Automata and Grammars (BIE-AAG)

5. Finite state 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 $Z \subseteq L \times 2^V$, L and V are languages.
The *domain* is set L and the range is set 2^V.
Relation Z maps a set of translations $Z(w)$, which is itself from V, to an 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

Le T and D be alphabets. A *homomorphism* is every mapping h from T to D^*. The domain of homomorphism h can be extended to T^* like this:
$h(\varepsilon) = \varepsilon$,
$h(xa) = h(x)h(a), \forall x, x \in T^*, a \in T$.

Formal translations

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 notation of an expression 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, where E_1' and E_2' are prefix notations.
   (b) $E_1'' E_2'' op$ is postfix notation, where E_1'' and E_2'' are postfix notations.
3. If E is an expression of the form (E_1), then
   (a) prefix notation of (E_1) is prefix notation of expression E_1,
   (b) postfix notation of (E_1) is postfix notation of expression 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 infix notation of an expression, y is postfix notation of the same expression$\}$.

Translation grammars

Definition

Translation grammar is $TG = (N, T, D, R, S)$, where

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

It holds that $T \cap D = \emptyset$ and $(T \cup D) \cap N = \emptyset$.

Translation grammars

Definition

$TG = (N, T, D, R, S), \alpha, \beta, \gamma \in (N \cup T \cup D)^*, A \in N$.

1. α derives *in one step* β, $\alpha \Rightarrow \beta$ if and only if
   $\exists \tau, \omega, \gamma \in (N \cup T \cup D)^*$, $A \in N$, $\alpha = \tau A \, \omega, \beta = \tau \gamma \omega$,
   $A \; \to \gamma \in R$
2. α *derives* β, $\alpha \Rightarrow^* \beta$ if and only if
   $\exists \alpha_1, \alpha_2, ..., \alpha_n \in (N \cup T \cup D)^*, (n \geq 1)$
   $\alpha = \alpha_1 \Rightarrow \alpha_2 \Rightarrow ... \Rightarrow \alpha_n = \beta$.

The sequence $\alpha_1, \alpha_2, ..., \alpha_n$ is called translation derivation of length n of string β from string α.

reflexive and transitive closure: $\Rightarrow^*$

transitive closure: $\Rightarrow^+$

Translation grammars

Example

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

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

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

Translation grammars

Definition

$TG = (N, T, D, R, S)$.

Input homomorphism h_i^{TG}: $h_i^{TG}(a) = \begin{cases} a & \text{for } a \in T \cup N \\ \varepsilon & \text{for } a \in D \end{cases}$

Output homomorphism h_o^{TG}: $h_o^{TG}(a) = \begin{cases} \varepsilon & \text{for } a \in T \\ a & \text{for } a \in D \cup N \end{cases}$

Definition

Translation defined by translation grammar $TG = (N, T, D, R, S)$:

$Z(TG) = \{(h_i^{TG}(w), h_o^{TG}(w)) : S \Rightarrow^* w, w \in (T \cup D)^*\}$.

Translation grammars

Example

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

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

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

$TG = (N, T, D, R, S)$.

Input grammar of translation grammar TG is CFG $G_i = (N, T, P_i, S)$, where $P_i = \{A \to h_i(\alpha) : A \to \alpha \in R\}$.

Output grammar of translation grammar TG is CFG $G_o = (N, D, P_o, S)$, where $P_o = \{A \to h_o(\alpha) : A \to \alpha \in R\}$.

Definition

CFG $G = (N, T \cup D, R, S)$ is *characteristic grammar of translation grammar* $TG = (N, T, D, R, S)$.

$L(G)$ 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

$TG = (N, T, D, R, S)$ is *regular*, if all rules in R are of the form:

- $A \rightarrow axB$ or $A \rightarrow ax$, where $A, B \in N, a \in T, x \in D^*$
- $S \rightarrow \varepsilon$ in case that S is not present in the right-hand side of any rule.

Translation grammars

Example

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

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

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

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

Finite transducers

Definition

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

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

Finite transducers

Definition

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

Transition relation $\vdash$ a defined on the set of configurations:
If $\delta(q, a)$ contains (r, z), then $(q, ax, y) \vdash (r, x, yz), x \in T^*, y \in D^*$.
Transitive closure: $\vdash^+$
Reflexive and transitive closure: $\vdash^*$
k-th power: $\vdash^k$

Finite transducers

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

Definition
FT is *deterministic*, if for all its states it holds:

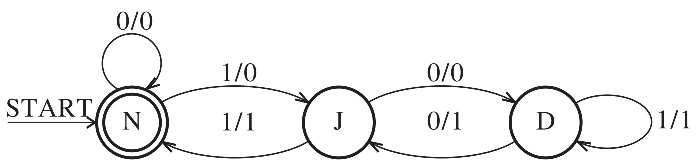
1. $|\delta(q, a)| \leq 1, \forall a \in T$ and $\delta(q, \varepsilon) = \emptyset$ or
2. $|\delta(q, \varepsilon)| \leq 1$ and $\delta(q, a) = \emptyset, \forall a \in T$.

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:

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

δ	0	1
N	$(N, 0)$	$(J, 0)$
J	$(D, 0)$	$(N, 1)$
D	$(J, 1)$	$(D, 1)$



Sequential mapping

Definition
Sequential mapping S:

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 $S(xx_1) = yy_1$ and $S(xx_2) = yy_2$ and $|x| = |y|$.

Sequential mapping

Definition
Mealy automaton $M = (Q, T, D, \delta, \lambda, q_o, F)$, where

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

Sequential mapping

Definition

Moore automaton $M = (Q, T, D, \delta, \lambda, q_o, F)$, where

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

Attributed translations

- Attribute – quantity that has a value from some set (range of the attribute).
  E.g. a variable in source code that has a defined type.
- Attributed symbol – symbol of alphabet that has a mapped (possibly empty) set of attributes.
- Attributed string – string of attributed symbols.

E.g.

$x.a$

$x[x.a_1, x.a_2, ..., x.a_n]$

Attributed translations

Definition

Attributed translation is a relation $Z_A \subseteq T^* \times D^*$, where T^* is a set of input attributed strings, D^* is a set of output attributed strings.

Example

Input: attributed strings over alphabet $\{a, +, *, (,)\}$. Symbol a has an attribute x (its range are all integers).
Output: v with an attribute y (its range are all integers).

$(a[10] \qquad\qquad\quad , v[10]),$

$(a[5] + a[6] \qquad , v[11]),$

$(a[3] * a[4] + a[2], v[14]).$

Attributed translations

Example

$a.x$ is a relative address, where the value of operand a is stored.
Address p, on which the relative operand addresses are based.
Function $\texttt{cho0se}(x)$ that chooses value from address x.

$p = 100$, $\texttt{choose}(102) = 3$, $\texttt{choose}(103) = 5$, $\texttt{choose}(104) = 6$

$(a[3] * a[4] + a[2], v[33])$

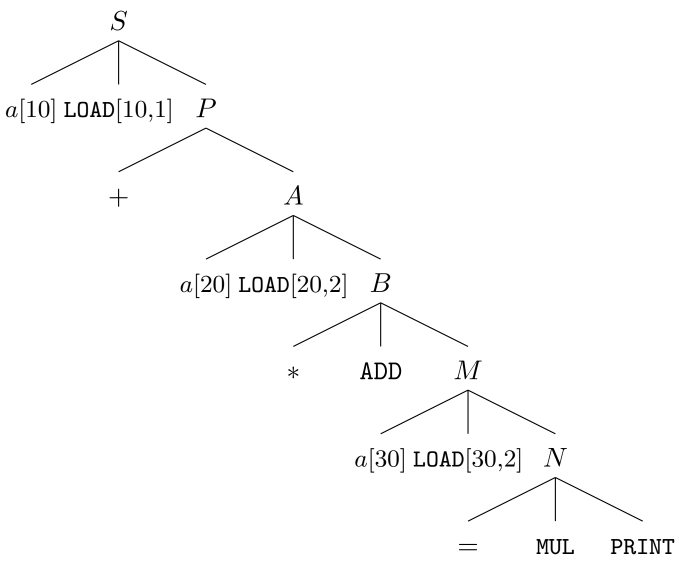
Attributed translations

The values of attributed output symbols can depend on:

- values of attributes of input symbols,
- structure of the input string,
- provided parameters.

Attributed translation tree

Attributed translation tree



Attributed translations

Attribute:

- synthesized — its value depends on information contained inside the subtree,
- inherited — its value depends on (outside) context of the subtree.

Attributed translations

attributed translation sentence:

- attributed string of output symbols that belong to the leaves of the attributed translation tree AT

semantic evaluation of translation tree:

- calculation of values of attributes in a translation tree
  - we set the values of inherited attributes of the root of AT
  - we set the values of the synthesized attributes of the leaves
  - we go through the nodes of the tree in some order and evaluate their attributes

Attributed translations

Example

Perform a translation of expression with operators $+$, $*$, $=$ and operands. Provided are addresses on which the operands are stored. The output language of the translation is a machine code that has the following instructions:

LOAD	$addr, r$ …	loads the contents of $addr$ into register r,
ADD	…	adds two registers 1 a 2 and stores the result in register 1,
MUL	…	multiplies two registers 1 a 2 and stores the result in register 1,
PRINT	…	prints the register 1.

$(a[20] =$, LOAD[20,1] PRINT[])

$(a[10] + a[30] =$, LOAD[10,1] LOAD[30,2] ADD[] PRINT[])

$(a[10] + a[20] * a[30] =$, LOAD[10,1] LOAD[20,2] ADD[] LOAD[30,2] MUL[] PRINT[])

Attributed translations

Example (continued)

Base grammar of attributed translation grammar

TG $= (\{S, P, A, B, M, N\}, \{a, +, *, =\},$
$\{\mathtt{LOAD}, \mathtt{ADD}, \mathtt{MUL}, \mathtt{PRINT}\}, R, S)$, where R:

1. $S \to a$ LOAD P,
2. $P \to +\ A$,
3. $P \to *\ M$,
4. $P \to =$ PRINT,
5. $A \to a$ LOAD B,
6. $M \to a$ LOAD N,
7. $B \to +$ ADD A,
8. $B \to *$ ADD M,
9. $B \to =$ ADD PRINT,
10. $N \to +$ MUL A,
11. $N \to *$ MUL M,
12. $N \to =$ MUL PRINT.

Symbols	Inherited attributes	Synthesized attributes
a		adr
LOAD	adr, r	
ADD		
MUL		
PRINT		

Attributed translations

Example (continued)

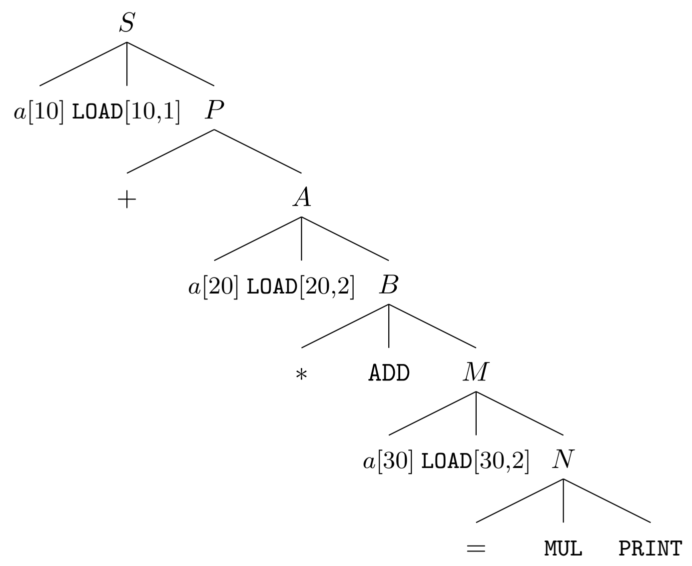
Semantic rules and their mapping to rules:

Syntax	Semantics
1. $S \to a$ LOAD P,	LOAD.$adr \leftarrow a.adr$ LOAD.$r \leftarrow 1$
2. $P \to +\ A$,	
3. $P \to *\ M$,	
4. $P \to =$ PRINT,	
5. $A \to a$ LOAD B,	LOAD.$adr \leftarrow a.adr$ LOAD.$r \leftarrow 2$
6. $M \to a$ LOAD N,	LOAD.$adr \leftarrow a.adr$ LOAD.$r \leftarrow 2$
7. $B \to +$ ADD A,	
8. $B \to *$ ADD M,	
9. $B \to =$ ADD PRINT,	
10. $N \to +$ MUL A,	
11. $N \to *$ MUL M,	
12. $N \to =$ MUL PRINT.	

Attributed translation tree

Example (continued)

Attributed translation tree for input string $a[10] + a[20] * a[30] =$.



Attributed translation tree

Definition
Attributed translation tree AT of attributed input sentence w in $ATG = (TG, A, V, F)$ is a translation tree of this sentence w constructed in TG (without considering attributes) and extended in the following manner:

1. Attributes defined by set $A(X)$ are assigned to every node evaluated with a symbol $X \in N \cup T \cup D$.
2. Values of inherited attributes of the root of the tree AT are set.
3. Values of the synthesized attributes of leaves of AT are determined by the input sentence w.
4. Let u_0 be an inner node X and $u_1, u_2, \ldots, u_n$ $(n \geq 0)$ be the direct descendants of u_0 evaluated with $X_1, X_2, \ldots, X_n$ and let $X_0 \to X_1 X_2 \ldots X_n$ be a syntax rule (r). Then it holds for the values of attributes that belong to nodes u_k, $0 \leq k \leq n$ that: if $t := f_{rtk}(a_1, a_2, \ldots, a_m)$ is a semantic rule for calculation of value of attribute t that belongs to node u_k, then the value of the attribute t is given by this semantic rule.

Attributed translation grammar

Definition
Attributed translation grammar $ATG = (TG, A, V, F)$, where:

- $TG = (N, T, D, R, S_o)$ – *base translation grammar*, where R:
  $$(r)\ X_0 \to X_1 X_2 \ldots X_{n_r},$$
  where $n_r \geq 0$, $X_0 \in N$, $X_k \in (N \cup T \cup D)$ for $1 \leq k \leq n_r$.
- A – a finite set of attributes ($A = S \cup I, S \cap I = \emptyset$, S is a set of synthesized attributes, I is a set of inherited attributes). For every attribute a its *range* $H(a)$ is given.
- V – mapping: to every $X \in N$ it maps a set $A(X) \in A$. The input symbols have synthesized attributes, output symbols have inherited attributes.
- F – finite set of semantic rules. $\forall X_k$ $(1 \leq k \leq n_r)$ on the right-hand side of rule $r \in R$ and its inherited attribute d:
  $d := f_{rdk}(a_1, a_2, ..., a_m)$, where $a_1, a_2, \ldots, a_m$ are attributes of symbols in rule r. $\forall$ synthesized attribute s of symbol X_o on the left-hand side of rule $r \in R$:
  $s := f_{rso}(a_1, a_2, \ldots, a_m)$, where $a_1, a_2, \ldots, a_m$ are

Attributed translation grammar

Determining the values of all attributes is not possible unless:

- the values of inherited attributes of the start symbol are given,
- the values of synthesized attributes of input symbols are given.

Regular attributed translations

Definition
ATG is *regular* ATG if it holds that:

1. The base translation grammar is regular.
2. Nonterminal symbols have only inherited attributes.

Regular attributed translations

Example

Translation of decimal numbers: $RTG = (\{C\}, \{d\}, \{ⓥ\}, P, C)$,
where P: $\quad C \to dC|dⓥ$.

Symbols	Inherited attributes	Synthesized attributes
d		*code*
C	$value$	
ⓥ	$value$	

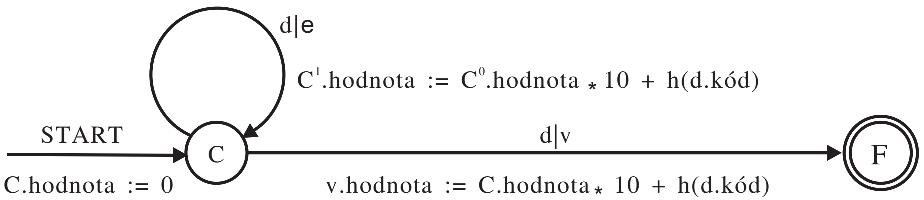
Syntax	Semantics
$C^0 \to dC^1$	$C^1.value := C^0.value * 10 + h(d.\textit{code})$
$C \to dⓥ$	$ⓥ.value := C.value * 10 + h(d.\textit{code})$

Function $h(x)$ has one argument – code of a digit – whose value is decimal value of the digit. The initial value of attribute $C.value$ of the start symbol is 0.

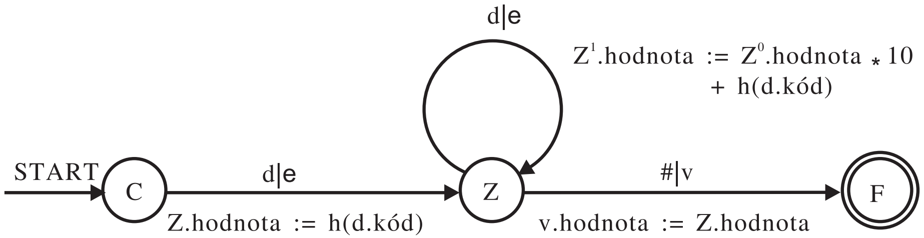
Regular attributed translations

Example (continued)

NFT



DFT



Regular attributed translations

Example (continued)

Modified translation grammar

Syntax	Semantics
$C \to dZ$	$Z.value := h(d.\textit{code})$
$Z^0 \to dZ^1$	$Z^1.value := Z^0.value * 10 + h(d.\textit{code})$
$Z \to \#ⓥ$	$ⓥ.value := Z.value$

Regular attributed translations

Example

RATG for a model of a calculator with keys 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, $+$, $*$, $=$. Input are the expressions with operators $+$ and $*$. Symbol $=$ will be the end of the expression. Operators $+$ a $*$ have the same priority.
two registers = two attributes x and y

$$RTG = (\{S, P, A, M, N\}, \{d, +, *, =\}, \{ⓥ\}, P, S).$$

Notation $d.h$ expresses the value of the input digit.

Regular attributed translations

Example (continued)

Syntax	Semantics	
$S \rightarrow dP$	$P.x := d.h$	$P.y := 0$
$P^0 \rightarrow dP^1$	$P^1.x := P^0.x * 10 + d.h$	$P^1.y := P_0.y$
$P \rightarrow + A$	$A.x := P.x + P.y$	
$P \rightarrow * M$	$M.x := P.x + P.y$	
$P \rightarrow = ⓥ$	$ⓥ.x := P.x + P.y$	
$A \rightarrow dP$	$P.x := d.h$	$P.y := A.x$
$M \rightarrow dN$	$N.x := d.h$	$N.y := M.x$
$N^0 \rightarrow dN^1$	$N^1.x := N^0.x * 10 + d.h$	$N^1.y := N_0.y$
$N \rightarrow + A$	$A.x := N.x * N.y$	
$N \rightarrow * M$	$M.x := N.x * N.y$	
$N \rightarrow = ⓥ$	$ⓥ.x := N.x * N.y$	