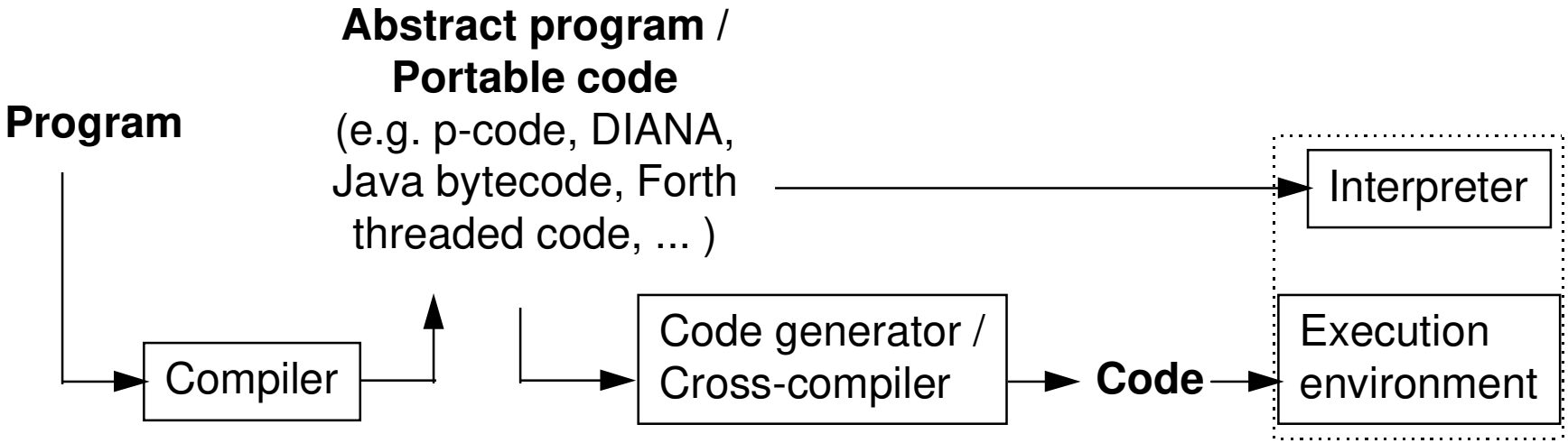
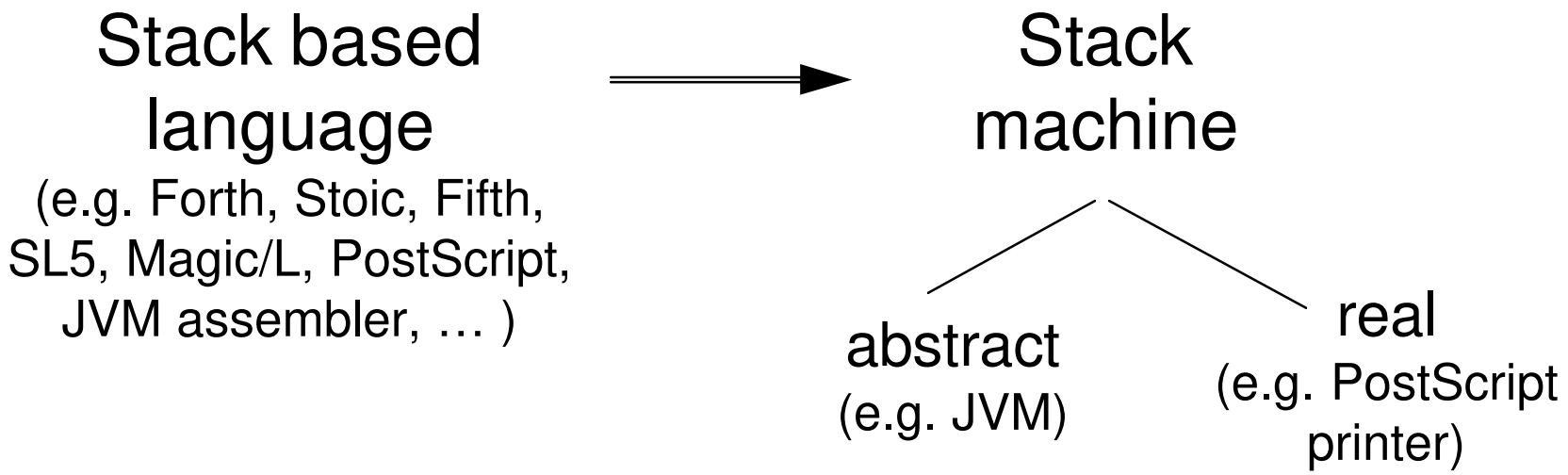
Alternative Syntactic Methods for Defining Stack Based Languages

Jaanus Pöial
Institute of Computer Science
University of Tartu, Estonia
e-mail: **jaanus@cs.ut.ee**



Language ⟹ Abstract machine ⟹ Real machine



- Data exchange through the stack(s)
- “Postfix” code for stack operations
- Extensibility

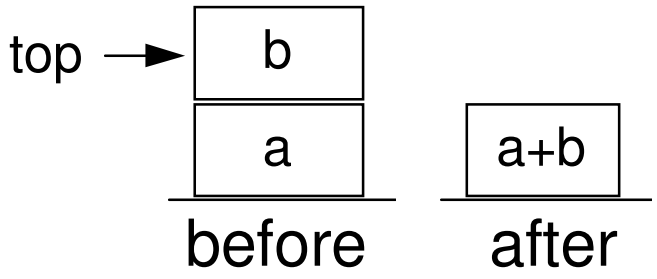
Main goal: early checking for possible problems when using stack machines and stack based languages

- Type checking in stack based languages
- Stack code validation (industrial and safety critical applications, validation of Java bytecode at receivers end, …)
- “Postfix” is hard to read and maintain by human programmer => programming tools
- Validation of code generation tools

Syntax of stack based languages

- **Context free grammars** - hardly ever used
- **Dictionary of stack operations** - most common but quite informal

	OPERATION	STACK EFFECT	DESCRIPTION
e.g.	**`plus`**	(a b -- a+b)	add two topmost elements



- **Stack effect calculus** - method for type checking based on formal descriptions of stack operations (attempt to create a formal model for the stack effect part of the dictionary)

- **Syntactic equations** - valid programs are defined by valid code patterns and valid substitutions

Example

Dictionary (stack effects)

p (-- α) push α to the stack

q (α --) pop α from the stack

Context free grammar (Λ denotes empty string)

$S_0 \rightarrow S \mid \Lambda$

$S \rightarrow pq \mid pSq \mid SS$

Syntactic equation pq = Λ

Stack effect calculus

T - operand types (`int, float, double, ...`)

$\mathbf{T}^*$ - type lists (last type on the top)

0 - type clash symbol (stack error)

The set of stack effects:

$$\mathbf{S} = (\mathbf{T}^* \times \mathbf{T}^*) \cup \{ \mathbf{0} \}$$

$$(\alpha \text{ -- } \beta)$$

input parameters (types) output parameters (types)

1 = (--) empty stack effect (no inputs, no outputs)

Composition (multiplication)

For all s in **S**: $s\mathbf{0} = \mathbf{0}s = \mathbf{0}$

For all $s_1, s_2, t_1, t_2, \alpha, \beta$ in $\mathbf{T}^*$:

$$\begin{cases} (s_1 \text{ -- } s_2)(\alpha s_2 \text{ -- } t_2) = (\alpha s_1 \text{ -- } t_2) \\ (s_1 \text{ -- } \beta t_1)(t_1 \text{ -- } t_2) = (s_1 \text{ -- } \beta t_2) \\ \mathbf{0}, \text{ otherwise} \end{cases}$$

0 is zero and **1** is unity for this operation

S is polycyclic monoid (inverse semigroup)

Examples

```
iadd            ( int int -- int )
fastore         ( address int float -- )
```

(a -- b c)(c -- d e) = (a -- b d e)

(a b -- c)(d c -- e) = (d a b -- e)

(a -- b)(b c -- d) = **0**

(-- a b)(a b --) = (--) = **1**

Inverse element:

$s = \mathbf{0} \quad \Rightarrow \quad s^{-1} = \mathbf{0}$

$s = (s_1 \text{ -- } s_2) \quad \Rightarrow \quad s^{-1} = (s_2 \text{ -- } s_1)$

Partial order (for all inverse semigroups):

$$s \leq t \iff st^{-1} = ss^{-1}$$

In case of stack effects

for each s in **S**: $\mathbf{0} \leq s$

for each α,β,γ in $\mathbf{T}^*$: $(\gamma\alpha \text{ -- } \gamma\beta) \leq (\alpha \text{ -- } \beta)$

The following assertions are equivalent:

$(s_1 \text{ -- } s_2) \leq (t_1 \text{ -- } t_2)$

$\exists \alpha \in \mathbf{T}^*\colon\ (s_1 \text{ -- } s_2) = (\alpha t_1 \text{ -- } \alpha t_2)$

$(\text{ -- } s_1)\ (t_1 \text{ -- } t_2)\ (s_2 \text{ -- }) = \mathbf{1}$

$(\text{ -- } s_1)\ (t_1 \text{ -- } t_2) = (\text{ -- } s_2)$

$(t_1 \text{ -- } t_2)\ (s_2 \text{ -- }) = (s_1 \text{ -- })$

Validation of stack machine programs

Δ - set of stack operations

Δ^* - set of "programs"

Let us define $sig(p) \in \mathbf{S}$ for each program p:

$sig(\Lambda) = \mathbf{1}$

$sig(pq) = sig(p)\ sig(q)$

sig is homomorphism that maps concatenation of stack operations ("a program") into the composition of stack effects ("specification"): $sig: \Delta^* \rightarrow \mathbf{S}$

$$Valid\ (\Delta, sig) = \{\ \omega \in \Delta^* |\ sig(\omega) \neq \mathbf{0}\}$$

$$Closed\ (\Delta, sig) = \{\ \omega \in \Delta^* |\ sig(\omega) = \mathbf{1}\}$$

$$\Lambda \in Closed(\Delta, sig) \subseteq Valid(\Delta, sig) \subseteq \Delta^*$$

Other definitions for this language?

Syntactic equations

$$\omega_1 = \omega_2 \quad \Leftrightarrow \quad sig(\omega_1) = sig(\omega_2) \neq \mathbf{0}$$

Basic types of equations

$st = \mathbf{1}$	$s=(-- \alpha)\ \ t=(\alpha --)$
$rst = \mathbf{1}$	$r=(-- \alpha\beta)\ \ s=(\beta -- \gamma)\ \ t=(\alpha\gamma --)$
$rs = r$	$r=(\alpha -- \beta\gamma)\ \ s=(\gamma -- \gamma)$
$st = t$	$s=(\alpha -- \alpha)\ \ t=(\beta\alpha -- \gamma)$
$s \leq rst$	$\alpha r_1=s_1\ \ \alpha r_2=\beta s_1\ \ \gamma t_1=\beta s_2\ \ \gamma t_2=s_2$