

Alban Reynaud

1 Jeremy's Checker

The current OCaml's recursive-value checker has been written by Jeremy Yallop. [2].

1.1 Modes

This checker uses three *access modes* to describe the way variables are accessed in an expression.

These modes are:

Deref : the value of a variable is accessed.

Guarded : the address of a variable is either placed in a constructor, either in an expression that is lazily evaluated, either unused.

Unguarded : the address of a variable is not used in a guarded context.

1.2 Types and Environments

Access modes are used to describe a type-system.

In this system, the type of a variable x is a map that associate every variable used in x 's definition to its access mode.

An environment is a map that associates variables to a type.

TODO: describe operations of types and environments (guard, discard, inspect, ...)

1.3 Inference Rules

This checker can be formalized by inference rules, as pointed out by Gabriel Scherer [1].

$$\frac{}{\Gamma \vdash c : \emptyset} \text{ where } c \text{ is a constant.} \quad \frac{}{\Gamma, x : A \vdash x : A} \quad \frac{}{\Gamma \vdash x : \emptyset} \text{ when } x \notin \Gamma$$

TODO: complete the rules

1.4 The Recursive Check Algorithm

When an expression of the form `let rec $x_1 = e_1$ and ... and $x_n = e_n$` is encountered, an envr,

2 A new system

2.1 Overview

The checker we propose use a simpler type system. Types are just access modes rather than maps from variables to modes. Consequently, an environment is a map that associate to variables a mode.

On the previous system, with a deduction of the form: $\Gamma \vdash expr : A$, the environment Γ is the input and the type A is the output.

On this new system, with a deduction of the form: $\Gamma \vdash expr : m$, the mode m is the input and the environment Γ is the output. The idea is that m represents the mode in which the expression e will be evaluated, and the environment Γ associates each free variable of e to their use in e .

2.2 Modes

As the mode **Guarded** has different meanings, the mode **Guarded** is split into three modes:

Guarded : a variable is *guarded* if its address is placed in a constructor or stored in the environment of a closure (*TODO: add an example about this subtlety, or remove it*). An expression is evaluated in a guarded context if its value is going to be used in a guarded way.

Delayed : an expression is *delayed* if it is lazily evaluated. Variables contained in a delayed expression are used in a delayed mode.

Unused : a variable that is unused.

2.3 Operations on modes

TODO: describe comparison between modes, mode composition and the rule let ... in ...

2.4 Inference Rules

$$\begin{array}{c}
 \frac{}{\emptyset \vdash c : m} \text{ where } c \text{ is a constant} \qquad \frac{}{x : m \vdash x : m} \\
 \\
 \frac{\Gamma_1 \vdash e_1 : m[Deref] \quad \Gamma_2 \vdash e_2 : m[Deref]}{\Gamma_1 + \Gamma_2 \vdash e_1 \ e_2 : m} \\
 \\
 \frac{\Gamma_1 \vdash e_1 : m[Guarded] \quad \dots \quad \Gamma_n \vdash e_n : m[Guarded]}{\Gamma_1 + \dots + \Gamma_n \vdash K(e_1, \dots, e_n) : m} \\
 \\
 \frac{\Gamma, x : m_x \vdash e : m[Delayed]}{\Gamma \vdash fun \ x \leftarrow e : m}
 \end{array}$$

References

- [1] Gabriel Scherer. <https://github.com/ocaml/ocaml/pull/556#issuecomment-329750085>. Accessed: 2018-07-02.
- [2] Jeremy Yallop. A new check that 'let rec' bindings are well formed. <https://github.com/ocaml/ocaml/pull/556>. Accessed: 2018-07-02.