Initial Algebra Semantics for Cyclic Sharing Structures

Makoto Hamana

Department of Computer Science,
Gunma University, Japan

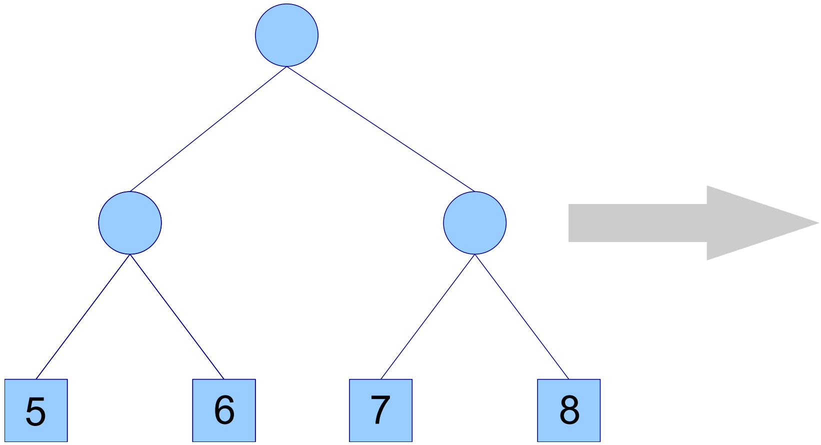
August 2009, GoI Workshop, Kyoto
http://www.cs.gunma-u.ac.jp/~hamana/

This Work

- ▷ How to inductively capture cylces and sharing
- ▷ Intended to apply it to functional programming

Introduction

▷ Terms are a representation of tree structures



bin(bin(5,6),bin(7,8))

(i) Reasoning: structural induction

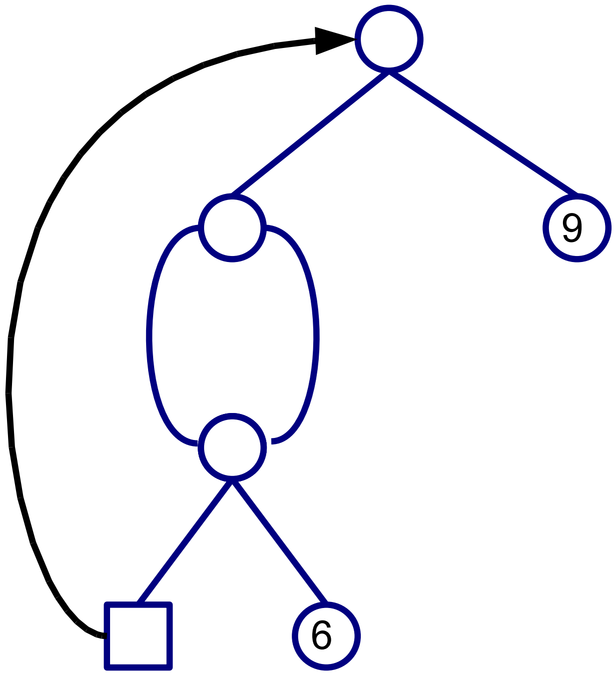
(ii) Functional programming:
pattern matching, structural recursion

(iii) Type: inductive type

(iv) Initial algebra property

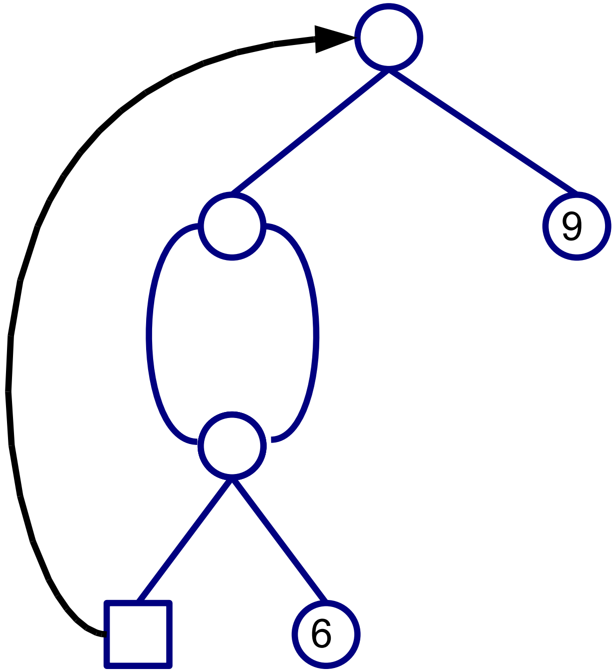
Introduction

- ▷ What about tree-like structures?



- ▷ How can we represent this data in functional programming?
- ▷ Maybe: vertices and edges set, adjacency lists, etc.
- ▷ Give up to use pattern matching, structural induction
- ▷ Not inductive

Introduction



Are really no inductive structures in tree-like structures?

This Work

▷ Gives an initial algebra characterisation of cyclic sharing structures

Aim

▷ To derive the following from ↑ :

[I] A simple term syntax that admits structural induction / recursion

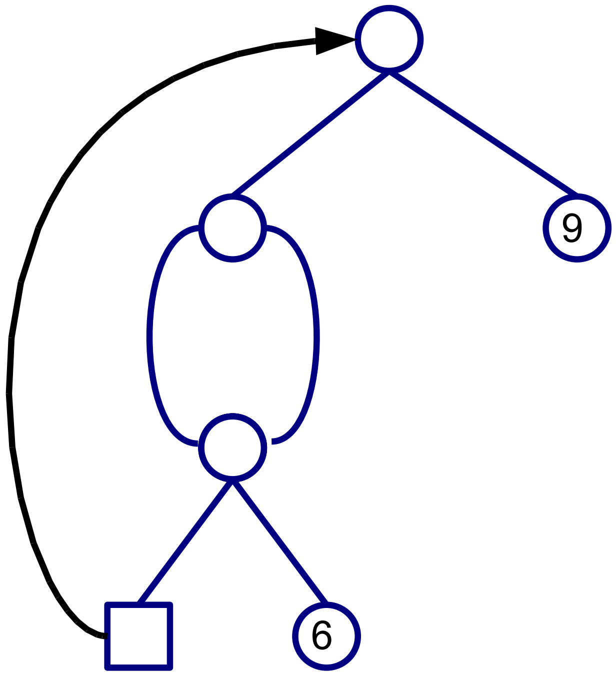
[II] To give an inductive type that represents cyclic sharing structures uniquely in functional languages and proof assistants

Variations of Initial Algebra Semantics

▷ Various computational structures are formulated as initial algebras by varying the base category

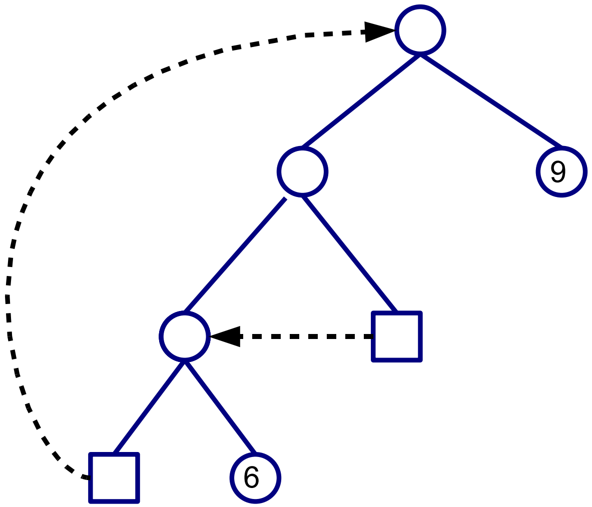
Abstract syntax	**Set**	ADJ	1975
S-sorted abstract syntax	$\mathbf{Set}^{S}$	Robinson	1994
Abstract syntax with binding	$\mathbf{Set}^{\mathbb{F}}$	Fiore,Plotkin,Turi	1999
Recursive path ordring	**LO**	R. Hasegawa	2002
S-sorted 2nd-order abs. syn.	$(\mathbf{Set}^{\mathbb{F}\downarrow S})^{S}$	Fiore	2003
2nd-order rewriting systems	$\mathbf{Pre}^{\mathbb{F}}$	Hamana	2005
Explicit substitutions	$[\mathbf{Set}, \mathbf{Set}]_{f}$	Ghani,Uustalu,Hamana	2006
Cyclic sharing structures	$(\mathbf{Set}^{\mathbb{T}^*})^{\mathbb{T}}$	Hamana	2009

Basic Idea



Basic Idea: Graph Algorithmic View

▷ Traverse a graph in a depth-first search manner:



Depth-First Search tree

▷ DFS tree consists of 3 kinds of edges:

(i) Tree edge (ii) Back edge

(iii) Right-to-left cross edge

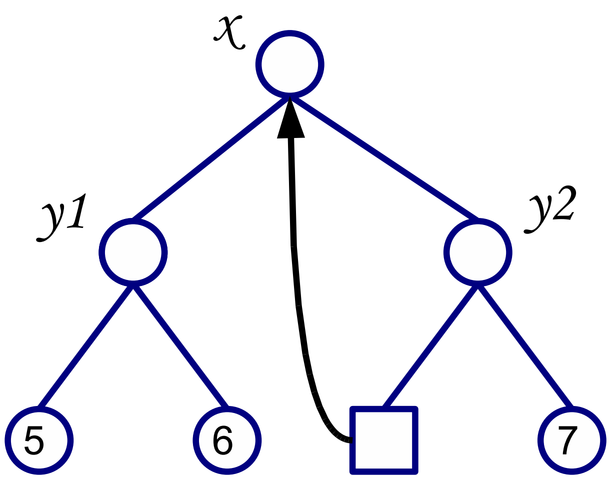
▷ Characterise pointers for back and cross edges

Formulation: Cycles by μ-terms

Idea

- ▷ *Binders* as *pointers*
- ▷ Back edges = bound variables

Cycles



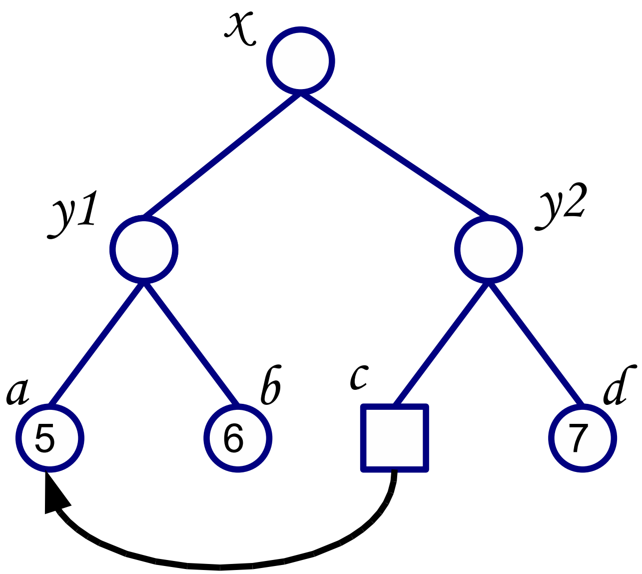
$$\mu x.\mathbf{bin}(\mu y_1.\mathbf{bin}(\mathbf{lf}(5), \mathbf{lf}(6)), \ \mu y_2.\mathbf{bin}(\ x, \ \mathbf{lf}(7)\ ))$$

Formulation: Sharing via ?

Idea

- ▷ *Binders as pointers*
- ▷ Back edges = bound variables
- ▷ Right-to-left Cross edges = ?

Sharing

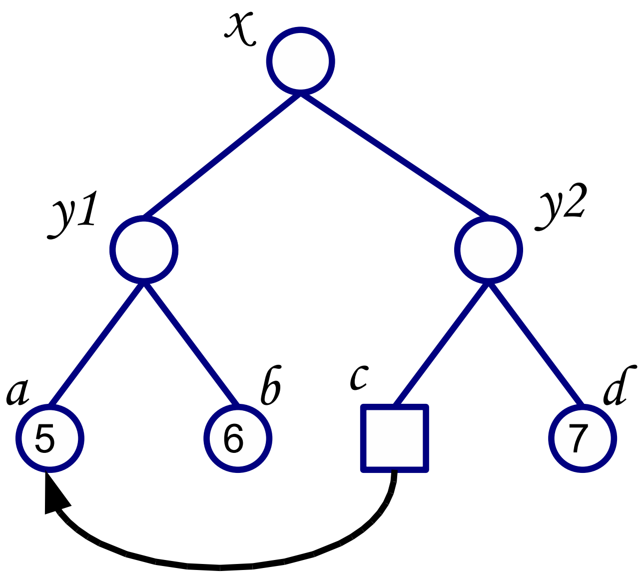


$$\mu x.\mathbf{bin}(\mu y_1.\mathbf{bin}(\mathbf{lf}(5), \mathbf{lf}(6)),\ \mu y_2.\mathbf{bin}(\boxed{}, \mathbf{lf}(7))).$$

Can we fill the blank to refer the node 5 by a bound variable?

Formulation: Sharing via Pointer

▷ Cross edges = pointers by a new notation



$$\mu x.\mathbf{bin}(\mu y_1.\mathbf{bin}(\mathbf{lf}(5), \mathbf{lf}(6)),\ \mu y_2.\mathbf{bin}(\boxed{\swarrow 11 \uparrow x}, \mathbf{lf}(7)))$$

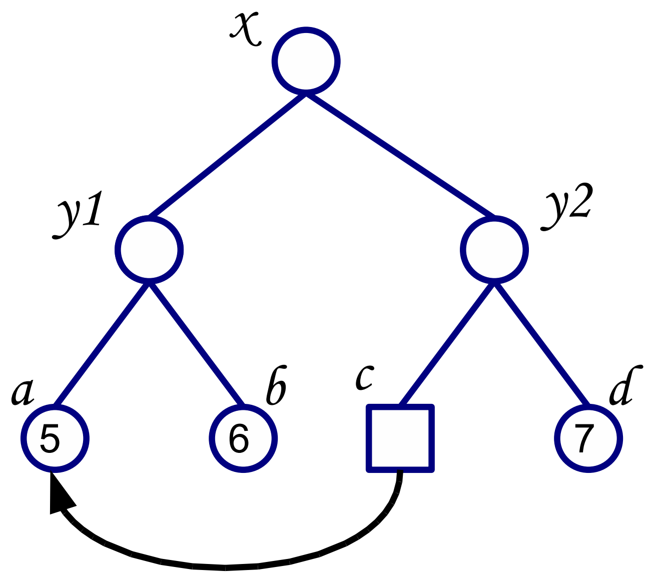
Pointer $\swarrow 11 \uparrow x$ means

▷ going back to the node x, then

▷ going down through the left child twice (by position $\mathbf{11}$)

Formulation: Sharing via Pointer

▷ Cross edges = pointers by a new notation



$$\mu x.\mathbf{bin}(\mu y_1.\mathbf{bin}(\mathbf{lf}(5), \mathbf{lf}(6)),\ \mu y_2.\mathbf{bin}(\boxed{\swarrow 11 \uparrow x}, \mathbf{lf}(7)))$$

Pointer $\swarrow 11 \uparrow x$ means Need to ensure a correct pointer only!!

▷ going back to the node x, then

▷ going down through the left child twice (by position **11**)

Typed Abstract Syntax

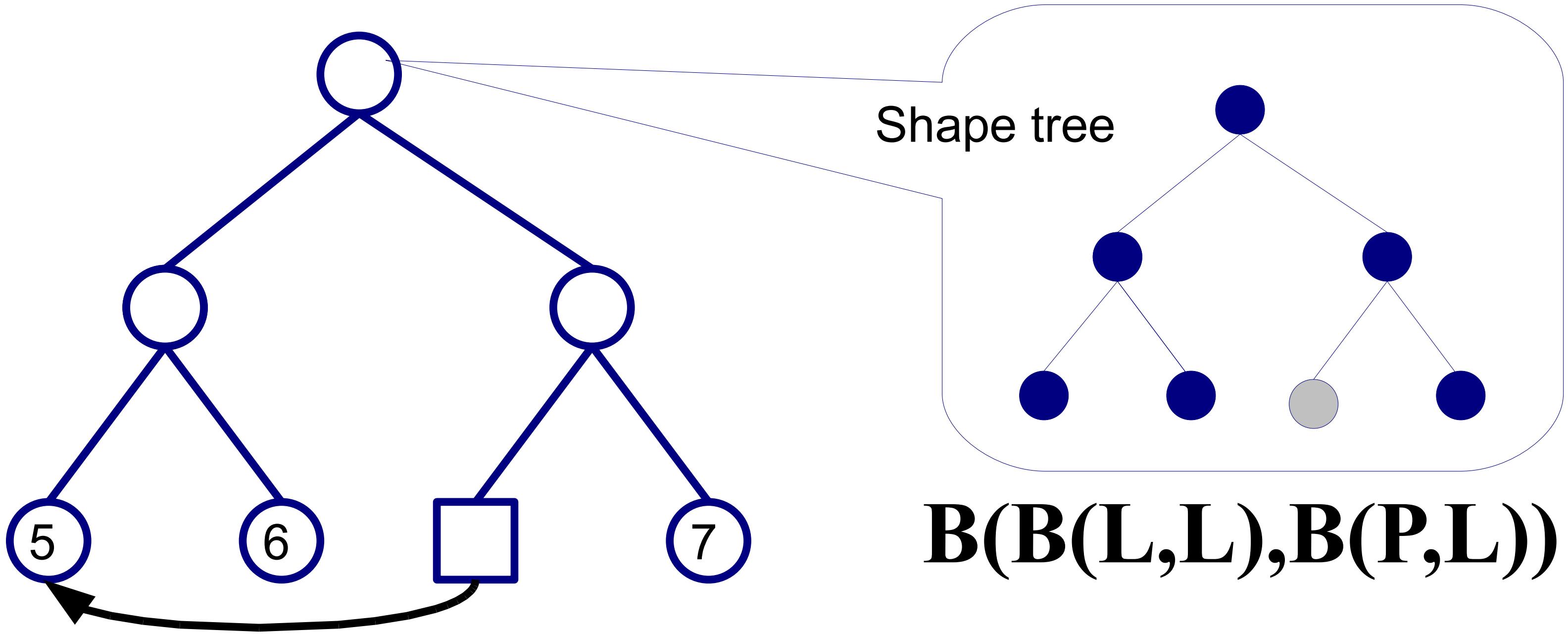
for

Cyclic Sharing Structures

Shape Trees

▷ Skeltons of cyclic sharing trees

Shape trees $\tau ::= \mathbf{E} \mid \mathbf{P} \mid \mathbf{L} \mid \mathbf{B}(\tau_1, \tau_2)$



▷ Used as types

▷ Blue nodes represent possible positions for sharing pointers.

Syntax and Types

Typing rules

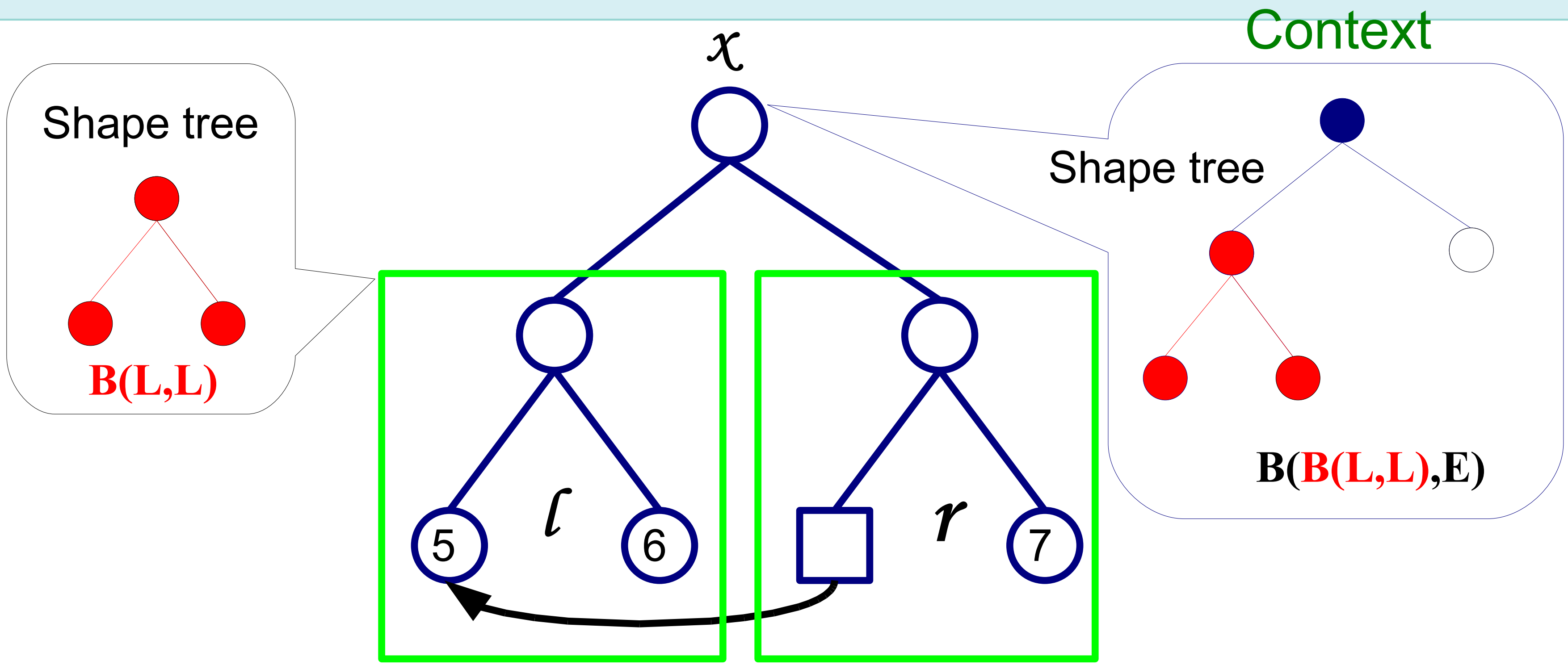
$$\frac{p \in \mathcal{P}os(\sigma)}{\Gamma, x : \sigma, \Gamma' \vdash \swarrow p{\uparrow}x : \mathrm{P}} \qquad \frac{k \in \mathbb{Z}}{\Gamma \vdash \mathsf{lf}(k) : \mathrm{L}}$$

$$\frac{x : \mathrm{B}(\mathrm{E}, \mathrm{E}), \Gamma \vdash \ell : \sigma \qquad x : \mathrm{B}(\sigma, \mathrm{E}), \Gamma \vdash r : \tau}{\Gamma \vdash \mu x.\mathsf{bin}(\ell, r) : \mathrm{B}(\sigma, \tau)}$$

▷ A type declaration $x : \sigma$ means:
"σ is the shape of a subtree headed by μx".

▷ Taking a position $p \in \mathcal{P}os(\sigma)$ safely refers to a position in the subtree.

Example: making bin-node



$$\frac{x{:}\mathrm{B}(\mathrm{E},\mathrm{E}) \vdash \mu y_1.\mathsf{bin}(5,6) : \mathrm{B}(\mathrm{L},\mathrm{L}) \qquad x{:}\mathrm{B}(\mathrm{B}(\mathrm{L},\mathrm{L}),\mathrm{E}) \vdash \mu y_2.\mathsf{bin}(\swarrow 11 \uparrow x, 7) : \mathrm{B}(\mathrm{P},\mathrm{L})}{\vdash \mu x.\mathsf{bin}(\mu y_1.\mathsf{bin}(5,6), \mu y_2.\mathsf{bin}(\swarrow 11 \uparrow x, 7)) : \mathrm{B}(\mathrm{B}(\mathrm{L},\mathrm{L}),\mathrm{B}(\mathrm{P},\mathrm{L}))}$$

Syntax and Types

Typing rules (de Bruijn version)

$$\frac{|\Gamma| = i - 1 \quad p \in \mathcal{P}os(\sigma)}{\Gamma, \sigma, \Gamma' \vdash \swarrow p \uparrow i : \mathrm{P}} \qquad \frac{k \in \mathbb{Z}}{\Gamma \vdash \mathsf{lf}(k) : \mathrm{L}}$$

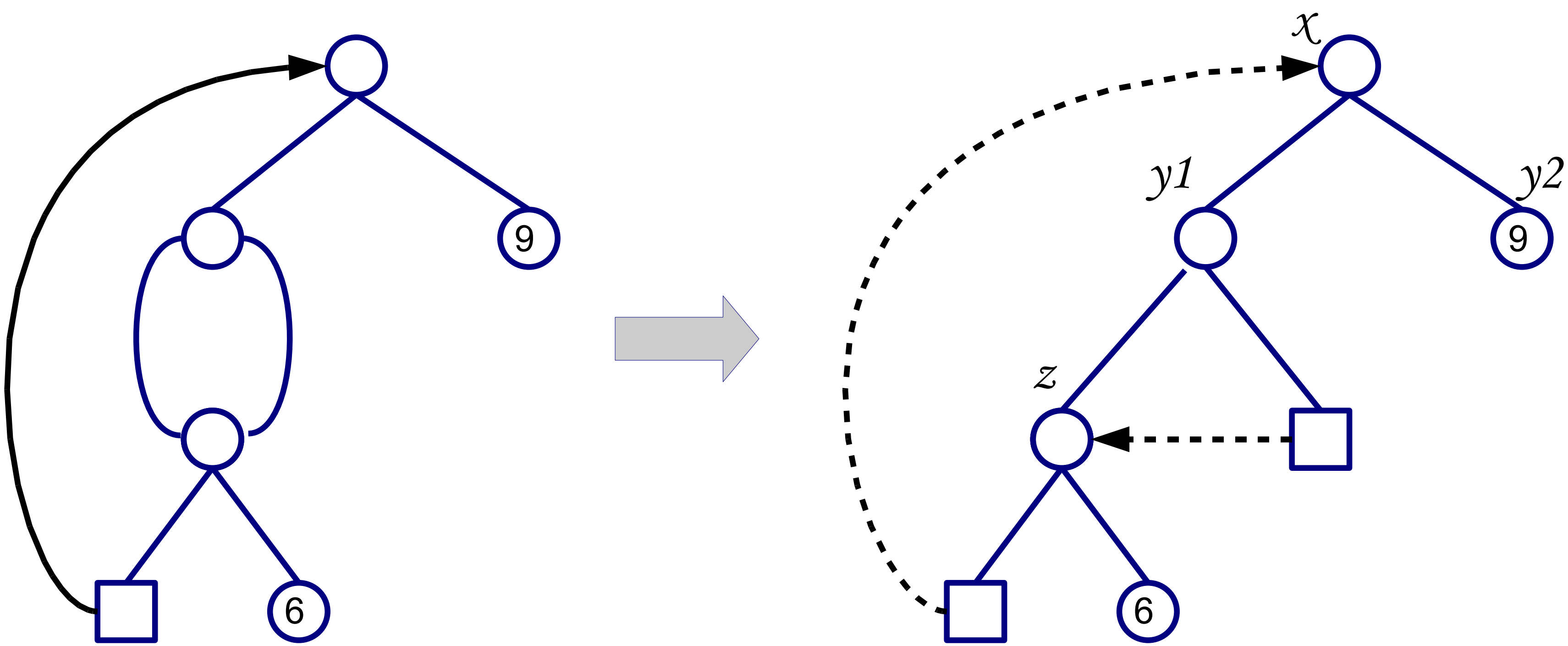
$$\frac{\mathrm{B}(\mathrm{E}, \mathrm{E}), \Gamma \vdash s : \sigma \quad \mathrm{B}(\sigma, \mathrm{E}), \Gamma \vdash t : \tau}{\Gamma \vdash \mathsf{bin}(s, t) : \mathrm{B}(\sigma, \tau)}$$

Thm.

Given rooted, connected and edge-ordered graph, the term representation in de Bruijn is unique.

Cyclic Sharing Data Structures

▷ Sharing via cross edge



▷ Term

$$\mathbf{bin}(\mathbf{bin}(\mathbf{bin}(\uparrow\ 3, \mathbf{lf}(6)), \swarrow 1 \uparrow 1), \mathbf{lf}(9))$$

Initial Algebra Semantics

▷ Cyclic sharing trees are all well-typed terms:

$$\boldsymbol{T_{\tau}(\Gamma) = \{t \mid \Gamma \vdash t : \tau\}}$$

▷ Need: sets indexed by
contexts $\mathbb{T}^*$ and shape trees $\mathbb{T}$

Consider algebras in $(\mathbf{Set}^{\mathbb{T}^*})^{\mathbb{T}}$

Initial Algebra Semantics

- Algebra of an endofunctor $\mathbf{\Sigma}$:
  $\mathbf{\Sigma}$-algebra $(\boldsymbol{A}, \boldsymbol{\alpha} : \mathbf{\Sigma}\boldsymbol{A} \longrightarrow \boldsymbol{A})$

- Functor $\mathbf{\Sigma} : (\mathbf{Set}^{\mathbb{T}^*})^{\mathbb{T}} \longrightarrow (\mathbf{Set}^{\mathbb{T}^*})^{\mathbb{T}}$ for cyclic sharing trees is defined by

$$(\mathbf{\Sigma}\boldsymbol{A})_{\mathrm{E}} = \mathbf{0} \qquad (\mathbf{\Sigma}\boldsymbol{A})_{\mathrm{P}} = \mathbf{PO} \qquad (\mathbf{\Sigma}\boldsymbol{A})_{\mathrm{L}} = \boldsymbol{K}_{\mathbb{Z}}$$

$$(\mathbf{\Sigma}\boldsymbol{A})_{\mathrm{B}(\sigma,\tau)} = \boldsymbol{\delta}_{\mathrm{B}(\mathrm{E},\mathrm{E})}\boldsymbol{A}_{\sigma} \times \boldsymbol{\delta}_{\mathrm{B}(\sigma,\mathrm{E})}\boldsymbol{A}_{\tau}$$

Initial Algebra Semantics

- $\boldsymbol{\Sigma}$-algebra $(\boldsymbol{A},\ \boldsymbol{\alpha} : \boldsymbol{\Sigma A} \longrightarrow \boldsymbol{A})$

- Functor $\boldsymbol{\Sigma} : (\mathbf{Set}^{\mathbb{T}^*})^{\mathbb{T}} \longrightarrow (\mathbf{Set}^{\mathbb{T}^*})^{\mathbb{T}}$ for cyclic sharing trees is given by

$$\mathbf{ptr}^{A} : \mathrm{PO} \longrightarrow A_{\mathrm{P}} \qquad \mathbf{lf}^{A} : K_{\mathbb{Z}} \longrightarrow A_{\mathrm{L}}$$

$$\mathbf{bin}^{\sigma,\tau\ A} : \delta_{\mathrm{B(E,E)}} A_{\sigma} \times \delta_{\mathrm{B}(\sigma,\mathrm{E})} A_{\tau} \longrightarrow A_{\mathrm{B}(\sigma,\tau)}$$

Typing rules (de Bruijn version)

$$\frac{|\Gamma| = i - 1 \quad p \in \mathcal{P}os(\sigma)}{\Gamma, \sigma, \Gamma' \vdash \swarrow p{\uparrow}i : \mathrm{P}} \qquad \frac{k \in \mathbb{Z}}{\Gamma \vdash \mathbf{lf}(k) : \mathrm{L}}$$

$$\frac{\mathrm{B(E,E)}, \Gamma \vdash s : \sigma \quad \mathrm{B}(\sigma,\mathrm{E}), \Gamma \vdash t : \tau}{\Gamma \vdash \mathbf{bin}(s,t) : \mathrm{B}(\sigma,\tau)}$$

Initial Algebra

▷ All cyclic sharing trees

$$\boldsymbol{T_{\tau}(\Gamma) = \{t \mid \Gamma \vdash t : \tau\}}$$

Thm. $\boldsymbol{T}$ forms an initial $\boldsymbol{\Sigma}$-algebra.

[Proof]

▷ Smith-Plotkin construction of an initial algebra

Principles

The initial algebra characterisation derives

(i) Structural recursion by the unique homomorphism

(ii) Structural induction by [Hermida,Jacobs I&C'98]

(iii) Inductive type (in Haskell)

Structural Recursion Principle

Thm. The unique homomorphism $\phi : T \longrightarrow A$ is:

$$\phi_{\mathrm{P}}(\Gamma)(\swarrow p\uparrow i) = \mathbf{ptr}^{A}(\Gamma)(\swarrow p\uparrow i)$$

$$\phi_{\mathrm{L}}(\Gamma)(\mathbf{lf}(k)) = \mathbf{lf}^{A}(\Gamma)(k)$$

$$\phi_{\mathrm{B}(\sigma,\tau)}(\Gamma)(\mathbf{bin}(s,t))$$
$$= \mathbf{bin}^{A}(\Gamma)(\phi_{\sigma}(\mathrm{B}(\mathrm{E},\mathrm{E}),\Gamma)(s),\ \phi_{\tau}(\mathrm{B}(\sigma,\mathrm{E}),\Gamma)(t))$$

▷ "fold" in Haskell

Structural Induction Principle

Thm. Let $\boldsymbol{P}$ be a predicate on $\boldsymbol{T}$.

To prove that $\boldsymbol{P}_{\boldsymbol{\tau}}^{\boldsymbol{\Gamma}}(\boldsymbol{t})$ holds for all $\boldsymbol{t} \in \boldsymbol{T}_{\boldsymbol{\tau}}(\boldsymbol{\Gamma})$,

it suffices to show

(i) $\boldsymbol{P}_{\mathbf{P}}^{\boldsymbol{\Gamma}}(\swarrow\boldsymbol{p}\uparrow\boldsymbol{i})$ holds for all $\swarrow\boldsymbol{p}\uparrow\boldsymbol{i} \in \mathbf{PO}(\boldsymbol{\Gamma})$,

(ii) $\boldsymbol{P}_{\mathbf{L}}^{\boldsymbol{\Gamma}}(\mathbf{lf}(\boldsymbol{k}))$ holds for all $\boldsymbol{k} \in \mathbb{Z}$,

(iii) If $\boldsymbol{P}_{\boldsymbol{\sigma}}^{\mathbf{B}(\mathbf{E},\mathbf{E}),\boldsymbol{\Gamma}}(\boldsymbol{s})$ & $\boldsymbol{P}_{\boldsymbol{\tau}}^{\mathbf{B}(\boldsymbol{\sigma},\mathbf{E}),\boldsymbol{\Gamma}}(\boldsymbol{t})$ holds, then $\boldsymbol{P}_{\mathbf{B}(\boldsymbol{\sigma},\boldsymbol{\tau})}^{\boldsymbol{\Gamma}}(\mathbf{bin}(\boldsymbol{s},\boldsymbol{t}))$ holds.

Inductive Type for Cyclic Sharing Structures

Constructors of the initial algebra $T \in (\mathbf{Set}^{\mathbb{T}^*})^{\mathbb{T}}$

$$\mathsf{ptr}^T(\Gamma) : \mathrm{PO}(\Gamma) \longrightarrow T_{\mathrm{P}}(\Gamma); \quad \swarrow p\!\uparrow\! i \mapsto \swarrow p\!\uparrow\! i.$$

$$\mathsf{lf}^T(\Gamma) : \mathbb{Z} \longrightarrow T_{\mathrm{L}}(\Gamma); \qquad k \mapsto \mathsf{lf}(k).$$

$$\mathsf{bin}^{\sigma,\tau\,T}(\Gamma) : T_\sigma(\mathrm{B}(\mathrm{E},\mathrm{E}),\Gamma) \times T_\tau(\mathrm{B}(\sigma,\mathrm{E}),\Gamma) \longrightarrow T_{\mathrm{B}(\sigma,\tau)}(\Gamma)$$

GADT in Haskell

```
data T :: * -> * -> * where
  Ptr :: Ctx n => n -> T n P
  Lf  :: Ctx n => Int -> T n L
  Bin :: (Ctx n, Shape s, Shape t) =>
         T (TyCtx (B E E) n) s -> T (TyCtx (B s E) n) t
                                  -> T n (B s t)
```

▷ Dependent type def. in Agda is more straightforward

Summary

▷ An initial algebra characterisation

Goals

▷ To derive the following from ↑ :

[I] A simple term syntax

[II] An inductive type

for cyclic sharing structures

Reference M. Hamana. Initial Algebra Semantics for Cyclic Sharing Structures, TLCA'09.

Connections to Other Works

There are interpretations:

$$\boldsymbol{T} \xrightarrow{\;!\;} \text{Equational Term Graphs} \longrightarrow \boldsymbol{\mathcal{S}}$$

where $\boldsymbol{\mathcal{S}}$ is any of

(i) Coalgebraic

(ii) Domain-theoretic

(iii) Categorical semantics:
Traced sym. monoidal categories [Hasegawa'97]

–

(Equational) term graphs [Barendregt et al.'87][Ariola,Klop'96]

Connection to Traced Categorical Semantics

▷ Interpretations

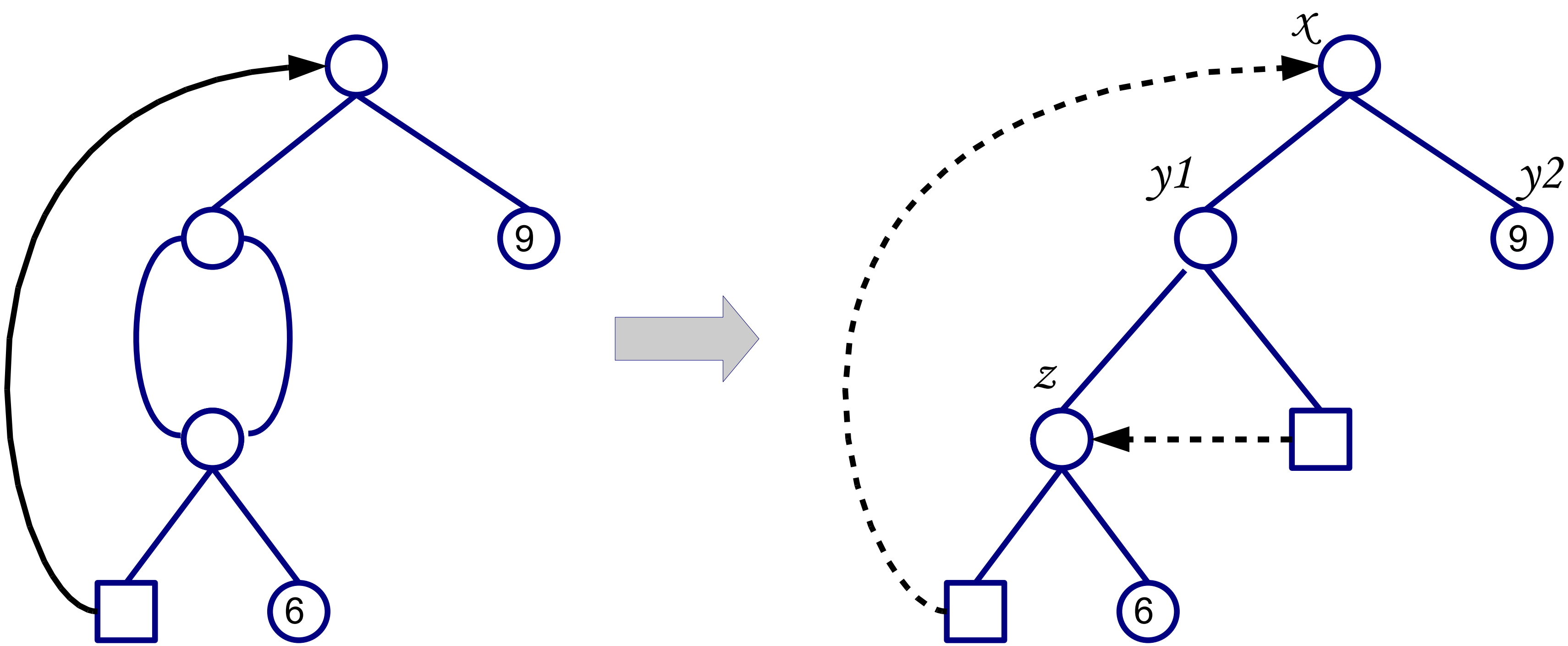
$$\mathcal{T} \xrightarrow{\ !\ } \text{Equational Term Graphs} \cong \text{letrec-Exprs} \longrightarrow (\mathcal{F} : \mathcal{C} \to \mathcal{M})$$

▷ Cartesian-center symmetric traced monoidal category
= identity-on-object functor $\mathcal{F} : \mathcal{C} \to \mathcal{M}$

- Cartesian $\mathcal{C}$
- Symmetric traced monoidal $\mathcal{M}$

Cyclic Sharing Data Structures

- ▷ Sharing via cross edge



- ▷ Term

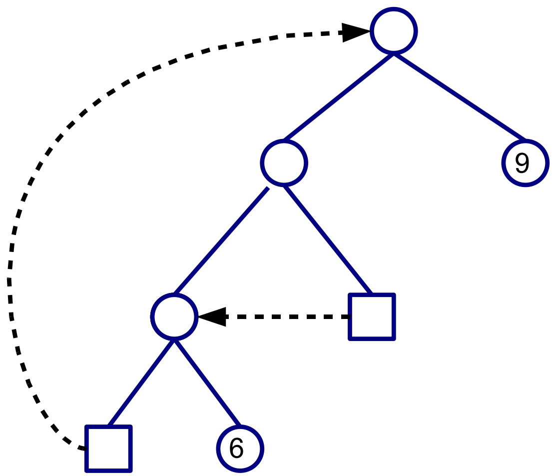
$$\mu x.\mathbf{bin}(\mu y_1.\mathbf{bin}(\mu z.\mathbf{bin}(\uparrow\ x, \mathbf{lf}(6)), \swarrow 1 \uparrow y_1), \mathbf{lf}(9))$$

$$\mu x.\mathsf{bin}(\mu y_1.\mathsf{bin}(\mu z.\mathsf{bin}(\uparrow\ x, \mathsf{lf}(6)), \swarrow 1{\uparrow}y_1), \mathsf{lf}(9))$$

$$\stackrel{\text{de Br.}}{=} \quad \mathsf{bin}(\mathsf{bin}(\mathsf{bin}(\uparrow\ 3, \mathsf{lf}(6)), \swarrow 1{\uparrow}1), \mathsf{lf}(9))$$

$$\longmapsto \quad \mathsf{bin}_{\epsilon}(\mathsf{bin}_1(\mathsf{bin}_{11}(\uparrow_{111}3, \mathsf{lf}_{112}(6)), \swarrow 1{\uparrow}_{12}\ 1), \mathsf{lf}_2(9))$$

$$\longmapsto \quad \{\epsilon \mid \begin{array}{lll} \epsilon & = & \mathsf{bin}(1,2) \\ 1 & = & \mathsf{bin}(11,12) \\ 11 & = & \mathsf{bin}(111,112) \\ 12 & = & 11 \\ 111 & = & \epsilon \\ 112 & = & \mathsf{lf}(6) \\ 2 & = & \mathsf{lf}(9)\} \end{array}$$



$$\longmapsto \quad \mathsf{letrec}\ (\epsilon, 1, 11, 12, 111, 112, 2)$$

$$= (\mathsf{bin}(1,2), \mathsf{bin}(1,12), \mathsf{bin}(111,112), 11, \epsilon, \mathsf{lf}(6), \mathsf{lf}(9))\ \mathsf{in}\ \epsilon$$

$$\stackrel{\text{Hasegawa}}{\longmapsto} \quad \mathcal{F}(\Delta); (\mathrm{id} \otimes \mathit{Tr}(\mathcal{F}\Delta_7; (\begin{array}{l} [\![\epsilon, 1, \ldots \vdash \mathsf{bin}(1,2)]\!] \otimes \\ [\![\epsilon, 1, \ldots \vdash \mathsf{bin}(11,12)]\!] \otimes \\ \cdots); \mathcal{F}\Delta)); \mathcal{F}\pi_1 \end{array}$$

Connection to Traced Categorical Semantics

- How useful?
- Application: Haskell's "arrows" [Hughes'00][Paterson'01]
  - Arrow-type in Haskell (or, Freyd category) is a cartesian-center premonoidal category [Heunen, Jacobs, Hasuo'06]
  - Arrow with loop is a cartesian-center traced premonoidal category [Benton, Hyland'03]
  - Cyclic sharing theory is interpreted in a cartesian-center traced monoidal category [Hasegawa'97]
- What impact for functional progrmming?