Relating Dataflow and Lambda-calculus

Zena M. Ariola
University of Oregon

18 May 2007

CSG in 80's

World leaders in Dataflow

Not quite world leaders but extremely interested in TRS's and λ-calculus

Token Pushing Semantics

Operational semantics of ID

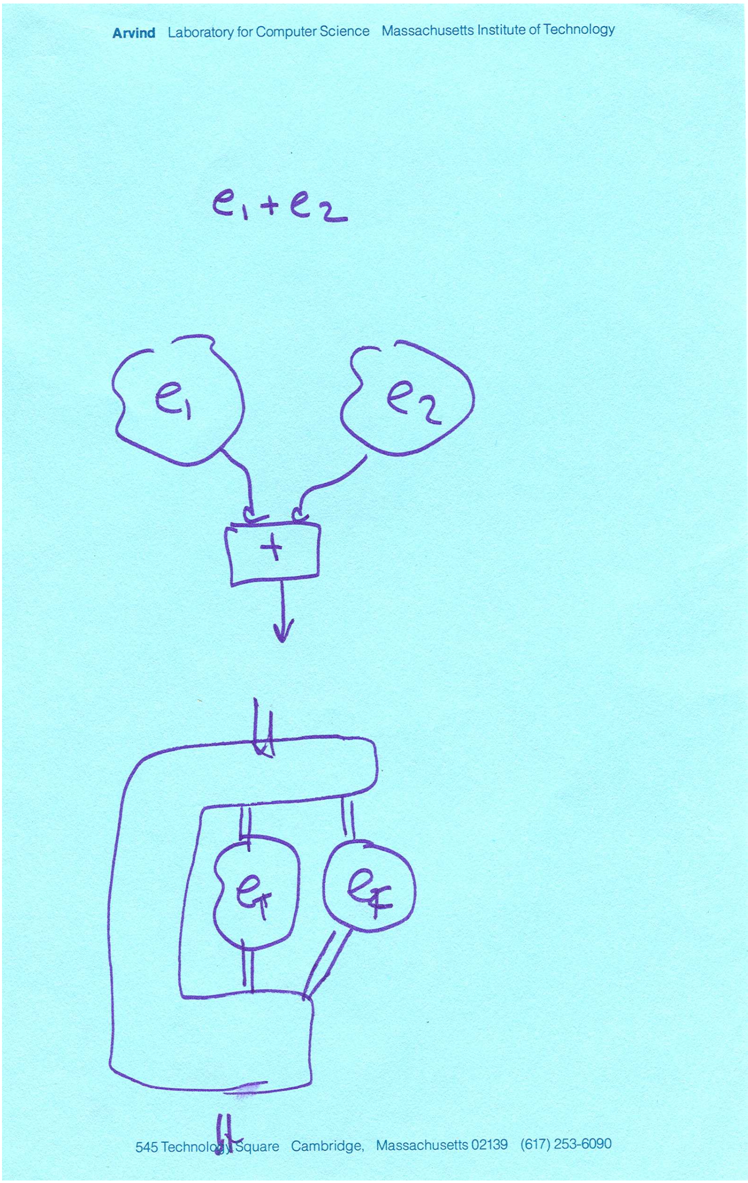
Compilation of Id

Optimizations of Id

were expressed in terms of

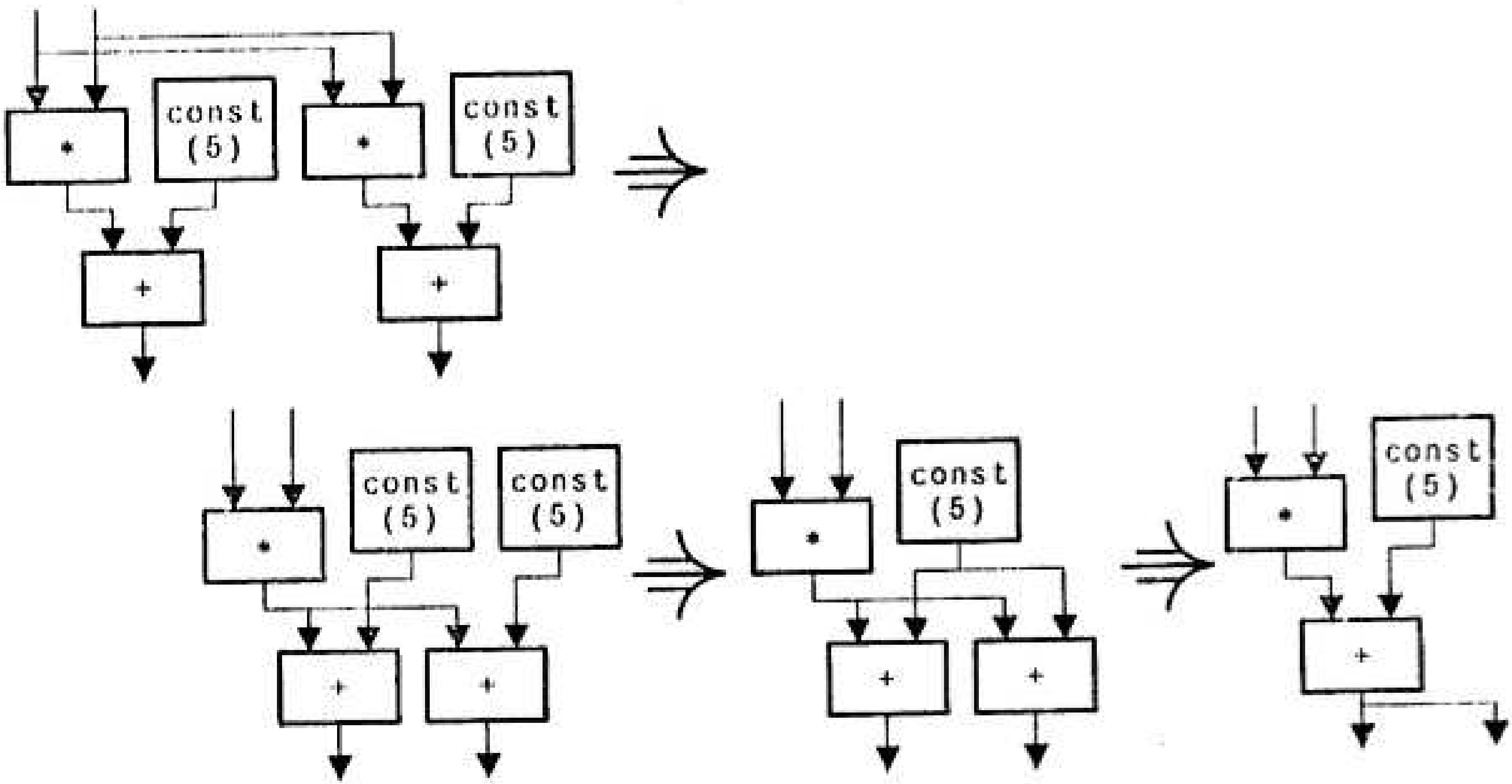
Dataflow Graphs

Dataflow Graph



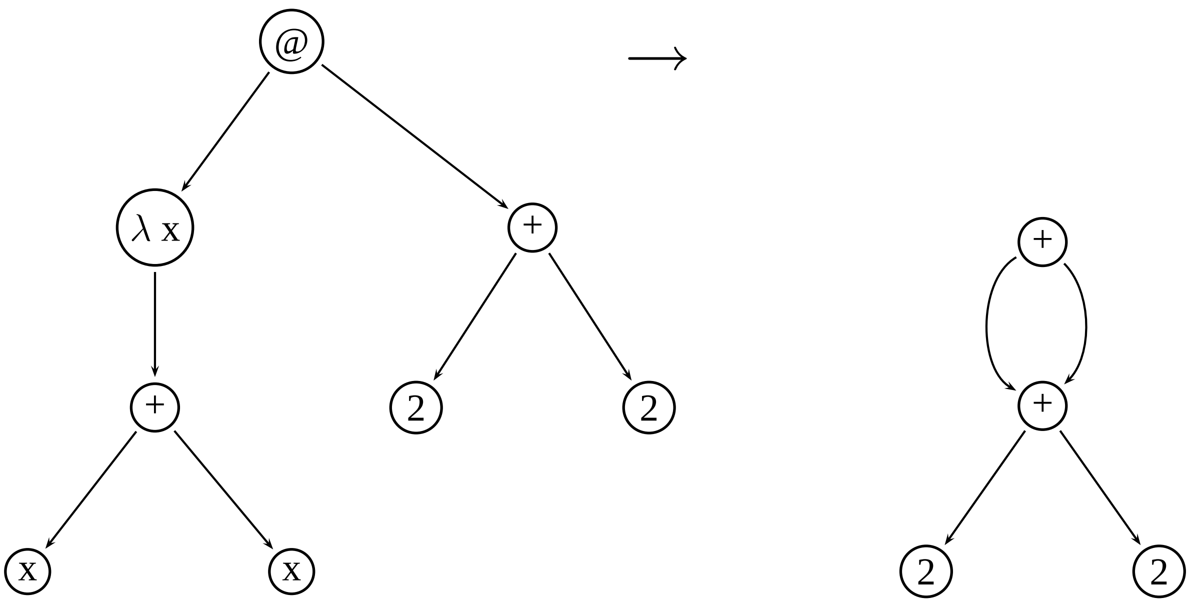
Transformations On Dataflow Graphs

Common Subexpression Elimination



Arvind, Pingali, Kathail: Graph Reduction

$$(\lambda x.x + x)(2 + 2) \rightarrow (2 + 2) + (2 + 2)$$



Sharing of Maximal Free Expressions

Wadsworth: do not repeat some obvious computation

$$fun\ f\ x = (2+2) + x$$

should you recompute 2+2 every time you apply f?

Extract Maximal Free Expressions at compile time (Arvind, Keshav, Pingali HLCA 1984)

Optimality

How much do we have to share to be optimal? Lévy theory of optimality

How do you implement that theory?

Vinod why aren't you here?

In 1985 Corrado Böhm invites Arvind to the First International Workshop on Reduction Machines in Ustica ....

What is Henk Barendregt trying to explain?

Again.... Arvind and Henk Barendregt

Discussions on TRS's with Jan Willem Klop were obviously enjoyable

Don’t worry - Gita was there!

Don’t Panic!

After Ustica

Arvind came back speaking λ-calculus and TRS's.....

Why not applying these ideas to Id?

Can we use TRS's and λ-calculus for Id?

I-structure: Logic Variables

Arvind Laboratory for Computer Science Massachusetts Institute of Technology

6.

```
def set A v = {A[1] = v}

{A,B = {X = array (1,10)
        Y = array (1,10)
        in
        X,Y};
   set A 10;
   set B 20;
in A[1] + B[1]}   =>   ?
     vs
{A,B = {X = array (1,10)
        in
        X,X};
   set A 10;
   set B 20;
in A[1] + B[1]}   =>   ?
```

545 Technology Square Cambridge, Massachusetts 02139 (617) 253-6090

We need to take sharing into account

We introduced an INNOVATIVE system called

Contextual Rewriting System (CRS)

Klop, Lévy $\Longrightarrow$ Graph Rewriting System

Graph Rewriting

Categorical approach: single or double pushouts – Too abstract

Implementation approach: allocations of nodes and redirections – Too low level

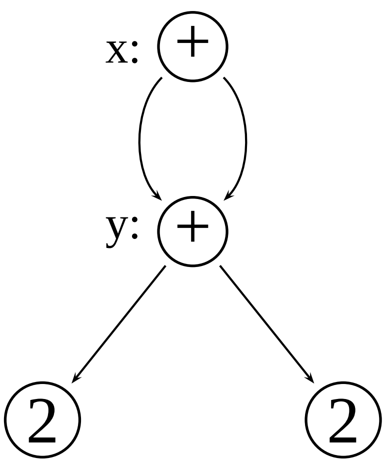
Equational Graph Rewriting System

TRS's + Letrec

λ-calculus + Letrec

Equational GRS

Give a name to each node of the graph and write down the interconnections via a system of recursive equations

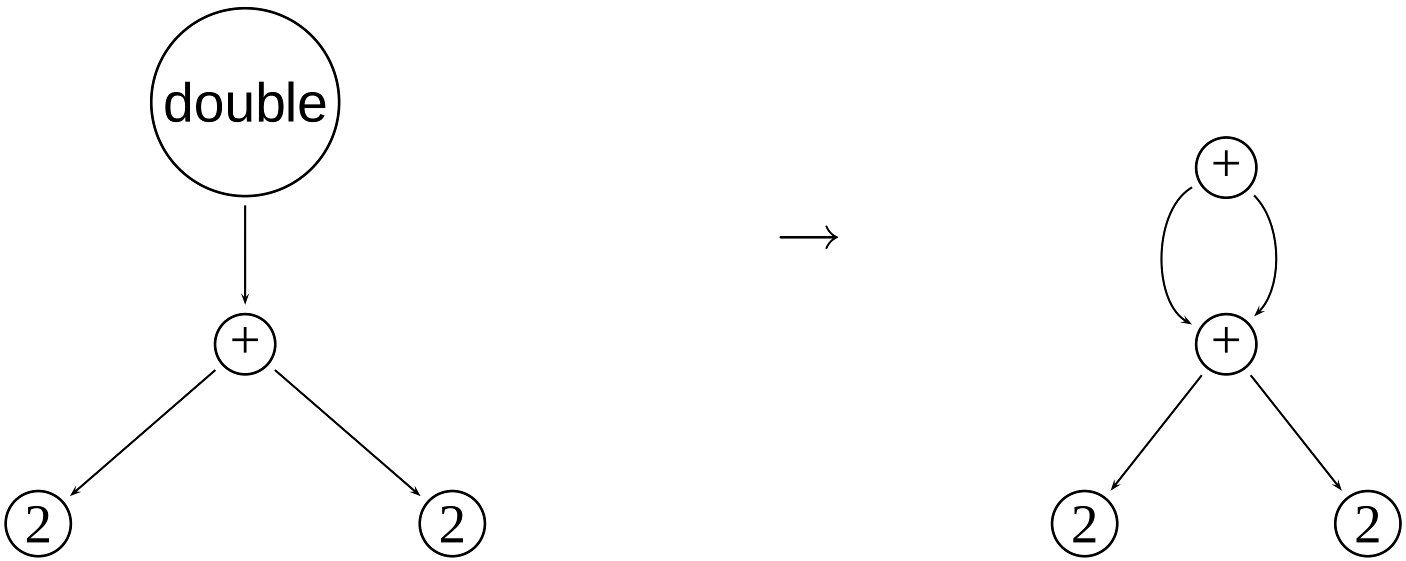


$$\langle x \mid x = y + y, y = 2 + 2\rangle$$
$$\langle x \mid x = y + y, y = z + w, z = 2, w = 2\rangle$$
$$\langle y + y \mid y = 2 + 2\rangle$$

Graph Reduction

$$\mathsf{double}\ x \longrightarrow x + x$$

$$\langle z \mid z = \mathsf{double}\ y, y = 2 + 2\rangle \longrightarrow \langle z \mid z = y + y, y = 2 + 2\rangle$$

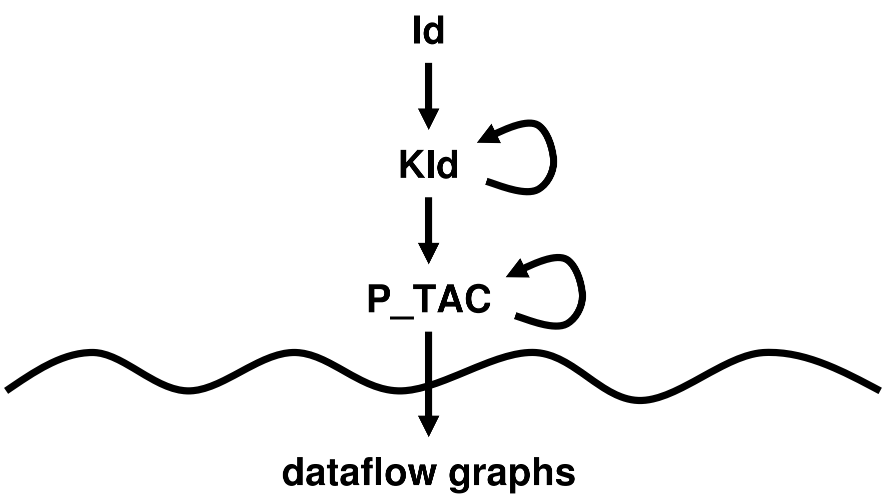


Two Intermediate Languages

P-TAC - Parallel Three Address Code - TRS + Letrec (FPCA'89)

Kid - Kernel Id - λ-calculus + Letrec (PEPM'91)

aaaaaaa



Optimizations as rewrite rules

Algebraic identities

$$
\begin{array}{lll}
\texttt{True} \wedge x \longrightarrow x & & \texttt{False} \wedge x \longrightarrow \texttt{False} \\
\texttt{False} \vee x \longrightarrow x & & \texttt{True} \vee x \longrightarrow \texttt{True} \\
x + 0 \longrightarrow x & & x = x \longrightarrow \texttt{True} \\
x * 1 \longrightarrow x & &
\end{array}
$$

$$
\begin{array}{l}
y < x \longrightarrow \texttt{True} \quad \text{if } x = y + m \\
y = x \longrightarrow \texttt{False} \quad \text{if } x = y + m
\end{array}
$$

Correctness of optimizations

We based the notion of correctness on the syntactic structure of terms: Optimizations are correct if they preserve the answer of a program (RTA'93, TCS '95)

$$
\begin{array}{llll}
\texttt{True} \wedge x & \to x & \texttt{False} \wedge x & \to \texttt{False} \\
\texttt{False} \vee x & \to x & \texttt{True} \vee x & \to \texttt{True} \\
x + 0 & \to x & x = x & \to \texttt{True} \\
x * 1 & \to x & y < x & \to \texttt{True} \quad \text{if } x = y + m \\
 & & y = x & \to \texttt{False} \ \text{if } x = y + m
\end{array}
$$

$$
\begin{array}{ll}
\langle z \mid x = \Omega, y = \texttt{True} \wedge x, z = \texttt{if } y \texttt{ then } 5 \texttt{ else } 7\rangle & \to \\
\langle z \mid x = \Omega, y = \texttt{True}, z = \texttt{if } y \texttt{ then } 5 \texttt{ else } 7\rangle & \twoheadrightarrow \\
\langle z \mid x = \Omega, y = \texttt{True}, z = 5\rangle & \twoheadrightarrow \\
5 &
\end{array}
$$

Confluence of Optimizations

$$\begin{array}{lll} y < x & \rightarrow \texttt{True} & x = y + m \\ y < x & \rightarrow \texttt{False} & y = x + m \end{array}$$

$$\begin{array}{l} \langle z \mid x = y + 3, y = x + 2, z = x < y \rangle \longrightarrow \langle z \mid x = y + 3, y = x + 2, z = \texttt{False} \rangle \\ \quad\quad \downarrow \\ \langle z \mid x = y + 3, y = x + 2, z = \texttt{True} \rangle \end{array}$$

Optimizations and Termination

Can lifting free expressions impact termination?

$$\langle a\ 1 \mid a = \lambda y.a\ 0\rangle \longrightarrow \langle(\lambda y.a\ 0)\ 1 \mid a = \lambda y.a\ 0\rangle \longrightarrow \langle a\ 0 \mid a = \lambda y.a\ 0\rangle \longrightarrow \cdots$$

$$\downarrow \text{lifting}$$

$$\langle a\ 1 \mid a = \lambda y.b, b = a\ 0\rangle$$

$$\downarrow \text{inlining}$$

$$\langle a\ 1 \mid a = \lambda y.b, b = (\lambda y.b)\ 0\rangle$$

$$\downarrow$$

$$\langle a\ 1 \mid a = \lambda y.b, b = b\rangle$$

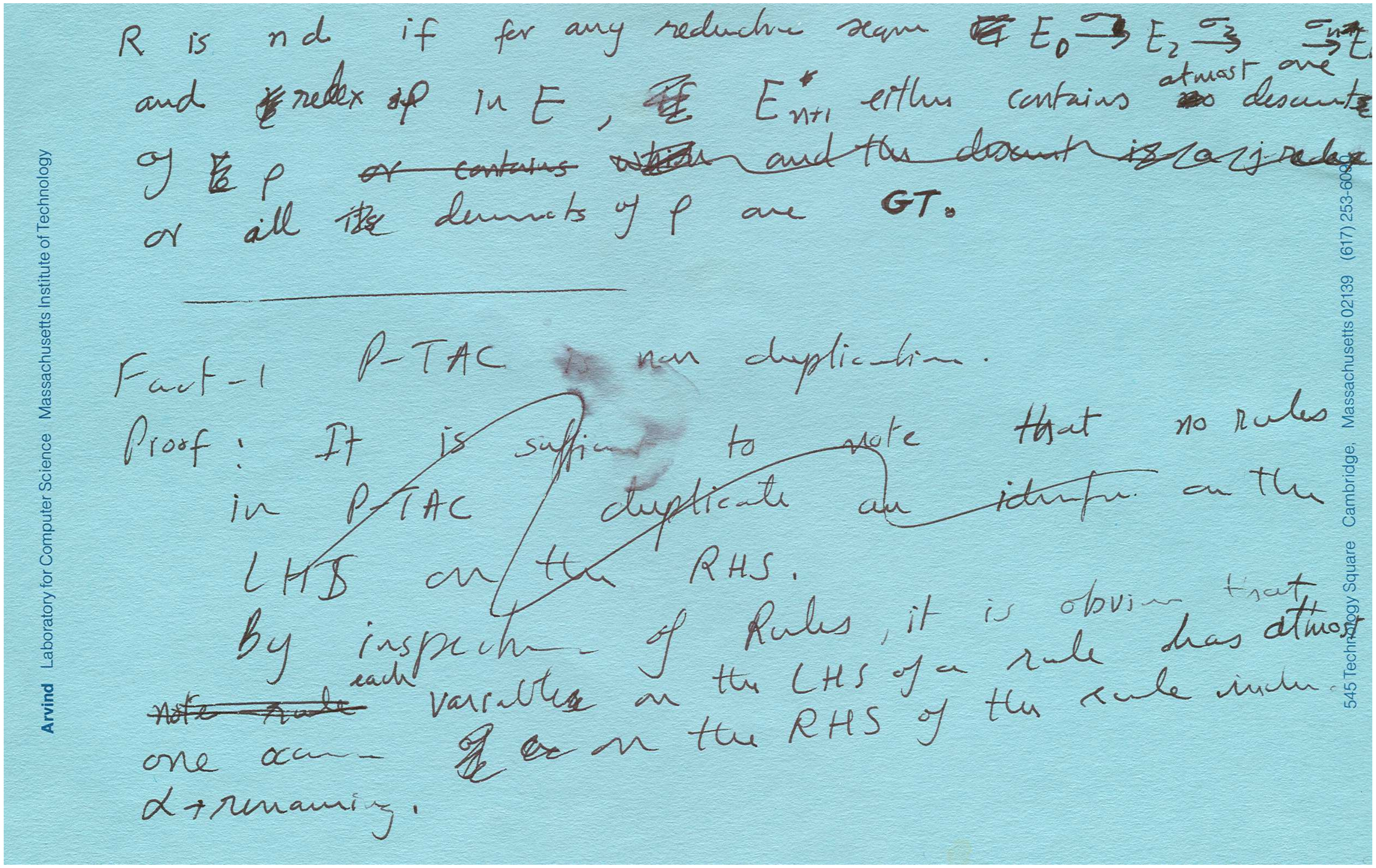
$$\downarrow$$

$$\langle a\ 1 \mid a = \lambda y.b, b = \bullet\rangle$$

$$\downarrow \text{constant folding}$$

$$\langle a\ 1 \mid a = \lambda y.\bullet\rangle \longrightarrow \langle \bullet \mid a = \lambda y.\bullet\rangle \longrightarrow \bullet$$

Properties



P-TAC is confluent

..... even false Properties!!!!

Propagation of $\top$

$$\{\text{m } X = \top;\ S_1; \cdots S_n \text{ In } \vec{Z_m}\ \} \quad \longrightarrow \quad \top$$
$$\{\text{m } \top_s;\ S_1; \cdots S_n \text{ In } \vec{Z_m}\ \} \quad \longrightarrow \quad \top$$

The rules for propagating $\top$ were motivated by a discussion with Vinod Kathail.

Theorem 4.1 *Kid is Confluent upto α-renaming on canonical terms.*

Proof: See [1]. ■

4.2 Printable Values and Answer of a Kid Term

We now define the printable information associated with a term. The grammar for printable values for Kid is given in Figure 3. A precise notion of printable values is essential to develop an interpreter for Kid as well as to discuss the correctness of optimizations (see Sections 4.3 and 5.3, respectively).

Atoms	::=	*Integers* | *Booleans* | Error | "Function" | Ω
List	::=	$\langle$List *PV List*$\rangle$ | Nil
Tuple	::=	$\langle$2_Tuple *PV PV*$\rangle$ | $\langle$3_Tuple *PV PV PV*$\rangle$ | $\cdots$
Array	::=	$\langle$2_Array *Tuple PV PV*$\rangle$ |$\rangle$3_Array *Tuple PV PV PV*$\rangle$ | $\cdots$
PV	::=	*Atoms* | *List* | *Tuple* | *Array* | $\top$

Figure 3: Grammar of Printable Values

The following procedure, $\mathcal{P}$, produces the printable value associated with a term. ρ and σ represent, respectively, the list of bindings that have as RHS either a λ-expression or an allocator, *i.e.* Make_tuple, I_array, Open_cons, and the list of store commands, *i.e.* the I-structure store. The procedure $\mathcal{L}$ is used to lookup the value of a variable or a location in ρ and σ, respectively. Given a program, *i.e.* a closed term, M, the $\mathcal{P}rint$ procedure is invoked as follows:

$$\mathcal{P}rint\ (M) = \mathcal{P}\ (\overline{M},\ \text{Nil},\ \text{Nil}) \quad \text{with } \overline{M} \text{ the canonical form of } M.$$

where $\mathcal{P}$ is:

$\mathcal{P}[\![\{S_s;\ A_s;\ B_s \text{ In } X\}]\!]\ \rho\ \sigma = \mathcal{P}[\![X]\!]\ (A_s : \rho)\ (S_s : \sigma)$

$\mathcal{P}[\![\underline{n}]\!]\ \rho\ \sigma = \underline{n}$

$\mathcal{P}[\![\text{True}]\!]\ \rho\ \sigma = \text{True}$

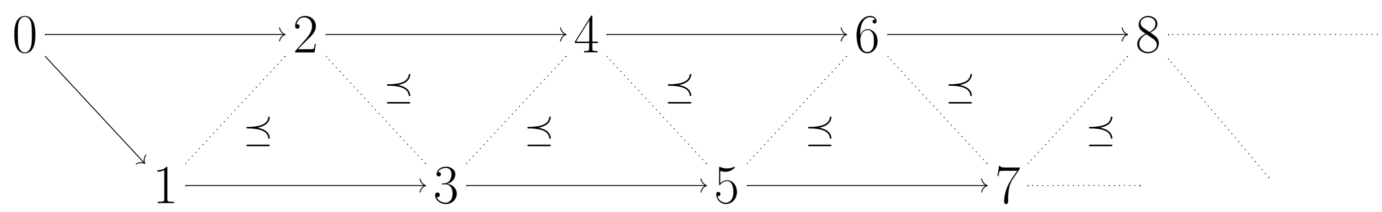
$\mathcal{P}[\![\text{False}]\!]\ \rho\ \sigma = \text{False}$

$\mathcal{P}[\![\top]\!]\ \rho\ \sigma = \top$

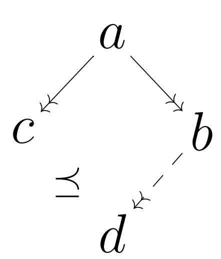
$\mathcal{P}[\![\text{Nil}]\!]\ \rho\ \sigma = \text{Nil}$

Skew Confluence (Blom,Klop)

$$0 \longrightarrow 1 \quad 0 \longrightarrow 2 \quad n \longrightarrow n+2$$



Notice that from 1 we can reach number 3 that is greater than 2 , and from 2 we can reach number 4 that is greater than 3, so on.



- Skew confluence guarantees unicity of infinite normal forms

KID is Skew Confluent

My life after CSG .....

Relate Calculi to Logic - Curry-Howard Isomorphism

Types as Formulae - Programs as Proofs

$$\lambda x.x : A \to A$$

Typing rules $\sim$ inference rules - Reduction rules $\sim$ normalization

Why?

How many years did it take to proves confluence and termination of λ-calculus?

Decidability, confluence and termination results can be reused

Logic and Simply Typed Lambda Calculus

$$\lambda\text{-}calculus: \quad M, N ::= x \mid \lambda x.M \mid M(N)$$

$$\Gamma, x : A \vdash x : A$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \to B}$$

$$\frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash M(N) : B}$$

Execution and Normalization

Detour:

$$\dfrac{\dfrac{y : A, x : A \vdash x : A}{y : A \vdash (\lambda x.x) : A \to A} \qquad y : A \vdash y : A}{y : A \vdash (\lambda x.x)\ y : A}$$

can be simplified to:

$$y : A \vdash y : A$$

The normalization rules correspond to how the program $(\lambda x.x)\ y$ is executed :

$$(\lambda x.x)\ y \mapsto y$$

Control Operators

Scheme callcc - C - Shift-reset - Abort - Catch and Throw

Control operators provide a general mechanism that allows the study of a variety of features:

- Jumps, exceptions, error handling
- Recursion, state, streams, irregular trees
- Coroutines, threads, multiprocessing
- Backtracking, logic programming, debuggers
- Web interactions, The Orbitz problem

Logic and Control Operators

How do you type control?

Do the typing rules correspond to inference rules of a known logic?

How do you reason about them?

We developed an elegant reduction theory for most of the control operators taken in isolation (ICALP’03,ICFP’04,HOSC’07)

The theory came out of the logical investigation

We do not have yet theories for the combined effects

Typing Abort

$$
\begin{array}{lcl}
(1 + (\mathcal{A}\ 5)) & \rightarrow & 5 \\
(\texttt{not}\ (\mathcal{A}\ 5)) & \rightarrow & 5 \\
(\mathcal{A}\ (\mathcal{A}\ 5)) & \rightarrow & 5 \\
(\texttt{"abc"} +\!\!+ (\mathcal{A}\ 5)) & \rightarrow & 5 \\
\\
(1 + (\mathcal{A}\ \texttt{true})) & \rightarrow & \texttt{true}
\end{array}
$$

What is the type of the top-level? $\bot = \forall X.X$

$$
\frac{\Gamma \vdash M : \bot}{\Gamma \vdash \mathcal{A}\ M : A}
$$

$\mathcal{A}$ corresponds to the Ex Falso Quodlibet

Proof by Contradiction

Prove A: assume A is false and try to derive a contradiction

$$\frac{\Gamma, \neg A \vdash \bot}{\Gamma \vdash A}$$

$$\frac{\Gamma, k : \neg A \vdash M : \bot}{\Gamma \vdash \mathcal{C}(\lambda k.M) : A}$$

Thanks Arvind!

Thanks Silvio Berlusconi!

Henk Barendregt: ”Who is paying for this?”

Corrado Böhm: ”It is sponsored by Fininvest”.

Henk Barendregt: ”But who is behind this?”

Corrado Böhm : ”A person called Berlusconi. You will hear from him!”

Special Thanks to Gita!