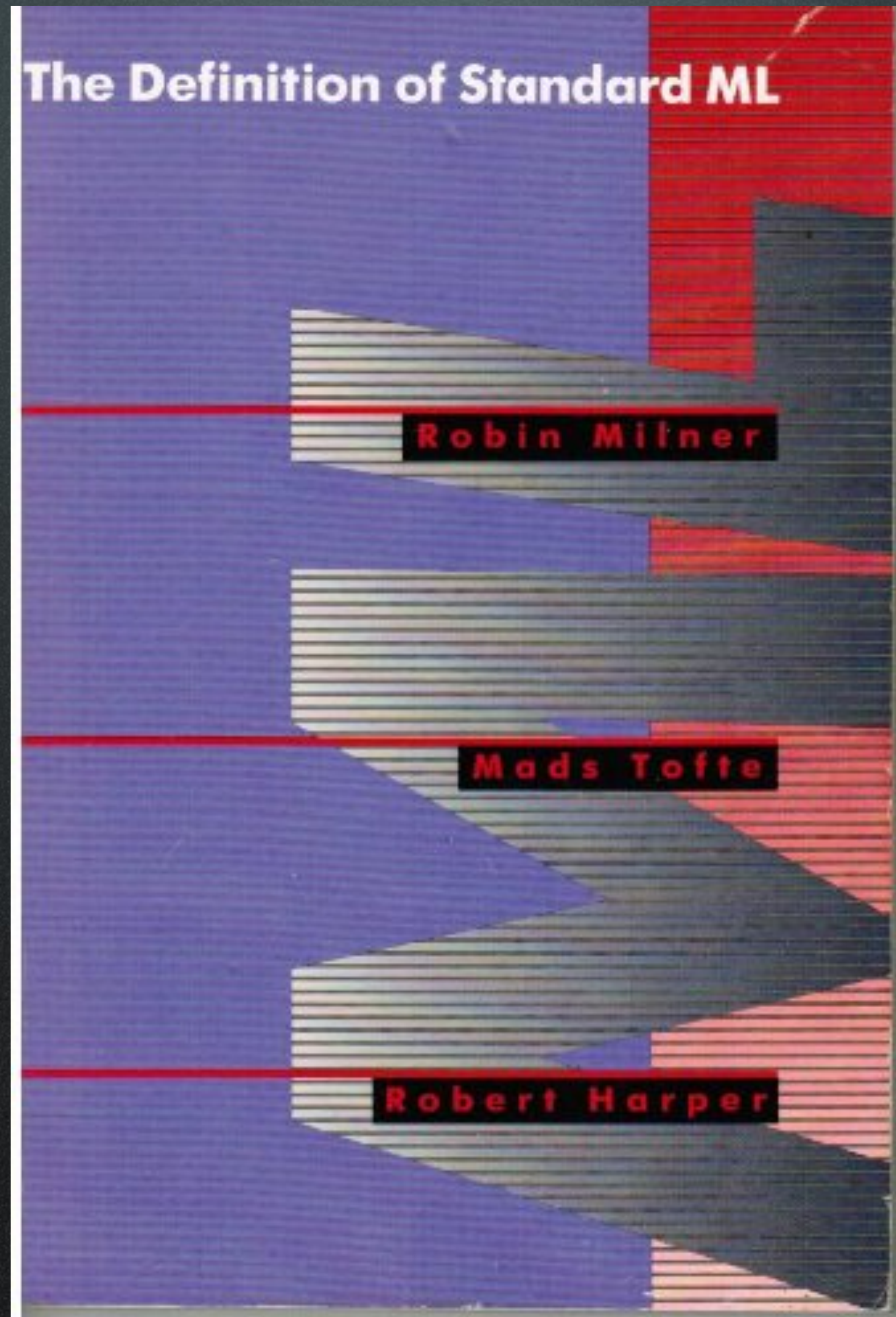
Language formalisation goes mainstream

Verified Software Workshop 2019

Andreas Rossberg
Dfinity


The Definition of Standard ML
Robin Milner
Mads Tofte
Robert Harper

Language formalisation goes mainstream

Verified Software Workshop 2019

Andreas Rossberg

Dfinity

WebAssembly
Language formalisation goes mainstream

Verified Software Workshop 2019

Andreas Rossberg
Dfinity

WebAssembly

Wasm

~~WASM~~

Wasm

a virtual instruction set architecture

that is fully sandboxed

and freely embeddable anywhere

a low-level virtual machine

open standard (W3C, github)

Goals & Constraints

Semantics

Language-independent

Platform-independent

Hardware-independent

Fast to execute

Safe to execute

Deterministic

Easy to reason about

Representation

Compact

Easy to generate

Fast to decode

Fast to validate

Fast to compile

Streamable

Parallelisable

modular & sandboxed

binaries are modules

function from explicit imports to exports

encapsulated, no access to internals

sandboxed, no ambient capabilities

type system

simple, linear-time type system

type-safe and memory-safe

no undefined behaviour

custom loader could implement extensions

formal semantics

from the get-go!

textbook techniques

meta-mission: raise the bar
for real-world language specs

static semantics

typing, a.k.a. validation

inference rules

$\Gamma \vdash e^* : t_1^* \rightarrow t_2^*$

stack machine

(**i32.const** 42)	: ε → i32
(**local.get** $x)	: ε → Γ($x)
(**i32.add**)	: i32 i32 → i32

} : ε → i32

$$\frac{}{\Gamma \vdash t.\mathbf{const}\ c : \varepsilon \rightarrow t}$$

$$\frac{}{\Gamma \vdash t.\mathbf{add} : t\ t \rightarrow t}$$

$$\frac{\Gamma(\$x) = t}{\Gamma \vdash \mathbf{local.get}\ \$x : \varepsilon \rightarrow t}$$

$$\frac{\Gamma \vdash e_0 : t_1{*} \rightarrow t_3{*} \qquad \Gamma \vdash e{*} : t_3{*} \rightarrow t_2{*}}{\Gamma \vdash e_0\ e{*} : t_1{*} \rightarrow t_2{*}}$$

$$\frac{\Gamma \vdash e{*} : t_1{*} \rightarrow t_2{*}}{\Gamma \vdash e{*} : t_0{*}\ t_1{*} \rightarrow t_0{*}\ t_2{*}}$$

structured control flow

$\$l : t_2{}^*$

(**block** $\$l\ (t_1{}^* \to t_2{}^*)$

…

(**br** $\$l$) $: t_2{}^* \to \bot$

…

end)

$\$l : t_1{}^*$

(**loop** $\$l\ (t_1{}^* \to t_2{}^*)$

…

(**br** $\$l$) $: t_1{}^* \to \bot$

…

end)

$$\frac{\Gamma, \$l : t_2^* \vdash e^* : t_1^* \rightarrow t_2^*}{\Gamma \vdash \textbf{block}\ \$l\ e^*\ \textbf{end} : t_1^* \rightarrow t_2^*}$$

$$\frac{\Gamma(\$l) = t^*}{\Gamma \vdash \textbf{br}\ \$l : t^* \rightarrow \bot}$$

$$\frac{\Gamma, \$l : t_1^* \vdash e^* : t_1^* \rightarrow t_2^*}{\Gamma \vdash \textbf{loop}\ \$l\ e^*\ \textbf{end} : t_1^* \rightarrow t_2^*}$$

$$\frac{\Gamma, \$l : t_2^* \vdash e^* : t_1^* \rightarrow t_2^*}{\Gamma \vdash \textbf{block}\ \$l\ e^*\ \textbf{end} : t_1^* \rightarrow t_2^*}$$

$$\frac{\Gamma(\$l) = t^*}{\Gamma \vdash \textbf{br}\ \$l : t^* \rightarrow \bot}$$

$$\frac{\Gamma, \$l : t_1^* \vdash e^* : t_1^* \rightarrow t_2^*}{\Gamma \vdash \textbf{loop}\ \$l\ e^*\ \textbf{end} : t_1^* \rightarrow t_2^*}$$

$$\frac{\Gamma, \$l : t_2^* \vdash e^* : t_1^* \to t_2^*}{\Gamma \vdash \textbf{block}\ \$l\ e^*\ \textbf{end} : t_1^* \to t_2^*}$$

$$\frac{\Gamma(\$l) = t^*}{\Gamma \vdash \textbf{br}\ \$l : t_1^*\ t^* \to t_2^*}$$

$$\frac{\Gamma, \$l : t_1^* \vdash e^* : t_1^* \to t_2^*}{\Gamma \vdash \textbf{loop}\ \$l\ e^*\ \textbf{end} : t_1^* \to t_2^*}$$

dynamic semantics

structured operational semantics

small-step reduction rules

$s; e^* \longmapsto^* s'; e'^*$

$$(\text{t.}\textbf{const}\ c_1)\ (\text{t.}\textbf{const}\ c_2)\ \text{t.}\textbf{add} \quad \longmapsto \quad (\text{t.}\textbf{const}\ c_1{+}c_2)$$

(values)	$v ::= (\text{t.}\textbf{const}\ c)$
(eval contexts)	$E ::= v^*\ [\,]\ e^* \mid (\textbf{label}\{e^*\}\ E\ \textbf{end})$

$$(\textbf{block}\ e^*\ \textbf{end}) \longmapsto (\textbf{label}\{\varepsilon\}\ e^*\ \textbf{end})$$

$$(\textbf{loop}\ e^*\ \textbf{end}) \longmapsto (\textbf{label}\{\textbf{loop}\ e^*\ \textbf{end}\}\ e^*\ \textbf{end})$$

$$(\textbf{label}\{e^*\}\ v^*\ \textbf{end}) \longmapsto v^*$$

$$(\textbf{label}\{e^*\}\ L^n[v^*\ (\textbf{br}\ n)]\ \textbf{end}) \longmapsto v^*\ e^*$$

$$\text{(label contexts)}\quad L^0 ::= v^*\ [\,]\ e^*$$

$$L^{n+1} ::= v^*\ (\textbf{label}\{e^*\}\ L^n\ \textbf{end})\ e^*$$

$$\frac{s.\mathrm{mem}[i..i+|t|-1] = \mathrm{bytes}_t(c)}{s;\ (\mathbf{i32.const}\ i)\ t.\mathbf{load} \quad \longmapsto \quad s;\ (t.\mathbf{const}\ c)}$$

$$\frac{s' = s\ \mathrm{with}\ \mathrm{mem}[i..i+|t|-1] = \mathrm{bytes}_t(c)}{s;\ (\mathbf{i32.const}\ i)\ (t.\mathbf{const}\ c)\ t.\mathbf{store} \quad \longmapsto \quad s';\ \varepsilon}$$

soundness

If $\vdash e^* : \varepsilon \rightarrow t^*$ and $\vdash s$,

then $s; e^* \longmapsto^* s'; v^*$

and $\vdash v^* : \varepsilon \rightarrow t^*$ and $\vdash s'$.

$$
\begin{array}{lrcl}
\text{(value types)} & t & ::= & \mathsf{i32} \mid \mathsf{i64} \mid \mathsf{f32} \mid \mathsf{f64} \\
\text{(packed types)} & pt & ::= & \mathsf{i8} \mid \mathsf{i16} \mid \mathsf{i32} \\
\text{(function types)} & ft & ::= & t^* \rightarrow t^* \\
\\
 & unop & ::= & \textbf{neg} \mid \textbf{abs} \mid \ldots \\
 & binop & ::= & \textbf{add} \mid \textbf{sub} \mid \textbf{mul} \mid \textbf{div\_s} \mid \textbf{div\_u} \mid \ldots \\
 & relop & ::= & \textbf{eq} \mid \textbf{ne} \mid \textbf{lt} \mid \textbf{gt} \mid \ldots \\
 & cvtop & ::= & \textbf{convert}/t \mid \textbf{reinterpret}/t \\
\\
\text{(instructions)} & e & ::= & t.\textbf{const}\ c \mid t.unop \mid t.binop \mid t.relop \mid t.cvtop \mid \\
 & & & \textbf{unreachable} \mid \textbf{nop} \mid \textbf{drop} \mid \textbf{select} \mid \\
 & & & \textbf{block}\ ft\ e^*\ \textbf{end} \mid \textbf{loop}\ ft\ e^*\ \textbf{end} \mid \textbf{if}\ ft\ e^*\ \textbf{else}\ e^*\ \textbf{end} \mid \\
 & & & \textbf{br}\ i \mid \textbf{br\_if}\ i \mid \textbf{br\_table}\ i^*\ i \mid \\
 & & & \textbf{call}\ i \mid \textbf{call\_indirect}\ ft \mid \textbf{return} \mid \\
 & & & \textbf{get\_local}\ i \mid \textbf{set\_local}\ i \mid \textbf{tee\_local}\ i \mid \\
 & & & \textbf{get\_global}\ i \mid \textbf{set\_global}\ i \mid \\
 & & & t.\textbf{load}\ pt^?\ n \mid t.\textbf{store}\ pt^?\ n \mid \textbf{current\_mem} \mid \textbf{grow\_mem} \\
\\
\text{(functions)} & func & ::= & \textbf{func}\ ft\ (\textbf{local}\ t)^*\ e^* \\
\text{(globals)} & glob & ::= & \textbf{global}\ \textbf{mut}^?\ t\ e^* \\
\text{(tables)} & tab & ::= & \textbf{table}\ n\ i^* \\
\text{(memories)} & mem & ::= & \textbf{memory}\ n \\
\text{(modules)} & m & ::= & \textbf{module}\ import^*\ func^*\ glob^*\ tab^?\ mem^?\ export^*
\end{array}
$$

(store)	s	$::=$	$\{\mathsf{inst}\ inst^*,\ \mathsf{tab}\ tabinst^*,\ \mathsf{mem}\ meminst^*\}$	
(instances)	$inst$	$::=$	$\{\mathsf{func}\ cl^*,\ \mathsf{glob}\ v^*,\ \mathsf{tab}\ i^?,\ \mathsf{mem}\ i^?\}$	
	$tabinst$	$::=$	cl^*	
	$meminst$	$::=$	b^*	
(closures)	cl	$::=$	$\{\mathsf{inst}\ i,\ \mathsf{code}\ f\}$	(where f is not an import and has all exports ex^* erased)
(values)	v	$::=$	$t.\mathbf{const}\ c$	
(administrative operators)	e	$::=$	$\dots \mid \mathbf{trap} \mid \mathbf{call}\ cl \mid \mathbf{label}\{t^*;e^*\}\ e^*\ \mathbf{end} \mid \mathbf{local}\{i;v^*\}\ e^*\ \mathbf{end}$	
(local contexts)	L^0	$::=$	$v^*\ [\_]\ e^*$	
	L^{k+1}	$::=$	$v^*\ \mathbf{label}\{t^*;e^*\}\ L^k\ \mathbf{end}\ e^*$	

Reduction $\dfrac{s;v^*;e^* \hookrightarrow_i s';v'^*;e'^*}{s;v^*;L^k[e^*] \hookrightarrow_i s';v'^*;L^k[e'^*]}$ $\dfrac{s;v^*;e^* \hookrightarrow_i s';v'^*;e'^*}{s;v_0^*;\mathbf{local}\{i;v^*\}\ e^*\ \mathbf{end} \hookrightarrow_j s';v_0^*;\mathbf{local}\{i;v'^*\}\ e'^*\ \mathbf{end}}$ $\boxed{s;v^*;e^* \hookrightarrow_i s;v^*;e^*}$

$(t.\mathbf{const}\ c)\ t.unop$	$\hookrightarrow$	$t.\mathbf{const}\ unop_t(c)$	
$(t.\mathbf{const}\ c_1)\ (t.\mathbf{const}\ c_2)\ t.binop$	$\hookrightarrow$	$t.\mathbf{const}\ c$	if $c = binop_t(c_1,c_2)$
$(t.\mathbf{const}\ c_1)\ (t.\mathbf{const}\ c_2)\ t.binop$	$\hookrightarrow$	$\mathbf{trap}$	otherwise
$(t.\mathbf{const}\ c)\ t.testop$	$\hookrightarrow$	$\mathbf{i32.const}\ testop_t(c)$	
$(t.\mathbf{const}\ c_1)\ (t.\mathbf{const}\ c_2)\ t.relop$	$\hookrightarrow$	$\mathbf{i32.const}\ relop_t(c_1,c_2)$	
$(t_1.\mathbf{const}\ c)\ t_2.\mathbf{convert}\ t_1\_sx^?$	$\hookrightarrow$	$t_2.\mathbf{const}\ c'$	if $c' = \mathrm{cvt}^{sx^?}_{t_1,t_2}(c)$
$(t_1.\mathbf{const}\ c)\ t_2.\mathbf{convert}\ t_1\_sx^?$	$\hookrightarrow$	$\mathbf{trap}$	otherwise
$(t_1.\mathbf{const}\ c)\ t_2.\mathbf{reinterpret}\ t_1$	$\hookrightarrow$	$t_2.\mathbf{const}\ \mathrm{const}_{t_2}(\mathrm{bits}_{t_1}(c))$	
$\mathbf{unreachable}$	$\hookrightarrow$	$\mathbf{trap}$	
$\mathbf{nop}$	$\hookrightarrow$	ϵ	
$v\ \mathbf{drop}$	$\hookrightarrow$	ϵ	
$v_1\ v_2\ (\mathbf{i32.const}\ 0)\ \mathbf{select}$	$\hookrightarrow$	v_2	
$v_1\ v_2\ (\mathbf{i32.const}\ k+1)\ \mathbf{select}$	$\hookrightarrow$	v_1	
$v^n\ \mathbf{block}\ (t_1^n \to t_2^m)\ e^*\ \mathbf{end}$	$\hookrightarrow$	$\mathbf{label}\{t_2^m;\epsilon\}\ v^n\ e^*\ \mathbf{end}$	
$v^n\ \mathbf{loop}\ (t_1^n \to t_2^m)\ e^*\ \mathbf{end}$	$\hookrightarrow$	$\mathbf{label}\{t_1^n;\mathbf{loop}\ (t_1^n \to t_2^m)\ e^*\ \mathbf{end}\}\ v^n\ e^*\ \mathbf{end}$	
$(\mathbf{i32.const}\ 0)\ \mathbf{if}\ tf\ e_1^*\ \mathbf{else}\ e_2^*\ \mathbf{end}$	$\hookrightarrow$	$\mathbf{block}\ tf\ e_2^*\ \mathbf{end}$	
$(\mathbf{i32.const}\ k+1)\ \mathbf{if}\ tf\ e_1^*\ \mathbf{else}\ e_2^*\ \mathbf{end}$	$\hookrightarrow$	$\mathbf{block}\ tf\ e_1^*\ \mathbf{end}$	
$\mathbf{label}\{t^*;e^*\}\ v^*\ \mathbf{end}$	$\hookrightarrow$	v^*	
$\mathbf{label}\{t^*;e^*\}\ \mathbf{trap}\ \mathbf{end}$	$\hookrightarrow$	$\mathbf{trap}$	
$\mathbf{label}\{t^n;e^*\}\ L^j[v^n\ (\mathbf{br}\ j)]\ \mathbf{end}$	$\hookrightarrow$	$v^n\ e^*$	
$(\mathbf{i32.const}\ 0)\ (\mathbf{br\_if}\ j)$	$\hookrightarrow$	ϵ	
$(\mathbf{i32.const}\ k+1)\ (\mathbf{br\_if}\ j)$	$\hookrightarrow$	$\mathbf{br}\ j$	
$(\mathbf{i32.const}\ k)\ (\mathbf{br\_table}\ j_1^k\ j\ j_2^*)$	$\hookrightarrow$	$\mathbf{br}\ j$	
$(\mathbf{i32.const}\ k+n)\ (\mathbf{br\_table}\ j_1^k\ j)$	$\hookrightarrow$	$\mathbf{br}\ j$	
$s;\mathbf{call}\ j$	$\hookrightarrow_i$	$\mathbf{call}\ s_{\mathsf{func}}(i,j)$	
$s;(\mathbf{i32.const}\ j)\ \mathbf{call\_indirect}\ tf$	$\hookrightarrow_i$	$\mathbf{call}\ s_{\mathsf{tab}}(i,j)$	if $s_{\mathsf{tab}}(i,j)_{\mathsf{code}} = (\mathbf{func}\ tf\ \mathbf{local}\ t^*\ e^*)$
$s;(\mathbf{i32.const}\ j)\ \mathbf{call\_indirect}\ tf$	$\hookrightarrow_i$	$\mathbf{trap}$	otherwise
$v^n\ (\mathbf{call}\ cl)$	$\hookrightarrow$	$\mathbf{local}\{cl_{\mathsf{inst}};v^n\ (t.\mathbf{const}\ 0)^k\}\ \mathbf{block}\ (\epsilon \to t_2^m)\ e^*\ \mathbf{end}\ \mathbf{end}$ ...	
$\mathbf{local}\{i;v_l^*\}\ v^*\ \mathbf{end}$	$\hookrightarrow$	v^*	$\mid$ ... if $cl_{\mathsf{code}} = (\mathbf{func}\ (t_1^n \to t_2^m)\ \mathbf{local}\ t^k\ e^*)$
$\mathbf{local}\{i;v_l^*\}\ \mathbf{trap}\ \mathbf{end}$	$\hookrightarrow$	$\mathbf{trap}$	
$\mathbf{local}\{i;v_l^*\}\ L^{k+1}[\mathbf{return}]\ \mathbf{end}$	$\hookrightarrow$	$\mathbf{local}\{i;v_l^*\}\ L^{k+1}[\mathbf{br}\ k]\ \mathbf{end}$	
$v_1^j\ v\ v_2^k;\mathbf{get\_local}\ j$	$\hookrightarrow$	v	
$v_1^j\ v\ v_2^k;v'\ (\mathbf{set\_local}\ j)$	$\hookrightarrow$	$v_1^j\ v'\ v_2^k;\epsilon$	
$v\ (\mathbf{tee\_local}\ j)$	$\hookrightarrow$	$v\ v\ (\mathbf{set\_local}\ j)$	
$s;\mathbf{get\_global}\ j$	$\hookrightarrow_i$	$s_{\mathsf{glob}}(i,j)$	
$s;v\ (\mathbf{set\_global}\ j)$	$\hookrightarrow_i$	$s';\epsilon$	if $s' = s$ with $\mathrm{glob}(i,j) = v$
$s;(\mathbf{i32.const}\ k)\ (t.\mathbf{load}\ a\ o)$	$\hookrightarrow_i$	$t.\mathbf{const}\ \mathrm{const}_t(b^*)$	if $s_{\mathsf{mem}}(i,k+o,\lvert t\rvert) = b^*$
$s;(\mathbf{i32.const}\ k)\ (t.\mathbf{load}\ tp\_sx\ a\ o)$	$\hookrightarrow_i$	$t.\mathbf{const}\ \mathrm{const}_t^{sx}(b^*)$	if $s_{\mathsf{mem}}(i,k+o,\lvert tp\rvert) = b^*$
$s;(\mathbf{i32.const}\ k)\ (t.\mathbf{load}\ tp\_sx^?\ a\ o)$	$\hookrightarrow_i$	$\mathbf{trap}$	otherwise
$s;(\mathbf{i32.const}\ k)\ (t.\mathbf{const}\ c)\ (t.\mathbf{store}\ a\ o)$	$\hookrightarrow_i$	$s';\epsilon$	if $s' = s$ with $\mathrm{mem}(i,k+o,\lvert t\rvert) = \mathrm{bits}_t^{\lvert t\rvert}(c)$
$s;(\mathbf{i32.const}\ k)\ (t.\mathbf{const}\ c)\ (t.\mathbf{store}\ tp\ a\ o)$	$\hookrightarrow_i$	$s';\epsilon$	if $s' = s$ with $\mathrm{mem}(i,k+o,\lvert tp\rvert) = \mathrm{bits}_t^{\lvert tp\rvert}(c)$
$s;(\mathbf{i32.const}\ k)\ (t.\mathbf{const}\ c)\ (t.\mathbf{store}\ tp^?\ a\ o)$	$\hookrightarrow_i$	$\mathbf{trap}$	otherwise
$s;\mathbf{current\_memory}]$	$\hookrightarrow_i$	$\mathbf{i32.const}\ \lvert s_{\mathsf{mem}}(i,*)\rvert/64\,\mathrm{Ki}$	
$s;(\mathbf{i32.const}\ k)\ \mathbf{grow\_memory}$	$\hookrightarrow_i$	$s';\mathbf{i32.const}\ \lvert s_{\mathsf{mem}}(i,*)\rvert/64\,\mathrm{Ki}$	if $s' = s$ with $\mathrm{mem}(i,*) = s_{\mathsf{mem}}(i,*)\ (0)^{k\cdot 64\,\mathrm{Ki}}$
$s;(\mathbf{i32.const}\ k)\ \mathbf{grow\_memory}$	$\hookrightarrow_i$	$\mathbf{i32.const}\ (-1)$	

Figure 1. Small-step reduction rules

[PLDI 2017]

$$\text{(contexts)}\quad C \quad ::= \quad \{\mathsf{func}\ tf^*,\ \mathsf{global}\ tg^*,\ \mathsf{table}\ n^?,\ \mathsf{memory}\ n^?,\ \mathsf{local}\ t^*,\ \mathsf{label}\ (t^*)^*\}$$

Typing Instructions $\quad \boxed{C \vdash e^* : tf}$

$$\frac{}{C \vdash t.\mathbf{const}\ c : \epsilon \to t} \qquad \frac{}{C \vdash t.unop : t \to t} \qquad \frac{}{C \vdash t.binop : t\ t \to t} \qquad \frac{}{C \vdash t.testop : t \to \mathsf{i32}} \qquad \frac{}{C \vdash t.relop : t\ t \to \mathsf{i32}}$$

$$\frac{t_1 \neq t_2 \qquad sx^? = \epsilon \Leftrightarrow (t_1 = \mathsf{i}n \wedge t_2 = \mathsf{i}n' \wedge |t_1| < |t_2|) \vee (t_1 = \mathsf{f}n \wedge t_2 = \mathsf{f}n')}{C \vdash t_1.\mathbf{convert}\ t_2\_sx^? : t_2 \to t_1} \qquad \frac{t_1 \neq t_2 \qquad |t_1| = |t_2|}{C \vdash t_1.\mathbf{reinterpret}\ t_2 : t_2 \to t_1}$$

$$\frac{}{C \vdash \mathbf{unreachable} : t_1^* \to t_2^*} \qquad \frac{}{C \vdash \mathbf{nop} : \epsilon \to \epsilon} \qquad \frac{}{C \vdash \mathbf{drop} : t \to \epsilon} \qquad \frac{}{C \vdash \mathbf{select} : t\ t\ \mathsf{i32} \to t}$$

$$\frac{tf = t_1^n \to t_2^m \qquad C, \mathsf{label}\ (t_2^m) \vdash e^* : tf}{C \vdash \mathbf{block}\ tf\ e^*\ \mathbf{end} : tf} \qquad \frac{tf = t_1^n \to t_2^m \qquad C, \mathsf{label}\ (t_1^n) \vdash e^* : tf}{C \vdash \mathbf{loop}\ tf\ e^*\ \mathbf{end} : tf}$$

$$\frac{tf = t_1^n \to t_2^m \qquad C, \mathsf{label}\ (t_2^m) \vdash e_1^* : tf \qquad C, \mathsf{label}\ (t_2^m) \vdash e_2^* : tf}{C \vdash \mathbf{if}\ tf\ e_1^*\ \mathbf{else}\ e_2^*\ \mathbf{end} : t_1^n\ \mathsf{i32} \to t_2^m}$$

$$\frac{C_{\mathsf{label}}(i) = t^*}{C \vdash \mathbf{br}\ i : t_1^*\ t^* \to t_2^*} \qquad \frac{C_{\mathsf{label}}(i) = t^*}{C \vdash \mathbf{br\_if}\ i : t^*\ \mathsf{i32} \to t^*} \qquad \frac{(C_{\mathsf{label}}(i) = t^*)^+}{C \vdash \mathbf{br\_table}\ i^+ : t_1^*\ t^*\ \mathsf{i32} \to t_2^*}$$

$$\frac{C_{\mathsf{label}}(|C_{\mathsf{label}}| - 1) = t^*}{C \vdash \mathbf{return} : t_1^*\ t^* \to t_2^*} \qquad \frac{C_{\mathsf{func}}(i) = tf}{C \vdash \mathbf{call}\ i : tf} \qquad \frac{tf = t_1^* \to t_2^* \qquad C_{\mathsf{table}} = n}{C \vdash \mathbf{call\_indirect}\ tf : t_1^*\ \mathsf{i32} \to t_2^*}$$

$$\frac{C_{\mathsf{local}}(i) = t}{C \vdash \mathbf{get\_local}\ i : \epsilon \to t} \qquad \frac{C_{\mathsf{local}}(i) = t}{C \vdash \mathbf{set\_local}\ i : t \to \epsilon} \qquad \frac{C_{\mathsf{local}}(i) = t}{C \vdash \mathbf{tee\_local}\ i : t \to t} \qquad \frac{C_{\mathsf{global}}(i) = \mathsf{mut}^?\ t}{C \vdash \mathbf{get\_global}\ i : \epsilon \to t} \qquad \frac{C_{\mathsf{global}}(i) = \mathsf{mut}\ t}{C \vdash \mathbf{set\_global}\ i : t \to \epsilon}$$

$$\frac{C_{\mathsf{memory}} = n \qquad 2^a \leq (|tp| <)^? |t| \qquad (tp\_sz)^? = \epsilon \vee t = \mathsf{i}m}{C \vdash t.\mathbf{load}\ (tp\_sz)^?\ a\ o : \mathsf{i32} \to t} \qquad \frac{C_{\mathsf{memory}} = n \qquad 2^a \leq (|tp| <)^? |t| \qquad tp^? = \epsilon \vee t = \mathsf{i}m}{C \vdash t.\mathbf{store}\ tp^?\ a\ o : \mathsf{i32}\ t \to \epsilon}$$

$$\frac{C_{\mathsf{memory}} = n}{C \vdash \mathbf{current\_memory} : \epsilon \to \mathsf{i32}} \qquad \frac{C_{\mathsf{memory}} = n}{C \vdash \mathbf{grow\_memory} : \mathsf{i32} \to \mathsf{i32}}$$

$$\frac{}{C \vdash \epsilon : \epsilon \to \epsilon} \qquad \frac{C \vdash e_1^* : t_1^* \to t_2^* \qquad C \vdash e_2 : t_2^* \to t_3^*}{C \vdash e_1^*\ e_2 : t_1^* \to t_3^*} \qquad \frac{C \vdash e^* : t_1^* \to t_2^*}{C \vdash e^* : t^*\ t_1^* \to t^*\ t_2^*}$$

Typing Modules

$$\frac{tf = t_1^* \to t_2^* \qquad C, \mathsf{local}\ t_1^*\ t^*, \mathsf{label}\ (t_2^*) \vdash e^* : \epsilon \to t_2^*}{C \vdash ex^*\ \mathbf{func}\ tf\ \mathbf{local}\ t^*\ e^* : ex^*\ tf} \qquad \frac{tg = \mathsf{mut}^?\ t \qquad C \vdash e^* : \epsilon \to t \qquad ex^* = \epsilon \vee tg = t}{C \vdash ex^*\ \mathbf{global}\ tg\ e^* : ex^*\ tg}$$

$$\frac{(C_{\mathsf{func}}(i) = tf)^n}{C \vdash ex^*\ \mathbf{table}\ n\ i^n : ex^*\ n} \qquad \frac{}{C \vdash ex^*\ \mathbf{memory}\ n : ex^*\ n}$$

$$\frac{}{C \vdash ex^*\ \mathbf{func}\ tf\ im : ex^*\ tf} \qquad \frac{tg = t}{C \vdash ex^*\ \mathbf{global}\ tg\ im : ex^*\ tg} \qquad \frac{}{C \vdash ex^*\ \mathbf{table}\ n\ im : ex^*\ n} \qquad \frac{}{C \vdash ex^*\ \mathbf{memory}\ n\ im : ex^*\ n}$$

$$\frac{\begin{array}{c}(C \vdash f : ex_{\mathrm{f}}^*\ tf)^* \qquad (C_i \vdash glob_i : ex_{\mathrm{g}}^*\ tg_i)_i^* \qquad (C \vdash tab : ex_{\mathrm{t}}^*\ n)^? \qquad (C \vdash mem : ex_{\mathrm{m}}^*\ n)^? \\ (C_i = \{\mathsf{global}\ tg^{i-1}\})_i^* \qquad C = \{\mathsf{func}\ tf^*, \mathsf{global}\ tg^*, \mathsf{table}\ n^?, \mathsf{memory}\ n^?\} \qquad ex_{\mathrm{f}}^{**}\ ex_{\mathrm{g}}^{**}\ ex_{\mathrm{t}}^{*?}\ ex_{\mathrm{m}}^{*?}\ \text{distinct}\end{array}}{\vdash \mathbf{module}\ f^*\ glob^*\ tab^?\ mem^?}$$

Figure 1. Typing rules

[PLDI 2017]

mechanisation

OCaml [myself]

Isabelle [Conrad Watt, Cambridge]

Coq (ongoing) [Dave Swasey, MPI]

K (ongoing) [Everett Hildenbrandt, Illinois]

standard

structured around formalisation

adding prose rendering for the uninitiated

webassembly.github.io/spec/

proposal process

must include spec text

must include formalisation!

must include OCaml reference interpreter

must include comprehensive test suite

must be implemented in 2 production engines

road map

v1 (shipped): support low-level languages

v2 (next+ year): support high-level languages

v3 (maybe...): support “dynamic” languages

future features

tail calls

references

threads

vector instructions

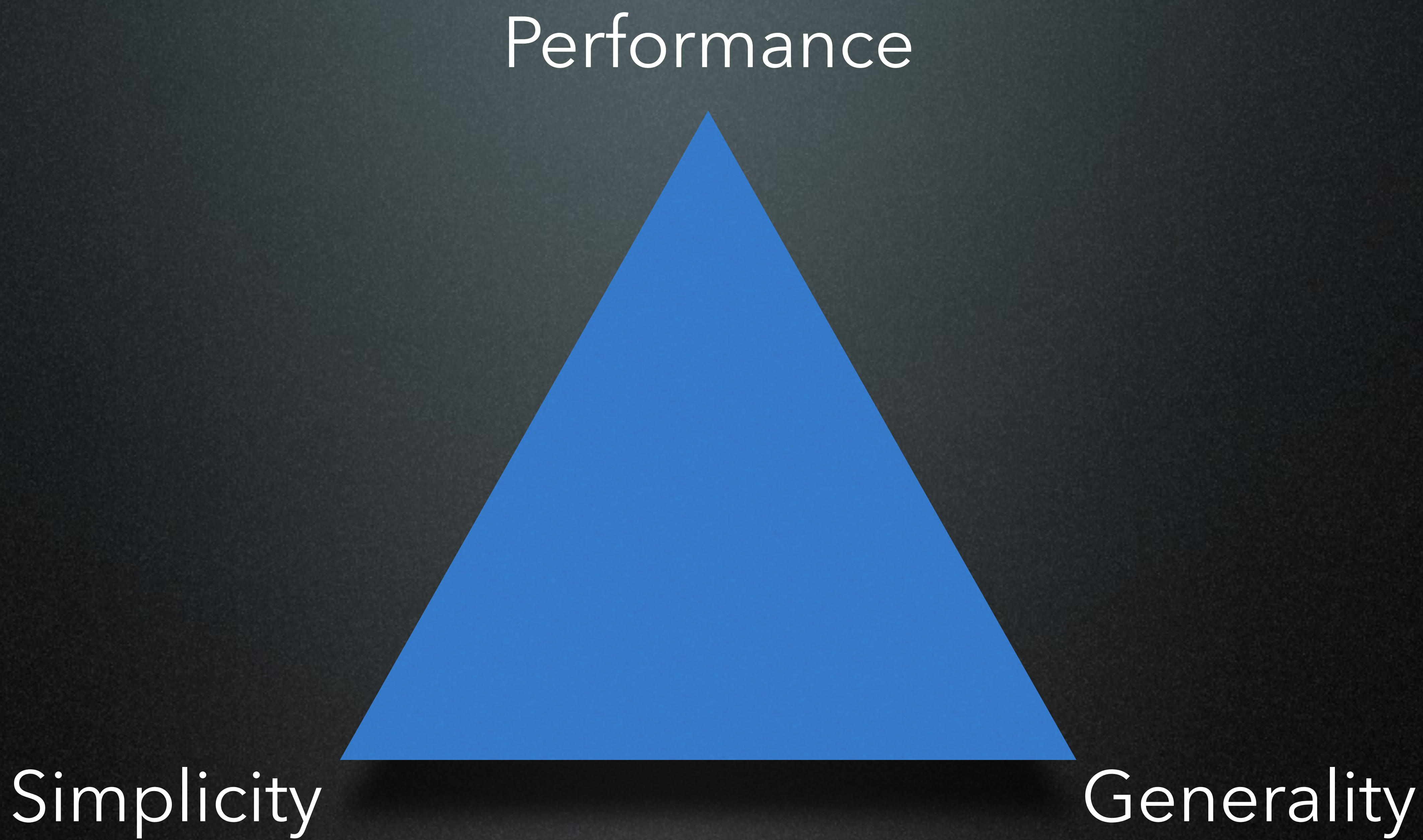
exception handling

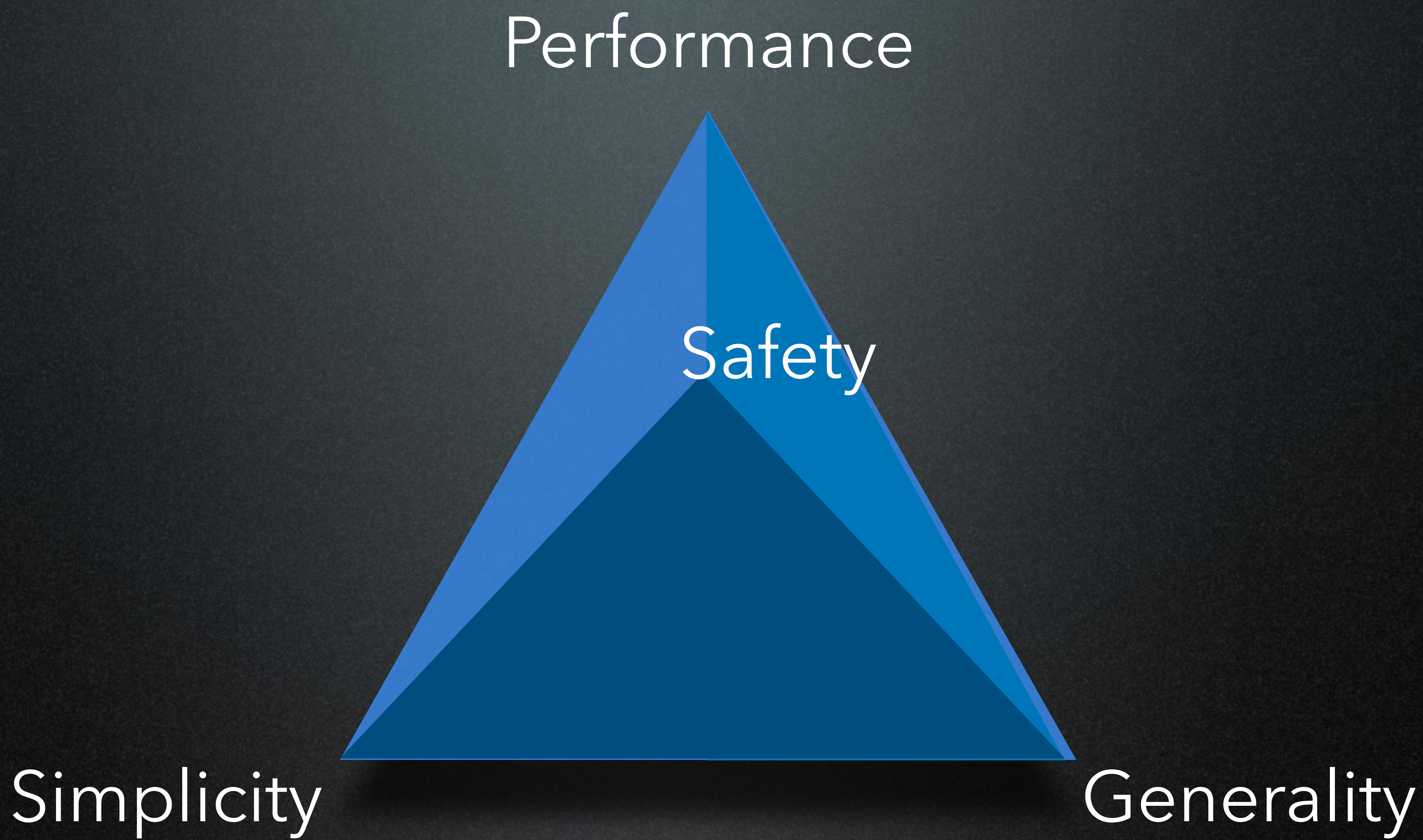
coroutines / effect handlers

garbage collection

…

https://github.com/WebAssembly/proposals


Performance
Simplicity
Generality


Performance
Safety
Simplicity
Generality

inside job

backing from key players

patience & stubbornness

adjust to audience

stay realistic

bus factor

Summary

Formal rigour and machine verification in the mainstream

Led to a clean and simple design

Progress is slow and brittle

But there is hope