- Another approach to specify the meaning of a Programming Language
- More* widely used method.
  * if it is used.

IDEA:
For each syntactic entity in a programming Language, define a function that maps entity to a known mathematical object.

M: Syntactic entity → known Mathematical Object.

◆ Example

Alphabet (Terminals) { • : ⁝ ⁞ | || ||| (shell) a b c d e ... z , ← △ ▽ ▲ ௐ ⊔ ⅅ ஸ ஸ
}

- Constants (Literals)

Grammar
DI → • | : | ⁝ | ⁞
DV → | | || | |||
C → DI | DV | DI DV | (shell)

Meaning:
$M_{DI}(\bullet) = 1$ $M_{DV}(|) = 5$
$M_{DI}(:) = 2$ $M_{DV}(||) = 10$
$M_{DI}(⁝) = 3$ $M_{DV}(|||) = 15$
$M_{DI}(⁞) = 4$

$M_C(DI) = M_{DI}(DI)$ $M_C(DI\ DV) = M_{DI}(DI) + M_{DV}(DV)$
$M_C(DV) = M_{DV}(DV)$ $M_C(\text{shell}) = 0$

E.G strings from C
"⁝||" "(shell)" "|" "•|||"
13 0 5 16

$M_C(⁝||) = M_{DI}(⁝) + M_{DV}(||)$
$= 3 + 10$
$= 13$

$M_C(\bullet|||) = M_{DI}(\bullet) + M_{DV}(|||)$
$= 1 + 15$
$= 16$

Grammar
Num → C | Num , C

Meaning:
$M_{Num}(C) = M_C(C)$
$M_{Num}(Num, C) = M_C(C) + 20 * M_{Num}(Num)$

Base 20, "Vigesimal"

E.G Strings from Num
"||" "⁝||, (shell)" "•, |, ⁝|" "•, (shell), (shell)"
10 260 507 400

$M_{Num}(\bullet, |, ⁝|) = M_C(⁝|) + 20 \cdot M_{Num}(\bullet, |)$
$= 7 + 20 \cdot (M_C(|) + 20 \cdot M_{Num}(\bullet))$
$= 7 + 20 \cdot 5 + 20 \cdot 20 \cdot M_C(\bullet)$
$= 7 + 20 \cdot 5 + 20^2 \cdot 1$
$= 507$

• Variables:

Grammar:
Var_name → a | b | c | d | e | ... | z

The meaning of a variable is dependent on STATE
STATE: a set of (var_name, value) pairs

the meaning of a variable is dependent on STATE

STATE: a set of ⟨var_name, value⟩ pairs
When var_name is unique: "mapping"
(Let the domain be integers)

$s = \{ \langle a, 8 \rangle \langle b, 13 \rangle \langle x, 9 \rangle \}$

Meaning:

$M_{var}(\text{var\_name}, s: \text{state}) = \text{val s.t. } \langle \text{var\_name}, \text{val} \rangle \in S$

e.g.

$M_{var}(r, \{ \langle p, 8 \rangle, \langle r, 9 \rangle, \langle q, 10 \rangle \}) = 9$

- Expressions:

Grammar.

$\text{Expr} \rightarrow \text{var\_name} \mid \text{Num} \mid \text{Expr} \vartriangle \text{Expr} \mid \text{Expr} \triangledown \text{Expr} \mid \text{Expr} \blacktriangle \text{Expr}$

Expr. Strings.
"a" "•|" "•| △ •" "|| ▽ c" ":| ▲ :"

÷ ||
% &&
a * b
a ** b a ^ b

Meaning:

$M_{Expr}(\text{var\_name}, s) = M_{var}(\text{var\_name}, s)$

$M_{Expr}(\text{Num}, s) = M_{num}(\text{Num})$

$M_{Expr}(\text{Expr}_1 \vartriangle \text{Expr}_2, S) = M_{Exp}(\text{Expr}_1, S) + M_{Exp}(\text{Expr}_2, S)$

$M_{Expr}(\text{Expr}_1 \triangledown \text{Expr}_2, S) = M_{Exp}(\text{Expr}_1, S) - M_{Exp}(\text{Expr}_2, S)$

$M_{Expr}(\text{Expr}_1 \blacktriangle \text{Expr}_2, S) = M_{Exp}(\text{Expr}_1, S) \cdot M_{Exp}(\text{Expr}_2\ S)$

Example

$M_{Expr}(|| \triangledown c, S_1) \qquad S_1 = \{ \langle a, 3 \rangle \langle c, 2 \rangle \langle x, 7 \rangle \}$

$= M_{Exp}(||, S_1) - M_{Exp}(c, S_1)$

$= M_{NUM}(||) - M_{var}(c, S_1)$

$= 10 - 2$

$= 8$

- ASSIGNMENT

Grammar:

$A \rightarrow \text{var\_name} \leftarrow \text{Expr}.$

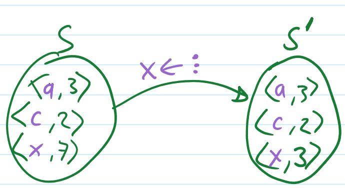
assignment strings:
"x ← :"
"y ← : ▲ x △ :||"
"e ← :| ▽ | △ y"

Meaning:

$M_A(\text{var\_name} \leftarrow \text{Expr}, s) = s'$

$s' = (s - \{ \langle \text{var\_name}, v \rangle \}) \cup \{ \langle \text{var\_name}, M_{Exp}(\text{Expr}, s) \rangle \}$

E.g.



E.g.

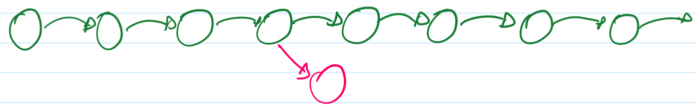
$M_A(x \leftarrow y \blacktriangle \vdots\ , s_1) = s'$ $\quad s_1 = \{\langle x,7\rangle\langle y,4\rangle\langle z,9\rangle\}$

$s' = (s - \{x,7\}) \cup \{\langle x, M_{Exp}(y \blacktriangle \vdots, s_1)\rangle\}$

$= \{\langle y,4\rangle\langle z,9\rangle\} \cup \{\langle x, \underbrace{M_{Exp}(y, s_1)}_{4} \bullet \underbrace{M_{Exp}(\vdots, s_1)}_{3}\rangle\}$

$\cup \{\langle x, 12\rangle\}$

$= \{\langle y,4\rangle\langle z,9\rangle\langle x,12\rangle\}$



- CONDITIONALS

Grammar:

$If \rightarrow$ ♋ Expr ⊔ a_1 ⊔ a_2 $\quad$ "♋ $\underbrace{x \triangledown 1}_{Expr}$ ⊔ $\underbrace{x \leftarrow \vdots \blacktriangle \vdots}_{a_1}$ ⊔ $\underbrace{x \leftarrow 0}_{a_2}$"

Meaning Function:

M_{If}(♋ Expr ⊔ a_1 ⊔ a_2 $, s) = s'$

$$s' = \begin{cases} M_A(a_1, s) & \text{if } M_{Expr}(Expr, s) \neq 0 \\ M_A(a_2, s) \end{cases}$$

Example:

M_{If}(♋ $\underbrace{x \triangledown 1}_{1}$ ⊔ $x \leftarrow \vdots \blacktriangle \vdots 1$ ⊔ $x \leftarrow 0$", $s)$ $\quad s = \{\langle x, 6\rangle\langle y, 9\rangle\}$

$M_{Expr}(x \triangledown 1, s)$

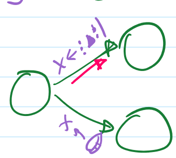
$= M_{Expr}(x, s) - M_{Exp}(1, s)$

$= \quad 6 \quad - \quad 5$

$= 1$

$s' = M_A(x \leftarrow \vdots \blacktriangle \vdots\ , s)$ which depends on $M_{Exp}(\vdots \blacktriangle \vdots 1, s)$ $= 21$

$s' = \{\langle x, 21\rangle, \langle y, 9\rangle\}$



- ITERATION

Grammar: $\quad$ " ♑ y ♌ y ← y ▽ • "

W → ♑ Expr ♌ a

Meaning:

Meaning:

$$M_W(\text{ooT Expr } \omega\ a,\ s) = s'$$

$$s' \begin{cases} s & M_{Expr}(Expr) = 0 \\ M_W(\text{ooT Expr } \omega\ a,\ M_A(a, s)) \end{cases}$$

Example:

$$M_W(\text{'ooT } y\ \omega\ y \leftarrow y \nabla \bullet,\ \{\langle y, 2\rangle\})$$

$M_{Expr}(y, \{\langle y, 2\rangle\}) = 2$

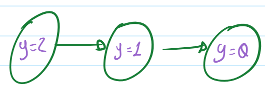
$$= M_W(\text{ooT } y\ \omega\ y \leftarrow y \nabla \bullet,\ \underbrace{M_A(y \leftarrow y \nabla \bullet, \langle y, 2\rangle)}_{\langle y, 1\rangle})$$

$M_{Expr}(y, \langle y, 1\rangle) = 1$

$$= M_W(\text{ooT } y\ \omega\ y \leftarrow y \nabla \bullet,\ \underbrace{M_A(y \leftarrow y \nabla \bullet, \langle y, 1\rangle)}_{\langle y, 0\rangle})$$

$M_{Expr}(y, \langle y, 0\rangle) = 0$

$$= \{\langle y, 0\rangle\}$$



while (y)
  y = y - 1

Fibbonacci n

```
q ← 0
p ← •
ooT n ω
    p ← p Δ q
    q ← p ∇ q
    n ← n ∇ •
```

——o—— EOL