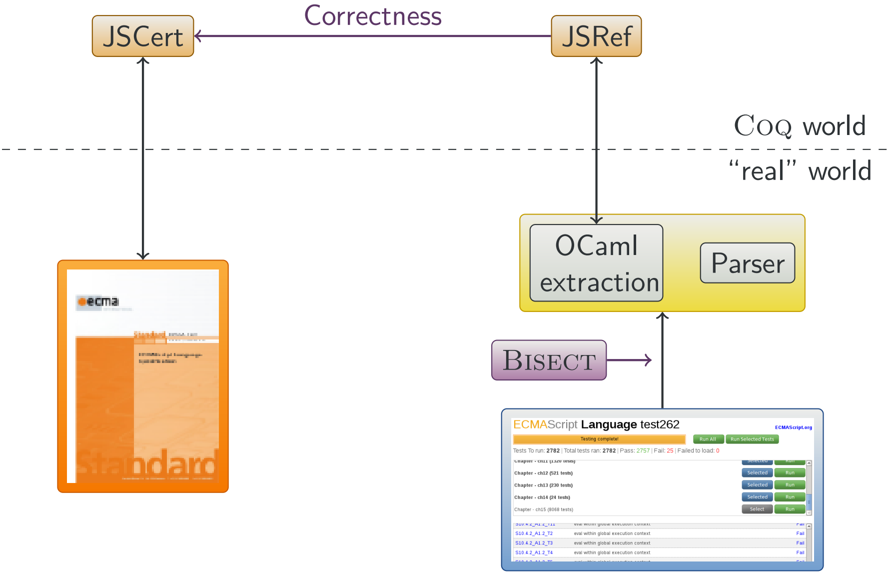
From JSCert to an Abstract Interpreter

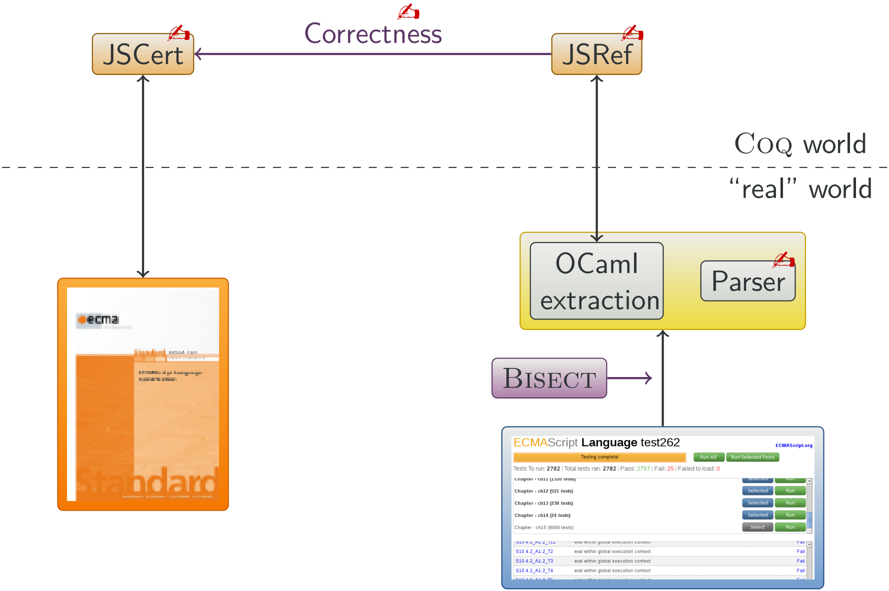
Martin Bodin

Inria

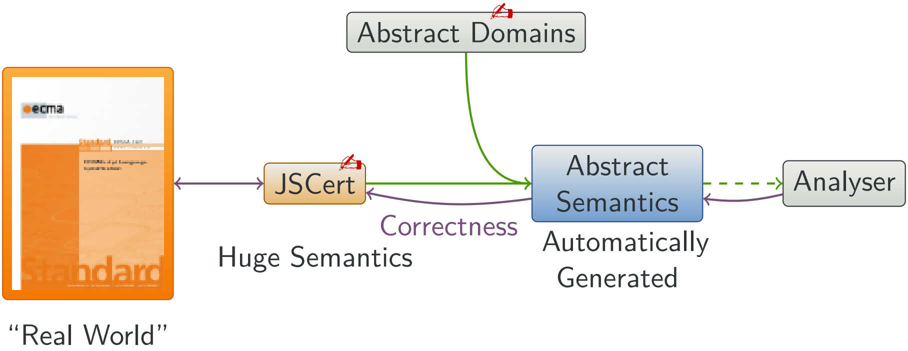
JSCert and JSRef



JSCert and JSRef



Next Step


Abstract Domains
JSCert
Abstract Semantics
Analyser
Correctness
Huge Semantics
Automatically Generated
“Real World”

JSCert, Summing Up

`jscert.org`

- An operational semantics for JavaScript;
- Trusted;
- *Huge* ($\sim$ 800 reduction rules).
- it's *moving*!

How to derive✍ an abstract interpreter from such a huge semantics?

... proven in Coq?

How to derive an abstract interpreter from such a huge semantics?

... proven in COQ?

Let's make it correct *by construction*!

Concrete Domains

`int`, `bool`

Concrete Operations

`+`, `=`

Concrete Semantics

$t, \sigma \Downarrow r$

Abstract Domains

Sign

Abstract Operations

$+^\sharp, =^\sharp$

Abstract Semantics

$t, \sigma^\sharp \Downarrow^\sharp r^\sharp$

Abstract Interpreter

$f\left(t, \sigma^\sharp\right) = r^\sharp$

Concrete Domains

`int`, `bool`

Concrete Operations

`+`, `=`

Concrete Semantics

$t, \sigma \Downarrow r$

Abstract Domains

Sign

Abstract Operations

$+^\sharp$, $=^\sharp$

Abstract Semantics

$t, \sigma^\sharp \Downarrow^\sharp r^\sharp$

Abstract Interpreter

$f\left(t, \sigma^\sharp\right) = r^\sharp$

Concrete Domains

`int`, `bool`

Concrete Operations

`+`, `=`

Concrete Semantics

$t, \sigma \Downarrow r$

Abstract Domains

Sign

Abstract Operations

$+^\sharp, =^\sharp$

Abstract Semantics

$t, \sigma^\sharp \Downarrow^\sharp r^\sharp$

Abstract Interpreter

$f\left(t, \sigma^\sharp\right) = r^\sharp$

Defining an Abstract Semantics, the Direct Approach

IfTrue

$$\frac{s_1, E \Downarrow E'}{if\ s_1\ s_2, (v, E) \Downarrow E'} \quad v \in \mathbb{Z}^\star$$

IfFalse

$$\frac{s_2, E \Downarrow E'}{if\ s_1\ s_2, (v, E) \Downarrow E'} \quad v \in \{0\}$$

Defining an Abstract Semantics, the Direct Approach

$$\textsc{IfTrue}\ \frac{s_1, E \Downarrow E'}{\mathit{if}\ s_1\ s_2, (v, E) \Downarrow E'}\ v \in \mathbb{Z}^\star \qquad \textsc{IfFalse}\ \frac{s_2, E \Downarrow E'}{\mathit{if}\ s_1\ s_2, (v, E) \Downarrow E'}\ v \in \{0\}$$

Let's just add $\sharp$ everywhere!

$$\textsc{IfTrue}\ \frac{s_1, E^\sharp \Downarrow^\sharp E'^\sharp}{\mathit{if}\ s_1\ s_2, \left(v^\sharp, E^\sharp\right) \Downarrow^\sharp E'^\sharp}\ \gamma\left(v^\sharp\right) \cap \mathbb{Z}^\star \neq \emptyset$$

$$\textsc{IfFalse}\ \frac{s_2, E^\sharp \Downarrow^\sharp E'^\sharp}{\mathit{if}\ s_1\ s_2, \left(v^\sharp, E^\sharp\right) \Downarrow^\sharp E'^\sharp}\ \gamma\left(v^\sharp\right) \cap \{0\} \neq \emptyset$$

Defining an Abstract Semantics, the Direct Approach

$$\textsc{IfTrue}\ \frac{s_1, E \Downarrow E'}{\mathit{if}\ s_1\ s_2, (v, E) \Downarrow E'}\ v \in \mathbb{Z}^\star \qquad \textsc{IfFalse}\ \frac{s_2, E \Downarrow E'}{\mathit{if}\ s_1\ s_2, (v, E) \Downarrow E'}\ v \in \{0\}$$

Let's just add ♯ everywhere!

$$\textsc{IfAdHoc}^{✍}\ \frac{s_1, E^\sharp \Downarrow^\sharp E_1^\sharp \qquad s_2, E^\sharp \Downarrow^\sharp E_2^\sharp}{\mathit{if}\ s_1\ s_2, \left(v^\sharp, E^\sharp\right) \Downarrow^\sharp E_1^\sharp \sqcup E_2^\sharp}\ v^\sharp = \top$$

Objective of the Abstract Semantics

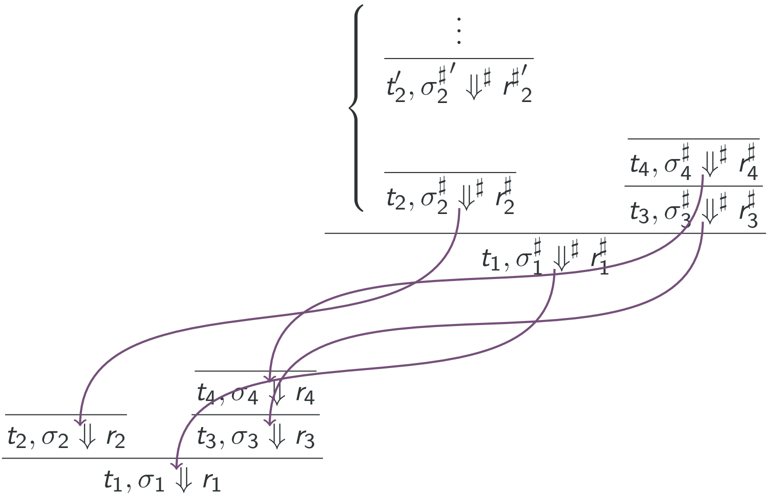
This is like David SCHMIDT's approach:

$$\dfrac{\left\{\begin{array}{c}\dfrac{\vdots}{t_2', \sigma_2^{\sharp\,\prime} \Downarrow^\sharp r^{\sharp\,\prime}_2} \\ \\ \dfrac{}{t_2, \sigma_2^\sharp \Downarrow^\sharp r_2^\sharp}\end{array}\right. \qquad \dfrac{\dfrac{}{t_4, \sigma_4^\sharp \Downarrow^\sharp r_4^\sharp}}{t_3, \sigma_3^\sharp \Downarrow^\sharp r_3^\sharp}}{t_1, \sigma_1^\sharp \Downarrow^\sharp r_1^\sharp}$$

$$\dfrac{\dfrac{}{t_2, \sigma_2 \Downarrow r_2} \qquad \dfrac{\dfrac{}{t_4, \sigma_4 \Downarrow r_4}}{t_3, \sigma_3 \Downarrow r_3}}{t_1, \sigma_1 \Downarrow r_1}$$

Objective of the Abstract Semantics

This is like David SCHMIDT's approach:



Abstract Rules

Shared between the concrete and abstract semantics

Each rule has

- A structural part: identifier, terms;
- A semantic part: side-conditions, transfer functions.

To be specified in the abstract semantics.
To be *locally* proved correct.

- The abstract semantics will follow the exact same structure as the concrete semantics.

Abstract Semantics

But we don't define $\Downarrow$ and $\Downarrow^\sharp$ the same way from the rules!

Concrete Semantics $\Downarrow$	Abstract Semantics $\Downarrow^\sharp$
At each step, apply *one* rule that applies	At each step, apply *all* the rules that apply

$$\dfrac{s_1, E_0^\sharp \Downarrow E_1^\sharp \;\uparrow \textsc{IfTrue} \qquad s_2, E_0^\sharp \Downarrow E_2^\sharp \;\uparrow \textsc{IfFalse}}{\mathit{if}\ s_1\ s_2, \left(v^\sharp, E_0^\sharp\right) \Downarrow E_1^\sharp \sqcup E_2^\sharp}$$

Abstract Semantics

But we don't define $\Downarrow$ and $\Downarrow^\sharp$ the same way from the rules!

Concrete Semantics $\Downarrow$	Abstract Semantics $\Downarrow^\sharp$
At each step, apply *one* rule that applies	At each step, apply *all* the rules that apply
	Allow approximations

$$\frac{s_1, E_0^\sharp \Downarrow E_1^\sharp \uparrow \textsc{IfTrue} \qquad s_2, E_0^\sharp \Downarrow E_2^\sharp \uparrow \textsc{IfFalse}}{if\ s_1\ s_2, \left(v^\sharp, E_0^\sharp\right) \Downarrow E_1^\sharp \sqcup E_2^\sharp}$$

Abstract Semantics

But we don't define $\Downarrow$ and $\Downarrow^\sharp$ the same way from the rules!

Concrete Semantics $\Downarrow$	Abstract Semantics $\Downarrow^\sharp$
At each step, apply *one* rule that applies	At each step, apply *all* the rules that apply
	Allow approximations
Inductive interpretation of the rules $\Downarrow = \mathit{lfp}\,(\mathcal{F})$	Co-inductive interpretation of the rules $\Downarrow^\sharp = \mathit{gfp}\left(\mathcal{F}^\sharp\right)$

$$\dfrac{s_1, E_0^\sharp \Downarrow E_1^\sharp \;\; \uparrow \textsc{IfTrue} \qquad s_2, E_0^\sharp \Downarrow E_2^\sharp \;\; \uparrow \textsc{IfFalse}}{\mathit{if}\ s_1\ s_2, \left(v^\sharp, E_0^\sharp\right) \Downarrow E_1^\sharp \sqcup E_2^\sharp}$$

Example of Concrete Rules

$$\textsc{While}(e, s)\quad \frac{\mathit{while}_1\ e\,s, \mathit{ret}\ E \Downarrow o}{\mathit{while}\ e\,s, E \Downarrow o}$$

$$\textsc{While1}(e, s)\quad \frac{e, E \Downarrow o \qquad \mathit{while}_2\ e\,s, (E, o) \Downarrow o'}{\mathit{while}_1\ e\,s, \mathit{ret}\ E \Downarrow o'}$$

$$\textsc{While2True}(e, s)\quad \frac{s, E \Downarrow o \qquad \mathit{while}_1\ e\,s, o \Downarrow o'}{\mathit{while}_2\ e\,s, (E, \mathit{val}\ v) \Downarrow o'} \quad v \in \mathbb{Z}^\star$$

$$\textsc{While2False}(e, s)\quad \frac{}{\mathit{while}_2\ e\,s, (E, \mathit{val}\ v) \Downarrow \mathit{ret}\ E} \quad v \in \{0\}$$

Example of a Concrete Derivation Tree

$$s = (x := x - 1)$$

$$\frac{\textsc{Var}(x)}{x, \{x \mapsto 1\} \Downarrow 1}$$

$$\frac{\textsc{Var}(x)}{x, \{x \mapsto 0\} \Downarrow 0}$$

$$\frac{\vdots}{\mathit{while}_2\, x\, s, (\{x \mapsto 0\}, \mathit{val}\, 0) \Downarrow \{x \mapsto 0\}}\ \textsc{While2False}(x, s)$$

$$\frac{\vdots}{\mathit{while}_1\, x\, s, \{x \mapsto 1\} \Downarrow \{x \mapsto 0\}}\ \textsc{While1}(x, s)$$

$$\frac{s, \{x \mapsto 1\} \Downarrow \{x \mapsto 0\} \qquad \vdots}{\mathit{while}_2\, x\, s, (\{x \mapsto 1\}, \mathit{val}\, 1) \Downarrow \{x \mapsto 0\}}\ \textsc{While2True}(x, s)$$

$$\frac{\vdots}{\mathit{while}_1\, x\, s, \{x \mapsto 1\} \Downarrow \{x \mapsto 0\}}\ \textsc{While1}(x, s)$$

$$\frac{}{\mathit{while}\, x\, s, \{x \mapsto 1\} \Downarrow \{x \mapsto 0\}}\ \textsc{While}(x, s)$$

Example of Abstract Rules

$$\textsc{While}(e, s)\quad \frac{\mathit{while}_1\ e\ s, E^\sharp \Downarrow^\sharp o^\sharp}{\mathit{while}\ e\ s, E^\sharp \Downarrow^\sharp o^\sharp}$$

$$\textsc{While1}(e, s)\quad \frac{e, E^\sharp \Downarrow^\sharp v^\sharp \qquad \mathit{while}_2\ e\ s, (E^\sharp, v^\sharp) \Downarrow^\sharp o^\sharp}{\mathit{while}_1\ e\ s, E^\sharp \Downarrow^\sharp o^\sharp}$$

$$\textsc{While2True}(e, s)\quad \frac{s, E^\sharp \Downarrow^\sharp o \qquad \mathit{while}_1\ e\ s, o^\sharp \Downarrow^\sharp o'^\sharp}{\mathit{while}_2\ e\ s, (E^\sharp, v^\sharp) \Downarrow^\sharp o'^\sharp}\quad \gamma\left(v^\sharp\right) \cap \mathbb{Z}^\star \neq \emptyset$$

$$\textsc{While2False}(e, s)\quad \frac{}{\mathit{while}_2\ e\ s, (E^\sharp, v^\sharp) \Downarrow^\sharp E^\sharp}\quad \gamma\left(v^\sharp\right) \cap \{0\} \neq \emptyset$$

Example of an Abstract Derivation Tree

$s = (x := x - 1)$

$$\dfrac{e, E \Downarrow o \qquad while_2\ e\,s, (E, o) \Downarrow o'}{while_1\ e\,s, ret\ E \Downarrow o'}\ \textsc{While1}(e, s)$$

$$\dfrac{\dfrac{}{while_1\ x\,s, \{x \mapsto +_0\} \Downarrow^\sharp}\ \textsc{While1}(x, s)}{while\ x\,s, \{x \mapsto +_0\} \Downarrow^\sharp}\ \textsc{While}(x, s)$$

Example of an Abstract Derivation Tree

$$s = (x := x - 1)$$

$$\dfrac{\textsc{Var}(x)}{x, \{x \mapsto +_0\} \Downarrow^\sharp +_0}$$

$$\vdots$$

$$\dfrac{\dfrac{\vdots \qquad \mathit{while}_2\, x\, s, (\{x \mapsto +_0\}, +_0) \Downarrow^\sharp}{\mathit{while}_1\, x\, s, \{x \mapsto +_0\} \Downarrow^\sharp}\ \textsc{While1}(x, s)}{\mathit{while}\, x\, s, \{x \mapsto +_0\} \Downarrow^\sharp}\ \textsc{While}(x, s)$$

Example of an Abstract Derivation Tree

$$s = (x := x - 1)$$

$$\dfrac{\textsc{Var}(x)}{x, \{x \mapsto +_0\} \Downarrow^\sharp +_0}$$

$$\textsc{While2False}(e, s)\quad \dfrac{}{while_2\ e\, s, (E^\sharp, v^\sharp) \Downarrow^\sharp E^\sharp}\ \gamma\left(v^\sharp\right) \cap \{0\} \neq \emptyset$$

$$\textsc{While2True}(e, s)\quad \dfrac{s, E^\sharp \Downarrow^\sharp o \qquad while_1\ e\, s, o^\sharp \Downarrow^\sharp o'^\sharp}{while_2\ e\, s, (E^\sharp, v^\sharp) \Downarrow^\sharp o'^\sharp}\ \gamma\left(v^\sharp\right) \cap \mathbb{Z}^\star \neq \emptyset$$

$$\dfrac{\dfrac{\vdots \qquad while_2\ x\, s, (\{x \mapsto +_0\}, +_0) \Downarrow^\sharp}{while_1\ x\, s, \{x \mapsto +_0\} \Downarrow^\sharp}\ \textsc{While1}(x, s)}{while\ x\, s, \{x \mapsto +_0\} \Downarrow^\sharp}\ \textsc{While}(x, s)$$

Example of an Abstract Derivation Tree

$s = (x := x - 1)$

$$\frac{\textsc{Var}(x)}{x, \{x \mapsto +_0\} \Downarrow^\sharp +_0}$$

$$\vdots$$

$$\left\{ \frac{}{while_2\, x\, s, (\{x \mapsto +_0\}, +_0) \Downarrow^\sharp \{x \mapsto +_0\}}\ \textsc{While2False}(x, s) \right.$$

$$\frac{\dfrac{\vdots}{while_1\, x\, s, \{x \mapsto +_0\} \Downarrow^\sharp}\ \textsc{While1}(x, s)}{while\, x\, s, \{x \mapsto +_0\} \Downarrow^\sharp}\ \textsc{While}(x, s)$$

Example of an Abstract Derivation Tree

$s = (x := x - 1)$

$$\dfrac{\textsc{Var}(x)}{x, \{x \mapsto +_0\} \Downarrow^\sharp +_0}$$

$\vdots$

$$\left\{ \begin{array}{l} \dfrac{}{\mathit{while}_2\, x\, s, (\{x \mapsto +_0\}, +_0) \Downarrow^\sharp \{x \mapsto +_0\}}\ \textsc{While2False}(x, s) \\ \\ \dfrac{\dfrac{\ddots}{\mathit{while}_1\, x\, s, \{x \mapsto \top\} \Downarrow^\sharp \{x \mapsto \top\}}\ \textsc{While1}(x, s)}{} \\ \quad \vdots \\ \dfrac{s, \{x \mapsto +_0\} \Downarrow^\sharp \{x \mapsto \top\} \qquad \vdots}{\mathit{while}_2\, x\, s, (\{x \mapsto +_0\}, +_0) \Downarrow^\sharp \{x \mapsto \top\}}\ \textsc{While2True}(x, s) \end{array} \right.$$

$$\dfrac{\dfrac{\vdots}{\mathit{while}_1\, x\, s, \{x \mapsto +_0\} \Downarrow^\sharp}\ \textsc{While1}(x, s)}{\mathit{while}\, x\, s, \{x \mapsto +_0\} \Downarrow^\sharp}\ \textsc{While}(x, s)$$

Example of an Abstract Derivation Tree

$s = (x := x - 1)$

$$\dfrac{\textsc{Var}(x)}{x, \{x \mapsto +_0\} \Downarrow^\sharp +_0}$$

$\vdots$

$$\dfrac{}{\mathit{while}_2\, x\, s, (\{x \mapsto +_0\}, +_0) \Downarrow^\sharp \{x \mapsto +_0\}}\ \textsc{While2False}(x, s)$$

$\ddots$

$$\dfrac{}{\mathit{while}_1\, x\, s, \{x \mapsto \top\} \Downarrow^\sharp \{x \mapsto \top\}}\ \textsc{While1}(x, s)$$

$\vdots$

$$\dfrac{s, \{x \mapsto +_0\} \Downarrow^\sharp \{x \mapsto \top\}}{\mathit{while}_2\, x\, s, (\{x \mapsto +_0\}, +_0) \Downarrow^\sharp \{x \mapsto \top\}}\ \textsc{While2True}(x, s)$$

$$\dfrac{\dfrac{\cdots}{\mathit{while}_1\, x\, s, \{x \mapsto +_0\} \Downarrow^\sharp \{x \mapsto \top\}}\ \textsc{While1}(x, s)}{\mathit{while}\, x\, s, \{x \mapsto +_0\} \Downarrow^\sharp \{x \mapsto \top\}}\ \textsc{While}(x, s)$$

An Abstract Semantics Correct by Construction

Hypotheses:

- Correctness of the side-conditions,
- Correctness of the transfer functions.

Theorem (Correctness)

The abstract semantics won't miss any behaviour.

An Abstract Semantics Correct by Construction

Hypotheses:

- Correctness of the side-conditions,
- Correctness of the transfer functions.

Theorem (Correctness)

Let t a term, σ and $\sigma^\sharp$ a concrete and an abstract semantic contexts, and r and $r^\sharp$ a concrete and an abstract results.

$$\text{If } \begin{cases} \sigma \in \gamma\left(\sigma^\sharp\right) \\ t, \sigma \Downarrow r \\ t, \sigma^\sharp \Downarrow^\sharp r^\sharp \end{cases} \quad \text{then } r \in \gamma\left(r^\sharp\right).$$

An Abstract Semantics Correct by Construction

Hypotheses:

- Correctness of the side-conditions,
- Correctness of the transfer functions.

Theorem (Correctness)

Let t a term, σ and $\sigma^\sharp$ a concrete and an abstract semantic contexts, and r and $r^\sharp$ a concrete and an abstract results.

$$\text{If } \begin{cases} \sigma \in \gamma\left(\sigma^\sharp\right) \\ t, \sigma \Downarrow r \\ t, \sigma^\sharp \Downarrow^\sharp r^\sharp \end{cases} \quad \text{then } r \in \gamma\left(r^\sharp\right).$$

Proven independently of the rules!

Concrete Domains

`int`, `bool`

Concrete Operations

`+`, `=`

Concrete Semantics

$t, \sigma \Downarrow r$

Abstract Domains

Sign

Abstract Operations

$+^\sharp$, $=^\sharp$

Abstract Semantics

$t, \sigma^\sharp \Downarrow^\sharp r^\sharp$

Abstract Interpreter

$f\left(t, \sigma^\sharp\right) = r^\sharp$

Defining Abstract Interpreters: a Verifier

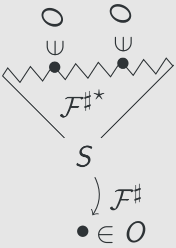
- An abstract interpreter is a function building an abstract derivation.

Defining Abstract Interpreters: a Verifier

- An abstract interpreter is a function building an abstract derivation.
- But this abstract semantic tree can be infinite!

A Verifier

- It takes an oracle, i.e., a set O of triples $t, \sigma^\sharp, r^\sharp$.



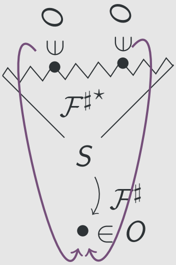
It tries to prove $O \subseteq {\mathcal{F}^\sharp}^+(O)$.
By PARK's principle, this implies $O \subseteq \Downarrow^\sharp$.

Defining Abstract Interpreters: a Verifier

- An abstract interpreter is a function building an abstract derivation.
- But this abstract semantic tree can be infinite!

A Verifier

- It takes an oracle, i.e., a set O of triples $t, \sigma^\sharp, r^\sharp$.



It tries to prove $O \subseteq {\mathcal{F}^\sharp}^+(O)$.
By PARK's principle, this implies $O \subseteq \Downarrow^\sharp$.

Generic Abstract Interpreters

- We have built some *generic* abstract interpreters.
- We extracted them to OCaml and run them (on a toy language).

$a := 6; b := 7; r := 0; n := a; \mathit{while}\, n\ (r := r + b; n := n - 1)$

$$(\{r \mapsto +, b \mapsto +, a \mapsto +, n \mapsto \top\}, \bot)$$

Generic Abstract Interpreters

- We have built some *generic* abstract interpreters.
- We extracted them to OCaml and run them (on a toy language).

$$a := 6; b := 7; prod\,(n) := \{if\, n\ (prod\,(n-1)\,; r := r + b)\,(r := 0)\,\}; prod\,(a)$$

$$(\{r \mapsto +, b \mapsto +, a \mapsto +\}\,, \bot)$$

Generic Abstract Interpreters

- We have built some *generic* abstract interpreters.
- We extracted them to OCaml and run them (on a toy language).

$$a := 6; b := 7; prod\,(n) := \{if\, n\ (prod\,(n-1)\,; \mathbf{r} := \mathbf{r} + \mathbf{b})\,(\mathbf{r} := \mathbf{0})\,\}; prod\,(a)$$

$$(\{r \mapsto +, b \mapsto +, a \mapsto +\}\,, \bot)$$

What to Remember of All This

Recipe

1. define the concrete semantics;
2. define the abstract domains and operations on the abstract domain,
   - this automatically defines an abstract semantics;
3. prove the abstract operations are correct,
   - this implies the abstract semantics is correct;
4. define an analysis.

What to Remember of All This

Concrete Domains

`int`, `bool`

Concrete Operations

`+`, `=`

Concrete Semantics

$t, \sigma \Downarrow r$

Abstract Domains

Sign

Abstract Operations

$+^\sharp, =^\sharp$

Abstract Semantics

$t, \sigma^\sharp \Downarrow^\sharp r^\sharp$

Abstract Interpreter

$f\left(t, \sigma^\sharp\right) = r^\sharp$

1. JSCert
2. Pretty-Big-Step
3. Defining an Abstract Semantics Correct by Construction
4. Running Abstract Interpreters