Automata and Grammars (BIE-AAG)

9. Transducers

Jan Holub

Department of Theoretical Computer Science
Faculty of Information Technology
Czech Technical University in Prague

Formal translations

Definition

Formal translation is a binary relation $Z \subseteq L \times V$.
The *domain* is set L and the *range* is set V.
Relation Z maps a set of translations $Z(w) \subseteq V$ to each element w of set L.
If $Z(w)$ contains at most one element for every $w \in L$, set Z is a function (could be partial) and the translation is said to be unambiguous.

Formal translations

Definition
Let Σ and D be alphabets. A *homomorphism* is every mapping h from Σ to D^*. The domain of homomorphism h can be extended to Σ^* as follows:
$h(\varepsilon) = \varepsilon$,
$h(xa) = h(x)h(a)$, $\forall x \in \Sigma^*, \forall a \in \Sigma$.

Example
translation of a string of decimal digits to a string of binary digits

a	0	1	2	3	4	5	6	7	8	9
$h(a)$	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

$h(1996) = 0001100110010110$

Formal translations

Definition (Prefix and postfix notation of expression)
Prefix and *postfix notation of an expression* E:

1. If E is a variable or a constant, then prefix notation and postfix notation of E is E.
2. If E is an expression of the form $E_1 \; op \; E_2$, where op is a binary operator, then
   (a) $op \; E_1' E_2'$ is prefix notation of E, where E_1' and E_2' are prefix notations of E_1 and E_2, respectively.
   (b) $E_1'' E_2'' op$ is postfix notation of E, where E_1'' and E_2'' are postfix notations of E_1 and E_2, respectively.
3. If E is an expression of the form (E_1), then
   (a) E_1' is prefix notation of E, where E_1' is prefix notation of E_1,
   (b) E_1'' is postfix notation of E, where E_1'' is postfix notation of E_1.

Formal translations

Example

Infix notation: $a * (b + c)$

Prefix notation: $*a + bc$

Postfix notation: $abc + *$

Example of a translation:

$\{(x, y)$: x is an expression in infix notation, y is the same expression in postfix notation$\}$.

Translation grammars

Definition
Translation grammar is $TG = (N, \Sigma, D, R, S)$, where

- N is a finite set of nonterminal symbols,
- Σ is a finite set of input symbols,
- D is a finite set of output symbols, $\Sigma \cap D = \emptyset$, $(\Sigma \cup D) \cap N = \emptyset$,
- R is a finite set of rules in the form $A \to \alpha$, where $A \in N$, $\alpha \in (N \cup \Sigma \cup D)^*$,
- $S \in N$ is the starting symbol.

Translation grammars

Definition

$TG = (N, \Sigma, D, R, S)$, $\alpha, \beta \in (N \cup \Sigma \cup D)^*$.
We say that α derives *in one step* β, $\alpha \Rightarrow \beta$ if $\exists \tau, \omega, \gamma \in (N \cup \Sigma \cup D)^*$,
$\exists A \in N$, $\alpha = \tau A \omega, \beta = \tau \gamma \omega$, $(A \rightarrow \gamma) \in R$
We say that α *derives* β, $\alpha \Rightarrow^* \beta$ if $\exists \alpha_1, \alpha_2, \ldots, \alpha_n \in (N \cup \Sigma \cup D)^*$, $n \geq 1$,
$\alpha = \alpha_1 \Rightarrow \alpha_2 \Rightarrow \ldots \Rightarrow \alpha_n = \beta$.
The sequence $\alpha_1, \alpha_2, \ldots, \alpha_n$ is called translation derivation of length n of string β from string α.
reflexive and transitive closure: $\Rightarrow^*$
transitive closure: $\Rightarrow^+$

Translation grammars

Example

$TG = (\{E\}, \{+, *, a\}, \{\oplus, \circledast, @\}, R, E)$, where R:

$(1)\ E \to +EE\oplus \qquad (2)\ E \to *EE\circledast \qquad (3)\ E \to a@$

$$\begin{aligned} E &\Rightarrow +EE\oplus \\ &\Rightarrow +a@E\oplus \\ &\Rightarrow +a@*EE\circledast\oplus \\ &\Rightarrow +a@*a@E\circledast\oplus \\ &\Rightarrow +a@*a@a@\circledast\oplus \end{aligned}$$

Translation grammars

Definition (Input/Output homomorphism)
$TG = (N, \Sigma, D, R, S)$.
Input homomorphism h_i^{TG}:

$$h_i^{TG}(a) = \begin{cases} a, & \forall a \in \Sigma \cup N \\ \varepsilon, & \forall a \in D \end{cases}$$

$$h_i^{TG}(x) = h_i^{TG}(x_1).h_i^{TG}(x_2)\dots h_i^{TG}(x_n),\ \forall x = x_1x_2\dots x_n \in (\Sigma \cup N \cup D)^*$$

Output homomorphism h_o^{TG}:

$$h_o^{TG}(a) = \begin{cases} \varepsilon, & \forall a \in \Sigma \\ a, & \forall a \in D \cup N \end{cases}$$

$$h_o^{TG}(x) = h_o^{TG}(x_1).h_o^{TG}(x_2)\dots h_o^{TG}(x_n),\ \forall x = x_1x_2\dots x_n \in (\Sigma \cup N \cup D)^*$$

Definition
Translation defined by translation grammar $TG = (N, \Sigma, D, R, S)$:
$Z(TG) = \{(h_i^{TG}(w), h_o^{TG}(w)) : w \in (\Sigma \cup D)^*, S \Rightarrow^* w\}$.

Translation grammars

Example

$TG = (\{E\}, \{+, *, a\}, \{\oplus, \circledast, @\}, R, E)$, where R:

$(1)\ E \to +EE\oplus \qquad (2)\ E \to *EE\circledast \qquad (3)\ E \to a@$

TG generates a translation of expressions in prefix notation to expressions in postfix notation.

$$
\begin{aligned}
E &\Rightarrow +EE\oplus \\
&\Rightarrow +a@E\oplus \\
&\Rightarrow +a@*EE\circledast\oplus \\
&\Rightarrow +a@*a@E\circledast\oplus \\
&\Rightarrow +a@*a@a@\circledast\oplus
\end{aligned}
$$

Derivation generates a pair $(+a*aa, @@@\circledast\oplus)$ that belongs to translation $Z(TG)$.

Translation grammars

Definition (Input/Output grammar of translation grammar)
$TG = (N, \Sigma, D, R, S)$.
Input grammar of translation grammar TG is CFG $G_i = (N, \Sigma, P_i, S)$, where $P_i = \{A \rightarrow h_i(\alpha) : (A \rightarrow \alpha) \in R\}$.
Output grammar of translation grammar TG is CFG $G_o = (N, D, P_o, S)$, where $P_o = \{A \rightarrow h_o(\alpha) : (A \rightarrow \alpha) \in R\}$.

Definition (Characteristic grammar/sentence/language)
CFG $G = (N, \Sigma \cup D, R, S)$ is *characteristic grammar of translation grammar* $TG = (N, \Sigma, D, R, S)$.
$L(G)$ is *characteristic language of translation Z(TG)*.
$w \in L(G)$ is *characteristic sentence of pair* (x, y), where $x = h_i(w)$, $y = h_o(w)$.

Translation grammars

Definition (Regular translation grammar)
$TG = (N, \Sigma, D, R, S)$ is *regular*, if all rules in R are of the form:

- $A \ \rightarrow axB$ or $A \rightarrow ax$, where $A, B \in N, a \in \Sigma, x \in D^*$, or
- $S \rightarrow \varepsilon$ in case S is not present in the right-hand side of no rule.

Translation grammars

Example

$RTG = (\{S, A, P, K\}, \{a, +, *\}, \{ⓐ, \oplus, \circledast\}, R, S)$, where R:

$$
\begin{array}{ll}
S \rightarrow a ⓐ A & A \rightarrow *K \\
S \rightarrow a ⓐ & A \rightarrow +P \\
K \rightarrow a ⓐ \circledast A & P \rightarrow a ⓐ \oplus A \\
K \rightarrow a ⓐ \circledast & P \rightarrow a ⓐ \oplus
\end{array}
$$

$$
\begin{aligned}
S &\Rightarrow a ⓐ A \\
&\Rightarrow a ⓐ + P \\
&\Rightarrow a ⓐ + a ⓐ \oplus A \\
&\Rightarrow a ⓐ + a ⓐ \oplus * K \\
&\Rightarrow a ⓐ + a ⓐ \oplus * a ⓐ \circledast.
\end{aligned}
$$

Translation: $(a + a * a, ⓐⓐ\oplusⓐ\circledast)$.

Finite transducers

Definition
Finite transducer $FT = (Q, \Sigma, D, \delta, q_0, F)$, where

- Q is a finite set of states,
- Σ is a finite set of input symbols,
- D is a finite set of output symbols,
- δ is a mapping from $Q \times (\Sigma \cup \{\varepsilon\})$ into a set of finite subsets $2^{Q \times D^*}$,
- $q_0 \in Q$ is the start state,
- $F \subseteq Q$ is the set of final states.

Finite transducers

Definition

Configuration of finite transducer $FT = (Q, \Sigma, D, \delta, q_0, F)$ is a triple $(q, x, y) \in Q \times \Sigma^* \times D^*$.
(q_0, x, ε) is the *initial configuration*.
(q, ε, y), $q \in F$ is the *final configuration*.

Let $\vdash_{FT}$ is a relation over $Q \times \Sigma^* \times D^*$ such that $(q, aw, y) \vdash_{FT} (p, w, yz)$ iff $\delta(q, a) = (p, z)$ for some $a \in \Sigma \cup \{\varepsilon\}$, $w \in \Sigma^*, y, z \in D^*$. An element of relation $\vdash_{FT}$ is called a *move* in finite transducer FT.

$\vdash^k_{FT}$ – the k-th power of relation $\vdash_{FT}$
$\vdash^+_{FT}$ – the transitive closure of relation $\vdash_{FT}$
$\vdash^*_{FT}$ – the transitive and reflexive closure of relation $\vdash_{FT}$

Finite transducers

Definition

Translation defined by a finite transducer $FT = (Q, \Sigma, D, \delta, q_0, F)$:
$Z(FT) = \{(u, v) : u \in \Sigma^*, v \in D^*, \exists q \in F, (q_0, u, \varepsilon) \vdash^*_{FT} (q, \varepsilon, v)\}$

Definition

FT is *deterministic*, if for all its states $q \in Q$ it holds:

1. $|\delta(q, a)| \leq 1$, $\forall a \in \Sigma$ and $\delta(q, \varepsilon) = \emptyset$ or
2. $|\delta(q, \varepsilon)| \leq 1$ and $\delta(q, a) = \emptyset$, $\forall a \in \Sigma$.

Finite transducers

Example

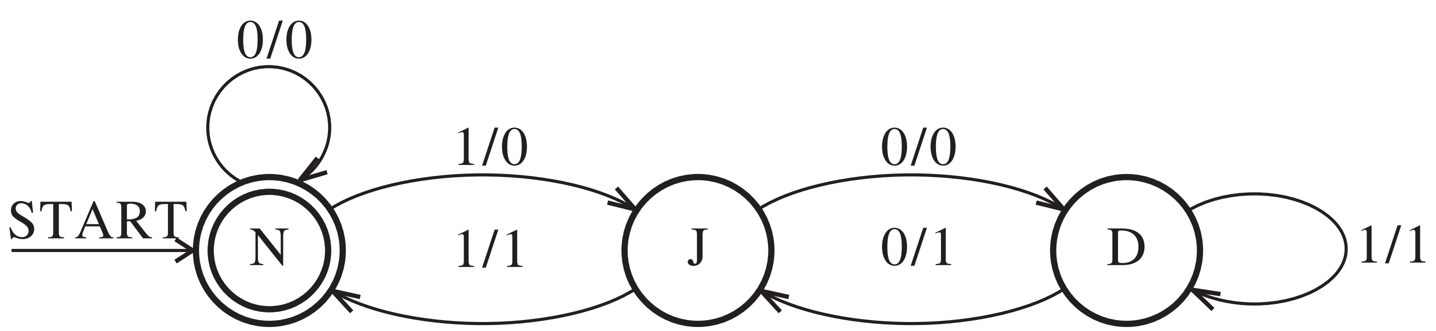
FT, that divides by three binary numbers divisible by three.

$FT = (\{N, J, D\}, \{0, 1\}, \{0, 1\}, \delta, N, \{N\})$, where δ:

$\delta(N, 0) = \{(N, 0)\}$ $\quad$ $\delta(J, 1) = \{(N, 1)\}$ $\quad$ $\delta(D, 0) = \{(J, 1)\}$

$\delta(N, 1) = \{(J, 0)\}$ $\quad$ $\delta(J, 0) = \{(D, 0)\}$ $\quad$ $\delta(D, 1) = \{(D, 1)\}$

δ	0	1
N	$\{(N, 0)\}$	$\{(J, 0)\}$
J	$\{(D, 0)\}$	$\{(N, 1)\}$
D	$\{(J, 1)\}$	$\{(D, 1)\}$



Transforming RTG to FT

Theorem

Given $RTG = (N, \Sigma, D, R, S)$. There exists $FT = (Q, \Sigma, D, \delta, q_0, F)$ such that $Z(RTG) = Z(FT)$.

Proof

For a given $RTG = (N, \Sigma, D, R, S)$ we create $FT = (Q, \Sigma, D, \delta, q_0, F)$, where $Q = N \cup \{X\}$, $X \notin N$.

1. Mapping δ is defined as ($y \in D^*, B, C \in N$):
   (a) $\delta(B, a) = \{(C, y) : (B \to ayC) \in R\} \cup \{(X, y) : (B \to ay) \in R\}$, $\forall B \in N, \forall a \in \Sigma$
   (b) $\delta(X, a) = \emptyset$, $\forall a \in \Sigma$
2. $q_0 = S$
3. $F = \{S, X\}$, if $(S \to \varepsilon) \in R$
   $F = \{X\}$, if $(S \to \varepsilon) \notin R$.

Proof that $Z(RTG) = Z(FT)$: by induction by the length of derivative in RTG and the length of sequence of transitions in FT. $\square$

Transforming RTG to FT

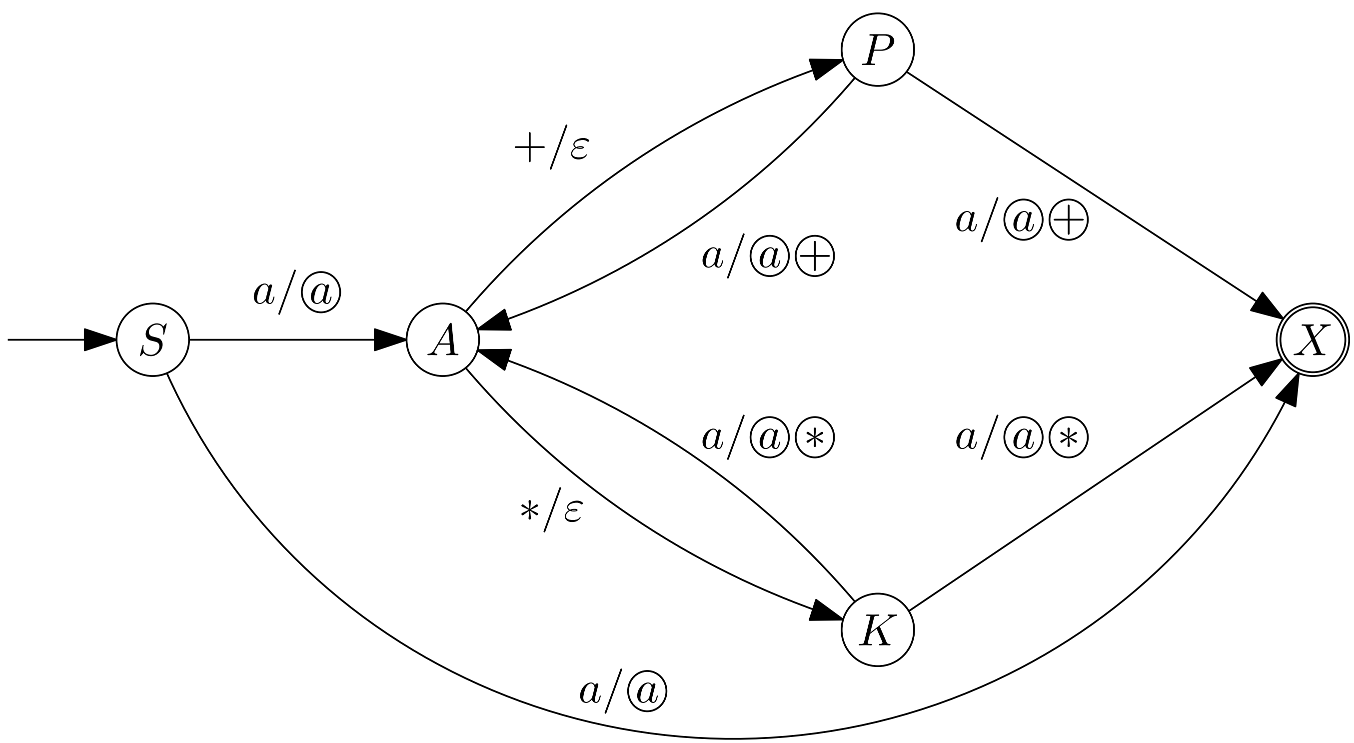
Example

$RTG = (\{S, A, P, K\}, \{a, +, *\}, \{@, \oplus, \circledast\}, R, S)$, where R:

$S \rightarrow a@A \qquad A \rightarrow *K \qquad P \rightarrow a@\oplus A \qquad K \rightarrow a@\circledast A$

$S \rightarrow a@ \qquad A \rightarrow +P \qquad P \rightarrow a@\oplus \qquad K \rightarrow a@\circledast$

$FT = (\{S, X, A, P, K\}, \{a, +, *\}, \{@\oplus\circledast\}, \delta, S, \{X\})$



Transforming FT to RTG

Theorem
Given a finite transducer FT. There exists a regular translation grammar RTG such that $Z(FT) = Z(RTG)$.

Proof
For a given $FT = (Q, \Sigma, D, \delta, q_0, F)$ we construct $RTG = (Q \cup \{S\}, \Sigma, D, R, S)$, where $S \notin Q$ as follows (Suppose that $\Sigma \cap D = \emptyset$, $(\Sigma \cup D) \cap Q = \emptyset$):

1. $R \leftarrow \{B \to ayC : B, C \in Q, a \in \Sigma, y \in D^*, (C, y) \in \delta(B, a)\}$
2. $R \leftarrow R \cup \{B \to ay : B \in Q, C \in F, a \in \Sigma, y \in D^*, (C, y) \in \delta(B, a)\}$
3. $R \leftarrow R \cup \{S \to x : x \in (N \cup D \cup \Sigma)^*, (q_0 \to x) \in R\}$
4. $R \leftarrow R \cup \{S \to \varepsilon\}$, $q_0 \in F$

The proof of equivalence $Z(RTG) = Z(FT)$: by induction on the length of derivation in RTG and on the length of sequence of transitions in FT. $\square$

Transforming FT to RTG

Example

$FT = (\{N, J, D\}, \{0, 1\}, \{⓪, ①\}, \delta, N, \{N\})$, where δ:

$$\begin{array}{lll}
\delta(N,0) = \{(N,⓪)\} & \delta(J,0) = \{(D,⓪)\} & \delta(D,0) = \{(J,①)\} \\
\delta(N,1) = \{(J,⓪)\} & \delta(J,1) = \{(N,①)\} & \delta(D,1) = \{(D,①)\}
\end{array}$$

$RTG = (\{S, N, J, D\}, \{0, 1\}, \{⓪, ①\}, R, S)$, where R:

$$\begin{array}{llll}
N \rightarrow 0⓪N & J \rightarrow 0⓪D & D \rightarrow 0①J & S \rightarrow 0⓪N \\
N \rightarrow 1⓪J & J \rightarrow 1①N & D \rightarrow 1①D & S \rightarrow 1⓪J \\
N \rightarrow 0⓪ & J \rightarrow 1① & S \rightarrow 0⓪ & S \rightarrow \varepsilon
\end{array}$$

Sequential mapping

Definition

Sequential mapping S is a mapping such that:

1. preserves the length of the string, i.e.; if $y = S(x)$, then $|x| = |y|$,
2. if two input strings have the same prefix of length $k > 0$, then also the corresponding output strings have identical prefixes of length at least k. That means, if $S(xx_1) = y_1$, $S(xx_2) = y_2$, then there exist y, y_1', y_2' such that $|y| \geq |x|$, $y_1 = yy_1'$, and $y_2 = yy_2'$.

Mealy/Moore automaton

Definition

Mealy automaton M is a sixtuple $(Q, \Sigma, D, \delta, \lambda, q_0)$, where

- Q is a finite set of states,
- Σ is a finite set of input symbols,
- D is a finite set of output symbols,
- δ is a mapping from $Q \times \Sigma$ into Q called *transition function*,
- λ is a mapping from $Q \times \Sigma$ into D called *output function*,
- $q_0 \in Q$ is the start state.

Mealy/Moore automaton

Definition

Moore automaton M is a sixtuple $(Q, \Sigma, D, \delta, \lambda, q_0)$, where

- Q is a finite set of states,
- Σ is a finite set of input symbols,
- D is a finite set of output symbols,
- δ is a mapping from $Q \times \Sigma$ into Q called *transition function*,
- λ is a mapping from Q into D called *output function*,
- $q_0 \in Q$ is the start state.

Remark

For input $w \in \Sigma^*, n = |w|$, a Mealy automaton outputs[1] a string of length n, and a Moore automaton outputs a string of length $n + 1$. In some sense, the output of the initial state of the Moore automaton is not meaningful because it does not depend on the input at all.

[1] We suppose mapping λ to be total for both automata.

Mealy/Moore automaton

Definition (Equivalence of Mealy and Moore automata)
Let X is Mealy or Moore automaton. Denote by $\Lambda_X(u), u \in \Sigma^*$ as output of automaton X for input u (i.e., $\Lambda_X(u) = v, (u, v) \in Z(X), v \in D^*$).
We say a Moore automaton M is equivalent to a Mealy automaton M' (denoted as $Z(M) = Z(M')$) if $\Lambda_M(w) = \Lambda_M(\varepsilon).\Lambda_{M'}(w)$, $\forall w \in \Sigma^*$, i.e., their output strings are identical after we remove the first output symbol of Moore automaton.

Definition (Extended transition function)
Let δ be a transition function of Mealy or Moore automaton. Extended transition function is mapping $\hat{\delta} : Q \times \Sigma^*$ into Q defined as follows:

1. $\hat{\delta}(q, \varepsilon) = q, \forall q \in Q$
2. $\hat{\delta}(q, a) = \delta(q, a), \forall a \in \Sigma, \forall q \in Q$
3. $\hat{\delta}(q, ua) = \delta(\hat{\delta}(q, u), a), \forall a \in \Sigma, u \in \Sigma^*, \forall q \in Q$

Mealy/Moore automaton

Theorem
Let $M = (Q, \Sigma, D, \delta, \lambda, q_0)$ be a Moore automaton. Then there exists Mealy automaton M' with the same number of states such that $Z(M) = Z(M')$.

Proof
Mealy automaton M' is constructed as follows:

1. $\lambda'(q, a) \leftarrow \lambda(\delta(q, a)), \forall q \in Q, \forall a \in \Sigma$
2. $M' \leftarrow (Q, \Sigma, D, \delta, \lambda', q_0)$

If the input of M is $w = a_1 a_2 \dots a_n, a_i \in \Sigma$, then M' outputs
$\lambda'(q_0, a_1).\lambda'(\hat{\delta}(q_0, a_1), a_2).\lambda'(\hat{\delta}(q_0, a_1 a_2), a_3) \dots \lambda'(\hat{\delta}(q_0, a_1 a_2 a_3 \dots a_{n-1}), a_n)$.
But by definition of λ', this is
$\lambda(\hat{\delta}(q_0, a_1)).\lambda(\hat{\delta}(q_0, a_1 a_2)).\lambda(\hat{\delta}(q_0, a_1 a_2 a_3)) \dots \lambda(\hat{\delta}(q_0, a_1 a_2 a_3 \dots a_n))$. $\square$

Mealy/Moore automaton

Theorem

Let $M' = (Q', \Sigma, D, \delta', \lambda', q'_0)$ be a Mealy automaton. Then there exists Moore automaton M with $|Q'||D|$ states such that $Z(M) = Z(M')$.

Proof

Moore automaton M is constructed as follows:

1. $Q \leftarrow Q' \times D$
2. $\delta((q,b),a) \leftarrow (\delta'(q,a), \lambda'(q,a)), \forall q \in Q', \forall a \in \Sigma, \forall b \in D$
3. $q_0 \leftarrow (q'_0, c)$, for some arbitrary fixed $c \in D$
4. $\lambda((q,b)) \leftarrow b, \forall q \in Q', \forall b \in D$
5. $M \leftarrow (Q, \Sigma, D, \delta, \lambda, q_0)$

If the input of M' is $w = a_1 a_2 \dots a_n, a_i \in \Sigma$, then M' enters the states $q'_0, \hat{\delta}'(q'_0, a_1), \hat{\delta}'(q'_0, a_1 a_2), \dots, \hat{\delta}'(q'_0, a_1 a_2 a_3 \dots a_n)$ and outputs $\lambda'(q_0, a_1)\lambda'(\hat{\delta}'(q_0, a_1), a_2) \dots \lambda'(\hat{\delta}'(q_0, a_1 a_2 a_3 \dots a_{n-1}), a_n)$

If the input of M is w, then M enters the states $(q'_0, c), (\hat{\delta}'(q'_0, a_1), \lambda'(q'_0, a_1)), (\hat{\delta}'(q'_0, a_1 a_2), \lambda'(\hat{\delta}'(q'_0, a_1), a_2)), \dots$ $(\hat{\delta}'(q'_0, a_1 a_2 \dots a_n), \lambda'(\hat{\delta}'(q'_0, a_1 a_2 \dots a_{n-1}), a_n)$ and outputs $c.\lambda'(q'_0, a_1).\lambda'(\hat{\delta}'(q'_0, a_1), a_2) \dots \lambda'(\hat{\delta}'(q'_0, a_1 a_2 \dots a_{n-1}), a_n)$ $\square$

Pushdown Translation Automata

Definition

The pushdown translation automaton is an octuple
$PTA = (Q, \Sigma, G, D, \delta, q_0, Z_0, F)$, where

- Q is a finite set of states,
- Σ is a finite set of input symbols,
- G is a finite set of symbols (*pushdown symbols*),
- D is a finite set of output symbols,
- δ is a finite mapping from $Q \times (\Sigma \cup \{\varepsilon\}) \times G^*$ to a set of subsets $Q \times G^* \times D^*$,
- $q_0 \in Q$ is the start state,
- $Z_0 \in G$ is an initial symbol of the pushdown store,
- $F \subseteq Q$ is a set of final states.

Pushdown Translation Automata

Definition

The configuration of a pushdown translation automaton $PTA = (Q, \Sigma, G, D, \delta, q_0, Z_0, F)$ is defined as a quadruple $(q, x, \alpha, y) \in Q \times \Sigma^* \times G^* \times D^*$.

$(q_0, x, Z_0, \varepsilon)$ – *the initial configuration* (q_0 is the initial state and Z_0 is the initial pushdown symbol)

Transition relation $\vdash$: $(q, ax, u\gamma, y) \vdash (r, x, \alpha\gamma, yv)$, $\forall a \in \Sigma \cup \{\varepsilon\}$, $\forall x \in \Sigma^*$, $\forall \gamma, \alpha \in G^*$, $\forall y, u, v \in D^*$, $\forall q \in Q$, $\forall r \in Q$, $(r, \alpha, v) \in \delta(q, a, u)$.

Pushdown Translation Automata

Definition

The translation defined by a pushdown translation automaton $PTA = (Q, \Sigma, G, D, \delta, q_0, Z_0, F)$ *by move into final state* is a set of pairs $Z(PTA) = \{(x, y) : x \in \Sigma^*, y \in D^*, \exists q \in F, \exists \alpha \in G^*, (q_0, x, Z_0, \varepsilon) \vdash^* (q, \varepsilon, \alpha, y)\}$.

The translation defined by a pushdown translation automaton $PTA = (Q, \Sigma, G, D, \delta, q_0, Z_0, \emptyset)$ *by move into configuration with empty pushdown store* is a set of pairs $Z_\varepsilon(PTA) = \{(x, y) : x \in \Sigma^*, y \in D^*, \exists q \in Q, (q_0, x, Z_0, \varepsilon) \vdash^* (q, \varepsilon, \varepsilon, y)\}$.

Pushdown Translation Automata

Example

$PTA = (\{q\}, \{a, +, *\}, \{+, *, E\}, \{a, +, *\}, \delta, q, E, \emptyset)$, where δ:

$\delta(q, a, E) = \{(q, \varepsilon, a)\}$
$\delta(q, +, E) = \{(q, EE+, \varepsilon)\}$
$\delta(q, *, E) = \{(q, EE*, \varepsilon)\}$
$\delta(q, \varepsilon, +) = \{(q, \varepsilon, +)\}$
$\delta(q, \varepsilon, *) = \{(q, \varepsilon, *)\}$.

The automaton makes the following moves for input $+ * aaa$:

$$
\begin{array}{lllrrr}
(q, + * aaa, E, \varepsilon) & \vdash (q, & *aaa, & EE+, & \varepsilon) \\
& \vdash (q, & aaa, & EE * E+, & \varepsilon) \\
& \vdash (q, & aa, & E * E+, & a) \\
& \vdash (q, & a, & *E+, & aa) \\
& \vdash (q, & a, & E+, & aa*) \\
& \vdash (q, & \varepsilon, & +, & aa * a) \\
& \vdash (q, & \varepsilon, & \varepsilon, & aa * a+).
\end{array}
$$

Pushdown Translation Automata

Definition

Pushdown translation automaton $PTA = (Q, \Sigma, G, D, \delta, q_0, Z_0, F)$ is called *deterministic* if:

1. $|\delta(q, a, \gamma)| \leq 1$, $\forall q \in Q, \forall a \in (\Sigma \cup \{\varepsilon\}), \forall \gamma \in G^*$.
2. If $\delta(q, a, \alpha) \neq \emptyset$, $\delta(q, a, \beta) \neq \emptyset$ and $\alpha \neq \beta$, then α is not a prefix of β and β is not a prefix of α (i.e., $\gamma\alpha \neq \beta, \alpha \neq \gamma\beta, \gamma \in G^*$).
3. If $\delta(q, a, \alpha) \neq \emptyset$, $\delta(q, \varepsilon, \beta) \neq \emptyset$, then α is not a prefix of β and β is not a prefix of α (i.e., $\gamma\alpha \neq \beta, \alpha \neq \gamma\beta, \gamma \in G^*$).

Pushdown Translation Automata

Theorem
Let $TG = (N, \Sigma, D, R, S)$ be a translation grammar. Then there exists pushdown translation automaton PTA such that $Z_\varepsilon(PTA) = Z(TG)$.

Proof: For given $TG = (N, \Sigma, D, R, S)$ we construct $PTA = (\{q\}, \Sigma, N \cup \Sigma \cup D, D, \delta, q, S, \emptyset)$, where δ:

1. $\delta(q, \varepsilon, A) \leftarrow \{(q, \alpha, \varepsilon) : (A \to \alpha) \in R\}$, $\forall A \in N$, $\triangleright$ $(\textbf{expansion})$
2. $\delta(q, a, a) \leftarrow \{(q, \varepsilon, \varepsilon)\}$, $\forall a \in \Sigma$, $\triangleright$ $(\textbf{comparison})$
3. $\delta(q, \varepsilon, b) \leftarrow \{(q, \varepsilon, b)\}$, $\forall b \in D$. $\triangleright$ $(\textbf{output})$

The proof that $Z(TG) = Z_\varepsilon(PTA)$ will be made by induction on the length of derivation in grammar TG and on length of sequence of transitions of automaton PTA. $\square$

Pushdown Translation Automata

Example

$TG = (\{E, T, F\}, \{+, *, (,), a\}, \{\oplus, \circledast, @\}, R, E)$, where R:

$E \to E + T\oplus$ $\qquad$ $E \to T$
$T \to T * F\circledast$ $\qquad$ $T \to F$
$F \to (E)$ $\qquad$ $F \to a@$.

$PTA = (\{q\}, \{+, *, (,), a\}, \{+, *, (,), a, E, T, F, \oplus, \circledast, @\},$
$\{\oplus, \circledast, @\}, \delta, q, E, \emptyset)$, where δ:
$\delta(q, \varepsilon, E) = \{(q, E + T\oplus, \varepsilon), (q, T, \varepsilon)\}$,
$\delta(q, \varepsilon, T) = \{(q, T * F\circledast, \varepsilon), (q, F, \varepsilon)\}$,
$\delta(q, \varepsilon, F) = \{(q, (E), \varepsilon), (q, a@, \varepsilon)\}$,
$\delta(q, c, c) = \{(q, \varepsilon, \varepsilon)\}$, $\forall c \in \{+, *, (,), a\}$,
$\delta(q, \varepsilon, b) = \{(q, \varepsilon, b)\}$, $\forall b \in \{\oplus, \circledast, @\}$.

Pushdown Translation Automata

$$
\begin{array}{lllrrr}
(q, a+a*a, E, \varepsilon) & \vdash(q, & a+a*a, & E+T\oplus, & \varepsilon) \\
& \vdash(q, & a+a*a, & T+T\oplus, & \varepsilon) \\
& \vdash(q, & a+a*a, & F+T\oplus, & \varepsilon) \\
& \vdash(q, & a+a*a, & a\text{ⓐ}+T\oplus, & \varepsilon) \\
& \vdash(q, & +a*a, & \text{ⓐ}+T\oplus, & \varepsilon) \\
& \vdash(q, & +a*a, & +T\oplus, & \text{ⓐ}) \\
& \vdash(q, & a*a, & T\oplus, & \text{ⓐ}) \\
& \vdash(q, & a*a, & T*F\circledast\oplus, & \text{ⓐ}) \\
& \vdash(q, & a*a, & F*F\circledast\oplus, & \text{ⓐ}) \\
& \vdash(q, & a*a, & a\text{ⓐ}*F\circledast\oplus, & \text{ⓐ}) \\
& \vdash(q, & *a, & \text{ⓐ}*F\circledast\oplus, & \text{ⓐ}) \\
& \vdash(q, & *a, & *F\circledast\oplus, & \text{ⓐⓐ}) \\
& \vdash(q, & a, & F\circledast\oplus, & \text{ⓐⓐ}) \\
& \vdash(q, & a, & a\text{ⓐ}\circledast\oplus, & \text{ⓐⓐ}) \\
& \vdash(q, & \varepsilon, & \text{ⓐ}\circledast\oplus, & \text{ⓐⓐ}) \\
& \vdash(q, & \varepsilon, & \circledast\oplus, & \text{ⓐⓐⓐ}) \\
& \vdash(q, & \varepsilon, & \oplus, & \text{ⓐⓐⓐ}\circledast) \\
& \vdash(q, & \varepsilon, & \varepsilon, & \text{ⓐⓐⓐ}\circledast\oplus).
\end{array}
$$