Università
degli Studi
di Padova

Value-passing CCS Compiler

Languages for Concurrency and Distribution exam project

Alberto Lazari

April 4, 2024

Syntax

Syntax to define

- vCCS, for the parser
- CCS, for encoding and output printing

Syntax to define

- vCCS, for the parser
- CCS, for encoding and output printing

Inspired from CAAL’s syntax

Value-passing CCS syntax

Constants

- $n \in \mathbb{N}$
- $k \in K$
- $x \in \mathrm{Var}$
- $a \in \mathcal{A}$

Expressions

$$
\begin{aligned}
e ::= &\quad n \\
| &\quad (e) \\
| &\quad x \\
| &\quad e \text{ abop } e
\end{aligned}
$$

$$\text{abop} ::= + \mid - \mid * \mid /$$

Booleans

$$
\begin{aligned}
b ::= \quad & \mathrm{true} \mid \mathrm{false} \\
\mid \quad & (b) \\
\mid \quad & \mathrm{not}\ b \\
\mid \quad & b \ \mathrm{and}\ b \mid b \ \mathrm{or}\ b \\
\mid \quad & e \ \mathrm{bbop}\ e
\end{aligned}
$$

$$
\mathrm{bbop} ::= \ = \mid \neq \mid < \mid > \mid \leqslant \mid \geqslant
$$

Processes

$$
\begin{aligned}
P ::= {} & 0 \\
\mid {} & (P) \\
\mid {} & \text{act}.P \\
\mid {} & k \mid k(\text{args}) \\
\mid {} & \text{if } b \text{ then } P \\
\mid {} & P + P \mid P \mid P \\
\mid {} & P[f] \mid P \setminus L
\end{aligned}
$$

$$\text{act} ::= \tau \mid a(x) \mid {'a(e)}$$

$$\text{args} ::= \varepsilon \mid e \mid e, \text{args}$$

$$f ::= \varepsilon \mid a/a \mid a/a, f$$

$$\text{channels} ::= \varepsilon \mid a \mid a, \text{channels}$$

$$L ::= a \mid \{\} \mid \{\text{channels}\}$$

Program

$$
\begin{aligned}
\pi \ ::= & \quad P \\
\mid & \quad k = P; \pi \\
\mid & \quad k(\text{params}) = P; \pi
\end{aligned}
$$

$$\text{params} ::= \varepsilon \mid x \mid x, \text{params}$$

Pure CCS syntax

Constants

- $k \in K$
- $a \in \mathcal{A}$

Processes

$$
\begin{aligned}
P ::= \quad & 0 \\
\mid \quad & (P) \\
\mid \quad & \mathrm{act}.P \\
\mid \quad & k \\
\mid \quad & P + P \\
\mid \quad & P \mid P \\
\mid \quad & P[f] \mid P \setminus L
\end{aligned}
$$

$$\mathrm{act} ::= \tau \mid a \mid {'a}$$

$$f ::= a/a \mid a/a, f$$

$$\mathrm{channels} ::= a \mid a, \mathrm{channels}$$

$$L ::= a \mid \{\mathrm{channels}\}$$

Program

$$
\begin{aligned}
\pi \ ::= &\quad P \\
&\mid k = P; \pi
\end{aligned}
$$

Compiler components

Architecture

- vCCS parser
- CCS interface
- vCCS to CCS encoder
- vCCS encoding utilities

vCCS parser

Classic components to parse a language

- Abstract syntax tree (AST)
- Parser
- Lexer

CCS interface

Basic CCS support for the encoder results

- AST
- Pretty printer

vCCS to CCS encoder

Implementation of $[\![\quad]\!]$

Defined by structural induction on vCCS processes

vCCS to CCS encoder

Implementation of $[\![\ \ ]\!]$

Defined by structural induction on vCCS processes

Considerations:

- Programs are the root nodes of my syntax

vCCS to CCS encoder

Implementation of $[\![\quad]\!]$

Defined by structural induction on vCCS processes

Considerations:

- Programs are the root nodes of my syntax
- More syntax cases than just π and P to consider in practice

vCCS encoding utilities

Functions that solve CCS encoding sub-tasks

vCCS encoding utilities

Functions that solve CCS encoding sub-tasks

- Booleans/expressions evaluation

  $$\text{'out}((1+3)/2) \quad \longrightarrow \quad \text{'out}(2)$$

vCCS encoding utilities

Functions that solve CCS encoding sub-tasks

- Booleans/expressions evaluation

  $$\text{'out}((1+3)/2) \quad \longrightarrow \quad \text{'out}(2)$$

- Variable substitution

  $$k(x) \quad \longrightarrow \quad k(1)$$

vCCS encoding utilities

Functions that solve CCS encoding sub-tasks

- Booleans/expressions evaluation

  $$\text{'out}((1+3)/2) \quad \longrightarrow \quad \text{'out}(2)$$

- Variable substitution

  $$k(x) \quad \longrightarrow \quad k(1)$$

- Variable expansion

  $$\mathrm{in}(x).k(x) \quad \longrightarrow \quad \mathrm{in}_1(x).k(1) + \mathrm{in}_2(x).k(2) + \dots$$

Technological stack

Programming language

Choice: OCaml

Widely used ML dialect

Programming language

Choice: OCaml

Widely used ML dialect

Pros

- Pattern matching!

Programming language

Choice: OCaml

Widely used ML dialect

Pros

- Pattern matching!
- I know ML

Programming language

Choice: OCaml

Widely used ML dialect

Pros

- Pattern matching!
- I know ML
- Popular parser generators available

Parser generator

OCaml versions of lex and yacc available

- ocamllex, the standard
- Menhir, more recent twist on ocamlyacc

Package manager

- Language setup with `opam init`
- Project build and installation with `opam install .`
- Powerful build system with Dune

Implementation

Abstract syntax tree

Store syntax elements

```
type act =
   | Tau
   | Input of string * string
   | Output of string * expr
type proc =
   | Nil
   | Act of act * proc
   | Const of string * expr list
   | If of boolean * proc
   | Sum of proc * proc
   | Paral of proc * proc
   | Red of proc * (string * string) list
```

Parser

Tokens

↓

AST nodes

```
%token <string> ID
%token TAU
%token POINT
%token IF THEN
%token PIPE
act:
  | TAU { Tau }
  | a = ID LPAREN x = ID RPAREN { Input (a, x) }
proc:
  | a = act POINT p = proc { Act (a, p) }
  | IF b = boolean THEN p = proc { If (b, p) }
  | p1 = proc PIPE p2 = proc { Paral (p1, p2) }
```

Lexer

Characters sequences

↓

Parser tokens

```
let blank    = [' ' '\t' '\n']+
let letter   = ['a'-'z' 'A'-'Z']
let tau      = "τ" | "tau"
rule read = parse
  | blank+  { read lexbuf }
  | '='     { EQ }
  | tau     { TAU }
  | '.'     { POINT }
  | "if"    { IF }
  | "then"  { THEN }
  | '|'     { PIPE }
  | id      { ID (Lexing.lexeme lexbuf) }
```

Encoder

Trivial cases

$$\llbracket\ \ \rrbracket_\pi : \mathrm{Prog}_{\mathrm{vCCS}} \to \mathrm{Prog} \qquad \llbracket\ \ \rrbracket : \mathrm{Proc}_{\mathrm{vCCS}} \to \mathrm{Proc}$$

$$\llbracket\, P \,\rrbracket_\pi = \llbracket\, P \,\rrbracket$$

$$\llbracket\, k = P; \pi \,\rrbracket_\pi = (k = \llbracket\, P \,\rrbracket; \mathrm{encode}(\pi))$$

Trivial cases

$[\![\ \ ]\!]_\pi : \mathrm{Prog}_{\mathrm{vCCS}} \to \mathrm{Prog}$	$[\![\ \ ]\!] : \mathrm{Proc}_{\mathrm{vCCS}} \to \mathrm{Proc}$
$[\![\, P \,]\!]_\pi = [\![\, P \,]\!]$	$[\![\, 0 \,]\!] = 0$
$[\![\, k = P; \pi \,]\!]_\pi = (k = [\![\, P \,]\!]; \mathrm{encode}(\pi))$	$[\![\, \tau.P \,]\!] = \tau.[\![\, P \,]\!]$
	$[\![\, k \,]\!] = k$
	$[\![\, P + Q \,]\!] = [\![\, P \,]\!] + [\![\, Q \,]\!]$
	$[\![\, P \mid Q \,]\!] = [\![\, P \,]\!] \mid [\![\, Q \,]\!]$

Evaluation – expressions

$$\text{eval}_e : \text{expr} \;\rightarrow\; \mathbb{N}$$

Evaluation – expressions

$$\mathrm{eval}_e : \mathrm{expr} \;\rightarrow\; \mathbb{N}$$

$$\mathrm{eval}_e(n) = n$$

$$\mathrm{eval}_e(e_1 \ \mathrm{op}\ e_2) = \mathrm{eval}_e(e_1)\ \mathrm{op}\ \mathrm{eval}_e(e_2)$$

$$\boxed{\mathrm{op} ::= \ + \mid - \mid * \mid /}$$

Evaluation – expressions

$$\text{eval}_e : \text{expr} \ \rightarrow \ \mathbb{N}$$

$$\text{eval}_e(n) = n$$

$$\text{eval}_e(e_1 \ \text{op} \ e_2) = \text{eval}_e(e_1) \ \text{op} \ \text{eval}_e(e_2)$$

$$\text{eval}_e(x) = ?$$

$$\boxed{\text{op} ::= \ + \ | \ - \ | \ * \ | \ /}$$

Evaluation – expressions

$$\text{eval}_e : \text{expr} \to \mathbb{N}$$

$$\text{eval}_e(n) = n$$

$$\text{eval}_e(e_1 \text{ op } e_2) = \text{eval}_e(e_1) \text{ op } \text{eval}_e(e_2)$$

$$\text{eval}_e(x) = ? \to$$ `error: unbound variable x`

$$\boxed{\text{op} ::= + \mid - \mid * \mid /}$$

Evaluation – booleans

$$\mathrm{eval}_b : \mathrm{boolean} \rightarrow \{\mathrm{true}, \mathrm{false}\}$$

Evaluation – booleans

$$\mathrm{eval}_b : \mathrm{boolean} \rightarrow \{\mathrm{true}, \mathrm{false}\}$$

$$\mathrm{eval}_b(\mathrm{true}) = \mathrm{true} \quad \mathrm{eval}_b(\mathrm{false}) = \mathrm{false}$$

$$\mathrm{eval}_b(\mathrm{not}\ b) = \neg b$$

$$\mathrm{eval}_b(b_1\ \mathrm{or}\ b_2) = b_1 \vee b_2 \qquad \mathrm{eval}_b(b_1\ \mathrm{and}\ b_2) = b_1 \wedge b_2$$

$$\mathrm{eval}_b(e_1\ \mathrm{op}\ e_2) = \mathrm{eval}_e(e_1)\ \mathrm{op}\ \mathrm{eval}_e(e_2)$$

$$\boxed{\mathrm{op} ::= \ = \mid \neq \mid < \mid > \mid \leqslant \mid \geqslant}$$

Evaluation

$$[\![\ 'a(e).P\ ]\!] = 'a_n.[\![\ P\ ]\!] \qquad n = \mathrm{eval}_e(e)$$

$$[\![\ k(e_1, \ldots, e_h)\ ]\!] = k_{n_1,\ldots,n_h} \qquad n_i = \mathrm{eval}_e(e_i)$$

$$[\![\ \text{if } b \text{ then } P\ ]\!] = \begin{cases} [\![\ P\ ]\!] & \mathrm{eval}_b(b) = \text{true} \\ 0 & \mathrm{eval}_b(b) = \text{false} \end{cases}$$

Let’s start a small digression…

Expansion

Missing cases

$$\llbracket\, k(x_1, \ldots, x_h) = P; \pi \,\rrbracket_\pi = \; ?$$

$$\llbracket\, a(x).P \,\rrbracket = \; ?$$

Missing cases

Variable binders

$$[\![\, k(x_1, \dots, x_h) = P; \pi \,]\!]_\pi = \ ?$$

$$[\![\, a(x).P \,]\!] = \ ?$$

Missing cases

Variable binders

$$[\![\, k(x_1, \ldots, x_h) = P; \pi \,]\!]_\pi = \ ?$$

$$[\![\, a(x).P \,]\!] = \ ?$$

Also channel manipulators

$$[\![\, P \setminus L \,]\!] = \ ?$$

$$[\![\, P[f] \,]\!] = \ ?$$

The problem

Cannot expand for infinite number of values

$$\mathrm{in}(x).k(x) \;\longrightarrow\; \mathrm{in}_0.k_0 + \mathrm{in}_1.k_1 + \mathrm{in}_2.k_2 + \mathrm{in}_3.k_3 + \ldots$$

The problem

Cannot expand for infinite number of values

$$\text{in}(x).k(x) \;\longrightarrow\; \text{in}_0.k_0 + \text{in}_1.k_1 + \text{in}_2.k_2 + \text{in}_3.k_3 + \ldots$$

$\Longrightarrow$ Finite value domain needed

$$\text{in}(x).k(x) \;\xrightarrow{D=\{0,1,2\}}\; \text{in}_0.k_0 + \text{in}_1.k_1 + \text{in}_2.k_2$$

Variable substitution

Let's introduce variable substitution first:

$P\{^n/_x\} \quad \longrightarrow \quad$ replace all (free) occurrences of x with value n

Variable substitution – booleans/expressions

$$\{/\}_b : \text{boolean} \to \text{Var} \to \mathbb{N} \to \text{boolean} \qquad \{/\}_e : \text{expr} \to \text{Var} \to \mathbb{N} \to \text{expr}$$

Variable substitution – booleans/expressions

$$\{/\}_b : \text{boolean} \to \text{Var} \to \mathbb{N} \to \text{boolean} \qquad \{/\}_e : \text{expr} \to \text{Var} \to \mathbb{N} \to \text{expr}$$

$$(\text{not } b)\{^n/_x\}_b = \text{not } b\{^n/_x\}_b$$

$$(b_1 \text{ and } b_2)\{^n/_x\}_b = b_1\{^n/_x\}_b \text{ and } b_2\{^n/_x\}_b$$

$$(b_1 \text{ or } b_2)\{^n/_x\}_b = b_1\{^n/_x\}_b \text{ or } b_2\{^n/_x\}_b$$

$$(e_1 \text{ op } e_2)\{^n/_x\}_b = e_1\{^n/_x\}_e \text{ op } e_2\{^n/_x\}_e$$

$$b\{^n/_x\}_b = b$$

Variable substitution – booleans/expressions

$$\{/\}_b : \text{boolean} \to \text{Var} \to \mathbb{N} \to \text{boolean} \qquad \{/\}_e : \text{expr} \to \text{Var} \to \mathbb{N} \to \text{expr}$$

$$(\text{not } b)\{^n/_x\}_b = \text{not } b\{^n/_x\}_b$$

$$(b_1 \text{ and } b_2)\{^n/_x\}_b = b_1\{^n/_x\}_b \text{ and } b_2\{^n/_x\}_b$$

$$(b_1 \text{ or } b_2)\{^n/_x\}_b = b_1\{^n/_x\}_b \text{ or } b_2\{^n/_x\}_b$$

$$(e_1 \text{ op } e_2)\{^n/_x\}_b = e_1\{^n/_x\}_e \text{ op } e_2\{^n/_x\}_e$$

$$b\{^n/_x\}_b = b$$

$$n\{^n/_x\}_e = n$$

$$y\{^n/_x\}_e = \begin{cases} n & \text{if } x = y \\ y & \text{otherwise} \end{cases}$$

$$(e_1 \text{ op } e_2)\{^n/_x\}_e = e_1\{^n/_x\}_e \text{ op } e_1\{^n/_x\}_e$$

Variable substitution – processes

$$\{/\} : \mathrm{Proc}_{\mathrm{vCCS}} \to \mathrm{Var} \to \mathbb{N} \to \mathrm{Proc}_{\mathrm{vCCS}}$$

Variable substitution – processes

$$\{/\} : \mathrm{Proc}_{\mathrm{vCCS}} \to \mathrm{Var} \to \mathbb{N} \to \mathrm{Proc}_{\mathrm{vCCS}}$$

$$0\{^n/_x\} = 0$$

$$(P + Q)\{^n/_x\} = P\{^n/_x\} + Q\{^n/_x\}$$

$$(P \mid Q)\{^n/_x\} = P\{^n/_x\} \mid Q\{^n/_x\}$$

$$(P[f])\{^n/_x\} = P\{^n/_x\}[f]$$

$$(P \setminus L)\{^n/_x\} = P\{^n/_x\} \setminus L$$

Variable substitution – processes

$$\{/\} : \mathrm{Proc}_{\mathrm{vCCS}} \to \mathrm{Var} \to \mathbb{N} \to \mathrm{Proc}_{\mathrm{vCCS}}$$

$$0\{{}^{n}/_{x}\} = 0$$

$$(P + Q)\{{}^{n}/_{x}\} = P\{{}^{n}/_{x}\} + Q\{{}^{n}/_{x}\}$$

$$(P \mid Q)\{{}^{n}/_{x}\} = P\{{}^{n}/_{x}\} \mid Q\{{}^{n}/_{x}\}$$

$$(P[f])\{{}^{n}/_{x}\} = P\{{}^{n}/_{x}\}[f]$$

$$(P \setminus L)\{{}^{n}/_{x}\} = P\{{}^{n}/_{x}\} \setminus L$$

$$(a(y).P)\{{}^{n}/_{x}\} = \begin{cases} a(y).P\{{}^{n}/_{x}\} & \text{if } y \neq x \\ a(y).P & \text{otherwise} \end{cases}$$

$$('a(e).P)\{{}^{n}/_{x}\} = 'a\left(e\{{}^{n}/_{x}\}_{e}\right).P$$

$$k(e_1, ..., e_n)\{{}^{n}/_{x}\} = k\left(e_1\{{}^{n}/_{x}\}_{e}, ..., e_n\{{}^{n}/_{x}\}_{e}\right)$$

$$(\text{if } b \text{ then } P)\{{}^{n}/_{x}\} = \text{if } b\{{}^{n}/_{x}\}_{b} \text{ then } P\{{}^{n}/_{x}\}$$

Constant parameter

Expand first parameter: $k(x_1, x_2) \xrightarrow{D=\{0,1\}} k_0(x_2); k_1(x_2)$

Constant parameter

Expand first parameter: $k(x_1, x_2) \xrightarrow{D=\{0,1\}} k_0(x_2); k_1(x_2)$

$$^{D}\langle \quad \rangle_k : 2^{\mathbb{N}} \to \mathrm{Prog}_{\mathrm{vCCS}} \to \mathrm{Prog}_{\mathrm{vCCS}}$$

Constant parameter

Expand first parameter: $k(x_1, x_2) \xrightarrow{D=\{0,1\}} k_0(x_2); k_1(x_2)$

$${}^{D}\langle\ \ \rangle_k : 2^{\mathbb{N}} \to \mathrm{Prog}_{\mathrm{vCCS}} \to \mathrm{Prog}_{\mathrm{vCCS}}$$

$${}^{\emptyset}\langle\, k(x_1, \dots, x_h) = P; \pi \,\rangle_k = \pi \qquad {}^{D}\langle\, \pi \,\rangle_k = \pi$$

$${}^{\{n\} \cup S}\langle\, k(x_1, x_2, \dots, x_h) = P; \pi \,\rangle_k =$$

$$k_n(x_2, \dots, x_h) = P\left\{{}^{n}/_{x_1}\right\}; {}^{S}\langle\, k(x_1, x_2, \dots, x_h) = P; \pi \,\rangle_k$$

Input variable

$${}^{D}\langle\ \ \rangle_a : 2^{\mathbb{N}} \to \mathrm{Proc}_{\mathrm{vCCS}} \to \mathrm{Proc}_{\mathrm{vCCS}}$$

Input variable

$$^{D}\langle\ \ \rangle_a : 2^{\mathbb{N}} \rightarrow \mathrm{Proc}_{\mathrm{vCCS}} \rightarrow \mathrm{Proc}_{\mathrm{vCCS}}$$

$$^{\varnothing}\langle\, a(x).P \,\rangle_a = 0$$

$$^{\{n\}}\langle\, a(x).P \,\rangle_a = a_n(x).P\{^n/_x\}$$

$$^{\{n\} \cup S}\langle\, a(x).P \,\rangle_a = a_n(x).P\{^n/_x\} + {}^{S}\langle\, a(x).P \,\rangle_a$$

$$^{D}\langle\, P \,\rangle_a = P$$

Redirection function

$$^{D}\langle \quad \rangle_f : 2^{\mathbb{N}} \to \mathrm{Proc}_{\mathrm{vCCS}} \to \mathrm{Proc}_{\mathrm{vCCS}}$$

Redirection function

$$^{D}\langle\ \ \rangle_f : 2^{\mathbb{N}} \to \mathrm{Proc}_{\mathrm{vCCS}} \to \mathrm{Proc}_{\mathrm{vCCS}}$$

$$^{\varnothing}\langle\, P[f]\, \rangle_f = P$$

$$^{\{n_1, n_2, \dots, n_h\}}\langle\, P[f]\, \rangle_f = P\left[f_{n_1}, f_{n_2}, \dots, f_{n_h}\right]$$

$$^{D}\langle\, P\, \rangle_f = P$$

Where $f = a/b, c/d, \dots \implies f_n = a_n/b_n, c_n/d_n, \dots$

Restricted channels

$$^{D}\langle\ \ \rangle_L : 2^{\mathbb{N}} \to \mathrm{Proc}_{\mathrm{vCCS}} \to \mathrm{Proc}_{\mathrm{vCCS}}$$

Restricted channels

$$^{D}\langle\ \ \rangle_L : 2^{\mathbb{N}} \to \text{Proc}_{\text{vCCS}} \to \text{Proc}_{\text{vCCS}}$$

$$^{\emptyset}\langle\, P \setminus L \,\rangle_L = P$$

$$^{\{n_1, n_2, \ldots, n_h\}}\langle\, P \setminus L \,\rangle_L = P \setminus \left(L_{n_1} \cup L_{n_2} \cup \ldots \cup L_{n_h}\right)$$

$$^{D}\langle\, P \,\rangle_L = P$$

Where $L = a, b, \ldots \implies L_n = a_n, b_n \ldots$

Now, back to the encoder

Expansion – constants

Given a finite domain $D \subseteq \mathbb{N}$

$$[\![\ k(x_1, ..., x_h) = P; \pi\ ]\!]_\pi = [\![\ ^D\langle\ k(x_1, ..., x_h) = P\ \rangle_k ; \pi\ ]\!]_\pi$$

Expansion – constants

Given a finite domain $D \subseteq \mathbb{N}$

$$\llbracket\, k(x_1, \ldots, x_h) = P; \pi \,\rrbracket_\pi = \left\llbracket\, {}^{D}\langle\, k(x_1, \ldots, x_h) = P \,\rangle_k ; \pi \,\right\rrbracket_\pi$$

$$\longrightarrow \left\llbracket\, k_{n_1}(x_2, \ldots, x_h) = P; k_{n_2}(x_2, \ldots, x_h) = P; \ldots; \pi \,\right\rrbracket_\pi$$

$$\longrightarrow \left\llbracket\, k_{n_1,m_1}(x_3, \ldots, x_h) = P; k_{n_1,m_2}(x_3, \ldots, x_h) = P; \ldots; \pi \,\right\rrbracket_\pi$$

$\ldots$

$$\longrightarrow \left\llbracket\, k_{n_1,m_1,\ldots} = P; \ldots; \pi \,\right\rrbracket_\pi$$

Expansion – input

Given a finite domain $D \subseteq \mathbb{N}$

$$[\![\, a(x).P \,]\!] = \left[\!\!\left[\, {}^{D}\langle\, a(x).P \,\rangle_a \,\right]\!\!\right]$$

$$[\![\, a_n(x).P \,]\!] = a_n.[\![\, P \,]\!]$$

Expansion – redirection

Given a finite domain $D \subseteq \mathbb{N}$

$$[\![P[f]]\!] = \left[\!\!\left[{}^{D}\langle P[f] \rangle_f \right]\!\!\right]$$

$$\left[\!\!\left[P\left[f_{n_1}, ..., f_{n_h}\right] \right]\!\!\right] = [\![P]\!]\left[f_{n_1}, ..., f_{n_h}\right]$$

Expansion – restriction

Given a finite domain $D \subseteq \mathbb{N}$

$$[\![\, P \setminus L \,]\!] = \left[\!\left[\, {}^{D}\langle \, P \setminus L \, \rangle_L \, \right]\!\right]$$

$$\left[\!\left[\, P \setminus \left(L_{n_1} \cup L_{n_2} \cup \ldots \cup L_{n_h}\right) \, \right]\!\right] = [\![\, P \,]\!] \setminus \left(L_{n_1} \cup L_{n_2} \cup \ldots \cup L_{n_h}\right)$$

Bounded evaluation

Given a finite domain $D \subseteq \mathbb{N}$

$$\frac{}{{}^{D}\mathrm{eval}_e \, : \, 2^{\mathbb{N}} \, \to \, \mathrm{expr} \, \to \, \mathbb{N}}$$

Bounded evaluation

Given a finite domain $D \subseteq \mathbb{N}$

$$\frac{{}^{D}\mathrm{eval}_e : 2^{\mathbb{N}} \rightarrow \mathrm{expr} \rightarrow \mathbb{N}}{\begin{aligned} {}^{D}\mathrm{eval}_e(e) &= \mathrm{eval}_e(e), \quad \mathrm{eval}_e(e) \in D \\ {}^{D}\mathrm{eval}_e(e) &= \mathrm{eval}_e(e), \quad \mathrm{eval}_e(e) \notin D \end{aligned}}$$

Bounded evaluation

Given a finite domain $D \subseteq \mathbb{N}$

$$\frac{{}^{D}\mathrm{eval}_e : 2^{\mathbb{N}} \to \mathrm{expr} \to \mathbb{N}}{\begin{array}{l} {}^{D}\mathrm{eval}_e(e) = \mathrm{eval}_e(e), \quad \mathrm{eval}_e(e) \in D \\ {}^{D}\mathrm{eval}_e(e) = \mathrm{eval}_e(e), \quad \mathrm{eval}_e(e) \notin D \end{array}}$$

$\longrightarrow$ `error: out of bounds value evaluated`

Demo time!