

# Foundations of Software Fall 2022

Week 14

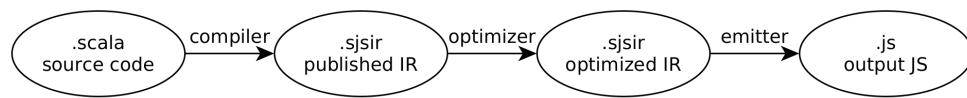
Sébastien Doeraene

1

## Elements of the Scala.js IR type system

2

## Scala.js compilation pipeline



3

## Why formally study an IR

4

## Why formally study an IR

- ▶ Optimizations may only be applicable if the type system is sound
- ▶ Prove that certain optimizations are correct
- ▶ Prove that the translation from source and to the target language are correct
- ▶ etc.

## Mixing primitives and objects

## Motivation

Featherweight Java only has objects. How do we model primitives, for example, `int` and `bool`?

## Motivation

Featherweight Java only has objects. How do we model primitives, for example, `int` and `bool`?

Moreover, in Scala, primitive types are “object-like”. We can use them in arbitrary type parameters, and they should behave like objects.

On the JVM, this is implemented with *boxing*. In Scala.js, however, boxing would be detrimental to *interoperability* with JavaScript. How do we make primitives object-like without boxing?

## Motivation

Featherweight Java only has objects. How do we model primitives, for example, `int` and `bool`?

Moreover, in Scala, primitive types are “object-like”. We can use them in arbitrary type parameters, and they should behave like objects.

On the JVM, this is implemented with *boxing*. In Scala.js, however, boxing would be detrimental to *interoperability* with JavaScript. How do we make primitives object-like without boxing?

Idea: make primitive types *subtypes* of their “representative classes”.

6

## Types and subtyping

`T ::=`

`C`  
`int`  
`bool`

*types*  
*class*  
*primitive int*  
*primitive bool*

$$\frac{CT(C) = \text{class } C \text{ extends } D \{ \dots \}}{C <: D}$$
$$T <: T$$
$$\frac{S <: W \quad W <: T}{S <: T}$$

`int <: Integer`

`bool <: Boolean`

7

## Representative classes

$tpcls(C) = C$

$tpcls(int) = Integer$

$tpcls(bool) = Boolean$

$T <: tpcls(T)$

8

## Syntax (terms)

$t ::=$

$x$

$t.f$

$t.m(\bar{t})$

$new\ C(\bar{t})$

$(T)\ t$

$false$

$true$

$if\ t\ then\ t\ else\ t$

$0$

$succ\ t$

$pred\ t$

$iszero\ t$

*terms*

*variable*

*field access*

*method invocation*

*object creation*

*cast*

9

## Syntax (values)

$v ::=$	$\text{new } C(\bar{v})$	<i>values</i>
	$nv$	<i>object creation</i>
	$bv$	<i>numeric value</i>
		<i>boolean value</i>
$nv ::=$	$0$	<i>numeric values</i>
	$\text{succ } nv$	<i>zero</i>
		<i>non-zero</i>
$bv ::=$	$\text{false}$	<i>boolean values</i>
	$\text{true}$	<i>false</i>
		<i>true</i>

10

## Typing rules: method calls

Adapting from Featherweight Java:

$$\frac{\begin{array}{l} \Gamma \vdash t_0 : C_0 \\ mtype(m, C_0) = \bar{S} \rightarrow T \\ \Gamma \vdash \bar{t} : \bar{S}_1 \quad \bar{S}_1 <: \bar{S} \end{array}}{\Gamma \vdash t_0.m(\bar{t}) : T} \quad (\text{T-INVK})$$

What if  $t_0$  is a primitive?

11

## Typing rules: method calls

Adapting from Featherweight Java:

$$\frac{\begin{array}{c} \Gamma \vdash t_0 : C_0 \\ mtype(m, C_0) = \bar{S} \rightarrow T \\ \Gamma \vdash \bar{t} : \bar{S}_1 \quad \bar{S}_1 <: \bar{S} \end{array}}{\Gamma \vdash t_0.m(\bar{t}) : T} \quad (T\text{-INVK})$$

What if  $t_0$  is a primitive?

$$\frac{\begin{array}{c} \Gamma \vdash t_0 : T_0 \\ mtype(m, tpcls(T_0)) = \bar{S} \rightarrow T \\ \Gamma \vdash \bar{t} : \bar{S}_1 \quad \bar{S}_1 <: \bar{S} \end{array}}{\Gamma \vdash t_0.m(\bar{t}) : T} \quad (T\text{-INVK})$$

If  $\Gamma \vdash x : \text{int}$ , the call  $x.m(\dots)$  is typed by looking up  $m$  in `Integer`.

11

## Example

```
class Boolean extends Object { Boolean() { super(); } }
class Integer extends Object {
  Integer() { super(); }
  int plus(int that) {
    return if (iszero that) then ((int) this)
      else (succ this.plus(pred that)); }
}
class Pair extends Object {
  Object fst;
  Object snd;
  Pair(Object fst, Object snd) {
    super(); this.fst=fst; this.snd=snd; }
  int sum() {
    return ((int) this.fst).plus((int) this.snd); }
}

new Pair(5, 11).sum()
```

12



## Typing rules: fields

Adapting from Featherweight Java:

$$\frac{\Gamma \vdash t_0 : C_0 \quad \text{fields}(C_0) = \bar{T} \ \bar{f}}{\Gamma \vdash t_0.f_i : T_i} \quad (\text{T-FIELD})$$

What if  $t_0$  is a primitive?

13

## Typing rules: fields

Adapting from Featherweight Java:

$$\frac{\Gamma \vdash t_0 : C_0 \quad \text{fields}(C_0) = \bar{T} \ \bar{f}}{\Gamma \vdash t_0.f_i : T_i} \quad (\text{T-FIELD})$$

What if  $t_0$  is a primitive?

We can't have that!

13

## Typing rules: fields

Adapting from Featherweight Java:

$$\frac{\Gamma \vdash t_0 : C_0 \quad \text{fields}(C_0) = \bar{T} \ \bar{f}}{\Gamma \vdash t_0.f_i : T_i} \quad (\text{T-FIELD})$$

What if  $t_0$  is a primitive?

We can't have that!

Add additional well-formedness conditions for representative classes:

$$\frac{\text{fields(Integer)} = \emptyset \quad \text{fields(Boolean)} = \emptyset}{\text{repr classes OK}}$$

13

## Typing rules: casts

Straightforward generalization to all types.

$$\frac{\Gamma \vdash t_0 : S \quad S <: T}{\Gamma \vdash (T)t_0 : T} \quad (\text{T-UCAST})$$

$$\frac{\Gamma \vdash t_0 : S \quad T <: S \quad T \neq S}{\Gamma \vdash (T)t_0 : T} \quad (\text{T-DCAST})$$

$$\frac{\Gamma \vdash t_0 : S \quad T \not<: S \quad S \not<: T \quad \text{stupid warning}}{\Gamma \vdash (T)t_0 : T} \quad (\text{T-SCAST})$$

14

## Typing rules: casts

Since it is an Intermediate Representation, warnings are not relevant anymore. Therefore, we keep only one typing rule for casts.

$$\frac{\Gamma \vdash t_0 : S}{\Gamma \vdash (T)t_0 : T} \quad (\text{T-CAST})$$

15

## Typing rules: casts

Since it is an Intermediate Representation, warnings are not relevant anymore. Therefore, we keep only one typing rule for casts.

$$\frac{\Gamma \vdash t_0 : S}{\Gamma \vdash (T)t_0 : T} \quad (\text{T-CAST})$$

Question: can we remove the premise of that rule?

15

## Evaluation rules

$$\frac{fields(C) = \bar{T} \ \bar{f}}{(new \ C(\bar{v})) . f_i \longrightarrow v_i} \quad (E-PROJNEW)$$

$$\frac{mbody(m, tpcls(vtpe(v))) = (\bar{x}, t_0)}{v.m(\bar{u}) \longrightarrow [\bar{x} \mapsto \bar{u}, this \mapsto v]t_0} \quad (E-INVKVAL)$$

$$\frac{vtpe(v) <: T}{(T)v \longrightarrow v} \quad (E-CASTVAL)$$

$$vtpe(new \ C(\bar{v})) = C \quad vtpc(nv) = int \quad vtpc(bv) = bool$$

plus congruence rules and rules for `if`, `pred`, `succ` and `iszero` (omitted)