Callability Control

By Isaac Oscar Gariano[1] and Marco Servetto[2]

(Victoria University of Wellington)

The Problem

Consider an imperative language, such as C♯, which:

- is statically-typed
- has named/identifiable functions (e.g. static/instance methods or constructors)
- has dynamic dispatch (e.g. interfaces, delegates, and virtual methods)
- supports dynamic code loading and execution

Note: for brevity I will omit accessibility modifiers and allow free standing static functions

1. What *could* this method do?

```
static void M1() { Abs(0); }
```

The Problem

Consider an imperative language, such as C♯, which:

- is statically-typed
- has named/identifiable functions (e.g. static/instance methods or constructors)
- has dynamic dispatch (e.g. interfaces, delegates, and virtual methods)
- supports dynamic code loading and execution

Note: for brevity I will omit accessibility modifiers and allow free standing static functions

1. What *could* this method do?
```
static void M1() { Abs(0); }
```
2. What about this, what *could* it do?
```
interface I { void Run(); }
static void M2(I x) { x.Run(); }
```

The Problem

Consider an imperative language, such as C♯, which:

- is statically-typed
- has named/identifiable functions (e.g. static/instance methods or constructors)
- has dynamic dispatch (e.g. interfaces, delegates, and virtual methods)
- supports dynamic code loading and execution

Note: for brevity I will omit accessibility modifiers and allow free standing static functions

1. What *could* this method do?
```
static void M1() { Abs(0); }
```
2. What about this, what *could* it do?
```
interface I { void Run(); }
static void M2(I x) { x.Run(); }
```
3. How about this?
```
static void M3(String url) {
    var code = Assembly.LoadFrom(url); // Load code(possibly
    code.GetMethod("M").Invoke(null, null); } // call M()
```

Callability

Callability is the *ability* to *call* a function.

Callability

Callability is the *ability* to *call* a function.

A function's callability is the set of things it can call.

Callability

Callability is the *ability* to *call* a function.

A function's callability is the set of things it can call.

Restatement of the Problem:

1. What is the callability of `Abs`?
2. What is the callability of `x.Run`?
3. What is the callability of `M`?

The Basics

The `calls[`$f_1, \ldots, f_n$`]` annotation

Example

```
static void Main(String[] argv) calls[WriteLine] {
    WriteLine("Hello World"); }
```

The Can-Call Relation: $f \rightsquigarrow g$

A function f can call a function g iff: $f \rightsquigarrow g$

1. g is in the `calls[ ... ]` annotation of f: $g \in Calls(f) \Rightarrow f \rightsquigarrow g$

Example

```
static void Write(String s) calls[WriteChar] {
    foreach (Char c in s) WriteChar(c); }
```

2. f can call every function in the `calls[ ... ]` annotation of g:
   $\forall h \in Calls(f)\ f \rightsquigarrow h) \Rightarrow f \rightsquigarrow g$

Example

```
static void WriteLine(String s) calls[WriteChar] {
    Write(s + "\n"); }
```

Note that these rules apply transitively.

Example