$\mathbb{K}$ Framework

Vincenzo Arceri

vincenzo.arceri@univr.it

Ph.D. Student

Lesson 1 - Introduction

November 3, 2017

UNIVERSITÀ
di **VERONA**
Dipartimento
di **INFORMATICA**

Formal semantics

Consider a generic language L

- Syntax of L: always given
- Semantics of L: always skipped, considered useless

C

```
int main(void) {
   int x = 0;
   return (x = 1) + (x = 2);
}
```

According to the C standard, it is undefined

GCC3, clang	GCC4
3	4

PHP

```
if ($p == 0)
   echo $p;
```

May print any string!
(that does not start with a number)

Of course, Javascript

```
var f = function () {
  if (true)
    function g() { return 1; }
  else
    function g() { return 2; }
  function g() { return 3; }
  return g;
  function g() { return 4; }
}

f();
```

SpiderMonkey

JScript

(… MicroSoft)

Rhino
Safari

Programming languages must have formal semantics

Reference manual are too informal and not sufficient

$\mathbb{K}$ Framework

Semantic framework which makes it easy and fun (?) to define programming languages design and semantics

A language to implement languages

K Framework


php


Java


C


python


JS JavaScript


ethereum


LLVM
COMPILER
INFRASTRUCTURE


OCaml


Haskell


VHDL

Language of expressions

$$\mathtt{e} \ni \mathtt{Exp} ::= n \mid \mathtt{e} + \mathtt{e} \mid \mathtt{e} * \mathtt{e} \mid \mathtt{e} - \mathtt{e}$$

$\mathbb{K}$ Introduction

Some syntactic classes are builtin in $\mathbb{K}$

Int, Float, Bool, String…

Evaluation strategies

```
syntax Exp ::=  Int
              |   Exp "*" Exp      [seqstrict]
              |   Exp "/" Exp      [seqstrict]
              >   Exp "+" Exp      [seqstrict]
              |   Exp "-" Exp      [seqstrict]
              |   "(" Exp ")"      [bracket]
```

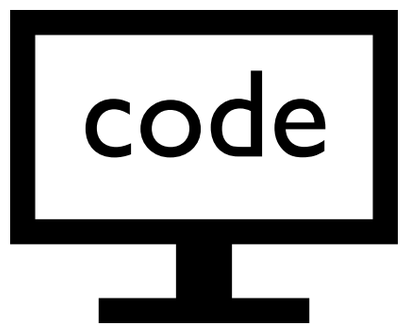
`[seqstrict]`: left-to-right evaluation up to a KResult

`[strict]`: *random* evaluation up to a KResult

`[strict(`n`)]`: first it evaluates the expression in the n position then search for a semantic rule to apply, without evaluating the other expressions

Language of expressions

$$\texttt{e} \ni \texttt{Exp} ::= \ldots \mid \texttt{if (e) e1 else e2} \mid \texttt{Bool}$$



Exercise 1

Extend the Exp language with:

$$\texttt{e} \ni \texttt{Exp} ::= \ldots \mid +(\texttt{e}) \mid -(\texttt{e})$$

$$\texttt{e} \ni \texttt{Exp} ::= \ldots \mid \texttt{e1} \ ? \ \texttt{e2} : \texttt{e3}$$

Dynamic and static semantics

The semantics defined so far is the dynamic semantics

$$\frac{\rho \vdash_\Delta \langle e, \sigma\rangle \to_e \langle e_0, \sigma\rangle}{\rho \vdash_\Delta \langle e \ \mathbf{bop}\ e', \sigma\rangle \to_e \langle e_0 \ \mathbf{bop}\ e', \sigma\rangle}$$

$$\frac{\rho \vdash_\Delta \langle e', \sigma\rangle \to_e \langle e_1, \sigma\rangle}{\rho \vdash_\Delta \langle k \ \mathbf{bop}\ e', \sigma\rangle \to_e \langle k \ \mathbf{bop}\ e_1, \sigma\rangle}$$

$$\rho \vdash_\Delta \langle k \ \mathbf{bop}\ k', \sigma\rangle \to_e \langle k'', \sigma\rangle, k'' = k \ \mathbf{bop}\ k'$$

Any ideas about how to define the static one?

$$\frac{\Delta \vdash_I e : \tau_0, e' : \tau_1}{\Delta \vdash_I e \ \mathbf{bop}\ e' : \tau_{bop}(\tau_0, \tau_1)}$$

Static semantics

We can replace the KResult of the dynamic semantics

```
syntax KResult ::= Int | Bool
```

with the types

```
syntax Type ::= "int" | "bool"
syntax KResult ::= Type
```

Exercise 2

Write the static semantics of Exp for:

$$\texttt{e} \ni \texttt{Exp} ::= \ldots \mid \texttt{if (e) e1 else e2} \mid \texttt{Bool}$$

Exercise 3

Extend Exp with

$$\texttt{e} \ni \texttt{Exp} ::= \ldots \mid \texttt{e1} <= \texttt{e2} \mid \texttt{e1} >= \texttt{e2} \mid \texttt{e1} > \texttt{e2} \mid \texttt{e1} < \texttt{e2}$$

giving the static and the dynamic semantics

Exercise 4

Extend Exp with String

$$\texttt{e} \ni \texttt{Exp} ::= \ldots \mid \texttt{String}$$

Then, overload the + semantics in order to perform the string concatenation. Define the static and the dynamic semantics.

$\mathbb{K}$ Framework

http://www.kframework.org/