilon,\#,\#\overleftarrow{\gamma}^{n}\gamma_n p) \text{ právě tehdy, když } \delta(q,\gamma_n)=(p,R)\\
&\vdash \quad (F,\varepsilon,\#,\#\overleftarrow{\gamma}^{n+1}p\Delta).
\end{aligned}
$$

Opět si všimněme, že výstup je řetězec jazyka $\{\#\}\Gamma^{*}Q\Gamma^{+}$. Pro vstup z tohoto jazyka zaručeno, že je přeložitelný tehdy a jen tehdy, když je možná akce pohybu hlavy doprava v simulovaném Turingově stroji. Výsledný překlad této komponenty tedy charakterizujeme takto:

$$
\begin{aligned}
T &= \{(\#\overleftarrow{\gamma}^{n}q\gamma_n\overrightarrow{\gamma}^{n},\#\overleftarrow{\gamma}^{n+1}p\gamma_{n+1}\overrightarrow{\gamma}^{n+1}) \mid \overleftarrow{\gamma}^{n}\in\Gamma^{*},q\in Q,\gamma_n\in\Gamma,\overrightarrow{\gamma}^{n}\in\Gamma^{+},\delta(q,\gamma_n)=(p,R)\} \cup\\
&\quad \{(\#\overleftarrow{\gamma}^{n}q\gamma_n,\#\overleftarrow{\gamma}^{n+1}p\Delta) \mid \overleftarrow{\gamma}^{n}\in\Gamma^{*},q\in Q,\gamma_n\in\Gamma,\delta(q,\gamma_n)=(p,R)\},\\
T(P_R^M) &\supseteq T,\\
T &= T(P_R^M)\cap\{(\#\overleftarrow{\gamma}^{n}q\gamma_n\overrightarrow{\gamma}^{n},\gamma) \mid \overleftarrow{\gamma}^{n}\in\Gamma^{*},q\in Q,\gamma_n\in\Gamma,\overrightarrow{\gamma}^{n},\gamma\in(\Gamma\cup\{\#\})^{*}\},\\
\emptyset &= T(P_R^M)\cap\{(w,\gamma) \mid w\in\Sigma^{*},\gamma\in(\Gamma\cup\{\#\})^{*}\}.
\end{aligned}
$$

Algoritmus 4.7

Třetí komponentu pro simulaci posunu hlavy doleva P_L^M vytvoříme takto:

$P_L^M=(Q_L,\Gamma\cup\{\#\},\{\#\},\Gamma\cup\{\#\},\delta_L,S,\#,\{F\})$, kde

$Q_L=\{S,L,F\}\cup\{(a,q)\mid a\in\Gamma,q\in Q\}\cup\Gamma$, $S,L,F\notin Q\cup\Gamma$ a

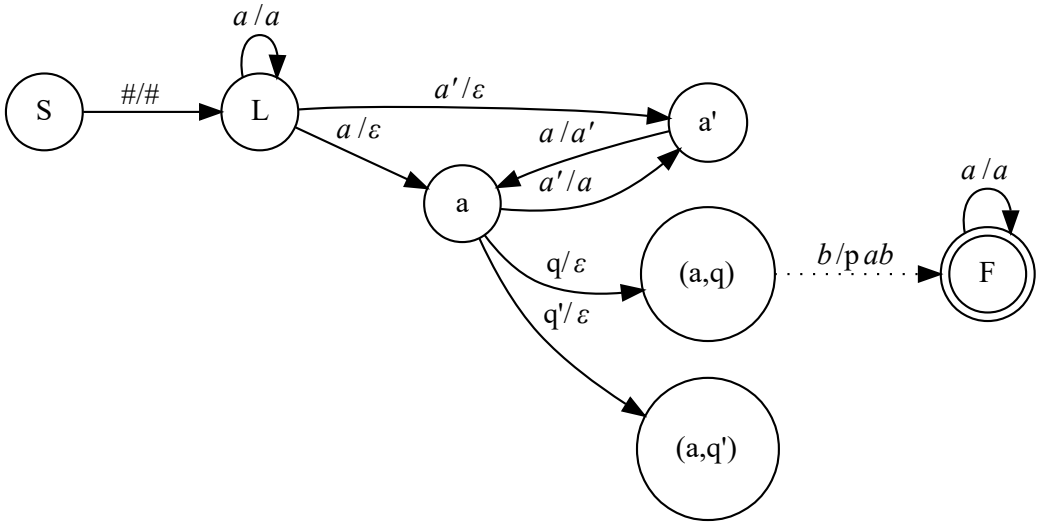
$\delta_L=\{(S,\#,\#,L,\#,\#)\}\cup$

$\{(L,a,\#,a,\#,\varepsilon)\mid a\in\Gamma\}\cup$

$\{(a,b,\#,b,\#,a)\mid a,b\in\Gamma\}\cup\{(a,q,\#,(a,q),\#,\varepsilon)\mid a\in\Gamma,q\in Q\}\cup$

$\{((a,q),b,\#,F,\#,pab)\mid a,b\in\Gamma,q\in Q,\delta(q,b)=(p,L)\}\cup$

$\{(F,a,\#,F,\#,a)\mid a\in\Gamma\}$.



Ilustrace 4.3: δ_L *pro různé* $q\in Q$ *a* $a\in\Gamma$

q zde opět zastupuje jeden z více stavů z Q, pro něhož je definována přechodová funkce $\delta(q,b)=(p,L)$, ale $\delta(q',b')\neq(p',L)$ pro žádné $b'\in\Gamma$ a $p'\in Q$.

Třetí komponenta opět zkontroluje *pojistku* a začne kopírovat symboly nalevo od hlavy. Kopírování zde však probíhá se zpožděním jednoho jednoho kroku přes vnitřní paměť realizovanou stavem. Pro kontrolu správnosti výběru komponenty a zapsání správného následného stavu je nutné mít k dispozici symbol pod novou pozicí hlavy, aktuální stav a symbol pod starou pozicí hlavy, takže pokud můžeme číst jen jeden symbol, zbylé dva si musíme uložit ve stavech. Symbol nalevo od hlavy sice není důležitý pro rozhodování, ale protože ve výstupním řetězci je tento symbol napravo od označení nového stavu, který se dá rozhodnout až po přečtení symbolu pod aktuální hlavou, je nutné si jej pamatovat. Po rozhodnutí a zapsání nového stavu můžeme zapsat symbol pod novou hlavou a symbol pod starou hlavou. Následuje už jen překopírování zbytku vstupu na výstup.

Opět pro libovolný vstup z jazyka Σ^* se komponenta zasekne v počátečním stavu, protože nezačíná *pojistkou*. Pro změnu ale předpokládejme slova $w\in\{\#\}\Gamma^+Q\Gamma^+$, protože nechceme přepadnout přes levý okraj pásky:

$$\begin{aligned}
(S,w,\#,\varepsilon)=(S,\#\overleftarrow{\gamma}^n q\gamma_n\vec{\gamma}^n,\#,\varepsilon) &\vdash (L,\overleftarrow{\gamma}^n q\gamma_n\vec{\gamma}^n,\#,\#)\\
&\vdash (\gamma_1,\gamma_2\dots\gamma_{n-1}q\gamma_n\vec{\gamma}^n,\#,\#)\\
&\vdash (\gamma_2,\gamma_3\dots\gamma_{n-1}q\gamma_n\vec{\gamma}^n,\#,\#\gamma_1)\\
&\vdash^{n-3} (\gamma_{n-1},q\gamma_n\vec{\gamma}^n,\#,\#\gamma_1\dots\gamma_{n-2})\\
&\vdash ((\gamma_{n-1},q),\gamma_n\vec{\gamma}^n,\#,\#\gamma_1\dots\gamma_{n-2})\\
&\vdash (F,\vec{\gamma}^n,\#,\#\overleftarrow{\gamma}^{n-1}p\gamma_{n-1}\gamma_n) \text{ pouze pokud } \delta(q,\gamma_n)=(p,L)\\
&\vdash^{|\vec{\gamma}|} (F,\varepsilon,\#,\#\overleftarrow{\gamma}^{n-1}p\gamma_{n-1}\vec{\gamma}^{n-1}).
\end{aligned}$$

Výstupem je řetězec jazyka $\{\#\}\Gamma^*Q\Gamma^+$. Kdyby $\overleftarrow{\gamma}^n=\varepsilon$ (a tedy $n=0$), komponenta by se zasekla ve stavu L, což odpovídá vyskočení mimo pásku v simulovaném Turingově stroji. Také se kontroluje, zda pro kombinaci q a γ_n je výsledek přechodové funkce ve tvaru (p,L) pro nějaké $p\in Q$. Jestliže není nelze projít do koncového stavu a přeložit vstupní řetězec. Výsledný překlad tedy charakterizujme takto:

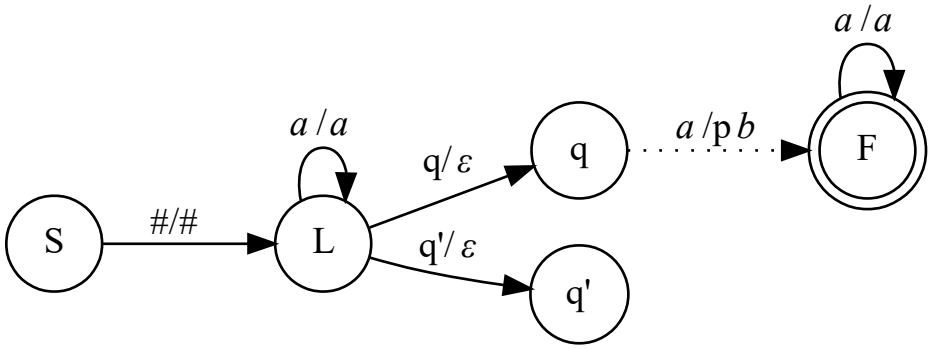
$$\begin{aligned}
T &= \{(\#\overleftarrow{\gamma}^n q\gamma_n\vec{\gamma}^n,\#\overleftarrow{\gamma}^{n-1}p\gamma_{n-1}\overleftarrow{\gamma}^{n-1})\mid\overleftarrow{\gamma}^n\in\Gamma^+,\vec{\gamma}^n\in\Gamma^*,\gamma_n\in\Gamma,q\in Q,\delta(q,\gamma_n)=(p,L)\},\\
T(P_L^M) &\supseteq T,\\
T &= T(P_L^M)\cap\{(\#\overleftarrow{\gamma}^n q\gamma_n\vec{\gamma}^n,\gamma)\mid\overleftarrow{\gamma}^n\in\Gamma^+,\vec{\gamma}^n\in\Gamma^*,\gamma_n\in\Gamma,q\in Q,\gamma\in(\Gamma\cup\{\#\})^*\},\\
\emptyset &= T(P_L^M)\cap\{(w,\gamma)\mid w\in\Sigma^*,\gamma\in(\Gamma\cup\{\#\})^*\},\\
\emptyset &= T(P_L^M)\cap\{(\#q\gamma_n\vec{\gamma}^n,\gamma)\mid q\in Q,\gamma_n\in\Gamma,\vec{\gamma}^n\in\Gamma^+,\gamma\in(\Gamma\cup\{\#\})^*\}.
\end{aligned}$$

Algoritmus 4.8

Čtvrtou komponentu pro simulaci přepisu symbolu P_S^M vytvoříme takto:

$P_S^M=(Q_S,\Gamma\cup\{\#\},\{\#\},\Gamma\cup\{\#\},\delta_S,S,\#,\{F\})$, kde
$Q_S=\{S,L,F\}\cup Q$, $S,L,F\notin Q$ a

$$\begin{aligned}\delta_S = {} & \{(S,\#,\#,L,\#,\#)\} \cup \\ & \{(L,a,\#,L,\#,a) \mid a\in\Gamma\} \cup \{(L,q,\#,q,\#,\varepsilon) \mid q\in Q\} \cup \\ & \{(q,a,\#,F,\#,pb) \mid q\in Q, p\in\Gamma, \delta(q,a)=(p,b)\} \cup \\ & \{(F,a,\#,F,\#,a) \mid a\in\Gamma\}.\end{aligned}$$



Ilustrace 4.4: δ_S *pro různé* $q\in Q$ *a* $a,b\in\Gamma$

q opět zastupuje jeden z více různých stavů z Q. Tentokrát je pro něj definována přechodová funkce $\delta(q,a)=(p,b)$ a pro stav q' opět $\delta(q',a')\neq(p',b')$ pro žádnou volbu $a',b'\in\Gamma$ a $p'\in Q$.

Čtvrtá komponenta nejprve jako vždy zkontroluje *pojistku* a poté zkopíruje symboly nalevo od hlavy. Po přečtení označení stavu si jej zapamatuje a pokud je pro symbol pod hlavou správně definována přechodová funkce, vypíše nový stav a nový symbol pod hlavou. Neobsahuje-li přechodová funkce přepisovací akci pro načtenou kombinaci stavu a symbolu pod hlavou, převodník se zasekne. Naopak po zapsání změněných údajů už jen zkopíruje symboly napravo od hlavy.

Opět řetězce jazyka Σ^* nejsou přijaty, protože neobsahují *pojistku*. Vezměme tedy řetězce jazyka $\{\#\}\Gamma^* Q \Gamma^+$:

$$\begin{aligned}(S,w,\#,\varepsilon)=(S,\#\overleftarrow{\gamma}^n q\gamma_n\vec{\gamma}^n,\#,\varepsilon) \quad &\vdash \quad (L,\overleftarrow{\gamma}^n q\gamma_n\vec{\gamma}^n,\#,\#) \\ &\vdash^{n-1} (L,q\gamma_n\vec{\gamma}^n,\#,\#\overleftarrow{\gamma}^n) \\ &\vdash \quad (q,\gamma_n\vec{\gamma}^n,\#,\#\overleftarrow{\gamma}^n) \\ &\vdash \quad (F,\vec{\gamma}^n,\#,\#\overleftarrow{\gamma}^n p a) \text{ právě tehdy, když } \delta(q,\gamma_n)=(p,a) \text{ pro } a\in\Gamma \\ &\vdash^{|\vec{\gamma}|} (F,\varepsilon,\#,\#\overleftarrow{\gamma}^n p\gamma_n\vec{\gamma}^n).\end{aligned}$$

Výstup opět spadá do jazyka $\{\#\}\Gamma^* Q\Gamma^+$. Přípustné jsou pouze odpovídající výstupy simulovaného Turingova stroje, což zajišťuje podmínka $\delta(q,\gamma_n)=(p,a)$ pro $a\in\Gamma$ u odpovídajícího kroku.

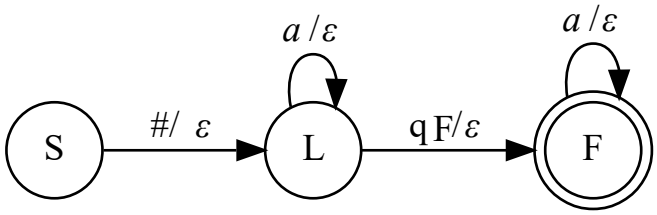
Výsledný překlad tedy obsahuje:

$$\begin{aligned}T &= \{(\#\overleftarrow{\gamma}^n q\gamma_n\vec{\gamma}^n,\#\overleftarrow{\gamma}^n pa\vec{\gamma}^n) \mid \overleftarrow{\gamma}^n,\vec{\gamma}^n\in\Gamma^*, q\in Q, a,\gamma_n\in\Gamma, \delta(q,\gamma_n)=(p,a)\}, \\ T(P_S^M) &\supseteq T, \\ T &= T(P_S^M) \cap \{(\#\overleftarrow{\gamma}^n q\gamma_n\vec{\gamma}^n,\gamma) \mid \overleftarrow{\gamma}^n,\vec{\gamma}^n\in\Gamma^*, q\in Q, a,\gamma_n\in\Gamma, \gamma\in(\Gamma\cup\{\#\})^*\}, \\ \varnothing &= T(P_S^M) \cap \{(w,\gamma) \mid w\in\Sigma^*, \gamma\in(\Gamma\cup\{\#\})^*\}.\end{aligned}$$

Algoritmus 4.9

Pátou komponentu pro ukončení simulace a rozhodnutí o přijetí P_F^M definujeme takto:

$P_F^M=(Q_F,\Gamma\cup\{\#\},\{\#\},\Gamma\cup\{\#\},\delta_F,S,\#,\{F\})$, kde
$Q_F=\{S,L,F\}$ a
$\delta_F=\{(S,\#,\#,L,\#,\varepsilon)\}\ \cup$
$\{(L,a,\#,L,\#,\varepsilon)|a\in\Gamma\}\ \cup\ \{(L,q_F,\#,F,\#,\varepsilon)\}\ \cup$
$\{(F,a,\#,F,\#,\varepsilon)|a\in\Gamma\}$



Ilustrace 4.5: δ_F pro různé $a\in\Gamma$

Po vymazání *pojistky* se začnou mazat i symboly vlevo od hlavy. Pokud je simulovaný Turingův stroj v koncovém stavu, vymaže se celá páska. Lépe řečeno přečte se celý vstup, zkontroluje se zda označení stavu odpovídá koncovému stavu stroje M a vypíše se prázdný řetězec. Jiný stav než koncový zapříčiní zaseknutí převodníku ve stavu L a nedokončený překlad.

Při pokusu o přijetí slova z jazyka Σ^* se komponenta opět zasekne na *pojistce* $\#$. Sledujme překlad při vstupu $w\in\{\#\}\Gamma^*Q\Gamma^+$:

$$\begin{aligned}(S,w,\#,\varepsilon)=(S,\#\overleftarrow{\gamma}^n q\gamma_n\vec{\gamma}^n,\#,\varepsilon) &\vdash (L,\overleftarrow{\gamma}^n q\gamma_n\vec{\gamma}^n,\#,\varepsilon)\\ &\vdash^{n-1} (L,q\gamma_n\vec{\gamma}^n,\#,\varepsilon)\\ &\vdash (F,\gamma_n\vec{\gamma}^n,\#,\varepsilon)\text{ právě tehdy, když } q=q_F\\ &\vdash (F,\vec{\gamma}^n,\#,\varepsilon)\\ &\vdash^{|\vec{\gamma}|} (F,\varepsilon,\#,\varepsilon).\end{aligned}$$

Výstupem je tentokrát pouze prázdný řetězec, kterého se dosáhne, pouze pokud stav ve vstupním řetězci je právě koncový stav simulovaného Turingova stroje. Výsledný překlad tedy charakterizujeme:

$$\begin{aligned}T &= \{(\#\overleftarrow{\gamma}^n q_F\gamma_n\vec{\gamma}^n,\varepsilon)|\overleftarrow{\gamma}^n,\vec{\gamma}^n\in\Gamma^*,\gamma_n\in\Gamma\},\\ T(P_F^M) &\supseteq T,\\ T &= T(P_F^M)\ \cap\ \{(\#\overleftarrow{\gamma}^n q\gamma_n\vec{\gamma}^n,\gamma)|\overleftarrow{\gamma}^n,\vec{\gamma}^n\in\Gamma^*,q\in Q,\gamma_n\in\Gamma,\gamma\in(\Gamma\cup\{\#\})^*\},\\ \varnothing &= T(P_F^M)\ \cap\ \{(w,\gamma)|w\in\Sigma^*,\gamma\in(\Gamma\cup\{\#\})^*\}.\end{aligned}$$

Tvrzení 4.10

Nejprve ukážeme, že zobrazení Φ z definice 4.4 mapuje konfigurace Turingova stroje na řetězce, které lze přeložit v jednom průběhu komponenty v systému převodníků. přesněji:

$$(q,\gamma,n) \vdash (p,\mu,m) \Leftrightarrow$$
$$(i,q_{0,i},\Phi(q,\gamma,n),Z_{0,i},\varepsilon) = (i,q_{0,i},\#\overleftarrow{\gamma}^{n} q\,\gamma_n \overrightarrow{\gamma}^{n},Z_{0,i},\varepsilon) \vdash^{P}$$
$$(i,f,\varepsilon,\alpha,\#\overleftarrow{\mu}^{m} p\,\mu_m \overrightarrow{\mu}^{m}) = (i,f,\varepsilon,\alpha,\Phi(p,\mu,m)),$$

pro všechna $q,p \in Q$, $\gamma,\mu \in \Gamma^{*}$, $0 \le n,m$, existuje $1 \le i \le 5$, $\alpha \in \Gamma_{\Pi^M}^{*}$ a $f \in F_i$.

Důkaz

Krok v Turingově stroji je lze provést třemi způsoby. Každý z těchto způsobů je realizován pomocí jedné z komponent P_R^M, P_L^M, P_S^M systému převodníků, přičemž protože Turingovy stroje jsou deterministické, vždy je možné použít nejvýše jednu z těchto komponent a dojít až do jejího koncového stavu. První komponenta P_1^M přijímá vstupní řetězce a upravuje je podle funkce Φ pro další zpracování a poslední komponenta P_F^M vyžaduje řetězec s koncovým stavem, ze kterého v Turingově stroji nelze provést další krok. Tyto dvě komponenty nás tedy nemusí zajímat.

Zaprvé posun doprava:

$$(q,\gamma,n) \vdash (p,\gamma,n+1), \text{ jestliže } \delta(q,\gamma_n) = (p,R).$$

Protože $\delta(q,\gamma_n) = (p,R)$, pro daný vstup nejsou překlady komponent P_L^M a P_S^M definované. Využijeme tedy komponentu P_R^M:

$$(2,S,\Phi(q,\gamma,n),\#,\varepsilon) = (2,S,\#\overleftarrow{\gamma}^{n} q\,\gamma_n \overrightarrow{\gamma}^{n},\#,\varepsilon) \vdash^{P}$$
$$(2,F,\varepsilon,\#,\#\overleftarrow{\gamma}^{n+1} p\,\gamma_{n+1} \overrightarrow{\gamma}^{n+1}) = (2,F,\varepsilon,\#,\Phi(q,\gamma,n+1)),$$

protože $\delta(q,\gamma_n) = (p,R)$, a tedy $T(P_R^M)(\#\overleftarrow{\gamma}^{n} q\,\gamma_n \overrightarrow{\gamma}^{n}) = \#\overleftarrow{\gamma}^{n+1} p\,\gamma_{n+1} \overrightarrow{\gamma}^{n+1}$.

Zadruhé posun doleva:

$$(q,\gamma,n) \vdash (p,\gamma,n-1), \text{ jestliže } \delta(q,\gamma_n) = (p,L) \ a \ 1 \le n.$$

Protože $\delta(q,\gamma_n) = (p,L)$, pro daný vstup nejsou překlady komponent P_R^M a P_S^M definované. Využijeme tedy komponentu P_L^M:

$$(3,S,\Phi(q,\gamma,n),\#,\varepsilon) = (3,S,\#\overleftarrow{\gamma}^{n} q\,\gamma_n \overrightarrow{\gamma}^{n},\#,\varepsilon) \vdash^{P}$$
$$(3,F,\varepsilon,\#,\#\overleftarrow{\gamma}^{n-1} p\,\gamma_{n-1} \overrightarrow{\gamma}^{n-1}) = (3,F,\varepsilon,\#,\Phi(q,\gamma,n-1)),$$

protože $\delta(q,\gamma_n) = (p,L)$ a $1 \le n$, a tedy $T(P_L^M)(\#\overleftarrow{\gamma}^{n} q\,\gamma_n \overrightarrow{\gamma}^{n}) = \#\overleftarrow{\gamma}^{n-1} p\,\gamma_{n-1} \overrightarrow{\gamma}^{n-1}$.

Zatřetí přepis symbolu pod hlavou:

$$(q,\overleftarrow{\gamma}^{n}\gamma_n\overrightarrow{\gamma}^{n},n) \vdash (p,\overleftarrow{\gamma}^{n} b\,\overrightarrow{\gamma}^{n},n), \text{ jestliže } \delta(q,\gamma_n) = (p,b) \ a \ b \in \Gamma.$$

Protože $\delta(q,\gamma_n) \notin \{(p,d) \mid p \in \Gamma, d \in \{L,R\}\}$, pro daný vstup nejsou překlady komponent P_R^M a P_L^M definované. Využijeme tedy komponentu P_S^M:

$$(4,S,\Phi(q,\gamma,n),\#,\varepsilon)=(4,S,\#\overleftarrow{\gamma}^{n}q\gamma_n\vec{\gamma}^{n},\#,\varepsilon)\vdash^{P}$$
$$(4,F,\varepsilon,\#,\#\overleftarrow{\gamma}^{n}pb\vec{\gamma}^{n})=(4,F,\varepsilon,\#,\Phi(q,\overleftarrow{\gamma}^{n}b\vec{\gamma}^{n},n)),$$

protože $\delta(q,\gamma_n)=(p,L)$ pro $b\in\Gamma$, a tedy $T(P_S^M)(\#\overleftarrow{\gamma}^{n}q\gamma_n\vec{\gamma}^{n})=\#\overleftarrow{\gamma}^{n}pb\vec{\gamma}^{n}$.

Tvrzení 4.10 tedy platí.

Tvrzení 4.11

Nyní dokážeme, že každá konfigurace Turingova stoje M dosažitelná z počáteční konfigurace $(q_0,\Delta w,0)$ a přepsaná pomocí funkce Φ na řetězec je vstupem libovolné komponenty systému převodníků Π^M s některou počáteční konfigurací $(i,q_{0,i},w,Z_{0,i},\varepsilon)$ pro $1\le i\le 5$. Formálně

$$(q_0,\Delta w,0)\vdash^{*}(p,u,n)\Leftrightarrow(i,q_{0,i},w,Z_{0,i},\varepsilon)\vdash^{*}(j,q_{0,j}\Phi(p,u,n),Z_{0,j},\varepsilon),$$

kde $w\in\Sigma^{*}$, $1\le j\le 5$ pro nějaké $1\le i\le 5$.

Důkaz

Budeme postupovat indukcí vzhledem k počtu kroků v Turingově stroji. Indukční hypotéza zní

$$(q_0,\Delta w,0)\vdash^{k}(q,w',n)\Leftrightarrow(i,q_{0,i},w,Z_{0,i},\varepsilon)(\vdash^{S}\circ\vdash^{P})^{k+1}(j,q_{0,j}\Phi(q,w',n),Z_{0,j},\varepsilon),$$

kde $w\in\Sigma^{*}$ pro nějaké $1\le i,j\le 5$.

Pro $k=0$ platí

$$\begin{aligned}(q_0,\Delta w,0) &\vdash^{0} (q_0,\Delta w,0),\\ (1,S,w,\#,\varepsilon) &\vdash^{P} (1,F,\varepsilon,\#,\#q_0\Delta w\Delta)\\ &\vdash^{S} (j,q_{0,i},\#q_0\Delta w,\#,\varepsilon)\\ &= (j,q_{0,i},\Phi(q_0,\Delta w,0),\#,\varepsilon).\end{aligned}$$

Libovolná jiná komponenta něž první (P_1^M) nedefinuje žádný překlad pro $w\in\Sigma^{*}$. Báze indukce je tedy dokázána.

Nyní předpokládejme, že indukční hypotéza platí pro všechna $l<k$ a dokažme ji pro $l=k$

$$(q_0,\Delta w,0)\vdash^{l}(p,u,n)\Leftrightarrow(q_0,\Delta w,0)\vdash^{l-1}(q,v,m)\vdash(p,u,n),$$

kde $u,v\in\Sigma^{*}$, $0\le n,m$ a $p,q\in Q$.

Z indukční hypotézy plyne

$$(q_0,\Delta w,0)\ \vdash^{l-1}\ (q,v,m)\ \Leftrightarrow\ (i,q_{0,i},w,Z_{0,i},\varepsilon)\ (\vdash^{S}\circ\vdash^{P})^{l}\ (i',q_{0,i'},\Phi(q,v,m),Z_{0,i'},\varepsilon)$$

pro $1\le i,i'\le 5$ a z tvrzení 4.10 plyne

$$(q,v,m)\vdash(p,u,n)\ \Leftrightarrow\ (i',q_{0,i'},\Phi(q,v,m),Z_{0,i'},\varepsilon)\vdash^{P}(j',q_{f,j'},\varepsilon,\alpha,\Phi(p,u,n))$$

pro $1\le j'\le 5$, $q_{f,j'}\in F_{j'}$ a $\alpha\in\Gamma_{\Pi^M}^{*}$.

Celkem tedy

$$\begin{aligned}(q_{0,}\Delta w,0)\vdash^{l}(p,u,n) \quad &\Leftrightarrow \quad (i,q_{0,i},w,Z_{0,i},\varepsilon)\\ &(\vdash^{S}\circ\vdash^{P})^{l} \quad (i',q_{0,i'},\Phi(q,v,m),Z_{0,i'},\varepsilon)\\ &\vdash^{P} \quad (j',q_{f,j'},\varepsilon,\alpha,\Phi(p,u,n))\\ &\vdash^{S} \quad (j,q_{0,j},\Phi(p,u,n),Z_{0,j},\varepsilon),\end{aligned}$$

neboli

$$(q_{0,}\Delta w,0)\vdash^{l}(p,u,n) \;\Leftrightarrow\; (i,q_{0,i},w,Z_{0,i},\varepsilon)(\vdash^{S}\circ\vdash^{P})^{l+1}(j,q_{0,j},\Phi(p,u,n),Z_{0,j},\varepsilon)\,.$$

Tvrzení 4.11 tedy je dokázané.

Věta 4.12

Pro daný Turingův stroj $M=(Q,\Sigma,\Gamma,\delta,q_0,q_F)$ sestrojíme systém převodníků Π^M z komponent vytvořených podle algoritmů 4.5 až 4.9

$$\Pi^M=(\Sigma,\{\#\},\Delta\cup\{\#\},\emptyset,P_1^M,P_R^M,P_L^M,P_S^M,P_F^M)\,.$$

Vstupní jazyk tohoto systému $L_I(\Pi^M)$ a jazyk přijímaný Turingovým strojem $L(M)$ jsou totožné.

Důkaz

Jazyk přijímaný Turingovým strojem M je definován jako

$$L(M)=\{w\,|\,w\in\Sigma^*,\exists\gamma\in\Gamma^*,0\le n:(q_{0,}\Delta w,0)\vdash^{*}(q_F,\gamma,n)\}\,.$$

Z předchozího tvrzení plyne pro $w\in L(M)$

$$(q_0,\Delta w,0)\vdash^{*}(q_F,\gamma,n) \;\Leftrightarrow\; (i,q_{0,i},w,Z_{0,i},\varepsilon)\vdash^{*}(j,q_{0,j},\Phi(q_F,\gamma,n),Z_{0,j},\varepsilon)\,.$$

Pokud vybereme $j=5$, tedy poslední komponentu P_F^M, můžeme provést průchod komponenty

$$(5,S,\Phi(q_F,\gamma,n),\#,\varepsilon) \;=\; (5,S,\#\overleftarrow{\gamma}^{n}q_F\gamma_n\vec{\gamma}^{n},\#,\varepsilon)\vdash^{P}(5,F,\varepsilon,\#,\varepsilon)\,.$$

Jiné komponenty nemají definovaný pro tento řetězec překlad. Tato komponenta jako jediná má na výstupu řetězce z jazyka $\emptyset^*$ – prázdné řetězce. Můžeme tedy určit překlad celkového systému převodníků Π^M

$$\begin{aligned}T(\Pi^M) &= \{(x,u)\,|\,x\in\Sigma^*,u\in\emptyset^*,\exists 1\le i,j\le 5,q\in F_j,\alpha\in\{\#\}^*:(i,q_{0,i},x,Z_{0,i},\varepsilon)\vdash^{*}(j,q,\varepsilon,\alpha,u)\}\\ &= \{(x,\varepsilon)\,|\,x\in\Sigma^*,\exists 1\le i\le 5,\alpha\in\{\#\}^*:(i,q_{0,i},x,Z_{0,i},\varepsilon)\vdash^{*}(5,F,\varepsilon,\alpha,\varepsilon)\}\\ &= \{(x,\varepsilon)\,|\,x\in L(M)\}.\end{aligned}$$

a vstupní jazyk je tedy

$$L_I(\Pi^M)=L(M)\,.$$

Samozřejmě, že i po vygenerování prázdného řetězce na výstupu poslední komponenty výpočet nemusí skončit a překlad se může přepnout na další komponentu. Jediná komponenta, která akceptuje prázdný řetězec, je však ta první P_1^M. Protože prázdný řetězec patří do jazyka Σ^*, k přijatelnému řetězci se dopracujeme pouze pokud $\varepsilon \in L(M)$. V takovém případě, je ale jen potvrzen překlad (w, ε) druhou možnou sekvencí kroků a vstupní jazyk se nemění.

Za povšimnutí rozhodně stojí fakt, že systém převodníků pro tento důkaz nevyužívá svůj zásobník. Kdybychom definovali systém konečných převodníků namísto zásobníkových, dospěli bychom ke stejnému závěru.

Můžeme tedy formulovat následující větu:

Věta 4.13

Rodina jazyků přijímaných Turingovými stroji je totožná s rodinou vstupních jazyků systémů převodníků.

Důkaz

Pro každý Turingův stroj existuje podle věty 4.13 systém převodníků, jehož vstupní jazyk je totožný s jazykem přijímaným Turingovým strojem. Naopak podle *Church-Turingovy* téze systémy převodníků jakožto matematické stroje nemohou být silnější než Turingovy stroje.

4.3 Překlad Turingova stroje

Na začátku této kapitoly jsme nastínili možnost definice překladu vedle klasické definice jazyka Turingova stroje. Velkou výhodou je, že konstrukce i sémantika přechodů může zůstat stejná, pouze se dodefinuje, co chápeme výstupem pro vstupní slovo. Jednou z možností je za výstup pokládat celou pásku, což je ovšem nevhodné, protože páska je neomezená a z definice je to nekonečný řetězec obsahující nekonečný sufix prázdných symbolů. Vhodnější je vybírat různé konečné prefixy pásky. V definici konfigurace pásky jsme využili nejdelšího prefixu pásky, který nekončí prázdným symbolem. Tato možnost je jistě validní, avšak výstup by mohl obsahovat prázdné symboly, které by překážely, kdybychom chtěli například spojit dva překlady za sebe. Už jen z důvodu kompatibility vstupu a výstupu by tedy bylo lepší vzít první řetězec na pásce, který neobsahuje prázdné symboly.

Definice 4.14

Mějme Turingův stroj $M = (Q, \Sigma, \Gamma, \delta, q_0, q_F)$ a dva řetězce $x, y \in \Sigma^*$. Řetězec y je *výstupem* pro vstup x, jestliže v M

$$(q_0, \Delta x, 0) \vdash^* (q_F, \Delta y \Delta \gamma, n),$$

pro nějaké $\gamma \in \Gamma^*$ a $0 \leq n$.

Překlad pro Turingův stroj M značený $T(M)$ je definován jako relace vstupů a odpovídajících výstupů, tedy

$$T(M) = \{(x, y) \mid x, y \in \Sigma^*, \exists \gamma \in \Gamma^*, 0 \leq n : (q_0, \Delta x, 0) \vdash^* (q_F, \Delta y \Delta \gamma, n)\}.$$

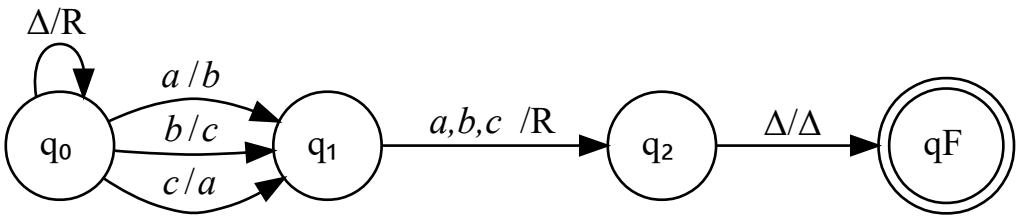
Zkusme nyní vytvořit jednoduchý příklad takového překladu a zkusit ho porovnat s mírně upravenou verzí systému převodníků z věty 4.12.

Příklad 4.15

Mějme Turingův stroj $M=(\{q_0,q_1,q_2,q_F\},\{a,b,c\},\{a,b,c,\Delta\},\delta,q_0,q_F)$, kde δ je nejmenší zobrazení takové, že

$$\begin{array}{llllll}\delta(q_0,\Delta) & = & (q_0,R), & \delta(q_1,a) & = & (q_2,R), \qquad \delta(q_2,\Delta) = (q_F,\Delta),\\ \delta(q_0,a) & = & (q_1,b), & \delta(q_1,b) & = & (q_2,R),\\ \delta(q_0,b) & = & (q_1,c), & \delta(q_1,c) & = & (q_2,R),\\ \delta(q_0,c) & = & (q_1,a). & & & \end{array}$$

Vyjádřeno graficky:



Turingův stroj nejprve přejde nad první symbol, změní ho a zkontroluje zda druhý symbol vstupu je prázdný. Pokud je na vstupu prázdný řetězec, tak se stroj nikdy nezastaví a bude neustále posouvat hlavu doprava. Při řetězcích délky dva a více se stroj zasekne ve stavu q_2. Přijímaným jazykem tedy je jazyk řetězců délky jedna – $\{a,b,c\}$.

Překlad je podobně jednoduchý. Každému řetězci je přiřazen některý jiný řetězec. Konkrétně $T(M)=\{(a,b),(b,c),(c,a)\}$.

Nyní vytvořme systém převodníků $\hat{\Pi}^M$, který se od systému Π^M liší v páté komponentě. Neformálně, tato nová komponenta namísto smazání vstupního řetězce a kontroly koncového stavu, překopíruje první řetězec neprázdných symbolů na výstup a zkontroluje koncový stav.

Mohli bychom předpokládat, že překlad generovaný systémem $\hat{\Pi}^M$ bude stejný jako překlad Turingova stroje M. Bohužel tomu tak není, protože po odsimulování celého Turingova stroje může překlad pokračovat první komponentou, která nemá šanci poznat, že její vstup je výstupem páté komponenty. Ve výsledku tedy přeložíme a na b, ale protože b můžeme přeložit na c, tak při překladu a můžeme pokračovat a přeložit ho i na c.

Nabízí se tedy otázka: jak upravit algoritmus pro vytváření ekvivalentního systému převodníků pro překlady? Odpověď však zní, že takový algoritmus neexistuje.

Věta 4.16

Překlad $\{(a,b),(b,c),(c,a)\}$ nelze definovat žádným z námi zavedených systémů převodníků.

Důkaz

Předpokládejme, že takový systém existuje a označme ho $\Pi_P=(\Sigma,\Gamma,K,\Delta,P_1,\ldots P_n)$. Jeho překlad je definován takto:

$$T(\Pi_P) = \{(a,b),(b,c),(c,a)\}$$
$$= \{(x,u) \mid x \in \Sigma^*, u \in \Delta^*, \exists 1 \le i, j \le n, q \in F_j, \alpha \in \Gamma^* : (i, q_{0,i}, x, Z_{0,i}, \varepsilon) \vdash^* (j, q, \varepsilon, \alpha, u)\}.$$

Můžeme tedy tvrdit, že existují tyto sekvence kroků pro $1 \le i_a, i_b, i_c, j_a, j_b, j_c \le n$, $q_a \in F_{j_a}$, $q_b \in F_{j_b}$, $q_c \in F_{j_c}$, $\alpha_a, \alpha_b, \alpha_c \in \Gamma^*$:

$$(i_a, q_{0,i_a}, a, Z_{0,i_a}, \varepsilon) \vdash^* (j_a, q_a, \varepsilon, \alpha_a, b),$$
$$(i_b, q_{0,i_b}, b, Z_{0,i_b}, \varepsilon) \vdash^* (j_b, q_b, \varepsilon, \alpha_b, c),$$
$$(i_c, q_{0,i_c}, c, Z_{0,i_c}, \varepsilon) \vdash^* (j_c, q_c, \varepsilon, \alpha_c, a).$$

Musí tedy existovat například i sekvence

$$(i_a, q_{0,i_a}, a, Z_{0,i_a}, \varepsilon) \vdash^* (j_a, q_a, \varepsilon, \alpha_a, b) \vdash^P (i_b, q_{0,i_b}, b, Z_{0,i_b}, \varepsilon) \vdash^* (j_b, q_b, \varepsilon, \alpha_b, c),$$

takže musí být možný překlad (a,c), který však není obsažen v $T(\Pi_P)$. Narazili jsme na spor, takže předpoklad, že existuje systém Π_P byl mylný. Tímto jsme dokázali, že žádný systém převodníků nedokáže definovat překlad $\{(a,b),(b,c),(c,a)\}$.

Věta 4.17

Turingovy stroje jsou silnější vzhledem k definovatelným překladům.

Důkaz

Plyne z příkladu 4.15 a věty 4.16.

5 Demonstrační skript

V rámci praktické demonstrace dosažených závěrů jsem se rozhodl implementovat systémy převodníků v jazyce Python 3.4. Jejich nejdůležitější funkcí je definice překladu, který ovšem nelze odvodit algoritmicky už jen proto, že bychom jinak mohli rozhodnout problémy Turingova stroje. Ani problém členství určitého slova v jazyce přijímaném Turingovým strojem není rozhodnutelný ([4], [5]). Avšak výpočetní procedura, která určuje odpovídající překlad k danému slovu, o kterém předpokládáme, že nějaký překlad má, je jednoduchá. Začneme v počáteční konfiguraci a podle definice kroku 3.3 pomocí přechodové relace zjišťujeme všechny dosažitelné konfigurace. Pro každou z nich testujeme, zda není (podle definice 3.4) koncovou konfigurací pro určitý prvek překladu. Pokud nemáme jistotu členství daného slova ve vstupním jazyce, tato procedura může skončit se záporným výsledkem, ale také skončit nemusí. Každopádně je tento problém vhodný pro demonstraci systémů převodníků a následně i algoritmů předložených v této práci.

Hlavním cílem demonstračního skriptu je ukázat aplikaci algoritmu pro redukci počtu komponent a algoritmu pro vytvoření systému převodníků pro Turingův stroj. Výsledné konstrukce pak můžeme porovnávat s jejich vstupy právě pomocí výpočetní procedury pro určení výstupu překladu, případně procedury, jež se pokusí ověřit náležitost vstupu do vstupního jazyka.

Pro demonstraci jsem tedy implementoval načítání a zápis systémů převodníků z, respektive do jeho textové reprezentace, nezaručené procedury pro testování prvku vstupního jazyka a výstupu překladu pro dané slovo a konstrukce systému s jednou komponentou a systému ekvivalentního zadanému Turingovu stroji. Kromě toho je pro porovnání výsledku poslední konstrukce vhodný simulátor Turingova stroje, který jsem rovněž implementoval i s načítáním a zápisem stroje.

V této kapitole představím možné režimy tohoto skriptu, formáty, které používá, a nakonec vnitřní principy a zajímavé problémy, které se při implementaci vyskytly.

5.1 Režimy

Samotný skript je konzolová aplikace, která ze souboru načítá popis systému převodníků, případně Turingova stroje, a opět do souboru vypisuje výsledek, ať je to prvek překladu nebo výsledek konstrukce. Existují čtyři režimy činnosti – nepočítáme-li vypsání nápovědy nebo přečteného systému, respektive stroje – testování přijmu řetězce, hledání výstupu překladu, redukce počtu komponent a konstrukce systému pro Turingův stroj.

5.1.1 Společné volby

Některé přepínače ovlivňují každý režim, například určení vstupního a výstupního souboru. Bez změny pomocí přepínačů `--input` a `--output` je využíván standardní vstup a výstup.
Další společný přepínač je `--format`, který určuje formát výstupu jak je definováno dále v podkapitole 5.2.6.

Vstup je ovlivněn přepínačem `--strict`, který způsobí, že žádná sémantická chyba není automaticky opravena. Mezi tyto opravitelné chyby patří využívání symbolů mimo definované abecedy, přechody obsahující neznámé stavy nebo třeba absence prázdného symbolu v páskové abecedě Turingova stroje. Všechny tyto chyby mají společné to, že je lze jednoduše opravit úpravou

patřičných abeced či množin. Volba striktního načítání umožňuje neopravovat tyto chyby a naopak je důsledně kontrolovat.

Posledním společným parametrem je přepínač `--turing`, který rozhoduje, jestli na vstupu je systém převodníků nebo právě Turingův stroj. U režimu redukce a konverze však nehraje roli.

5.1.2 Režim přijímání

Tento režim je volen přepínačem `-a`, respektive `--accepts` a vyžaduje na vstupu systém převodníků nebo Turingův stroj (podle přepínače `--turing`). Dalším povinným vstupem je samotné slovo, které se má testovat. Na výstup se zapíše pouze `1` v případě, že byla nalezena dosažitelná konfigurace vstupního, respektive přijímaného jazyka, a `0` v případě, že žádná taková konfigurace neexistuje.

Dalšími nepovinnými parametry jsou `--time` a `--sequence`. Protože neexistuje záruka, že procedura skončí, volba `--time` určuje v sekundách, kolik času je věnováno hledání. Pokud hledání skončí předčasně, není na výstup vypsáno nic. Volba `--sequence` pomáhá při rekonstrukci sekvence kroků, která vedla k přijetí. Kromě `0` nebo `1` vypíše i sekvenci komponent, respektive sekvenci akcí u Turingova stroje.

5.1.3 Režim překladu

Režim překladu je velmi podobný režimu přijímání, i když v principu režim přijímání využívá režim překladu. Vybírá se přepínačem `-t`, respektive `--translate` a další parametry má stejné jako předchozí režim, tedy `--turing`, `--time` a `--sequence`. Jedinou odlišností je výstup a možnost přepínače `-n` pro systém převodníků.

Na výstupu produkuje prvek překladu ve formátu `x -> u`, kde `x` je zadané a `u` je nalezené slovo. Pro Turingův stroj je díky determinismu takový výsledek nejvýše jeden, ale u systému převodníků může být takovýchto překladů více. Standardně se hledá pouze jeden, ale právě pomocí přepínače `-n` lze nastavit nejvýše kolik různých výsledků se má v časovém okně hledat.

5.1.4 Režim redukce

Další režim slouží pouze pro systémy převodníků. Pomocí přepínače `-r`, respektive `--reduce` je vytvořen ze zadaného systému nový jednokomponentní systém podle algoritmu 3.5. Tento režim nemá kromě společných parametrů žádné jiné.

5.1.5 Režim konverze

Poslední režim vytváří na základě Turingova stroje ekvivalentní systém převodníků podle věty 4.12. Aktivuje se pomocí přepínače `-c`, respektive `--convert`. Žádné další parametry kromě společných nemá.

5.2 Formáty

Systémy převodníků a Turingovy stroje vyžadují zápis čtyř druhů prvků a jejich množin. Jsou to stavy, symboly, řetězce a přechody. Systémy navíc tyto množiny a prvky musejí svazovat do komponent. Matematický zápis této struktury je sice možný, ale bez vhodného odsazování je nepřehledný. Navíc je poziční, což je velká nevýhoda pro ruční zadávání. Z těchto důvodů, jsem

vytvořil a implementoval jednoduchý formát založený na položkách ve tvaru `návěští: hodnota₁, hodnota₂, … , hodnotaₙ;`. Tedy pojmenování a seznam hodnot oddělených čárkou a ukončených středníkem Bílé znaky slouží pouze pro zvýšení přehlednosti. Tento systém je snadno čitelný a díky pojmenovaným návěštím je pořadí jednotlivých položek irelevantní.

Jednotlivé položky mohou být přiřazeny do komponent pomocí definice sekcí návěštími ve tvaru `[sekce]`. Takto je možné komponenty i pojmenovat. První sekce je bez označení a implicitně definuje celkový systém nebo Turingův stroj.

5.2.1 Symboly

Každá abeceda se skládá ze symbolů. Z těchto symbolů budeme později vytvářet řetězce, a proto je důležité je omezit tak, že výsledné řetězce budou jednoznačně rozdělitelné zpátky na symboly pouze z jejich textové reprezentace. Nejjednodušeji tohoto můžeme dosáhnout, pokud povolíme pouze symboly tvořené jedním znakem. Bohužel například algoritmus pro konverzi vyžaduje v komunikaci i stavy původního Turingova stroje a ty nechceme omezovat na jediný znak. Proto zavedeme speciální formát `<text>`, kde `text` je libovolný neprázdný řetězec, jež ovšem neobsahuje další úhlové závorky. Tento formát je jednoduše zpracovatelný na úrovni regulárních výrazů, což velmi usnadňuje implementaci. Symboly jsou tedy libovolné znaky a úhlovými závorkami ohraničené řetězce bez dalších úhlových závorek.

Zajímavým problémem při implementaci bylo právě omezení na absenci úhlových závorek i při konstrukcích, které to předpokládají. Například při redukci zkonvertovaného Turingova stroje druhý algoritmus předepisuje vytváření složených stavů z již složených stavů. Například ze stavu `<q, a>` v komponentně `i` se má vytvořit stav `<<q, a>, i>`. Toto jsem vyřešil přeznačováním vnitřních úhlových závorek na obyčejné, takže místo stavu `<<q, a>, i>` je využit stav `<(q, a), i>`. Hierarchie zůstala zachována a omezení nebylo porušeno.

V případě Turingova stroje jsou vyhrazeny symboly `R`, `L` a `_` pro akci posunu hlavy doprava, respektive doleva a pro prázdný symbol. Ten může být zapsán také v Unicode jako `\u0394` (Δ).

5.2.2 Řetězce

Jak bylo zmíněno výše, řetězce jsou jednoznačně rozdělitelné sekvence symbolů. Mohou se tedy střídat dlouhé symboly s jednoznakovými. Prázdný řetězec má zvláštní postavení, protože má dvojí reprezentaci. Přirozeně je určen absencí symbolů, ale také pomocí Unicode `\u03B5` (ϵ).

5.2.3 Stavy

Pro identifikaci stavů jsem zvolil řetězce identifikátorů, tedy řetězce začínající velkým či malým písmenem a pokračující libovolnou posloupností písmen, číslic a podtržítek. Pro reprezentaci složených stavů a stavů představující symbol na pásce je opět využito úhlových závorek.

5.2.4 Přechody systémů převodníků

Relace δ pro převodník je definována jako relace mezi $Q\times(\Sigma\cup\{\varepsilon\})\times\Gamma$ a $Q\times\Gamma^*\times\Delta^*$, tedy šestice `(<q>, <i>, <Z>, <p>, <Zs>, <os>)`. Tento zápis není příliš přehledný, a proto jsem raději zvolil formát `<i> <q> -> <p> <os> [<Z> -> <Zs>]`.

Práce se zásobníkem je oddělena a nemůže se plést se vstupy a výstupy. Přechod mezi stavy je zdůrazněn jejich blízkostí v řetězci. Jediné povinné bílé znaky jsou mezery mezi symbolem a stavem, aby bylo možné je jednoznačně rozdělit.

`<i>`, `<os>` a `<Zs>` mohou být prázdné řetězce, a tedy, jak bylo řečeno výše, mohou být vynechány s výsledkem, který vypadá až takto: `<q> -> <p> [<Z> ->]`, nebo mohou být reprezentovány symboly malého epsilon, tedy takto: `ϵ <q> -> <p> ϵ [<Z> -> ϵ]`.

5.2.5 Přechody Turingových strojů

Turingovy stroje definují přechody deterministicky pomocí zobrazení z $(Q\setminus\{qF\})\times\Gamma$ do $Q\times(\Gamma\cup\{L,R\})$. Místo čtveřice `(<q>, <i>, <p>, <a>)` píšeme přehledněji `<q> <i> -> <p> <a>`. Opět mezi stavem a symbolem, respektive akcí je nutný alespoň jeden bílý znak.

5.2.6 Návěští

Definovali jsme jednotlivé prvky a obecnou strukturu položky s několika hodnotami pro zápis množiny. Nyní musíme ještě definovat, která návěští jsou rozpoznávána a které položky jsou povinné. Pro systém převodníků jsou povinné počáteční stavy a počáteční vrcholy zásobníků všech komponent. Pro Turingův stroj je povinný pouze počáteční a koncový stav. Nedefinovaná položka se bere jako prázdná množina. Vícekrát definovaná položka je syntaktická chyba.

Jsou definovány čtyři formáty návěští: matematický popis využívající ASCII znaky, matematický popis využívající řecká písmena z Unicode, český popis a anglický popis. Na vstupu není předpokládán žádný z nich – odpovídající pojmenování návěští jsou chápána jako synonyma a mohou se míchat, i když to není vhodné. Výstup je pak ovlivněn přepínačem `--format`, přičemž implicitní volba je `words` ASCII opisu řeckých písmen. Tabulky 5.1 a 5.2 uvádí očekávané pojmenování návěští pro systémy převodníků, respektive Turingovy stroje.

words	cs	en	unicode
Sigma	Vstupní abeceda	Input	\u03A3 (Σ)
Gamma	Zásobníková abeceda	Pushdown	\u0393 (Γ)
Kappa	Komunikační abeceda	Communication	\u039A (K)
Delta	Výstupní abeceda	Output	\u0394 (Δ)
Q	Stavy	States	Q
q0	Počáteční stav	Start state	q\u2080 (q₀)
F	Koncové stavy	Final states	F
Z0	Počáteční vrchol zásobníku	Start pushdown top	Z\u2080 (Z₀)
delta	Přechody	Transitions	\u03B4 (δ)

Tabulka 5.1: Pojmenování návěští položek systémů převodníků podle formátu

words	cs	en	unicode
Sigma	Vstupní abeceda	Input	\u03A3 (Σ)
Gamma	Pásková abeceda	Tape	\u0393 (Γ)
Q	Stavy	States	Q
q0	Počáteční stav	Start state	q\u2080 (q₀)
qF	Koncový stav	Final state	qF
delta	Přechody	Transitions	\u03B4 (δ)

Tabulka 5.2: Pojmenování návěští položek Turingových strojů podle formátu

5.3 Sémantická kontrola

Po načtení správného formátu a vytvoření odpovídajících datových struktur zbývá ověřit, že nechybí nic podstatného a definované položky splňují náležitá omezení. Týká se to především přechodů a počátečních a koncových stavů. Každý stav, který se vyskytuje v přechodu, ale není definován v množině stavů, je sémantická chyba. Avšak protože je takováto chyba snadno napravitelná, v laxním režimu se bez vědomí uživatele napraví. Podobně fungují zásobníkové symboly vůči zásobníkové abecedě a symboly pásky vůči páskové abecedě. Vstupní a výstupní symboly v přechodech převodníků ale mohou být ze vstupní abecedy nebo komunikační abecedy. V tomto případě se volí konzervativní přístup a takovéto symboly se v laxním režimu přidávají do komunikační abecedy systému. Automaticky tedy nejsou určeny vstupní a výstupní abecedy. Toto je záměrné, protože právě tyto abecedy ovlivňují generovaný překlad, respektive jazyk.

Sémantická kontrola systému převodníků se skládá pouze z kontroly jeho komponent. Zde se kontroluje náležitost počátečního a všech koncových stavů mezi stavy, existence počátečního vrcholu zásobníku v zásobníkové abecedě a pro každý přechod náležitost položek do odpovídající množiny.

Pro Turingovy stroje je sémantická kontrola delší. Kromě počátečního a koncového stavu a položek všech přechodů je kontrolováno, zda vstupní abeceda je podmnožinou páskové abecedy, zda se prázdný symbol vyskytuje v páskové ale nikoli ve vstupní abecedě, zda v páskové abecedě nejsou symboly `R` a `L` vyhrazené akcím pohybu hlavy a zda koncový stav je opravdu koncový, tedy zda neexistuje nějaký přechod vedoucí z tohoto stavu.

Poslední sémantická kontrola nastává při zadání slova pro příjem nebo překlad. Toto slovo musí totiž být pouze ze symbolů vstupní abecedy.

5.4 Principy skriptu

Samotný skript je rozdělen do několika souborů, které implementují samostatné třídy pro práci s převodníky, systémy převodníků a Turingovými stroji. Navíc je zde oddělený soubor s definovanými konstantami pro znaky Unicode řecké abecedy a spustitelný soubor demo.py obsluhující příkazovou řádku.

Třída Transducer v souboru trans.py slouží pro uložení dat o převodníku a poskytuje jeho sémantickou kontrolu.

Třída TuringMachine v souboru turing.py slouží pro uložení dat o Turingově stroji. Poskytuje funkce pro načtení a uložení z, respektive do řetězce, sémantickou kontrolu a simulaci běhu pro přijetí řetězce i překlad.

Nejdůležitější je třída TransducerSystem v souboru transys.py, která slouží pro uložení dat o systému převodníků. Kromě načítání a ukládání poskytuje také funkce pro simulaci běhu pro přijetí i

překlad, ale hlavně funkce implementující algoritmy zavedené v této práci: pro redukci počtu komponent a pro vytvoření systému ekvivalentního danému Turingově stroji.

Zajímavým problémem z implementace bylo dodržení předpokladu $Q \cap \Gamma = \emptyset$ v Turingově stroji, pro který je vytvářen ekvivalentní systém. Každý stav je prodlužován o čárku ' tak dlouho dokud nespadá mimo Γ. Tímto způsobem je řešen i předpoklad $\# \notin \Gamma$ a $S, L, H, R, F \notin Q$.

5.4.1 Simulace běhu systému převodníků

Systémy převodníků jsou nedeterministické a to ve dvou směrech. Zaprvé přechody jsou definovány relací a nikoli zobrazením a zadruhé po průběhu komponenty se nedeterministicky vybírá další komponenta, na kterou se aktuální běh přepne. Kvůli tomu je nutné využít některý ze způsobů prohledávání stavového prostoru. Já jsem implementoval IDDFS, *iterative deepening deph-first search*. Tato varianta prohledávání do hloubky využívá zásobník pro ukládání stavů a tyto stavy rozgenerovává pouze do určité maximální hloubky, která se zvyšuje vždy po vyplýtvání tohoto omezeného stavového prostoru, dokud nebyl nalezen hledaný stav nebo nebyl prohledán celý stavový prostor.

Při průběhu komponenty se v každé konfiguraci procházejí všechna přechodová pravidla a pro ty vyhovující se generuje nová konfigurace na zásobník. Při přepnutí komponenty a při začátku algoritmu jsou vygenerovány počáteční konfigurace všech komponent s aktuálním slovem.

Funkce přijímání řetězce pouze obaluje funkci pro překlad – první nalezený prvek překladu rozhoduje o přijetí vstupu.

5.4.2 Simulace běhu Turingova stroje

Princip simulace běhu je díky determinismu velmi jednoduchý, i když ne nutně konečný. Z počáteční konfigurace $(q_{0,} \Delta w, 0)$ existuje vždy nejvýše jedna následující konfigurace. Pokud se v časovém limitu najde konfigurace s koncovým stavem, je slovo přijato, respektive je produkován překlad. Pokud se překročí levý okraj pásky nebo neexistuje další konfigurace, je slovo odmítnuto, respektive je ukončeno hledání překladu jako neúspěšné.

Funkce přijímání řetězce a funkce překladu řetězce využívají stejnou funkci pro simulaci běhu, která vrátí kromě rozhodnutí o přijetí i prefix obsahu pásky. Z těchto informací první funkce vrátí rozhodnutí o přijetí a druhá vyextrahuje přeložený řetězec z pásky.

Jediný zde zajímavý implementační problém byla reprezentace nekonečné pásky prázdných symbolů pomocí konečného prefixu a jeho prodlužování při posunu za jeho pravý konec.

6 Závěr

Tato práce zavedla systémy převodníků skloubením dvou různých oblastí teoretické informatiky. Hlavním předmětem zájmu pak byla rodina překladů takovýchto systémů a pro účely porovnání s Turingovými stroji i rodina vstupních jazyků.

Prvním závěrem byl důkaz ekvivalence síly systémů s různých počtem komponent vedený pomocí konstrukce jednokomponentního systému pro libovolný zadaný systém.

Druhým závěrem byl důkaz ekvivalence síly systémů a Turingových strojů s ohledem na rodinu vstupních, respektive přijímaných jazyků. Tento důkaz byl veden pomocí konstrukce systému, jehož vstupní jazyk je totožný s přijímaným jazykem zadaného Turingova stroje. Opačný směr, tedy že Turingovy stroje dokáží simulovat zadaný systém se opírá o *Church-Turingovu* tézi.

Třetí závěr porovnává sílu zavedených systémů a Turingových strojů s ohledem na jimi definovaný překlad. Pomocí představení Turingova stroje definujícího konečný překlad $\{(a,b),(b,c),(c,a)\}$, jež není definovatelný těmito systémy, je vyvrácena hypotéza o ekvivalenci.

Tuto práci doprovází aplikace v podobě skriptu v jazyce Python 3.4, která demonstruje uvedené konstrukční algoritmy a nabízí možnost porovnání systémů převodníků a Turingových strojů pomocí simulace jejich běhu pro zadaný řetězec.

Možnou další prací je úprava definice systémů převodníků tak, aby rodina překladů generovaných těmito systémy byla ekvivalentní rodině překladů Turingových strojů. Překlad systémů lze intuitivně chápat jako tranzitivní uzávěr překladů jednotlivých komponent tohoto systému, který je pomocí vstupní a výstupní abecedy filtrován od mezivýsledků, které v překladu nechceme. Bohužel, pokud výstupní abeceda je podmnožinou vstupní, překlad je stále tranzitivním uzávěrem, tentokrát však překladů sekvencí komponent. Tak jak jsem tyto systémy definoval, nemůžeme zaručit, že po nalezení překladu nebude systém pokračovat dále.

Mírnou úpravou definice přepnutí komponenty, která by zohledňovala, jestli je výsledek překladu ze vstupní abecedy nebo ne, bychom toto omezení dokázali obejít, ale tato úprava mi přišla dosti umělá. Přirozenější by bylo vytvořit obecné omezení na přepínání komponent, které by určovaly, kdy se které komponenty mohou v běhu použít. Podobný problém řeší regulované gramatiky nebo regulované zásobníkové automaty v [9]. V případě systémů by přidaný regulární jazyk určoval jediné povolené sekvence komponent a tím by dokázal vynutit například pouze jediné využití počáteční komponenty při celém překladu.

Možným pokračováním této práce by bylo i zavedení podobných systémů, které by však byly založeny namísto zásobníkových na konečných převodnících. Takováto práce by však byla nanejvýš formální, protože žádný algoritmus v této práci nevyžaduje využití zásobníku, a tak by analogické algoritmy měly bez problémů fungovat i pro konečné komponenty.

Literatura

[1] AHO, Alfred V. a Jeffrey D. ULLMAN. *The Theory of Parsing, Translation and Compiling* London: Prentice Hall, 1972, sv. 1, s. 212-238, ISBN 0-13-914556-7.

[2] APPEL, Andrew W. *Modern Compiler Implementation in C*. Cambridge, 1998, ISBN 0-521-60765-5.

[3] MOHRI, Mehryar. Weighted automata algorithms. **In:** Manfred DROSTE, Werner KUICH a Heiko VOGLER, eds. *Handbook of Weighted Automata. Monographs in Theoretical Computer Science.* Springer, 2009, s. 213-254.

[4] DASSOW, Jürgen, Gheorghe PĂUN a Grzegorz ROZENBERG. Grammar Systems. **In:** Grzegorz ROZENBERG a Arto SALOMAA, eds. *Handbook of Formal Languages.* Berlin: Springer, 1997, sv. 2, s. 155-214, ISBN 3-540-60648-3.

[5] MEDUNA, Alexander. *Automata and Languages: Theory and Applications.* London: Springer, 2000, ISBN 1-85233-074-0.

[6] PARKERS, Alan P. *Introduction to Languages, Machines and Logic.* London: Springer, 2002, ISBN 1-85233-464-9.

[7] ČEŠKA, Milan. *Teoretická informatika: Učební texty*[online]. Brno, 2002[cit. 2016-03-11]. Dostupné z: http://www.fit.vutbr.cz/study/courses/TIN/public/Texty/ti.pdf.

[8] MEDUNA, Alexander a Petr ZEMEK. *Regulated Grammars and Their Transformations*. Brno, 2010, ISBN 978-80-214-4203-0.

[9] MEDUNA Alexander, Dušan KOLÁŘ. *One-Turn Regulated Pushdown Automata and Their Reduction.* **In:** Fundamenta Informatica, 2002, č. 16, s. 399-405.

[10] MASOPUST, Tomáš, Alexander MEDUNA a Jiří TECHET. *Cooperating Distributed Grammar Systems*[online]. Brno, 2007[cit. 2015-11-21]. Dostupné z: http://www.fit.vutbr.cz/~meduna/work/lib/exe/fetch.php?media=lectures:phd:tid:frvs:09-cdgspres.pdf.

[11] MASOPUST, Tomáš, Alexander MEDUNA a Jiří TECHET. *Transducers and Translation Grammars*[online]. Brno, 2007[cit. 2015-11-21]. Dostupné z: http://www.fit.vutbr.cz/~meduna/work/lib/exe/fetch.php?media=lectures:phd:tid:frvs:15-transducerspres.pdf.