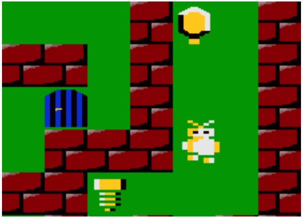
Which types have a unique inhabitant? Focusing on pure program equivalence

Gabriel Scherer,
under the supervision of Didier Rémy

Gallium – INRIA

March 30, 2016

Programming

Program execution		Program description
	$\Longleftarrow$	(code below)
ML Robbo		Tomasz Kokoszka

```
let rec game_loop () =
  let events = Input.process_events () in
  let new_action = Sdltimer.get_ticks () > !last_action + !delay in
  if new_action || events then begin
    last_action := Sdltimer.get_ticks ();
    Level.next_action selected_action;
    (* Odrysowanie *)
    Graph.hide_console ();
    Level.draw_image ();
    Level.draw_info ();
    Console.draw ();
    Video.flip ();
    Gc.major ();
  end else
    Sdltimer.delay 1;
  game_loop ()
```

Like writing cooking recipe for a very very literal cook.

Code inference

Some program fragments are **boring** to write.

Dear computer, please guess what I mean here.
It should be clear from the context.

Code inference

Some program fragments are **boring** to write.

Dear computer, please guess what I mean here.
It should be clear from the context.

> *Tamisez la farine dans un saladier. Ajoutez le sel, le sucre en poudre et le sucre vanillé puis les œufs et mélangez bien. Versez ensuite le lait petit à petit en mélangeant constamment pour éviter les grumeaux. Terminez en ajoutant le Cognac.*

http://www.papillesetpupilles.fr/2011/01/crepes-faciles.html

Code inference

Some program fragments are **boring** to write.

Dear computer, please guess what I mean here.
It should be clear from the context.

> *Tamisez la farine dans un saladier. Ajoutez le sel, le sucre en poudre et le sucre vanillé puis les œufs et mélangez bien. Versez ensuite le lait petit à petit en mélangeant constamment pour éviter les grumeaux. Terminez en ajoutant le Cognac.*
>
> *http://www.papillesetpupilles.fr/2011/01/crepes-faciles.html*

puis les œufs...

Code inference

Some program fragments are **boring** to write.

Dear computer, please guess what I mean here.
It should be clear from the context.

> *Tamisez la farine dans un saladier. Ajoutez le sel, le sucre en poudre et le sucre vanillé puis les œufs et mélangez bien. Versez ensuite le lait petit à petit en mélangeant constamment pour éviter les grumeaux. Terminez en ajoutant le Cognac.*
>
> *http://www.papillesetpupilles.fr/2011/01/crepes-faciles.html*

puis les œufs sans leur coquille

Implicit transformation, coercion: $\texttt{egg} \triangleright (\texttt{white} \times \texttt{yolk})$

Code inference

Some program fragments are **boring** to write.

Dear computer, please guess what I mean here.
It should be clear from the context.

> *Tamisez la farine dans un saladier. Ajoutez le sel, le sucre en poudre et le sucre vanillé puis les œufs et mélangez bien. Versez ensuite le lait petit à petit en mélangeant constamment pour éviter les grumeaux. Terminez en ajoutant le Cognac.*
>
> *http://www.papillesetpupilles.fr/2011/01/crepes-faciles.html*

puis les œufs sans leur coquille

Implicit transformation, coercion: $\texttt{egg} \triangleright (\texttt{white} \times \texttt{yolk})$

mélangez bien...

Code inference

Some program fragments are **boring** to write.

Dear computer, please guess what I mean here.
It should be clear from the context.

> *Tamisez la farine dans un saladier. Ajoutez le sel, le sucre en poudre et le sucre vanillé puis les œufs et mélangez bien. Versez ensuite le lait petit à petit en mélangeant constamment pour éviter les grumeaux. Terminez en ajoutant le Cognac.*
>
> *http://www.papillesetpupilles.fr/2011/01/crepes-faciles.html*

puis les œufs sans leur coquille

Implicit transformation, coercion: $\mathtt{egg} \triangleright (\mathtt{white} \times \mathtt{yolk})$

mélangez bien avec une cuillère

Implicit parameter.

Unicity

"Guess from the context"

What if there are several possible choices?

Refuse to guess? Ask the programmer?
(Heuristics studied in the literature.)

When is there only one choice?
When are all choices equivalent?

Type systems

Not all sentences are grammatically correct.
Not all grammatical sentences are meaningful:

verser le saladier dans le beurre

There are rules to follow to create well-formed, meaningful programs.

In this talk: simply-typed lambda-calculus.
Very simple, restrictive rules.
Realistic languages use richer systems.

"In this context, are all guesses equivalent?"
$\Longrightarrow$
"Given these type constraints, are all guesses equivalent?"

$$
\begin{array}{lll}
A, B, C, D ::= & & \text{(simple) types} \\
& \mid X, Y, Z & \text{atomic types: } \texttt{white}, \texttt{yolk}, \texttt{sugar}\ldots \\
& \mid A \times B & \text{pairs: } \texttt{white} \times \texttt{yolk} \\
& \mid A + B & \text{sums, choices: } \texttt{oil} + \texttt{butter} \\
& \mid A \rightarrow B & \text{functions: } \texttt{white} \rightarrow \texttt{foam} \\
& \mid \ldots &
\end{array}
$$

$$
\begin{aligned}
\mathtt{kitchen} &: \mathtt{salt} \times \mathtt{sugar} \times \mathtt{oil} \\
\mathtt{fridge} &: \mathtt{meat} \times \mathtt{egg} \times \mathtt{butter} \\
\mathtt{split} &: \mathtt{egg} \to \mathtt{white} \times \mathtt{yolk} \times \mathtt{shell} \\
\mathtt{whisk} &: \mathtt{white} \to \mathtt{foam} \\
\mathtt{oven} &: \mathtt{foam} \times \mathtt{sugar} \to \mathtt{meringue} \\
\mathtt{pan} &: \mathtt{meat} \times (\mathtt{oil} + \mathtt{butter}) \to \mathtt{steak}
\end{aligned}
$$

$$\vdash$$

$? : \mathtt{meringue}$

$$
\begin{aligned}
\texttt{kitchen} &: \texttt{salt} \times \texttt{sugar} \times \texttt{oil} \\
\texttt{fridge} &: \texttt{meat} \times \texttt{egg} \times \texttt{butter} \\
\texttt{split} &: \texttt{egg} \to \texttt{white} \times \texttt{yolk} \times \texttt{shell} \\
\texttt{whisk} &: \texttt{white} \to \texttt{foam} \\
\texttt{oven} &: \texttt{foam} \times \texttt{sugar} \to \texttt{meringue} \\
\texttt{pan} &: \texttt{meat} \times (\texttt{oil} + \texttt{butter}) \to \texttt{steak}
\end{aligned}
$$

$$\vdash$$

$? : \texttt{meringue}$

$$\begin{aligned}
\texttt{kitchen} &: \texttt{salt} \times \texttt{sugar} \times \texttt{oil} \\
\texttt{fridge} &: \texttt{meat} \times \texttt{egg} \times \texttt{butter} \\
\texttt{split} &: \texttt{egg} \to \texttt{white} \times \texttt{yolk} \times \texttt{shell} \\
\texttt{whisk} &: \texttt{white} \to \texttt{foam} \\
\texttt{oven} &: \texttt{foam} \times \texttt{sugar} \to \texttt{meringue} \\
\texttt{pan} &: \texttt{meat} \times (\texttt{oil} + \texttt{butter}) \to \texttt{steak}
\end{aligned}$$

$$\vdash$$

$$\texttt{oven}\left(? : \texttt{foam} \qquad , ? : \texttt{sugar} \qquad \right)$$

$$
\begin{aligned}
\texttt{kitchen} &: \texttt{salt} \times \texttt{sugar} \times \texttt{oil} \\
\texttt{fridge} &: \texttt{meat} \times \texttt{egg} \times \texttt{butter} \\
\texttt{split} &: \texttt{egg} \to \texttt{white} \times \texttt{yolk} \times \texttt{shell} \\
\texttt{whisk} &: \texttt{white} \to \texttt{foam} \\
\texttt{oven} &: \texttt{foam} \times \texttt{sugar} \to \texttt{meringue} \\
\texttt{pan} &: \texttt{meat} \times (\texttt{oil} + \texttt{butter}) \to \texttt{steak}
\end{aligned}
$$

$$\vdash$$

$$\texttt{oven}\left(? : \texttt{foam} \qquad\qquad , ? : \texttt{sugar} \qquad \right)$$

$$
\begin{aligned}
\texttt{kitchen} &: \texttt{salt} \times \texttt{sugar} \times \texttt{oil} \\
\texttt{fridge} &: \texttt{meat} \times \texttt{egg} \times \texttt{butter} \\
\texttt{split} &: \texttt{egg} \to \texttt{white} \times \texttt{yolk} \times \texttt{shell} \\
\texttt{whisk} &: \texttt{white} \to \texttt{foam} \\
\texttt{oven} &: \texttt{foam} \times \texttt{sugar} \to \texttt{meringue} \\
\texttt{pan} &: \texttt{meat} \times (\texttt{oil} + \texttt{butter}) \to \texttt{steak}
\end{aligned}
$$

$$\vdash$$

$$\texttt{oven}\left(? : \texttt{foam} \qquad\qquad , \pi_{\texttt{sugar}}\ \texttt{kitchen}\right)$$

$$
\begin{aligned}
\texttt{kitchen} &: \texttt{salt} \times \texttt{sugar} \times \texttt{oil} \\
\texttt{fridge} &: \texttt{meat} \times \texttt{egg} \times \texttt{butter} \\
\texttt{split} &: \texttt{egg} \to \texttt{white} \times \texttt{yolk} \times \texttt{shell} \\
\texttt{whisk} &: \texttt{white} \to \texttt{foam} \\
\texttt{oven} &: \texttt{foam} \times \texttt{sugar} \to \texttt{meringue} \\
\texttt{pan} &: \texttt{meat} \times (\texttt{oil} + \texttt{butter}) \to \texttt{steak}
\end{aligned}
$$

$$\vdash$$

$$\texttt{oven}\left(? : \texttt{foam} \qquad\qquad , \pi_{\texttt{sugar}}\ \texttt{kitchen}\right)$$

$$\begin{aligned}
\texttt{kitchen} &: \texttt{salt} \times \texttt{sugar} \times \texttt{oil} \\
\texttt{fridge} &: \texttt{meat} \times \texttt{egg} \times \texttt{butter} \\
\texttt{split} &: \texttt{egg} \to \texttt{white} \times \texttt{yolk} \times \texttt{shell} \\
\texttt{whisk} &: \texttt{white} \to \texttt{foam} \\
\texttt{oven} &: \texttt{foam} \times \texttt{sugar} \to \texttt{meringue} \\
\texttt{pan} &: \texttt{meat} \times (\texttt{oil} + \texttt{butter}) \to \texttt{steak}
\end{aligned}$$

$$\vdash$$

$$\texttt{oven}\left(\texttt{whisk}\,(? : \texttt{white} \qquad\qquad\qquad\qquad), \pi_{\texttt{sugar}}\,\texttt{kitchen}\right)$$

$$
\begin{aligned}
\texttt{kitchen} &: \texttt{salt} \times \texttt{sugar} \times \texttt{oil} \\
\texttt{fridge} &: \texttt{meat} \times \texttt{egg} \times \texttt{butter} \\
\texttt{split} &: \texttt{egg} \to \texttt{white} \times \texttt{yolk} \times \texttt{shell} \\
\texttt{whisk} &: \texttt{white} \to \texttt{foam} \\
\texttt{oven} &: \texttt{foam} \times \texttt{sugar} \to \texttt{meringue} \\
\texttt{pan} &: \texttt{meat} \times (\texttt{oil} + \texttt{butter}) \to \texttt{steak}
\end{aligned}
$$

$$\vdash$$

$$\texttt{oven}\left(\texttt{whisk}\,(? : \texttt{white} \qquad\qquad\qquad\qquad), \pi_{\texttt{sugar}}\ \texttt{kitchen}\right)$$

$$
\begin{aligned}
\texttt{kitchen} &: \texttt{salt} \times \texttt{sugar} \times \texttt{oil} \\
\texttt{fridge} &: \texttt{meat} \times \texttt{egg} \times \texttt{butter} \\
\texttt{split} &: \texttt{egg} \to \texttt{white} \times \texttt{yolk} \times \texttt{shell} \\
\texttt{whisk} &: \texttt{white} \to \texttt{foam} \\
\texttt{oven} &: \texttt{foam} \times \texttt{sugar} \to \texttt{meringue} \\
\texttt{pan} &: \texttt{meat} \times (\texttt{oil} + \texttt{butter}) \to \texttt{steak}
\end{aligned}
$$

$$\vdash$$

$$\texttt{oven}\left(\texttt{whisk}\,(\pi_{\texttt{white}}\,(\texttt{split}\,(? : \texttt{egg} \qquad))), \pi_{\texttt{sugar}}\,\texttt{kitchen}\right)$$

$$
\begin{aligned}
\texttt{kitchen} &: \texttt{salt} \times \texttt{sugar} \times \texttt{oil} \\
\texttt{fridge} &: \texttt{meat} \times \texttt{egg} \times \texttt{butter} \\
\texttt{split} &: \texttt{egg} \to \texttt{white} \times \texttt{yolk} \times \texttt{shell} \\
\texttt{whisk} &: \texttt{white} \to \texttt{foam} \\
\texttt{oven} &: \texttt{foam} \times \texttt{sugar} \to \texttt{meringue} \\
\texttt{pan} &: \texttt{meat} \times (\texttt{oil} + \texttt{butter}) \to \texttt{steak}
\end{aligned}
$$

$$\vdash$$

$$\texttt{oven}\left(\texttt{whisk}\,(\pi_{\texttt{white}}\,(\texttt{split}\,(? : \texttt{egg} \qquad))), \pi_{\texttt{sugar}}\,\texttt{kitchen}\right)$$

$$\begin{aligned}
\mathtt{kitchen} &: \mathtt{salt} \times \mathtt{sugar} \times \mathtt{oil} \\
\mathtt{fridge} &: \mathtt{meat} \times \mathtt{egg} \times \mathtt{butter} \\
\mathtt{split} &: \mathtt{egg} \to \mathtt{white} \times \mathtt{yolk} \times \mathtt{shell} \\
\mathtt{whisk} &: \mathtt{white} \to \mathtt{foam} \\
\mathtt{oven} &: \mathtt{foam} \times \mathtt{sugar} \to \mathtt{meringue} \\
\mathtt{pan} &: \mathtt{meat} \times (\mathtt{oil} + \mathtt{butter}) \to \mathtt{steak}
\end{aligned}$$

$$\vdash$$

$$\mathtt{oven}\left(\mathtt{whisk}\,(\pi_{\mathtt{white}}\,(\mathtt{split}\,(\pi_{\mathtt{egg}}\,\mathtt{fridge}))), \pi_{\mathtt{sugar}}\,\mathtt{kitchen}\right)$$

$$
\begin{aligned}
\texttt{kitchen} &: \texttt{salt} \times \texttt{sugar} \times \texttt{oil} \\
\texttt{fridge} &: \texttt{meat} \times \texttt{egg} \times \texttt{butter} \\
\texttt{split} &: \texttt{egg} \to \texttt{white} \times \texttt{yolk} \times \texttt{shell} \\
\texttt{whisk} &: \texttt{white} \to \texttt{foam} \\
\texttt{oven} &: \texttt{foam} \times \texttt{sugar} \to \texttt{meringue} \\
\texttt{pan} &: \texttt{meat} \times (\texttt{oil} + \texttt{butter}) \to \texttt{steak}
\end{aligned}
$$

$$\vdash$$

$$\texttt{oven}\left(\texttt{whisk}\,(\pi_{\texttt{white}}\,(\texttt{split}\,(\pi_{\texttt{egg}}\,\texttt{fridge}))), \pi_{\texttt{sugar}}\,\texttt{kitchen}\right)$$

$$
\begin{aligned}
\texttt{kitchen} &: \texttt{salt} \times \texttt{sugar} \times \texttt{oil} \\
\texttt{fridge} &: \texttt{meat} \times \texttt{egg} \times \texttt{butter} \\
\texttt{split} &: \texttt{egg} \to \texttt{white} \times \texttt{yolk} \times \texttt{shell} \\
\texttt{whisk} &: \texttt{white} \to \texttt{foam} \\
\texttt{oven} &: \texttt{foam} \times \texttt{sugar} \to \texttt{meringue} \\
\texttt{pan} &: \texttt{meat} \times (\texttt{oil} + \texttt{butter}) \to \texttt{steak}
\end{aligned}
$$

$$\vdash$$

$\texttt{?} : \texttt{steak}$

$$
\begin{aligned}
\texttt{kitchen} &: \texttt{salt} \times \texttt{sugar} \times \texttt{oil} \\
\texttt{fridge} &: \texttt{meat} \times \texttt{egg} \times \texttt{butter} \\
\texttt{split} &: \texttt{egg} \to \texttt{white} \times \texttt{yolk} \times \texttt{shell} \\
\texttt{whisk} &: \texttt{white} \to \texttt{foam} \\
\texttt{oven} &: \texttt{foam} \times \texttt{sugar} \to \texttt{meringue} \\
\texttt{pan} &: \texttt{meat} \times (\texttt{oil} + \texttt{butter}) \to \texttt{steak}
\end{aligned}
$$

$$\vdash$$

$? : \texttt{steak}$

$$\begin{aligned}
\texttt{kitchen} &: \texttt{salt} \times \texttt{sugar} \times \texttt{oil} \\
\texttt{fridge} &: \texttt{meat} \times \texttt{egg} \times \texttt{butter} \\
\texttt{split} &: \texttt{egg} \to \texttt{white} \times \texttt{yolk} \times \texttt{shell} \\
\texttt{whisk} &: \texttt{white} \to \texttt{foam} \\
\texttt{oven} &: \texttt{foam} \times \texttt{sugar} \to \texttt{meringue} \\
\texttt{pan} &: \texttt{meat} \times (\texttt{oil} + \texttt{butter}) \to \texttt{steak}
\end{aligned}$$

$$\vdash$$

$$\texttt{pan}\left(? : \texttt{meat} \qquad , ? : \texttt{oil} + \texttt{butter} \qquad \right)$$

$$\begin{aligned}
\texttt{kitchen} &: \texttt{salt} \times \texttt{sugar} \times \texttt{oil} \\
\texttt{fridge} &: \texttt{meat} \times \texttt{egg} \times \texttt{butter} \\
\texttt{split} &: \texttt{egg} \to \texttt{white} \times \texttt{yolk} \times \texttt{shell} \\
\texttt{whisk} &: \texttt{white} \to \texttt{foam} \\
\texttt{oven} &: \texttt{foam} \times \texttt{sugar} \to \texttt{meringue} \\
\texttt{pan} &: \texttt{meat} \times (\texttt{oil} + \texttt{butter}) \to \texttt{steak}
\end{aligned}$$

$$\vdash$$

$$\texttt{pan}\left(? : \texttt{meat} \qquad , ? : \texttt{oil} + \texttt{butter} \qquad\right)$$

$$
\begin{aligned}
\texttt{kitchen} &: \texttt{salt} \times \texttt{sugar} \times \texttt{oil} \\
\texttt{fridge} &: \texttt{meat} \times \texttt{egg} \times \texttt{butter} \\
\texttt{split} &: \texttt{egg} \to \texttt{white} \times \texttt{yolk} \times \texttt{shell} \\
\texttt{whisk} &: \texttt{white} \to \texttt{foam} \\
\texttt{oven} &: \texttt{foam} \times \texttt{sugar} \to \texttt{meringue} \\
\texttt{pan} &: \texttt{meat} \times (\texttt{oil} + \texttt{butter}) \to \texttt{steak}
\end{aligned}
$$

$$\vdash$$

$$\texttt{pan} \left(\pi_{\texttt{meat}}\, \texttt{fridge}, ? : \texttt{oil} + \texttt{butter} \qquad \right)$$

$$
\begin{aligned}
\texttt{kitchen} &: \texttt{salt} \times \texttt{sugar} \times \texttt{oil} \\
\texttt{fridge} &: \texttt{meat} \times \texttt{egg} \times \texttt{butter} \\
\texttt{split} &: \texttt{egg} \to \texttt{white} \times \texttt{yolk} \times \texttt{shell} \\
\texttt{whisk} &: \texttt{white} \to \texttt{foam} \\
\texttt{oven} &: \texttt{foam} \times \texttt{sugar} \to \texttt{meringue} \\
\texttt{pan} &: \texttt{meat} \times (\texttt{oil} + \texttt{butter}) \to \texttt{steak}
\end{aligned}
$$

$$\vdash$$

$$\texttt{pan}\left(\pi_{\texttt{meat}}\ \texttt{fridge}, ? : \texttt{oil} + \texttt{butter} \qquad \right)$$

$$
\begin{aligned}
\texttt{kitchen} &: \texttt{salt} \times \texttt{sugar} \times \texttt{oil} \\
\texttt{fridge} &: \texttt{meat} \times \texttt{egg} \times \texttt{butter} \\
\texttt{split} &: \texttt{egg} \to \texttt{white} \times \texttt{yolk} \times \texttt{shell} \\
\texttt{whisk} &: \texttt{white} \to \texttt{foam} \\
\texttt{oven} &: \texttt{foam} \times \texttt{sugar} \to \texttt{meringue} \\
\texttt{pan} &: \texttt{meat} \times (\texttt{oil} + \texttt{butter}) \to \texttt{steak}
\end{aligned}
$$

$$\vdash$$

$$
\texttt{pan}\left(\pi_{\texttt{meat}}\,\texttt{fridge}, \left\{\begin{array}{l} ? : \texttt{oil} \\ ? : \texttt{butter} \end{array}\right\}\right)
$$

$$
\begin{aligned}
\texttt{kitchen} &: \texttt{salt} \times \texttt{sugar} \times \texttt{oil} \\
\texttt{fridge} &: \texttt{meat} \times \texttt{egg} \times \texttt{butter} \\
\texttt{split} &: \texttt{egg} \to \texttt{white} \times \texttt{yolk} \times \texttt{shell} \\
\texttt{whisk} &: \texttt{white} \to \texttt{foam} \\
\texttt{oven} &: \texttt{foam} \times \texttt{sugar} \to \texttt{meringue} \\
\texttt{pan} &: \texttt{meat} \times (\texttt{oil} + \texttt{butter}) \to \texttt{steak}
\end{aligned}
$$

$$\vdash$$

$$
\texttt{pan}\left(\pi_{\texttt{meat}}\,\texttt{fridge}, \begin{Bmatrix} ? : \texttt{oil} \\ ? : \texttt{butter} \end{Bmatrix}\right)
$$

$$
\begin{aligned}
\texttt{kitchen} &: \texttt{salt} \times \texttt{sugar} \times \texttt{oil} \\
\texttt{fridge} &: \texttt{meat} \times \texttt{egg} \times \texttt{butter} \\
\texttt{split} &: \texttt{egg} \to \texttt{white} \times \texttt{yolk} \times \texttt{shell} \\
\texttt{whisk} &: \texttt{white} \to \texttt{foam} \\
\texttt{oven} &: \texttt{foam} \times \texttt{sugar} \to \texttt{meringue} \\
\texttt{pan} &: \texttt{meat} \times (\texttt{oil} + \texttt{butter}) \to \texttt{steak}
\end{aligned}
$$

$$\vdash$$

$$
\texttt{pan}\left(\pi_{\texttt{meat}}\ \texttt{fridge}, \left\{ \begin{array}{l} \pi_{\texttt{oil}}\ \texttt{kitchen} \\ ? : \texttt{butter} \end{array} \right\}\right)
$$

$$
\begin{aligned}
\texttt{kitchen} &: \texttt{salt} \times \texttt{sugar} \times \texttt{oil} \\
\texttt{fridge} &: \texttt{meat} \times \texttt{egg} \times \texttt{butter} \\
\texttt{split} &: \texttt{egg} \to \texttt{white} \times \texttt{yolk} \times \texttt{shell} \\
\texttt{whisk} &: \texttt{white} \to \texttt{foam} \\
\texttt{oven} &: \texttt{foam} \times \texttt{sugar} \to \texttt{meringue} \\
\texttt{pan} &: \texttt{meat} \times (\texttt{oil} + \texttt{butter}) \to \texttt{steak}
\end{aligned}
$$

$$\vdash$$

$$\texttt{pan}\left(\pi_{\texttt{meat}}\ \texttt{fridge}, \left\{ \begin{array}{l} \pi_{\texttt{oil}}\ \texttt{kitchen} \\ ? : \texttt{butter} \end{array} \right\}\right)$$

$$
\begin{aligned}
\texttt{kitchen} &: \texttt{salt} \times \texttt{sugar} \times \texttt{oil} \\
\texttt{fridge} &: \texttt{meat} \times \texttt{egg} \times \texttt{butter} \\
\texttt{split} &: \texttt{egg} \to \texttt{white} \times \texttt{yolk} \times \texttt{shell} \\
\texttt{whisk} &: \texttt{white} \to \texttt{foam} \\
\texttt{oven} &: \texttt{foam} \times \texttt{sugar} \to \texttt{meringue} \\
\texttt{pan} &: \texttt{meat} \times (\texttt{oil} + \texttt{butter}) \to \texttt{steak}
\end{aligned}
$$

$$\vdash$$

$$\texttt{pan}\left(\pi_{\texttt{meat}}\,\texttt{fridge}, \left\{\begin{array}{l} \pi_{\texttt{oil}}\,\texttt{kitchen} \\ \pi_{\texttt{butter}}\,\texttt{fridge} \end{array}\right\}\right)$$

Two different choices!

$$
\begin{array}{ll}
\texttt{kitchen} & : \texttt{salt} \times \texttt{sugar} \times \texttt{oil} \\
\texttt{fridge} & : \texttt{meat} \times \texttt{egg} \times \texttt{butter} \\
\texttt{split} & : \texttt{egg} \to \texttt{white} \times \texttt{yolk} \times \texttt{shell} \\
\texttt{whisk} & : \texttt{white} \to \texttt{foam} \\
\texttt{oven} & : \texttt{foam} \times \texttt{sugar} \to \texttt{meringue} \\
\texttt{pan} & : \texttt{meat} \times (\texttt{oil} + \texttt{butter}) \to \texttt{steak}
\end{array}
$$

$$\vdash$$

$$\texttt{pan}\left(\pi_{\texttt{meat}}\,\texttt{fridge}, \left\{\begin{array}{l} \pi_{\texttt{oil}}\,\texttt{kitchen} \\ \pi_{\texttt{butter}}\,\texttt{fridge} \end{array}\right\}\right)$$

Two different choices!

Additional difficulties: equivalent programs, infinite search.

Proofs-programs correspondence (Curry-Howard)

Another domain has to respect strict rules of well-formedness: mathematical proofs.

Types connect programming languages to logic (study of formal proofs).

Proofs-programs correspondence (Curry-Howard)

Another domain has to respect strict rules of well-formedness: mathematical proofs.

Types connect programming languages to logic (study of formal proofs).

Type system	**Logic**
Type	Proposition
Program	Formal proof
$A \times B$	A and B
$A + B$	A or B
$A \rightarrow B$	A implies B
Program search	Proof search

Proofs-programs correspondence (Curry-Howard)

Another domain has to respect strict rules of well-formedness: mathematical proofs.
Types connect programming languages to logic (study of formal proofs).

Type system	**Logic**
Type	Proposition
Program	Formal proof
$A \times B$	A and B
$A + B$	A or B
$A \to B$	A implies B
Program search	Proof search
Inhabitation	Provability

Proofs-programs correspondence (Curry-Howard)

Another domain has to respect strict rules of well-formedness: mathematical proofs.
Types connect programming languages to logic (study of formal proofs).

Type system	**Logic**
Type	Proposition
Program	Formal proof
$A \times B$	A and B
$A + B$	A or B
$A \rightarrow B$	A implies B
Program search	Proof search
Inhabitation	Provability
Unicity	N/A

Canonical proofs

Proof theory has studied the question of "canonical" proof representations.

Can we adopt these representations for programs?
Do they lose computational information?

Do logical techniques respect program equivalence?

Contribution

A functional programming problem:
Which types have a unique inhabitant?

A method from logic: focusing.
New bridge between the communities.
(Noam, Teyjus, Psyche, L. More elementary, new audience.)

A result: Unique inhabitation is decidable for the simply-typed lambda-calculus with atoms, sums and empty types.
An effective algorithm. For any $(\Gamma \vdash ? : A)$ returns 0, 1 or 2 programs.

Examples

Abstract library functions. Glue code.

$$\texttt{fst} : \alpha \times \beta \to \alpha \qquad \texttt{?} : \alpha \to \alpha \to \alpha$$

$$\texttt{curry} : (\alpha \times \beta \to \gamma) \to \alpha \to \beta \to \gamma$$

Examples

Abstract library functions. Glue code.

$$\texttt{fst} : \alpha \times \beta \to \alpha \qquad\qquad \texttt{?} : \alpha \to \alpha \to \alpha$$

$$\texttt{curry} : (\alpha \times \beta \to \gamma) \to \alpha \to \beta \to \gamma$$

$$\texttt{error}\ A \stackrel{\text{def}}{=} \texttt{exn} + A$$

$$\texttt{Monad error} \quad \begin{cases} \texttt{return} & : \ \alpha \to \texttt{error}\ \alpha \\ \texttt{bind} & : \ \texttt{error}\ \alpha \to (\alpha \to \texttt{error}\ \beta) \to \texttt{error}\ \beta \end{cases}$$

Examples

Abstract library functions. Glue code.

$$\texttt{fst} : \alpha \times \beta \to \alpha \qquad\qquad \texttt{?} : \alpha \to \alpha \to \alpha$$

$$\texttt{curry} : (\alpha \times \beta \to \gamma) \to \alpha \to \beta \to \gamma$$

$$\texttt{error}\ A \stackrel{\text{def}}{=} \texttt{exn} + A$$

$$\texttt{Monad error} \quad \begin{cases} \texttt{return} & : \ \alpha \to \texttt{error}\ \alpha \\ \texttt{bind} & : \ \texttt{error}\ \alpha \to (\alpha \to \texttt{error}\ \beta) \to \texttt{error}\ \beta \end{cases}$$

$\texttt{Functor error}$ (unique)

Examples

Abstract library functions. Glue code.

$$\texttt{fst} : \alpha \times \beta \to \alpha \qquad \texttt{?} : \alpha \to \alpha \to \alpha$$

$$\texttt{curry} : (\alpha \times \beta \to \gamma) \to \alpha \to \beta \to \gamma$$

$$\texttt{error}\ A \stackrel{\text{def}}{=} \texttt{exn} + A$$

$$\texttt{Monad error} \quad \begin{cases} \texttt{return} & : \ \alpha \to \texttt{error}\ \alpha \\ \texttt{bind} & : \ \texttt{error}\ \alpha \to (\alpha \to \texttt{error}\ \beta) \to \texttt{error}\ \beta \end{cases}$$

`Functor error` (unique)

`Applicative error` (not unique)

Examples

Abstract library functions. Glue code.

$$\texttt{fst} : \alpha \times \beta \to \alpha \qquad\qquad \texttt{?} : \alpha \to \alpha \to \alpha$$

$$\texttt{curry} : (\alpha \times \beta \to \gamma) \to \alpha \to \beta \to \gamma$$

$$\texttt{error}\ A \stackrel{\text{def}}{=} \texttt{exn} + A$$

$$\texttt{Monad error} \quad \begin{cases} \texttt{return} & : \ \alpha \to \texttt{error}\ \alpha \\ \texttt{bind} & : \ \texttt{error}\ \alpha \to (\alpha \to \texttt{error}\ \beta) \to \texttt{error}\ \beta \end{cases}$$

`Functor error` (unique)

`Applicative error` (not unique)

$$\texttt{ap} : \texttt{error}\ (\alpha \to \beta) \to \texttt{error}\ \alpha \to \texttt{error}\ \beta$$

General idea

Enumerate all possible terms in $(\Gamma \vdash ? : A)$:

$$t_1, t_2, \ldots, t_n$$

Stop if we find two distinct terms.

We must remove duplicates.

$$t_1, (\lambda x.\, x)\ t_1, (\lambda x.\, x)\ ((\lambda x.\, x)\ t_1), \ldots$$

Infinitely many duplicates $\Longrightarrow$ non-termination.

Enumerate terms without duplicate: canonical representations.

β-normal forms

Easy in the **negative** fragment $(\to, \times, 1)$. Values and neutrals.

$$
\begin{aligned}
v ::= &\ \text{values} \\
&\mid \lambda x.\, v \\
&\mid (v_1, v_2) \\
&\mid n
\end{aligned}
\qquad\qquad
\begin{aligned}
n ::= &\ \text{neutrals} \\
&\mid x \\
&\mid \pi_i\ n \\
&\mid n\ v
\end{aligned}
$$

Problem: no clear way to add **positives** $(+, 0)$.

β-normal forms

Easy in the **negative** fragment $(\to, \times, 1)$. Values and neutrals.

$$v ::= \text{values} \mid \lambda x.\, v \mid (v_1, v_2) \mid n$$

$$n ::= \text{neutrals} \mid x \mid \pi_i\ n \mid n\ v$$

Problem: no clear way to add **positives** $(+, 0)$.

$$v, w ::= \text{values} \mid \lambda x.\, v \mid (v_1, v_2) \mid n \mid\ ?\ \sigma_i\ v$$

$$n ::= \text{neutrals} \mid x \mid \pi_i\ n \mid n\ v$$
$$?\ \mid \texttt{match}\ n\ \texttt{with}\ \mid \sigma_1\ x_1 \to v_1 \mid \sigma_2\ x_2 \to v_2$$
$$?\ \mid \texttt{match}\ n\ \texttt{with}\ \mid \sigma_1\ x_1 \to n_1 \mid \sigma_2\ x_2 \to n_2$$

β-normal forms

Easy in the **negative** fragment $(\rightarrow, \times, 1)$. Values and neutrals.

$$
\begin{array}{rcl}
v & ::= & \text{values} \\
  & \mid & \lambda x.\, v \\
  & \mid & (v_1, v_2) \\
  & \mid & n
\end{array}
\qquad
\begin{array}{rcl}
n & ::= & \text{neutrals} \\
  & \mid & x \\
  & \mid & \pi_i\ n \\
  & \mid & n\ v
\end{array}
$$

Problem: no clear way to add **positives** $(+, 0)$.

$$
\begin{array}{rrl}
v, w & ::= & \text{values} \\
  & \mid & \lambda x.\, v \\
  & \mid & (v_1, v_2) \\
  & \mid & n \\
? & \mid & \sigma_i\ v
\end{array}
\qquad
\begin{array}{rrl}
n & ::= & \text{neutrals} \\
  & \mid & x \\
  & \mid & \pi_i\ n \\
  & \mid & n\ v \\
? & \mid & \texttt{match}\ n\ \texttt{with}\ \mid \sigma_1\ x_1 \rightarrow v_1 \mid \sigma_2\ x_2 \rightarrow v_2 \\
? & \mid & \texttt{match}\ n\ \texttt{with}\ \mid \sigma_1\ x_1 \rightarrow n_1 \mid \sigma_2\ x_2 \rightarrow n_2
\end{array}
$$

$$
\left(\texttt{match}\ n\ \texttt{with} \;\middle|\; \begin{array}{l} \sigma_1\ x_1 \rightarrow \lambda y.\, v_1 \\ \sigma_2\ x_2 \rightarrow \lambda y.\, v_2 \end{array} \right) w
$$

β-normal forms

Easy in the **negative** fragment $(\to, \times, 1)$. Values and neutrals.

$$v ::= \text{values} \mid \lambda x.\, v \mid (v_1, v_2) \mid n \qquad n ::= \text{neutrals} \mid x \mid \pi_i\ n \mid n\ v$$

Problem: no clear way to add **positives** $(+, 0)$.

$$v, w ::= \text{values} \mid \lambda x.\, v \mid (v_1, v_2) \mid n \mid^{?} \sigma_i\ v$$

$$n ::= \text{neutrals} \mid x \mid \pi_i\ n \mid n\ v \mid^{?} \texttt{match}\ n\ \texttt{with}\ \mid \sigma_1\ x_1 \to v_1 \mid \sigma_2\ x_2 \to v_2 \mid^{?} \texttt{match}\ n\ \texttt{with}\ \mid \sigma_1\ x_1 \to n_1 \mid \sigma_2\ x_2 \to n_2$$

$$\texttt{match}\ n\ \texttt{with} \left|\begin{array}{l} \sigma_1\ x_1 \to \sigma_1\ v_1 \\ \sigma_2\ x_2 \to \sigma_2\ v_2 \end{array}\right.$$

Focusing (overview)

$$\frac{}{x : X + Y \vdash Y + X} \quad ?$$

Focusing (overview)

$$\dfrac{\overline{x : X + Y \vdash Y}}{x : X + Y \vdash Y + X} \qquad \sigma_1 \; ?$$

Focusing (overview)

$$\dfrac{\dfrac{x_1 : X \vdash Y \qquad x_2 : Y \vdash Y}{x : X + Y \vdash Y}}{x : X + Y \vdash Y + X}$$

$$\sigma_1 \left(\texttt{match } x \texttt{ with} \left| \begin{array}{l} \sigma_1\, x_1 \to ? \\ \sigma_2\, x_2 \to ? \end{array} \right. \right)$$

Focusing (overview)

$$\dfrac{\dfrac{x_1 : X \vdash Y \qquad x_2 : Y \vdash Y}{x : X + Y \vdash Y}}{x : X + Y \vdash Y + X}$$

$$\sigma_1 \left(\texttt{match } x \texttt{ with } \middle| \begin{array}{l} \sigma_1\, x_1 \to ? \\ \sigma_2\, x_2 \to ? \end{array} \right)$$

Focusing (overview)

$$\dfrac{\dfrac{x_1 : X \vdash Y \qquad \dfrac{}{x_2 : Y \vdash Y}}{x : X + Y \vdash Y}}{x : X + Y \vdash Y + X}$$

$$\sigma_1 \left(\texttt{match } x \texttt{ with } \middle| \begin{array}{l} \sigma_1\, x_1 \to ? \\ \sigma_2\, x_2 \to x_2 \end{array} \right)$$

Focusing (overview)

$$\dfrac{x_1 : X \vdash Y \qquad \dfrac{}{x_2 : Y \vdash Y}}{\dfrac{x : X + Y \vdash Y}{x : X + Y \vdash Y + X}}$$

$$\sigma_1 \left(\texttt{match } x \texttt{ with } \left| \begin{array}{l} \sigma_1\, x_1 \to \,? \\ \sigma_2\, x_2 \to x_2 \end{array} \right. \right)$$

Focusing (overview)

$$\dfrac{\dfrac{x_1 : X \vdash Y \qquad \overline{x_2 : Y \vdash Y}}{x : X + Y \vdash Y}}{x : X + Y \vdash Y + X}$$

$$\sigma_1 \left(\texttt{match } x \texttt{ with } \middle| \begin{array}{l} \sigma_1\ x_1 \to ? \\ \sigma_2\ x_2 \to x_2 \end{array} \right)$$

If you apply rules in the wrong order, you can get stuck.

Focusing (overview)

$$\dfrac{\dfrac{x_1 : X \vdash Y \qquad \overline{x_2 : Y \vdash Y}}{x : X + Y \vdash Y}}{x : X + Y \vdash Y + X}$$

$$\sigma_1 \left(\texttt{match}\ x\ \texttt{with} \ \middle|\ \begin{array}{l} \sigma_1\, x_1 \to ? \\ \sigma_2\, x_2 \to x_2 \end{array} \right)$$

If you apply rules in the wrong order, you can get stuck.

invertible rule: cannot get stuck. `match` x `with` ...
non-invertible rule: risky **choice**. σ_i ...

Focusing (overview)

$$\cfrac{\cfrac{x_1 : X \vdash Y \qquad \overline{x_2 : Y \vdash Y}}{x : X + Y \vdash Y}}{x : X + Y \vdash Y + X}$$

$$\sigma_1 \left(\texttt{match}\ x\ \texttt{with} \;\middle|\; \begin{array}{l} \sigma_1\, x_1 \to ? \\ \sigma_2\, x_2 \to x_2 \end{array} \right)$$

If you apply rules in the wrong order, you can get stuck.

invertible rule: cannot get stuck. `match` x `with` ...
non-invertible rule: risky **choice**. σ_i ...

Focusing imposes restrictions based on invertibility.

Focused normal forms (Overview)

$$\begin{aligned} v ::= &\ \text{values} \\ &\mid \lambda x.\, v \\ &\mid (v_1, v_2) \\ &\mid n \end{aligned} \qquad \begin{aligned} n ::= &\ \text{neutrals} \\ &\mid x \\ &\mid \pi_i\ n \\ &\mid n\ v \end{aligned}$$

Remark that values are **invertible**, neutrals **non-invertible**.

Focused normal forms (Overview)

$$
\begin{aligned}
v ::=&\ \text{values} \\
&\mid \lambda x.\, v \\
&\mid (v_1, v_2) \\
&\mid n \\
&\mid \texttt{match}\ x\ \texttt{with}\ \mid \sigma_1\, x \to v_1 \mid \sigma_2\, x \to v_2
\end{aligned}
\qquad
\begin{aligned}
n ::=&\ \text{neutrals} \\
&\mid x \\
&\mid \pi_i\ n \\
&\mid n\ v
\end{aligned}
$$

Remark that values are **invertible**, neutrals **non-invertible**.

Focused normal forms (Overview)

$$
\begin{aligned}
v ::= &\ \text{values} \\
&\mid \lambda x.\, v \\
&\mid (v_1, v_2) \\
&\mid n \\
&\mid \texttt{match}\ x\ \texttt{with}\ \mid \sigma_1\, x \to v_1 \mid \sigma_2\, x \to v_2
\end{aligned}
\qquad
\begin{aligned}
n ::= &\ \text{neutrals} \\
&\mid x \\
&\mid \pi_i\ n \\
&\mid n\ v \\
&\mid \sigma_i\ n
\end{aligned}
$$

Remark that values are **invertible**, neutrals **non-invertible**.

Focused normal forms (Overview)

$$
\begin{aligned}
v ::= &\ \text{values} \\
&\mid \lambda x.\, v \\
&\mid (v_1, v_2) \\
&\mid n \\
&\mid \texttt{match}\ x\ \texttt{with}\ \mid \sigma_1\, x \to v_1 \mid \sigma_2\, x \to v_2
\end{aligned}
\qquad
\begin{aligned}
n ::= &\ \text{neutrals} \\
&\mid x \\
&\mid \pi_i\ n \\
&\mid n\ v \\
&\mid \sigma_i\ n
\end{aligned}
$$

Remark that values are **invertible**, neutrals **non-invertible**.

(The details are a bit different.)

Focusing is not enough

For

$$x : X + Y \vdash ? : Y + X$$

there is a unique focused normal form (good!).

But not for

$$f : 1 \to (X + Y) \vdash ? : Y + X$$

Applying a function is non-invertible, no imposed ordering.

Can be applied:

- before or after making choices
- zero, one, many times

Saturation

Focusing forces splitting of **values** of sum type.
Computations of sum type may be done at any point.
Duplicates (in a pure setting).

Saturation

Focusing forces splitting of **values** of sum type.
Computations of sum type may be done at any point.
Duplicates (in a pure setting).

Saturation: deduce **all** positives before making any choice.
Run computations of sum type as **early** as possible.

Saturation

Focusing forces splitting of **values** of sum type.
Computations of sum type may be done at any point.
Duplicates (in a pure setting).

Saturation: deduce **all** positives before making any choice.
Run computations of sum type as **early** as possible.

Stronger than focusing: no more duplicates.

Demo time

Implementation available:
`https://gitlab.com/gasche/unique-inhabitant`

$$f : (1 \to X + Y)$$

$$\vdash$$

$$? : Y + X$$

Demo time

Implementation available:
`https://gitlab.com/gasche/unique-inhabitant`

$$f : (1 \to X + Y)$$

$$\vdash$$

$$? : Y + X$$

Demo time

Implementation available:
`https://gitlab.com/gasche/unique-inhabitant`

$$f : (1 \to X + Y)$$

$$\vdash$$

$$\texttt{let } z^{X+Y} = f\ ()\ \texttt{in } ? : Y + X$$

Demo time

Implementation available:
`https://gitlab.com/gasche/unique-inhabitant`

$$f : (1 \to X + Y)$$

$$\vdash$$

$$\texttt{let}\ z^{X+Y} = f\ ()\ \texttt{in}\ ? : Y + X$$

Demo time

Implementation available:
`https://gitlab.com/gasche/unique-inhabitant`

$$f : (1 \to X + Y)$$

$$\vdash$$

$$\texttt{let } z^{X+Y} = f\ ()\ \texttt{in match } z \texttt{ with } \left|\ \begin{array}{l} \sigma_1\ x^X \to ? : Y + X \\ \sigma_2\ y^Y \to ? : Y + X \end{array}\right.$$

Demo time

Implementation available:
`https://gitlab.com/gasche/unique-inhabitant`

$$f : (1 \to X + Y)$$

$$\vdash$$

$$\texttt{let } z^{X+Y} = f\ ()\ \texttt{in match } z \texttt{ with} \left|\begin{array}{l} \sigma_1\ x^X \to\ ? : Y + X \\ \sigma_2\ y^Y \to\ ? : Y + X \end{array}\right.$$

Demo time

Implementation available:
`https://gitlab.com/gasche/unique-inhabitant`

$$f : (1 \to X + Y)$$

$$\vdash$$

$$\texttt{let } z^{X+Y} = f\ () \texttt{ in match } z \texttt{ with } \left| \begin{array}{l} \sigma_1\ x^X \to ? : Y + X \\ \sigma_2\ y^Y \to ? : Y + X \end{array} \right.$$

Demo time

Implementation available:
`https://gitlab.com/gasche/unique-inhabitant`

$$f : (1 \to X + Y)$$

$$\vdash$$

$$\boxed{\texttt{let } z^{X+Y} = f\ ()\ \texttt{in match } z \texttt{ with} \left|\begin{array}{l} \sigma_1\ x^X \to \sigma_2\ x \\ \sigma_2\ y^Y \to ? : Y + X \end{array}\right.}$$

Demo time

Implementation available:
`https://gitlab.com/gasche/unique-inhabitant`

$$f : (1 \to X + Y)$$

$$\vdash$$

$$\boxed{\texttt{let}\ z^{X+Y} = f\ ()\ \texttt{in}\ \texttt{match}\ z\ \texttt{with}\ \left|\ \begin{array}{l} \sigma_1\ x^X \to \sigma_2\ x \\ \sigma_2\ y^Y \to \sigma_1\ y \end{array}\right.}$$

Demo time

Implementation available:
`https://gitlab.com/gasche/unique-inhabitant`

$$f : (1 \to X + Y)$$

$$\vdash$$

$$\boxed{\texttt{let}\ z^{X+Y} = f\ ()\ \texttt{in}\ \texttt{match}\ z\ \texttt{with}\ \left|\ \begin{array}{l} \sigma_1\ x^X \to \sigma_2\ x \\ \sigma_2\ y^Y \to \sigma_1\ y \end{array}\right.}$$

Final result: zero, one or two (distinct) terms.

Two-or-more approximation

Does saturation terminate in general?

$$f : \mathbb{N} \to Y + Y, \ldots$$

Even when they are infinitely many programs, **proof** search terminates.

Subformula property: normal forms use finitely many types/formulas.

Two-or-more approximation

Does saturation terminate in general?

$$f : \mathbb{N} \to Y + Y, \ldots$$

Even when they are infinitely many programs, **proof** search terminates.

Subformula property: normal forms use finitely many types/formulas.

- proofs environments: sets of formulas $(A, B, \ldots)$
- typing environments: variable$\to$type mappings $(x : A, y : B, \ldots)$

Two-or-more approximation

Does saturation terminate in general?

$$f : \mathbb{N} \to Y + Y, \dots$$

Even when they are infinitely many programs, **proof** search terminates.

Subformula property: normal forms use finitely many types/formulas.

- proofs environments: sets of formulas $(A, B, \dots)$
- typing environments: variable→type mappings $(x : A, y : B, \dots)$

$$x_1 : X, x_2 : X, x_3 : X, \dots$$

Two-or-more approximation

Does saturation terminate in general?

$$f : \mathbb{N} \to Y + Y, \dots$$

Even when they are infinitely many programs, **proof** search terminates.

Subformula property: normal forms use finitely many types/formulas.

- proofs environments: sets of formulas $(A, B, \dots)$
- typing environments: variable→type mappings $(x : A, y : B, \dots)$

$$x_1 : X, x_2 : X, x_3 : X, \dots$$

Result: to detect unicity, keep at most **two** variables of each type.

Two-or-more approximation

Does saturation terminate in general?

$$f : \mathbb{N} \to Y + Y, \ldots$$

Even when they are infinitely many programs, **proof** search terminates.

Subformula property: normal forms use finitely many types/formulas.

- proofs environments: sets of formulas $(A, B, \ldots)$
- typing environments: variable$\to$type mappings $(x : A, y : B, \ldots)$

$$x_1 : X, x_2 : X, x_3 : X, \ldots$$

Result: to detect unicity, keep at most **two** variables of each type.

Consequence: Saturation can stop after **two** proofs of each positive:

$$0, 1, \text{many}$$

Sequent calculus (logic)

$$\frac{\Gamma \vdash A \qquad \Gamma, B \vdash C}{\Gamma, A \to B \vdash C} -$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}$$

Sequent calculus (logic)

$$\frac{\Gamma \vdash A \qquad \Gamma, B \vdash C}{\Gamma, A \to B \vdash C}$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}$$

$$\frac{\Gamma, A_i \vdash C}{\Gamma, A_1 \times A_2 \vdash C}$$

$$\frac{\Gamma \vdash A_1 \qquad \Gamma \vdash A_2}{\Gamma \vdash A_1 \times A_2}$$

$$\frac{}{\Gamma \vdash 1}$$

Sequent calculus (logic)

$$\frac{\Gamma \vdash A \qquad \Gamma, B \vdash C}{\Gamma, A \to B \vdash C} -$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}$$

$$\frac{\Gamma, A_i \vdash C}{\Gamma, A_1 \times A_2 \vdash C} -$$

$$\frac{\Gamma \vdash A_1 \qquad \Gamma \vdash A_2}{\Gamma \vdash A_1 \times A_2}$$

$$\frac{}{\Gamma \vdash 1}$$

$$\frac{\Gamma, A_1 \vdash C \qquad \Gamma, A_2 \vdash C}{\Gamma, A_1 + A_2 \vdash C}$$

$$\frac{\Gamma \vdash A_i}{\Gamma \vdash A_1 + A_2} +$$

$$\frac{}{\Gamma, 0 \vdash A}$$

Sequent calculus (logic)

$$\frac{\Gamma \vdash A \qquad \Gamma, B \vdash C}{\Gamma, A \to B \vdash C} -$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}$$

$$\frac{\Gamma, A_i \vdash C}{\Gamma, A_1 \times A_2 \vdash C} -$$

$$\frac{\Gamma \vdash A_1 \qquad \Gamma \vdash A_2}{\Gamma \vdash A_1 \times A_2}$$

$$\frac{}{\Gamma \vdash 1}$$

$$\frac{\Gamma, A_1 \vdash C \qquad \Gamma, A_2 \vdash C}{\Gamma, A_1 + A_2 \vdash C}$$

$$\frac{\Gamma \vdash A_i}{\Gamma \vdash A_1 + A_2} +$$

$$\frac{}{\Gamma, 0 \vdash A}$$

Negatives: $(\to, \times, 1)$

Positives: $(+, 0)$

Focusing

Invertible phase: apply invertible rules as early as possible.

$$\Gamma \vdash_{\mathsf{inv}} A$$

Focusing

Invertible phase: apply invertible rules as early as possible.

$$\Gamma \vdash_{\mathsf{inv}} A$$

When all invertible rules are done, we must **choose** a focus.

$$\Gamma \vdash_{\mathsf{foc}} A$$

(Γ negative, A positive)

Focusing

Invertible phase: apply invertible rules as early as possible.

$$\Gamma \vdash_{\mathsf{inv}} A$$

When all invertible rules are done, we must **choose** a focus.

$$\Gamma \vdash_{\mathsf{foc}} A$$

(Γ negative, A positive)

Focus on a formula:

- on the right: $\Gamma \vdash_{\mathsf{foc.r}} [A]$
- or on the left: $\Gamma, [A] \vdash_{\mathsf{foc.l}} B$

Then keep applying non-invertible rules on the formula under focus.

Focused λ-calculus

$$
\begin{aligned}
t, u, r ::= \quad & \textit{invertible} \\
\mid\ & \lambda x.\, t \\
\mid\ & (t, u) \\
\mid\ & \texttt{match } x \texttt{ with} \left| \begin{array}{l} \sigma_1\, x \to u_1 \\ \sigma_2\, x \to u_2 \end{array} \right. \\
\mid\ & () \\
\mid\ & \texttt{absurd}(x) \\
\mid\ & (f : A_{pa})
\end{aligned}
$$

$$
\begin{aligned}
f, g ::= \quad & \textit{choice of focus} \\
\mid\ & (n : X) \\
\mid\ & \texttt{let } (x : A_p) = n \texttt{ in } t \\
\mid\ & (p : A_p)
\end{aligned}
$$

$$
\begin{aligned}
n, m ::= \quad & \textit{negative neutral} \\
\mid\ & \pi_i\, n \\
\mid\ & n\, p \\
\mid\ & (x : A_n)
\end{aligned}
$$

$$
\begin{aligned}
p, q ::= \quad & \textit{positive neutral} \\
\mid\ & \sigma_i\, p \\
\mid\ & (x : X) \\
\mid\ & (t : A_{na})
\end{aligned}
$$

Saturation

$$\frac{\Gamma, \Delta \vdash_{\mathsf{foc}} A}{\Gamma; \Delta \vdash_{\mathsf{inv}} A}$$

Old and New.

Saturation

$$\frac{\Gamma, \Delta \vdash_{\text{foc}} A}{\Gamma; \Delta \vdash_{\text{inv}} A}$$

Old and New.

Left focusing: $\Gamma_{na} \vdash \texttt{let}\ (x : A_p) = n\ \texttt{in}\ t$

Saturation: $\Gamma; \Delta \vdash \texttt{let}\ (x_i)^{i \in I} = (n_i)^{i \in I}\ \texttt{in}\ t$

$$\text{where}\ (n_i)^{i \in I} \stackrel{\text{def}}{=} \{\ \Gamma, \Delta \vdash n \Downarrow A_p \ \mid\ n \text{ uses } \Delta\ \}_2$$

Saturation

$$\frac{\Gamma, \Delta \vdash_{\text{foc}} A}{\Gamma; \Delta \vdash_{\text{inv}} A}$$

Old and New.

Left focusing: $\Gamma_{na} \vdash \texttt{let}\ (x : A_p) = n\ \texttt{in}\ t$

Saturation: $\Gamma; \Delta \vdash \texttt{let}\ (x_i)^{i \in I} = (n_i)^{i \in I}\ \texttt{in}\ t$

where $(n_i)^{i \in I} \stackrel{\text{def}}{=} \{\ \Gamma, \Delta \vdash n \Downarrow A_p \mid n \text{ uses } \Delta\ \}_2$

All typed terms: $\Gamma \vdash t : A$

Focused terms: $\Gamma^{\text{at}} \vdash_{\text{foc}} f : P^{\text{at}}$

Saturated terms: $\Gamma^{\text{at}}; \Gamma^{\text{at}\prime} \vdash_{\text{sat}} g : P^{\text{at}}$

Canonicity

Completeness

Focusing: (known result) $\Gamma^{\mathsf{at}} \vdash_{\mathsf{foc}} t \rightsquigarrow f : P^{\mathsf{at}}$ with $t \approx_{\beta\eta} f$

Saturation: $\Gamma^{\mathsf{at}}; \Gamma^{\mathsf{at}\prime} \vdash_{\mathsf{sat}} f \rightsquigarrow g : P^{\mathsf{at}}$ with $f \approx_{\beta\eta} g$

Theorem (Canonicity)

$$\left.\begin{array}{l} \Gamma^{\mathsf{at}} \vdash_{\mathsf{foc}} f_1 \approx_{\beta\eta} f_2 : P^{\mathsf{at}} \\ \emptyset; \Gamma^{\mathsf{at}} \vdash_{\mathsf{sat}} f_1 \rightsquigarrow g_1 : P^{\mathsf{at}} \\ \emptyset; \Gamma^{\mathsf{at}} \vdash_{\mathsf{sat}} f_2 \rightsquigarrow g_2 : P^{\mathsf{at}} \end{array}\right\} \implies g_1 = g_2$$

Canonicity

Completeness

Focusing: (known result)	$\Gamma^{at} \vdash_{foc} t \rightsquigarrow f : P^{at}$	with $t \approx_{\beta\eta} f$
Saturation:	$\Gamma^{at}; \Gamma^{at\prime} \vdash_{sat} f \rightsquigarrow g : P^{at}$	with $f \approx_{\beta\eta} g$

Theorem (Canonicity)

$$\left.\begin{array}{l} \Gamma^{at} \vdash_{foc} f_1 \approx_{\beta\eta} f_2 : P^{at} \\ \emptyset; \Gamma^{at} \vdash_{sat} f_1 \rightsquigarrow g_1 : P^{at} \\ \emptyset; \Gamma^{at} \vdash_{sat} f_2 \rightsquigarrow g_2 : P^{at} \end{array}\right\} \implies g_1 = g_2$$

Result: searching among saturated terms decides unicity

Canonicity

Completeness

Focusing: (known result)	$\Gamma^{at} \vdash_{foc} t \rightsquigarrow f : P^{at}$	with $t \approx_{\beta\eta} f$
Saturation:	$\Gamma^{at}; \Gamma^{at\prime} \vdash_{sat} f \rightsquigarrow g : P^{at}$	with $f \approx_{\beta\eta} g$

Theorem (Canonicity)

$$\left.\begin{array}{l} \Gamma^{at} \vdash_{foc} f_1 \approx_{\beta\eta} f_2 : P^{at} \\ \emptyset; \Gamma^{at} \vdash_{sat} f_1 \rightsquigarrow g_1 : P^{at} \\ \emptyset; \Gamma^{at} \vdash_{sat} f_2 \rightsquigarrow g_2 : P^{at} \end{array}\right\} \implies g_1 = g_2$$

Result: searching among saturated terms decides unicity

Corollary: equivalence with sums and empty type is decidable

Future work

Semantic proof of equivalence with sums and empty type.

$$g_1 = g_2 \qquad \Rightarrow \qquad g_1 \approx_{\beta\eta} g_2 \qquad \Rightarrow \qquad g_1 \approx_{\texttt{ctx}} g_2$$

Future work

Semantic proof of equivalence with sums and empty type.

$$g_1 = g_2 \qquad \Leftarrow \qquad g_1 \approx_{\beta\eta} g_2 \qquad\qquad g_1 \approx_{\texttt{ctx}} g_2$$

Future work

Semantic proof of equivalence with sums and empty type.

$$g_1 = g_2 \quad \Longleftarrow \quad g_1 \approx_{\texttt{ctx}} g_2$$

Future work

Semantic proof of equivalence with sums and empty type.

$$g_1 = g_2 \qquad \Longleftarrow \qquad g_1 \approx_{\mathtt{ctx}} g_2$$

System L.

Future work

Semantic proof of equivalence with sums and empty type.

$$g_1 = g_2 \quad \Longleftarrow \quad g_1 \approx_{\mathtt{ctx}} g_2$$

System L.

Implement the feature in programming languages (ML).
Design choices.

Future work

Semantic proof of equivalence with sums and empty type.

$$g_1 = g_2 \qquad \Longleftarrow \qquad g_1 \approx_{\texttt{ctx}} g_2$$

System L.

Implement the feature in programming languages (ML).
Design choices.

Richer type systems (polymorphism, dependency...)

Undecidable; hopefully inspire semi-decision procedure.
Which equivalence to consider?
The richer the system, the more unique types.

Future work

Semantic proof of equivalence with sums and empty type.

$$g_1 = g_2 \qquad \Longleftarrow \qquad g_1 \approx_{\texttt{ctx}} g_2$$

System L.

Implement the feature in programming languages (ML).
Design choices.

Richer type systems (polymorphism, dependency...)
    Undecidable; hopefully inspire semi-decision procedure.
    Which equivalence to consider?
    The richer the system, the more unique types.

Unicity modulo equations?

Conclusion

Program inference is not yet fully understood;
we propose a **principled** approach.

Focusing and saturation
inspire powerful program representations.

Purity and types are not just restrictions

Conclusion

Program inference is not yet fully understood;
we propose a **principled** approach.

Focusing and saturation
inspire powerful program representations.

Purity and types are not just restrictions
they can make programming **easier**.