Space-Efficient Computation by Interaction

A Type System for Logarithmic Space

Ulrich Schöpp

LMU Munich

September 27, 2006

Implicit Computational Complexity

Characterisation of complexity classes

- by programming languages
- by logical concepts
- independent from machine models
- free from explicit resource bounds

Project *Pro.Platz* at LMU Munich

[Hofmann, Johannsen, Schwichtenberg, S]

Programming language aspects of sublinear space complexity classes (LOGSPACE, POLYLOG-SPACE, NLOGSPACE, …)

Programming Languages for Logarithmic Space

Existing Languages for LOGSPACE

[Lind 1974], [Bellantoni 1992], [Jones 1999],
[Møller-Neergaard & Mairson 2004], [Kristiansen 2005]
All minimal — by design

Richer languages?

function spaces, data types, convenient recursion schemes, …

Issues with extending existing languages

- Manageability
- Compositionality (implementation, soundness proof)

Space-Efficient Computation by Interaction

Address issues of manageability and compositionality by modelling *computation by interaction*.

Computation by Interaction as a general framework for space-efficient computation

Today's talk

1. Motivation: function algebra BC^{-}_{ε}
2. Modelling computation by interaction
3. Integrating resource bounds

The function algebra BC^-_ε

Origin

- BC [Bellantoni & Cook 1992]
- BC^- [Murawski & Ong 2000]
- $BC^- \subseteq$ LOGSPACE [Ong & Mairson 2003]
- $BC^-_\varepsilon =$ LOGSPACE [Møller-Neergaard 2004]

Definition

BC^-_ε is a set of functions on $\mathbb{N} = \{0,1\}^*$.

$$BC^-_\varepsilon \subseteq \{f : \mathbb{N}^m \times \mathbb{N}^n \to \mathbb{N} \mid m, n \in \mathbb{N}\}$$

Notation: $f(\vec{x}; \vec{y})$
(*normal* arguments $\vec{x}$; *safe* arguments $\vec{y}$)

Functions in BC^-

Base functions

Constants and basic functions like

$$\mathrm{c}(\cdot;\, y, u, v) = \begin{cases} u & \text{if } y = y'1 \\ v & \text{otherwise} \end{cases}$$

Recursion on notation

$$\begin{aligned} f(\vec{x}, \varepsilon;\, \vec{y}) &= g(\vec{x};\, \vec{y}) \\ f(\vec{x}, x0;\, \vec{y}) &= h(\vec{x}, x;\, f(\vec{x}, x;\, \vec{y})) \\ f(\vec{x}, x1;\, \vec{y}) &= k(\vec{x}, x;\, f(\vec{x}, x;\, \vec{y})) \end{aligned}$$

Composition

Normal arguments may be duplicated.

Safe arguments must be used *linearly*.

LOGSPACE-Evaluation of BC_ε^-

$$
\begin{aligned}
shift(\varepsilon,\, y) &= y & \mathrm{p}(\varepsilon) &= \varepsilon \\
shift(x0,\, y) &= \mathrm{p}(shift(x,\, y)) & \mathrm{p}(x0) &= x \\
shift(x1,\, y) &= \mathrm{p}(shift(x,\, y)) & \mathrm{p}(x1) &= x
\end{aligned}
$$

- Example: $shift(101;\, 1110) = 1$
- Naive call-by-value evaluation of $shift$ uses linear space.
- LOGSPACE-evaluation of BC_ε^- proceeds bit by bit.

LOGSPACE-Evaluation of BC_{ε}^{-}

$$
\begin{aligned}
shift(\varepsilon,\, y) &= y & \mathrm{p}(\varepsilon) &= \varepsilon \\
shift(x0,\, y) &= \mathrm{p}(shift(x,\, y)) & \mathrm{p}(x0) &= x \\
shift(x1,\, y) &= \mathrm{p}(shift(x,\, y)) & \mathrm{p}(x1) &= x
\end{aligned}
$$

Bit 0 of
$shift(101;\, 1110)$

LOGSPACE-Evaluation of BC_{ε}^{-}

$$
\begin{aligned}
shift(\varepsilon,\, y) &= y & \mathrm{p}(\varepsilon) &= \varepsilon \\
shift(x0,\, y) &= \mathrm{p}(shift(x,\, y)) & \mathrm{p}(x0) &= x \\
shift(x1,\, y) &= \mathrm{p}(shift(x,\, y)) & \mathrm{p}(x1) &= x
\end{aligned}
$$

Bit 0 of $shift(101;\, 1110)$

Bit 1 of $shift(10;\, 1110)$

LOGSPACE-Evaluation of BC_{ε}^{-}

$$shift(\varepsilon,\, y) = y \qquad \mathrm{p}(\varepsilon) = \varepsilon$$
$$shift(x0,\, y) = \mathrm{p}(shift(x,\, y)) \qquad \mathrm{p}(x0) = x$$
$$shift(x1,\, y) = \mathrm{p}(shift(x,\, y)) \qquad \mathrm{p}(x1) = x$$

Bit 0 of $shift(101;\, 1110)$ | Bit 1 of $shift(10;\, 1110)$ | Bit 2 of $shift(1;\, 1110)$

LOGSPACE-Evaluation of BC_{ε}^{-}

$$
\begin{aligned}
shift(\varepsilon,\, y) &= y & \mathrm{p}(\varepsilon) &= \varepsilon \\
shift(x0,\, y) &= \mathrm{p}(shift(x,\, y)) & \mathrm{p}(x0) &= x \\
shift(x1,\, y) &= \mathrm{p}(shift(x,\, y)) & \mathrm{p}(x1) &= x
\end{aligned}
$$

Bit 0 of	Bit 1 of	Bit 2 of	Bit 3 of
$shift(101;\, 1110)$	$shift(10;\, 1110)$	$shift(1;\, 1110)$	$shift(\varepsilon;\, 1110)$

LOGSPACE-Evaluation of BC_{ε}^{-}

$$
\begin{aligned}
\mathit{shift}(\varepsilon,\, y) &= y & \mathrm{p}(\varepsilon) &= \varepsilon \\
\mathit{shift}(x0,\, y) &= \mathrm{p}(\mathit{shift}(x,\, y)) & \mathrm{p}(x0) &= x \\
\mathit{shift}(x1,\, y) &= \mathrm{p}(\mathit{shift}(x,\, y)) & \mathrm{p}(x1) &= x
\end{aligned}
$$

Bit 0 of	Bit 1 of	Bit 2 of	Bit 3 of
$\mathit{shift}(101;\, 1110)$	$\mathit{shift}(10;\, 1110)$	$\mathit{shift}(1;\, 1110)$	$\mathit{shift}(\varepsilon;\, 1110)$ is 1

LOGSPACE-Evaluation of BC_{ε}^{-}

$$
\begin{aligned}
\mathit{shift}(\varepsilon,\, y) &= y & \mathrm{p}(\varepsilon) &= \varepsilon \\
\mathit{shift}(x0,\, y) &= \mathrm{p}(\mathit{shift}(x,\, y)) & \mathrm{p}(x0) &= x \\
\mathit{shift}(x1,\, y) &= \mathrm{p}(\mathit{shift}(x,\, y)) & \mathrm{p}(x1) &= x
\end{aligned}
$$

Bit 3 of
$\mathit{shift}(\varepsilon;\, 1110)$
is 1

LOGSPACE-Evaluation of BC_{ε}^{-}

$$
\begin{aligned}
\mathit{shift}(\varepsilon,\, y) &= y & \mathrm{p}(\varepsilon) &= \varepsilon \\
\mathit{shift}(x0,\, y) &= \mathrm{p}(\mathit{shift}(x,\, y)) & \mathrm{p}(x0) &= x \\
\mathit{shift}(x1,\, y) &= \mathrm{p}(\mathit{shift}(x,\, y)) & \mathrm{p}(x1) &= x
\end{aligned}
$$

Bit 0 of
$\mathit{shift}(101;\, 1110)$

Bit 3 of
$\mathit{shift}(\varepsilon;\, 1110)$
is 1

LOGSPACE-Evaluation of BC_{ε}^{-}

$$\begin{aligned} shift(\varepsilon,\, y) &= y & \mathrm{p}(\varepsilon) &= \varepsilon \\ shift(x0,\, y) &= \mathrm{p}(shift(x,\, y)) & \mathrm{p}(x0) &= x \\ shift(x1,\, y) &= \mathrm{p}(shift(x,\, y)) & \mathrm{p}(x1) &= x \end{aligned}$$

Bit 0 of $shift(101;\, 1110)$

Bit 1 of $shift(10;\, 1110)$

Bit 3 of $shift(\varepsilon;\, 1110)$ is 1

LOGSPACE-Evaluation of BC_{ε}^{-}

$$
\begin{aligned}
\mathit{shift}(\varepsilon,\, y) &= y & \mathrm{p}(\varepsilon) &= \varepsilon \\
\mathit{shift}(x0,\, y) &= \mathrm{p}(\mathit{shift}(x,\, y)) & \mathrm{p}(x0) &= x \\
\mathit{shift}(x1,\, y) &= \mathrm{p}(\mathit{shift}(x,\, y)) & \mathrm{p}(x1) &= x
\end{aligned}
$$

Bit 0 of	Bit 1 of	Bit 2 of	Bit 3 of
$\mathit{shift}(101;\, 1110)$	$\mathit{shift}(10;\, 1110)$	$\mathit{shift}(1;\, 1110)$	$\mathit{shift}(\varepsilon;\, 1110)$ is 1

LOGSPACE-Evaluation of BC_{ε}^{-}

$$\begin{aligned} shift(\varepsilon,\, y) &= y & \mathrm{p}(\varepsilon) &= \varepsilon \\ shift(x0,\, y) &= \mathrm{p}(shift(x,\, y)) & \mathrm{p}(x0) &= x \\ shift(x1,\, y) &= \mathrm{p}(shift(x,\, y)) & \mathrm{p}(x1) &= x \end{aligned}$$

Bit 0 of $shift(101;\, 1110)$

Bit 1 of $shift(10;\, 1110)$

Bit 2 of $shift(1;\, 1110)$ is 1

Bit 3 of $shift(\varepsilon;\, 1110)$

LOGSPACE-Evaluation of BC_{ε}^{-}

$$\begin{aligned} \mathit{shift}(\varepsilon,\, y) &= y & \qquad \mathrm{p}(\varepsilon) &= \varepsilon \\ \mathit{shift}(x0,\, y) &= \mathrm{p}(\mathit{shift}(x,\, y)) & \qquad \mathrm{p}(x0) &= x \\ \mathit{shift}(x1,\, y) &= \mathrm{p}(\mathit{shift}(x,\, y)) & \qquad \mathrm{p}(x1) &= x \end{aligned}$$

Bit 2 of
$\mathit{shift}(1;\, 1110)$
is 1

LOGSPACE-Evaluation of BC_{ε}^{-}

$$
\begin{aligned}
shift(\varepsilon,\, y) &= y & \mathrm{p}(\varepsilon) &= \varepsilon \\
shift(x0,\, y) &= \mathrm{p}(shift(x,\, y)) & \mathrm{p}(x0) &= x \\
shift(x1,\, y) &= \mathrm{p}(shift(x,\, y)) & \mathrm{p}(x1) &= x
\end{aligned}
$$

Bit 0 of
$shift(101;\, 1110)$

Bit 2 of
$shift(1;\, 1110)$
is 1

LOGSPACE-Evaluation of BC_{ε}^{-}

$$
\begin{aligned}
shift(\varepsilon,\, y) &= y & \mathrm{p}(\varepsilon) &= \varepsilon \\
shift(x0,\, y) &= \mathrm{p}(shift(x,\, y)) & \mathrm{p}(x0) &= x \\
shift(x1,\, y) &= \mathrm{p}(shift(x,\, y)) & \mathrm{p}(x1) &= x
\end{aligned}
$$

Bit 0 of $shift(101;\, 1110)$

Bit 1 of $shift(10;\, 1110)$

Bit 2 of $shift(1;\, 1110)$ is 1

LOGSPACE-Evaluation of BC_{ε}^{-}

$$
\begin{aligned}
\mathit{shift}(\varepsilon,\, y) &= y & \mathrm{p}(\varepsilon) &= \varepsilon \\
\mathit{shift}(x0,\, y) &= \mathrm{p}(\mathit{shift}(x,\, y)) & \mathrm{p}(x0) &= x \\
\mathit{shift}(x1,\, y) &= \mathrm{p}(\mathit{shift}(x,\, y)) & \mathrm{p}(x1) &= x
\end{aligned}
$$

Bit 0 of $\mathit{shift}(101;\, 1110)$

Bit 1 of $\mathit{shift}(10;\, 1110)$ is 1

Bit 2 of $\mathit{shift}(1;\, 1110)$

LOGSPACE-Evaluation of BC_{ε}^{-}

$$shift(\varepsilon,\, y) = y \qquad \mathrm{p}(\varepsilon) = \varepsilon$$
$$shift(x0,\, y) = \mathrm{p}(shift(x,\, y)) \qquad \mathrm{p}(x0) = x$$
$$shift(x1,\, y) = \mathrm{p}(shift(x,\, y)) \qquad \mathrm{p}(x1) = x$$

Bit 1 of
$shift(10;\, 1110)$
is 1

LOGSPACE-Evaluation of BC_{ε}^{-}

$$shift(\varepsilon,\, y) = y \qquad \mathrm{p}(\varepsilon) = \varepsilon$$
$$shift(x0,\, y) = \mathrm{p}(shift(x,\, y)) \qquad \mathrm{p}(x0) = x$$
$$shift(x1,\, y) = \mathrm{p}(shift(x,\, y)) \qquad \mathrm{p}(x1) = x$$

Bit 0 of $shift(101;\, 1110)$

Bit 1 of $shift(10;\, 1110)$ is 1

LOGSPACE-Evaluation of BC_{ε}^{-}

$$
\begin{aligned}
shift(\varepsilon,\, y) &= y & \mathrm{p}(\varepsilon) &= \varepsilon \\
shift(x0,\, y) &= \mathrm{p}(shift(x,\, y)) & \mathrm{p}(x0) &= x \\
shift(x1,\, y) &= \mathrm{p}(shift(x,\, y)) & \mathrm{p}(x1) &= x
\end{aligned}
$$

Bit 0 of $shift(101;\, 1110)$ is 1

Bit 1 of $shift(10;\, 1110)$

LOGSPACE-Evaluation of BC_{ε}^{-}

$$
\begin{aligned}
\mathit{shift}(\varepsilon,\, y) &= y & \mathrm{p}(\varepsilon) &= \varepsilon \\
\mathit{shift}(x0,\, y) &= \mathrm{p}(\mathit{shift}(x,\, y)) & \mathrm{p}(x0) &= x \\
\mathit{shift}(x1,\, y) &= \mathrm{p}(\mathit{shift}(x,\, y)) & \mathrm{p}(x1) &= x
\end{aligned}
$$

Bit 0 of
$\mathit{shift}(101;\, 1110)$
is 1

LOGSPACE-Evaluation of BC_{ε}^{-}

$$
\begin{aligned}
shift(\varepsilon,\, y) &= y & \mathrm{p}(\varepsilon) &= \varepsilon \\
shift(x0,\, y) &= \mathrm{p}(shift(x,\, y)) & \mathrm{p}(x0) &= x \\
shift(x1,\, y) &= \mathrm{p}(shift(x,\, y)) & \mathrm{p}(x1) &= x
\end{aligned}
$$

This evaluation strategy is …

- correct because one bit of $shift(xi;\, y)$ depends on only one bit of $shift(x;\, y)$, by linearity.
- in LOGSPACE because we only need to store
  - Bit numbers
  - Recursion depth

LOGSPACE-Evaluation of BC_ε^-

LOGSPACE-evaluation of BC_ε^- is a question/answer-dialogue.

$\Rightarrow$ *Game Semantics*

$\Rightarrow$ *Geometry of Interaction Situation*

Geometry of Interaction

Origin: syntax-free cut elimination for linear logic [Girard 1988]

Connection to game semantics [Abramsky, Jagadeesan 1992]

Applications in many areas

- Compilation of functional languages [Mackie 1995]
- Abstract machines [Danos, Herbelin, Regnier 1996]
- Implicit complexity theory [Baillot, Pedicini 2000]
- …

General categorical formulation
[Abramsky, Haghverdi, Scott 2000], [Hyland]

Geometry of Interaction

Objects

$$A = (A^-, A^+)$$

Example:

$$\mathbb{N}^- = \mathbb{N}$$
$$\mathbb{N}^+ = \{0, 1, *\}$$

Natural numbers are represented by functions $\mathbb{N}^- \longrightarrow \mathbb{N}^+$.

Geometry of Interaction

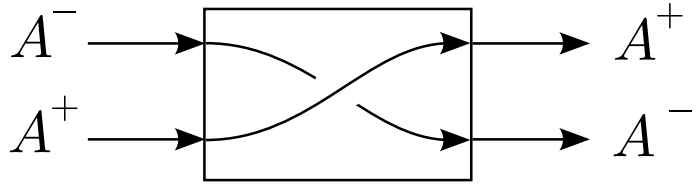
Morphisms

Morphism of type $f : A \to B$ is a partial function

$$f : B^- + A^+ \longrightarrow B^+ + A^-$$

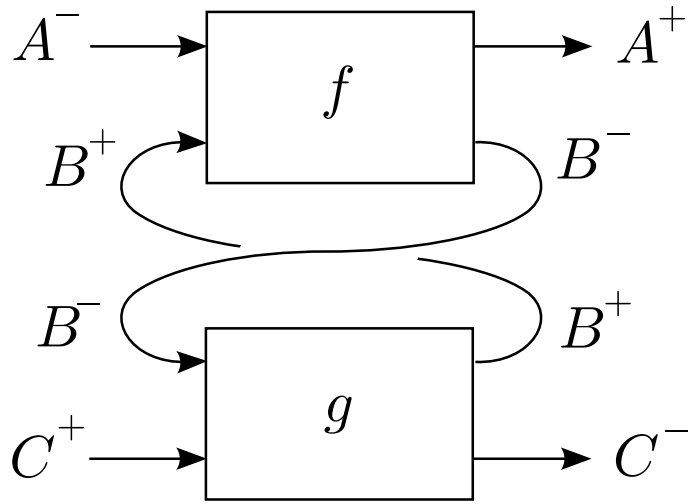
Identity $id : A \longrightarrow A$

$$A^- + A^+ \longrightarrow A^+ + A^-$$
$$inl(q) \mapsto inr(q)$$
$$inr(a) \mapsto inl(a)$$



Composition

$$\frac{f\colon B \longrightarrow A \qquad g\colon C \longrightarrow B}{f \cdot g\colon C \longrightarrow A}$$



Structure

- Pairs

$$A \otimes B = (A^- + B^-, A^+ + B^+)$$

- Functions

$$A \multimap B = (A^+ + B^-, A^- + B^+)$$

- Storage

$$!A = (A^- \times \mathbb{N},\ A^+ \times \mathbb{N})$$

Interpreting BC^-_ε

Compilation of BC^-_ε to LOGSPACE by interpretation in GoI

Conditional

$$\mathrm{c}\colon\ !\mathbb{N}\otimes!\mathbb{N}\otimes!\mathbb{N} \longrightarrow \mathbb{N}$$

$$\begin{aligned}
\mathbb{N}^- + !\mathbb{N}^+ + !\mathbb{N}^+ + !\mathbb{N}^+ &\longrightarrow \mathbb{N}^+ + !\mathbb{N}^- + !\mathbb{N}^- + !\mathbb{N}^- \\
in_1(q) &\longmapsto in_2(0,q) \\
in_2(1,s) &\longmapsto in_3(s,0) \\
in_2(_,s) &\longmapsto in_4(s,0) \\
in_3(a,s) &\longmapsto in_1(a) \\
in_4(a,s) &\longmapsto in_1(a)
\end{aligned}$$

Interpreting BC^{-}_{ε}

Recursion combinator

$$\mathrm{rec}\colon !\mathbb{N}\otimes!(!\mathbb{N}\multimap\mathbb{N})\otimes!(!\mathbb{N}\multimap\mathbb{N})\otimes!\mathbb{N}\multimap\mathbb{N}$$

$$\begin{aligned}
inl(q) &\longmapsto in_x(0,\langle q,q,0,-1,-1\rangle)\\
in_x(*,s) &\longmapsto in_g(s.q,s)\\
in_x(i,s) &\longmapsto in_{h_i}(s.q,s)\\
in_g(a,s) &\longmapsto \begin{cases} inr(a) & \text{if } s_3=0\\ in_x(q_0,\langle s_1,s_1,0,s_3,a\rangle) & \text{otherwise}\end{cases}\\
in_{h_i}(a,s) &\longmapsto in_g(a,s)\\
in_{r_i}((q,l),s) &\longmapsto \begin{cases} in_{r_i}((s_5,l),s) & \text{if } s_3+1=s_4\\ in_x(s_3+1,\langle s_1,q,s_3+1,s_4,s_5\rangle)) & \text{otherwise}\end{cases}
\end{aligned}$$

Computation by Interaction

What do we gain by modelling computation in GoI?

- Framework for implementing space-efficient programs in a compositional way and for proving resource-bounds
- Higher-order linear functions
- Data types
- Potential to reuse a large amount of existing work

What's not so good?

- GoI contains much more than LOGSPACE

$\Rightarrow$ We still have to establish resource bounds.

Integrating Resource Bounds

Restrict the interaction model to capture LOGSPACE

Aspects of LOGSPACE-soundness for BC_{ε}^{-}

- Bit-by-bit evaluation $\Rightarrow$ Realisability
- Each bit evaluated in linear space $\Rightarrow$ Linear space GoI
- Safe arguments queried at most once $\Rightarrow$ Co-Realisability
- Polynomial growth $\Rightarrow$ Non-size-increasing functions

Integrating Resource Bounds

Restrict the interaction model to capture LOGSPACE

Aspects of LOGSPACE-soundness for BC^{-}_{ε}

- Bit-by-bit evaluation $\Rightarrow$ Realisability
- Each bit evaluated in linear space
- Safe arguments queried at most once $\Rightarrow$ Co-Realisability
- Polynomial growth

Realisability

Object A

- Set $|A|$ (Elements)
- Each $x \in |A|$ has at least one realiser $e : R^- \to R^+$

Morphism $A \to B$

- Function $|f| : |A| \to |B|$

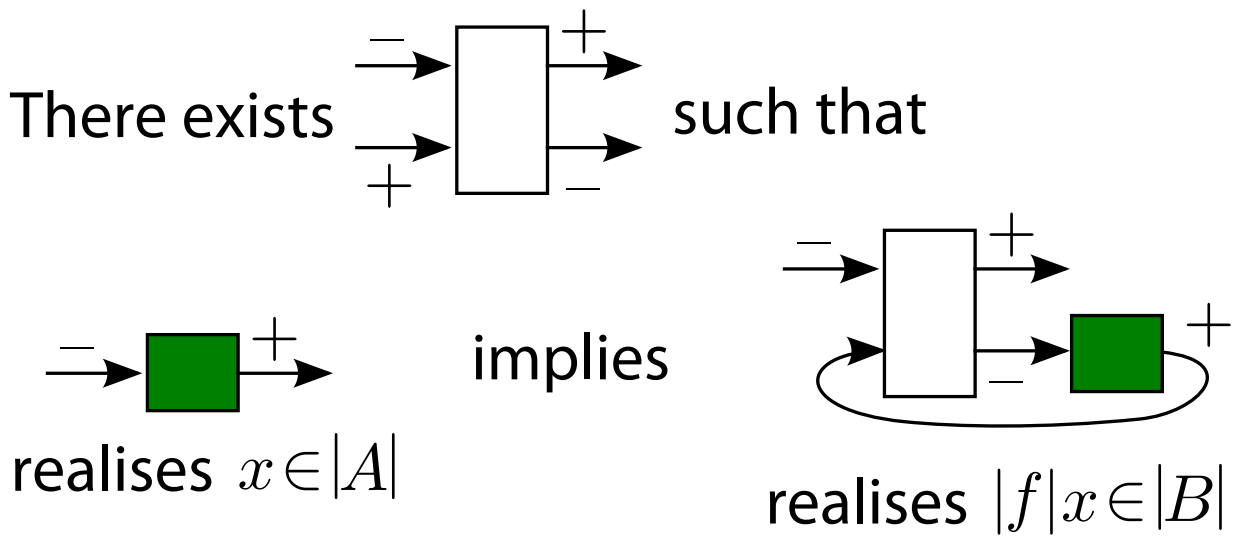
Realisability

Object A

- Set $|A|$ (Elements)
- Each $x \in |A|$ has at least one realiser $e : R^- \to R^+$

Morphism $A \to B$

- Function $|f| : |A| \to |B|$



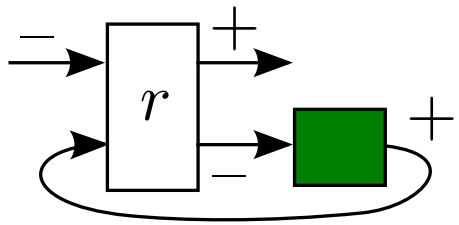
Realisability

Restriction to LOGSPACE

For each morphism $f : \mathbf{L}(\mathbf{Bool}) \to \mathbf{L}(\mathbf{Bool})$,
the underlying function $|f|$ is LOGSPACE-computable.

Not enough to model BC^{-}_{ε}-style recursion

Step functions can only be morphisms with realisers r that query their argument at most once!



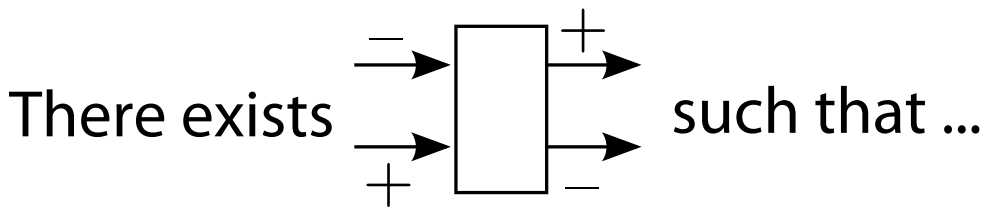
Co-Realisability

Object A

- Sets $|A|$ (Elements)
- Each $x \in |A|$ has at least one realiser $e : R^- \to R^+$
- Sets A^* (Co-Elements)
- Each $k \in A^*$ has at least one co-realiser $c : R^+ \to R^-$

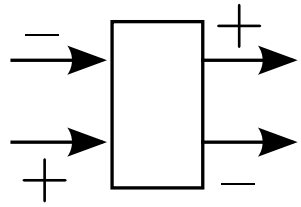
Morphism $A \to B$

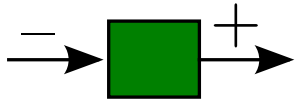
- Function $|f| : |A| \to |B|$
- Function $f^* : B^* \to A^*$



Morphism $A \to B$

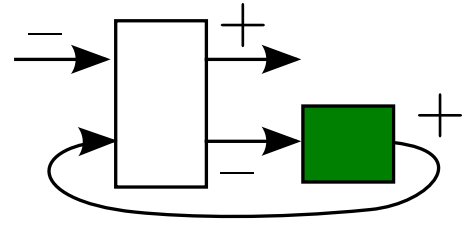
- Function $|f| : |A| \to |B|$
- Function $f^* : B^* \to A^*$

There exists  such that

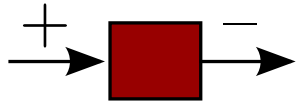


realises $x \in |A|$

implies

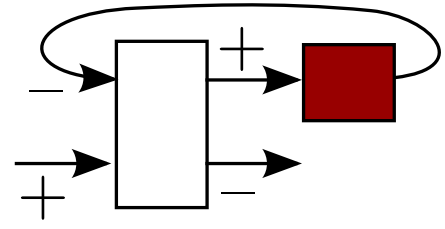


realises $|f| x \in |B|$



co-realises $k \in B^*$

implies



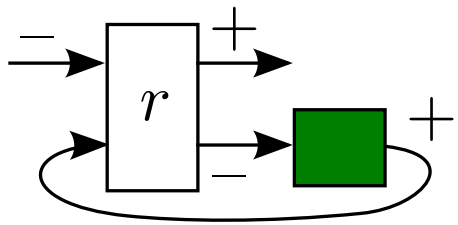
co-realises $f^* k \in A^*$

Co-Realisability

Co-Realisability controls how realisers use their arguments

$f : A \to A$ realised by r, where $A^* = \{\emptyset\}$ and $\emptyset$ is co-realised only by the empty function

How many questions does r send to its argument?

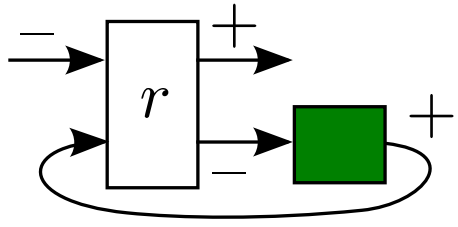


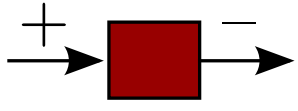
Co-Realisability

Co-Realisability controls how realisers use their arguments

$f : A \to A$ realised by r, where $A^* = \{\emptyset\}$ and $\emptyset$ is co-realised only by the empty function

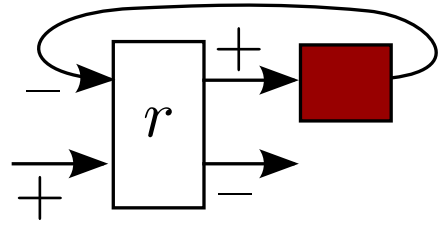
How many questions does r send to its argument?





co-realises $k \in A^*$

implies



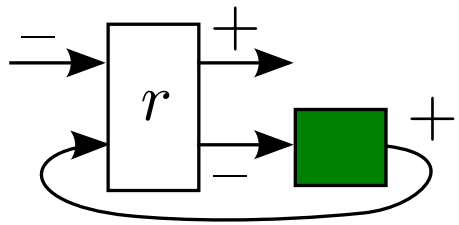
co-realises $f^*k \in A^*$

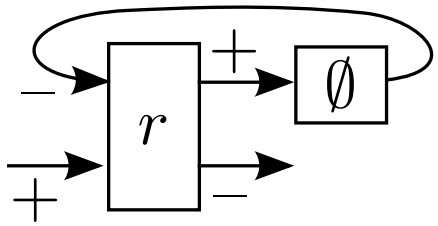
Co-Realisability

Co-Realisability controls how realisers use their arguments

$f : A \to A$ realised by r, where $A^* = \{\emptyset\}$ and $\emptyset$ is co-realised only by the empty function

How many questions does r send to its argument?





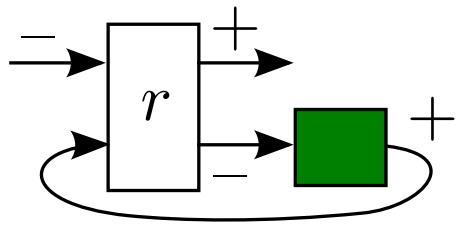
is the empty function

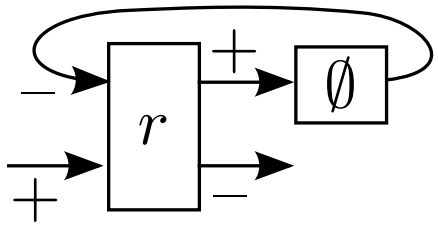
Co-Realisability

Co-Realisability controls how realisers use their arguments

$f : A \to A$ realised by r, where $A^* = \{\emptyset\}$ and $\emptyset$ is co-realised only by the empty function

How many questions does r send to its argument?





is the empty function

$\Rightarrow r$ queries its argument at most once

Structure

- Pairs ($\otimes$)
- Linear functions ($\multimap$)
- Basic data types ($\mathbf{Bool}$, $\mathbf{SN}$, $\mathbf{L}(A)$)
- Co-Realisability for controlling the number of questions
- Single-query pairs ($\otimes_1$)
- Modality ($\Box$)

$$f : \Box A \otimes A \longrightarrow A$$

- …

Can build a programming language on this structure

Conclusion and Further Work

Computation by interaction is a good way of modelling space-efficient computation!

Further Work

- Simplify and extend programming language in the paper
- Recursively defined functions
- Other complexity classes (e.g. polylog space)