Abstract computation over first-order structures

Sufficient conditions for the existence of universal BSS RAMs

Christine Gaßner

University of Greifswald

Verona 2025

Outline

From algorithms over first-order structures to universal machines

Motivation: Reasons for our generalization

Introduction: The BSS-RAM model

- Structures $\mathcal{A}$ of signature σ
- σ-Programs
- Transition systems and BSS RAMs
- An example

Semi-decidable decision problems

Universal machines of type 1 and their importance

Constants and their recognizability

Algorithms over First-Order Structures
Informal

Examples

IF $X > 1$ THEN $X := X + 1$.

$\Uparrow\Uparrow$ $\Uparrow\Uparrow$

describes an algorithm to compute $x + 1$ for x greater than 1.

We use

- an operation to transform an object,
- a relation for evaluating a condition,
- a constant.

The underlying structures

of our machines has this form:

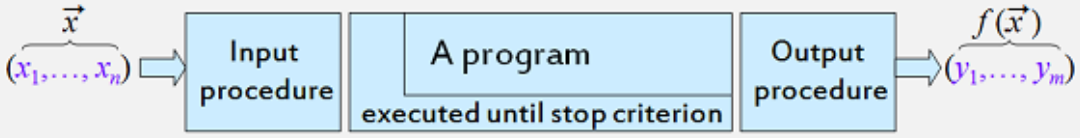
$$\mathcal{A} = (\underbrace{U_{\mathcal{A}}}_{\text{universe}}; \underbrace{(c_i)_{i\in N_1}}_{\text{constants}}; \underbrace{(f_i)_{i\in N_2}}_{\text{operations}}; \underbrace{(r_i)_{i\in N_3}}_{\text{relations}}).$$

Known Machines over First-Order Structures

First-order structures $\mathcal{A}$ and several types of machines

$(\{0,1\};0,1;;=)$	Turing machines (1), Type-2 machines (3)
$(\mathbb{N};0,1;+,-,\cdot;<,=)$	classical RAMs (2)
$(\mathbb{R};\mathbb{R};+,-,\cdot;<,=)$	real RAMs (2), algebraic models (2), BSS machines (1)
$(\mathbb{R};\mathbb{R};+,-;<,=)$	linear models (1), (2), additive BSS machines (1)

Minsky (1961), Scott (1967), Blum/Shub/Smale (1989),... (offline):



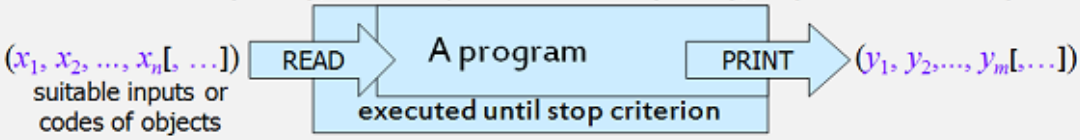
(1)

machines over A

$\mathcal{A}$-machines

BSS RAMs over $\mathcal{A}$

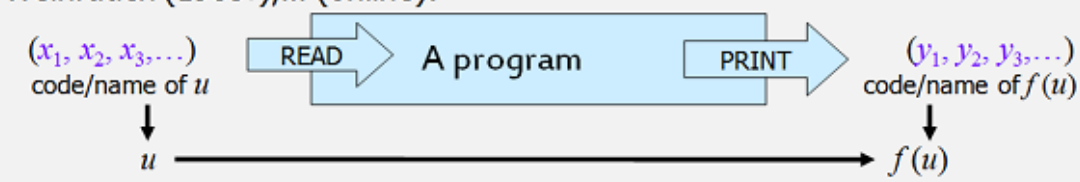
Cook/Reckhow (1973), Aho/Hopcroft/Ullman (1974),... (offline / online):



(2)

machines over A

Weihrauch (1985?),... (online):



(3)

machines over A

Known Machines over First-Order Structures

First-order structures

$(\{0,1\};0,1;;=)$	Turing machines (1)
$(\mathbb{N};0,1;+,-,\cdot;<,=)$	classical RAMs (2)
$(\mathbb{R};\mathbb{R};+,-,\cdot;<,=)$	real RAMs (3), algebraic models (3), BSS machines (4)
$(\mathbb{R};\mathbb{R};+,-;<,=)$	linear models (3), (5), additive BSS machines (5)

Introduced and/or investigated by

(1) Alan M. Turing,

(2) Alfred V. Aho, John E. Hopcroft, Jeffrey D. Ullman, . . .

(3) Franco P. Preparata, Michael I. Shamos, . . .

(4) Lenore Blum, Steve Smale, Michael Shub,

(5) Felipe Cucker, Klaus Meer, Pascal Koiran, ...

A Program for Semi-Deciding a Set

Over $(\mathbb{R}; \mathbb{Z}; +, \cdot; =)$

Examples (Semi-deciding the square roots)

Input $(x_1, \ldots, x_n) \in \mathbb{R}^\infty$.

1: if $I_1 = I_2$ then goto 2 else goto 1;

2: $Z_2 := -1$;

3: $Z_3 := 1$;

4: $Z_1 := Z_1 * Z_1$;

5: $Z_2 := Z_2 + Z_3$;

6: if $Z_1 = Z_2$ then goto 7 else goto 5;

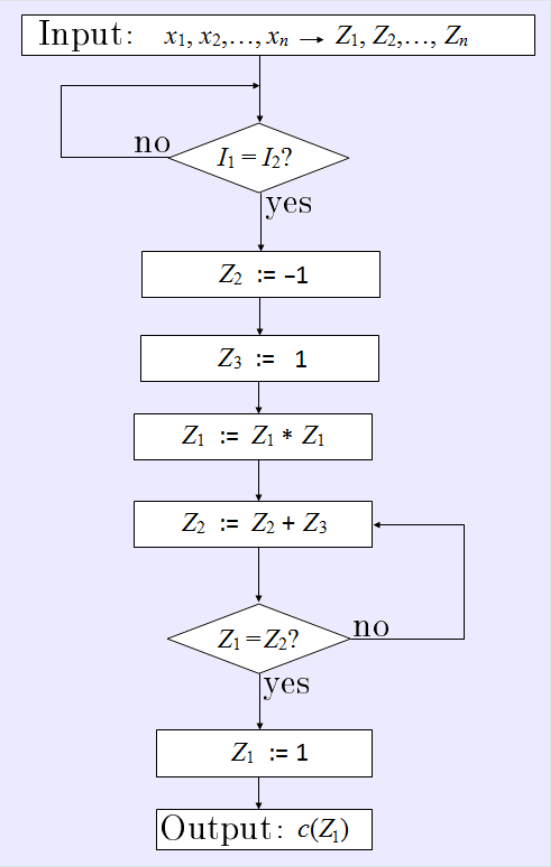
7: $Z_1 := 1$;

8: stop.

Output $c(Z_1)$.

Symbols: $+$ and $*$ for binary functions.
1 and -1 for a constant.

Infix notation $Z_1 + Z_2$ stands for $f_1^2(Z_1, Z_2)$ and so on.

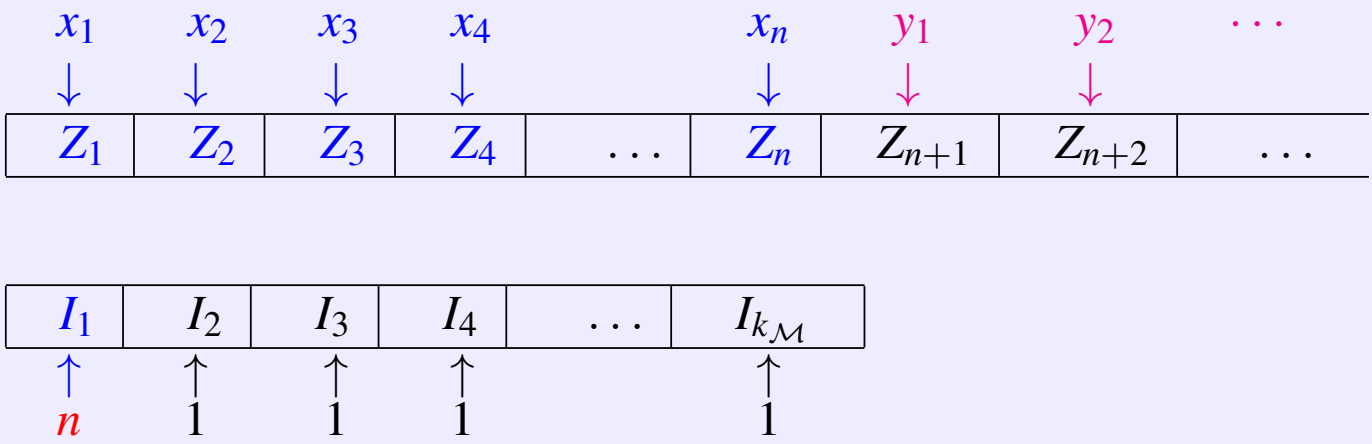


Uniform Computation over First-Order Structures

Input and output procedures for [digitally] non-deterministic BSS RAMs in $\mathrm{M}_{\mathcal{A}}^{[\mathrm{D}]\,\mathrm{ND}}$

Input and output space: $U_{\mathcal{A}}^{\infty} =_{\mathrm{df}} \bigcup_{i \geq 1} U_{\mathcal{A}}^{i}$

Input of $\vec{x} = (x_1, \ldots, x_n) \in U_{\mathcal{A}}^{\infty}$:



with guesses $y_1, \ldots, y_m \in U_{\mathcal{A}}$ or
"digits" $y_1, \ldots, y_m \in \{c_1, c_2\}$

Output after a stop-instruction:

$(c(Z_1), \ldots, c(Z_{c(I_1)}))$

Algorithms over First-Order Structures

σ-programs for [in]finite $\mathcal{A}$-machines $\mathcal{M}$

A σ-program can be interpreted by any structure of a finite or infinite suitable signature.

To simplify matters, let us consider

- a structure $\mathcal{A} = (U_{\mathcal{A}}; c_1, \ldots, c_{n_1}; f_1, \ldots, f_{n_2}; r_1, \ldots, r_{n_3})$
- a finite signature $\sigma = (n_1; m_1, \ldots, m_{n_2}; k_1, \ldots, k_{n_3})$
- the first-order symbols $c_1^0, \ldots, c_{n_1}^0, f_1^{m_1}, \ldots, f_{n_2}^{m_{n_2}}, r_1^{k_1}, \ldots, r_{n_3}^{k_{n_3}}$

Definition (σ-program $\mathcal{P}_{\mathcal{M}}$ of an $\mathcal{A}$-machine)

$1 :$ instruction$_1$; $\ldots$; $\ell_{\mathcal{P}_{\mathcal{M}}} - 1 :$ instruction$_{\ell_{\mathcal{P}_{\mathcal{M}}}-1}$; $\ell_{\mathcal{P}_{\mathcal{M}}} :$ stop.

- Any instruction$_i$ is a σ-instruction.
- The execution of $\mathcal{P}_{\mathcal{M}}$ is:
  the stepwise transformation of configurations
  by the transition system $(\mathrm{S}_{\mathcal{M}}, \rightarrow_{\mathcal{M}})$.

Algorithms over First-Order Structures

σ-instructions for infinite $\mathcal{A}$-machines $\mathcal{M}$ and transport instructions

Z_1	Z_2	Z_3	Z_4	Z_5	$\ldots$

Z-registers (for *individuals* in $U_{\mathcal{A}}$)

I_1	I_2	I_3	I_4	$\ldots$	$I_{k_{\mathcal{M}}}$

Index registers (for *indices* in $\mathbb{N}_+$)

Computation	$\ell\colon\ Z_j := f_i^{m_i}(Z_{j_1}, \ldots, Z_{j_{m_i}})$	(1)
	$\ell\colon\ Z_j := c_i^0$	(2)
Copy	$\ell\colon\ Z_{I_j} := Z_{I_k}$	(3)
Branching	$\ell\colon$ if $r_i^{k_i}(Z_{j_1}, \ldots, Z_{j_{k_i}})$ then goto ℓ_1 else goto ℓ_2	(4)
Index	$\ell\colon\ I_j := 1$	(5)
	$\ell\colon\ I_j := I_j + 1$	(6)
	$\ell\colon$ if $I_j = I_k$ then goto ℓ_1 else goto ℓ_2	(7)
Stop	$l\colon$ stop	(8)

The Z-registers contain individuals of the underlying structure. For copying these values from one Z-register into another Z-register, we will use index registers and index instructions.

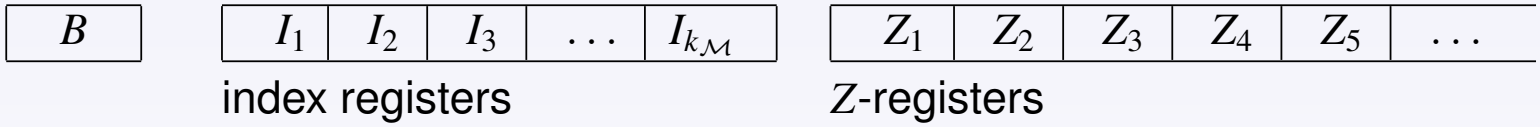
Overall States

Configurations of an $\mathcal{A}$-machine $\mathcal{M}$

Definition (Configurations in $S_{\mathcal{M}}$)

$$(\ell \,.\, \vec{\nu} \,.\, \bar{u}) \quad = \underbrace{(\ell, \nu_1, \ldots, \nu_{k_{\mathcal{M}}}, u_1, u_2, \ldots)}$$

a configuration of $\mathcal{M}$



index registers

Z-registers

pointers

read-and-write heads

tape

$\ell \in \mathcal{L}_{\mathcal{M}}$ $\quad \vec{\nu} = (\nu_1, \ldots, \nu_{k_{\mathcal{M}}}) \in \mathbb{N}_+^{k_{\mathcal{M}}}$ $\quad \bar{u} = (u_1, u_2, \ldots) \in U_{\mathcal{A}}^{\omega}$

- ℓ a label in $\mathcal{L}_{\mathcal{M}}$ — instruction counter, the internal state
- $\vec{\nu}$ indices of Z-registers — addresses, positions on the tape
- $\bar{u}$ a sequence of individuals

Unit Costs by a Computational System for $\mathcal{A}$-Machines

Transformation of configurations of an $\mathcal{A}$-machine $\mathcal{M}$

Definition (Transition system for any $\mathcal{A}$-machine $\mathcal{M}$)

$$\mathcal{S}_{\mathcal{M}} = (\mathsf{S}_{\mathcal{M}}, \rightarrow_{\mathcal{M}})$$

with a binary relation $\rightarrow_{\mathcal{M}} \subseteq \mathsf{S}^2_{\mathcal{M}}$

defined below.

$\rightarrow_{\mathcal{M}}$ depends on

- the types of the single instructions belonging to the program,
- the single parameters in these instructions, and
- the underlying structure.

(cf. Introduction 2020, . . .; see also Börger for finite machines, . . .)

Unit Costs of $\mathcal{A}$-machines

Transformation of configurations of an $\mathcal{A}$-machine $\mathcal{M}$

$\ell\colon\ Z_j := f_i^{m_i}(Z_{j_1}, \ldots, Z_{j_{m_i}})$

$$(\ell \,.\, \vec{\nu} \,.\, \bar{u}) \quad \to_{\mathcal{M}} \quad (\ell + 1 \,.\, \vec{\nu} \,.\, (u_1, \ldots, u_{j-1}, f_i(u_{j_1}, \ldots, u_{j_{m_i}}), u_{j+1}, \ldots))$$

$\ell\colon\ Z_j := c_i^0$

$$(\ell \,.\, \vec{\nu} \,.\, \bar{u}) \quad \to_{\mathcal{M}} \quad (\ell + 1 \,.\, \vec{\nu} \,.\, (u_1, \ldots, u_{j-1}, c_i, u_{j+1}, \ldots))$$

$\ell\colon\ Z_{I_j} := Z_{I_k}$

$$(\ell \,.\, \vec{\nu} \,.\, \bar{u}) \quad \to_{\mathcal{M}} \quad (\ell + 1 \,.\, \vec{\nu} \,.\, (u_1, \ldots, u_{\nu_j - 1}, u_{\nu_k}, u_{\nu_j + 1}, \ldots))$$

$\ell\colon$ if $r_i^{k_i}(Z_{j_1}, \ldots, Z_{j_{k_i}})$ then goto ℓ_1 else goto ℓ_2

$$\begin{array}{lll} (\ell \,.\, \vec{\nu} \,.\, \bar{u}) & \to_{\mathcal{M}} \quad (\ell_1 \,.\, \vec{\nu} \,.\, \bar{u}) & \text{if } (u_{j_1}, \ldots, u_{j_{k_i}}) \in r_i \\ (\ell \,.\, \vec{\nu} \,.\, \bar{u}) & \to_{\mathcal{M}} \quad (\ell_2 \,.\, \vec{\nu} \,.\, \bar{u}) & \text{if } (u_{j_1}, \ldots, u_{j_{k_i}}) \notin r_i \end{array}$$

Unit Costs of $\mathcal{A}$-machines

Transformation of configurations of an $\mathcal{A}$-machine $\mathcal{M}$

ℓ: if $I_j = I_k$ then goto ℓ_1 else goto ℓ_2

$$(\ell \,.\, \vec{\nu} \,.\, \bar{u}) \;\to_{\mathcal{M}}\; (\ell_1 \,.\, \vec{\nu} \,.\, \bar{u}) \qquad \text{if } \nu_j = \nu_k$$
$$(\ell \,.\, \vec{\nu} \,.\, \bar{u}) \;\to_{\mathcal{M}}\; (\ell_2 \,.\, \vec{\nu} \,.\, \bar{u}) \qquad \text{if } \nu_j \neq \nu_k$$

ℓ: $I_j := 1$

$$(\ell \,.\, \vec{\nu} \,.\, \bar{u}) \;\to_{\mathcal{M}}\; (\ell + 1 \,.\, (\nu_1, \ldots, \nu_{j-1}, 1, \nu_{j+1}, \ldots) \,.\, \bar{u})$$

ℓ: $I_j := I_j + 1$

$$(\ell \,.\, \vec{\nu} \,.\, \bar{u}) \;\to_{\mathcal{M}}\; (\ell + 1 \,.\, (\nu_1, \ldots, \nu_{j-1}, \nu_j + 1, \nu_{j+1}, \ldots) \,.\, \bar{u})$$

$\ell_{\mathcal{P}}$: stop

$$(\ell_{\mathcal{P}} \,.\, \vec{\nu} \,.\, \bar{u}) \;\to_{\mathcal{M}}\; (\ell_{\mathcal{P}} \,.\, \vec{\nu} \,.\, \bar{u})$$

Semi-Decidable Problems: Halting Sets

Some decision problems of BSS RAMs

Definition (Decision problems)

Any subset P of $U_{\mathcal{A}}^{\infty}$ (that contains all tuples) is a *decision problem*.

Definition (Halting process)

$\mathcal{M}$ *halts* on input $\vec{x}$ if $\ell_{\mathcal{P}}$: stop is reached [for some guesses].

$\mathcal{M}(\vec{x})\downarrow$ if $\mathcal{M}$ halts on input $\vec{x}$ after a finite number of steps [for some guesses].

$\mathcal{M}(\vec{x})\uparrow$ if $\mathcal{M}(\vec{x})\downarrow$ does not hold.

Definition (Halting set)

$H_{\mathcal{M}} = \{\vec{x} \in U_{\mathcal{A}}^{\infty} \mid \mathcal{M}(\vec{x})\downarrow\}$ is the *halting set* of $\mathcal{M}$.

$\vec{x} \in H_{\mathcal{M}} \Rightarrow$ $\mathcal{M}$ *accepts* $\vec{x}$.

$\mathrm{Res}_{\mathcal{M}}(\vec{x}) \in U_{\mathcal{A}}^{\infty}$ and $\emptyset \neq \mathrm{Res}_{\mathcal{M}}(\vec{x}) \subseteq U_{\mathcal{A}}^{\infty}$ (resp.)

$\mathrm{SDEC}_{\mathcal{A}} = \{H_{\mathcal{M}} \mid \mathcal{M} \in \mathsf{M}_{\mathcal{A}}\}$ $\quad$ $\mathrm{SDEC}_{\mathcal{A}}^{\mathrm{ND}} = \{H_{\mathcal{M}} \mid \mathcal{M} \in \mathsf{M}_{\mathcal{A}}^{\mathrm{ND}}\}$

Complete Problems: Halting Problems

Over first-order structures

Properties (Halting problems)

$Halt = \{\langle input, machine\rangle \subseteq U_{\mathcal{A}}^{\infty} \mid \langle machine\rangle \text{ halts on } \langle input\rangle\}$

$\Rightarrow$ It is useful to consider universal BSS RAMs.

Universal Machines of Type 1 and their Halting Sets

Halting problems

$$\mathcal{A} = (U_{\mathcal{A}}; (c_i)_{i \in N_1}; f_1, \ldots, f_{n_2}; r_1, \ldots, r_{n_3}) \qquad (\{1,2\} \subseteq N_1)$$

Type 1 here means encoding by using two constants

$$a = c_1,$$
$$b = c_2.$$

$$\mathrm{code}_{(a,b)}(\mathcal{M}) \in \{a,b\}^\infty \qquad \text{(for all BSS RAMs } \mathcal{M} \text{ over } \mathcal{A})$$

Definition (Halting problems of type 1)

$$\mathbb{H}_{\mathcal{A}} = \{(\mathrm{code}_{(a,b)}(\mathcal{M}) \,.\, \vec{x}) \in U_{\mathcal{A}}^\infty \mid \mathcal{M} \in \mathsf{M}_{\mathcal{A}} \quad \& \; \mathcal{M}(\vec{x}) \downarrow\}$$

$$\mathbb{H}_{\mathcal{A}}^{\mathrm{ND}} = \{(\mathrm{code}_{(a,b)}(\mathcal{M}) \,.\, \vec{x}) \in U_{\mathcal{A}}^\infty \mid \mathcal{M} \in \mathsf{M}_{\mathcal{A}}^{\mathrm{ND}} \quad \& \; \mathcal{M}(\vec{x}) \downarrow\}$$

for BSS RAMs over $\mathcal{A}$, in $\mathsf{M}_{\mathcal{A}}^{[\mathrm{ND}]}$, without guessing, with guessing

Simulation of $\mathcal{M}$ by a Universal Machine $\mathcal{M}_0$ (Type 1)

Computing $\mathrm{Res}_{\mathcal{M}} = \mathrm{Res}_{\mathcal{M}_0} \circ \mathrm{Res}_{\mathcal{N}_{\mathcal{M}}}$ for $\mathcal{A}$ with $\{1,2\} \subseteq N_1$

$\boxed{\vec{x}}$ $\Rightarrow_{\mathcal{N}_{\mathcal{M}}}$ $\boxed{(\mathrm{code}_{(a,b)}(\mathcal{M}) \,.\, \vec{x})}$

$\Downarrow \mathrm{Input}_{\mathcal{M}}$ $\Downarrow \mathrm{Input}_{\mathcal{M}_0}$

initial configuration for $\mathcal{M}$

$\boxed{(1 \,.\, (n, 1, \ldots, 1) \,.\, \vec{x} \,.\, (x_n, x_n, \ldots))}$

initial configuration for $\mathcal{M}_0$

$\boxed{(1 \,.\, (n_0, 1, \ldots, 1) \,.\, (\mathrm{code}_{(a,b)}(\mathcal{M}) \,.\, \vec{x} \,.\, (x_n, x_n, \ldots)))}$

with $n_0 = |(\mathrm{code}_{(a,b)}(\mathcal{M}) \,.\, \vec{x})|$

$\Downarrow$ The execution of $\mathcal{M}$ $\Downarrow$ The execution of $\mathcal{M}_0$

end configuration

$\boxed{(\ell_{\mathcal{P}_{\mathcal{M}}} \,.\, \vec{\nu} \,.\, \bar{u})}$

end configuration

$\boxed{(\ell_{\mathcal{P}_{\mathcal{M}_0}} \,.\, \vec{\mu} \,.\, \bar{z})}$

$\Downarrow \mathrm{Output}_{\mathcal{M}}$ $\Downarrow \mathrm{Output}_{\mathcal{M}_0}$

$\boxed{\mathrm{Res}_{\mathcal{M}}(\vec{x}) = (u_1, \ldots, u_{\nu_1})}$ $=$ $\boxed{(z_1, \ldots, z_{\mu_1}) = \mathrm{Res}_{\mathcal{M}_0}(\mathrm{code}_{(a,b)}(\mathcal{M}) \,.\, \vec{x})}$

Simulation of $\mathcal{M}$ by a Universal Machine $\mathcal{M}_0$ (Type 1)

The reduction of halting sets

Input of $\mathcal{M}$		Input of $\mathcal{M}_0$
$\vec{x}$	$\Rightarrow_{\mathcal{N}_\mathcal{M}}$	$(\mathrm{code}_{(a,b)}(\mathcal{M}) \,.\, \vec{x})$

$\Rightarrow$ $\mathcal{N}_\mathcal{M}$ computes the reduction of $H_\mathcal{M}$ to $H_{\mathcal{M}_0}$

for $\mathcal{M} \in \mathsf{M}_\mathcal{A}$ (in linear time).

$$
\begin{array}{rcl}
\vec{x} \in H_\mathcal{M} & \Rightarrow & (\mathrm{code}_{(a,b)}(\mathcal{M}) \,.\, \vec{x}) \in H_{\mathcal{M}_0} \\
\vec{x} \notin H_\mathcal{M} & \Rightarrow & (\mathrm{code}_{(a,b)}(\mathcal{M}) \,.\, \vec{x}) \notin H_{\mathcal{M}_0} \\
\mathcal{M} \text{ halts on } \vec{x} & \Leftrightarrow & \mathcal{M}_0 \text{ halts on } (\mathrm{code}_{(a,b)}(\mathcal{M}) \,.\, \vec{x})
\end{array}
$$

Note, we have also:

$\mathcal{N}_\mathcal{M}$ computes the reduction of $H_\mathcal{M}$ to $H_{\mathcal{M}_0^{\mathrm{ND}}}$ for $\mathcal{M} \in \mathsf{M}_\mathcal{A}^{\mathrm{ND}}$ (in linear time).

Encoding BSS RAMs $\mathcal{M}$ for their Simulation

Strings in $\{a,b\}^\kappa$ as codes for the σ-programs

Our instructions $\ell\colon\ Z_j := f_i^{m_i}(Z_{j_1},\ldots,Z_{j_{m_i}})$ (1), ... encoded by

$code_\lambda(\cdot)$ in $\{a,b\}^\lambda$ (codes for labels and indices)

$\Rightarrow (code_\lambda(\cdot)\,.\ code_\lambda(\cdot)\,.\ldots)$ in $\{a,b\}^\kappa$ (codes for instructions)

Type	e_1	e_2	e_3	e_4	e_5	e_6	e_7	$\cdots$	$e_{\ldots}$	Codes of length κ
(1)	1	i			j	j_1	j_2		j_{m_i}	$(code_\lambda(1)\,.\ code_\lambda(i)\,.\ldots)$
(2)	2				j					$(code_\lambda(2)\,.\ code_\lambda(0)\,.\ldots)$
(3)	3				j	k				
(4)	4	i	ℓ_1	ℓ_2		j_1	j_2		j_{k_i}	
(5)	5		ℓ_1	ℓ_2	j	k				
(6)	6				j					
(7)	7				j					
(8)	0									

Recall: $a = c_1$, $b = c_2$.

Are the codes (are c_1 and c_2) recognizable or distinguishable?

Subprograms for Evaluating Codes?

Subprograms for deciding codes?

The equality relation = can be used in executing a branching instruction of type (4) only if the relation = belongs to the underlying structure $\mathcal{A}$.

- A structure $\mathcal{A} = (U_\mathcal{A}; (c_i)_{i\in N_1}; f_1, \ldots, f_{n_2}; r_1, \ldots, r_{n_3})$ $\quad (\{1,2\} \subseteq N_1)$
- an expansion $\quad \mathcal{A}^+ = (U_\mathcal{A}; (c_i)_{i\in N_1}; f_1, \ldots, f_{n_2}; r_1, \ldots, r_{n_3}, =)$
- a new signature $\quad \sigma^+ = (n_1; m_1, \ldots, m_{n_2}; k_1, \ldots, k_{n_3}, 2)$
- new symbols (for the identity) $\quad =, \mathrm{id}, r^2_{n_3+1}$

Examples (Useful pseudo instructions for computation over $\mathcal{A}$?)

ℓ: if $\mathrm{id}(\mathrm{Z}_1, \mathrm{c}_1^0)$ then goto ℓ_1 else goto ℓ_2 (i)

ℓ: if $\mathrm{id}(\mathrm{Z}_1, \mathrm{c}_2^0)$ then goto ℓ_1 else goto ℓ_2 (ii)

id is a symbol for a binary relation. id could be interpreted, e.g., by the equality relation =.

If the equality relation is decidable over $\mathcal{A}$ by a subprogram, then both pseudo instructions can stand for subprograms allowing to decide $\{c_1\}$ and $\{c_2\}$.

Which properties must $\mathcal{A}$ possess in order to be able to evaluate the codes?

Subprograms for Evaluating Codes

Sufficient conditions for subprograms helping to decide codes

Definition (The classes $\mathrm{SDEC}_{\mathcal{A}}$ and $\mathrm{DEC}_{\mathcal{A}}$)

$$\mathrm{SDEC}_{\mathcal{A}} = \{H_{\mathcal{M}} \mid \mathcal{M} \in \mathsf{M}_{\mathcal{A}}\}$$
$$\mathrm{DEC}_{\mathcal{A}} = \{P \subseteq U_{\mathcal{A}}^{\infty} \mid P \in \mathrm{SDEC}_{\mathcal{A}} \ \& \ U_{\mathcal{A}}^{\infty} \setminus P \in \mathrm{SDEC}_{\mathcal{A}}\}$$

(a) $\mathcal{A}$ contains $\mathrm{id}_{U_{\mathcal{A}}}$,
(b) $\mathrm{id}_{U_{\mathcal{A}}} \in \mathrm{DEC}_{\mathcal{A}}$,
(c) $\mathrm{id}_{U_{\mathcal{A}}} \in \mathrm{SDEC}_{\mathcal{A}}$,
(d) $\{c_1\}, \{c_2\} \in \mathrm{SDEC}_{\mathcal{A}}$.

Theorem (Pseudo instructions for evaluating codes over $\mathcal{A}$)

Let one of conditions, (a) or (b) or (c) or (d), hold.

Then, $\{c_1\}, \{c_2\} \in \mathrm{DEC}_{\mathcal{A}}$ and

(i) and (ii) describe subprograms for deciding $\{c_1\}$ and $\{c_2\}$ over $\mathcal{A}$.

ℓ: if $\mathrm{id}(Z_1, c_1^0)$ then goto ℓ_1 else goto ℓ_2 (i)

ℓ: if $\mathrm{id}(Z_1, c_2^0)$ then goto ℓ_1 else goto ℓ_2 (ii)

More Background (for Evaluating Codes)
Identity, constants, and basic implications

Definition (The classes $\mathrm{SDEC}_{\mathcal{A}}$ and $\mathrm{DEC}_{\mathcal{A}}$)

$$\begin{aligned}\mathrm{SDEC}_{\mathcal{A}} &= \{H_{\mathcal{M}} \mid \mathcal{M} \in \mathsf{M}_{\mathcal{A}}\}\\ \mathrm{DEC}_{\mathcal{A}} &= \{P \subseteq U_{\mathcal{A}}^{\infty} \mid P \in \mathrm{SDEC}_{\mathcal{A}} \;\&\; U_{\mathcal{A}}^{\infty} \setminus P \in \mathrm{SDEC}_{\mathcal{A}}\}\end{aligned}$$

$$\begin{array}{lll}(a) & \mathcal{A} \text{ contains } & \mathrm{id}_{U_{\mathcal{A}}},\\ (b) & \mathrm{id}_{U_{\mathcal{A}}} & \in \mathrm{DEC}_{\mathcal{A}},\\ (c) & \mathrm{id}_{U_{\mathcal{A}}} & \in \mathrm{SDEC}_{\mathcal{A}},\\ (d) & \{c_1\}, \{c_2\} & \in \mathrm{SDEC}_{\mathcal{A}},\\ (e) & \{c_1, c_2\}^{\infty} & \in \mathrm{SDEC}_{\mathcal{A}}.\end{array}$$

Let $(a) \Rightarrow (b)$ mean (a) implies (b), ...

There hold

$$(a) \Rightarrow (b) \Rightarrow (c) \Rightarrow (d) \Rightarrow (e) \qquad \text{(cf. Part IIb)}$$

and there are structures with

$$(a) \not\Leftarrow (b) \quad \text{or} \quad (c) \not\Leftarrow (d) \quad \text{or} \quad (d) \not\Leftarrow (e). \qquad \text{(cf. Part IIb)}$$

What do we have, (b) $\not\Leftarrow$ (c) or (b) $\Leftarrow$ (c) ?

More Background (for Evaluating Codes)

Basic properties and some equivalences

Definition (The classes $\mathrm{SDEC}_{\mathcal{A}}$ and $\mathrm{DEC}_{\mathcal{A}}$)

$$\mathrm{SDEC}_{\mathcal{A}} = \{H_{\mathcal{M}} \mid \mathcal{M} \in \mathsf{M}_{\mathcal{A}}\}$$
$$\mathrm{DEC}_{\mathcal{A}} = \{P \subseteq U_{\mathcal{A}}^{\infty} \mid P \in \mathrm{SDEC}_{\mathcal{A}} \;\&\; U_{\mathcal{A}}^{\infty} \setminus P \in \mathrm{SDEC}_{\mathcal{A}}\}$$

$$\begin{array}{lll} (a) & \mathcal{A} \text{ contains} & \mathrm{id}_{U_{\mathcal{A}}}, \\ (b) & \mathrm{id}_{U_{\mathcal{A}}} & \in \mathrm{DEC}_{\mathcal{A}}, \\ (c) & \mathrm{id}_{U_{\mathcal{A}}} & \in \mathrm{SDEC}_{\mathcal{A}}, \\ (d) & \{c_1\}, \{c_2\} & \in \mathrm{SDEC}_{\mathcal{A}}, \\ (e) & \{c_1, c_2\}^{\infty} & \in \mathrm{SDEC}_{\mathcal{A}}. \end{array}$$

Let $(a) \Leftrightarrow (b)$ mean $(a) \Rightarrow (b)$ and $(b) \Rightarrow (a)$, ...

Then:

$$\begin{array}{llll} (c) & \Leftrightarrow & \mathrm{id}_{U_{\mathcal{A}}} & \in \mathrm{DEC}_{\mathcal{A}}. \\ (d) & \Leftrightarrow & \{c_1\}, \{c_2\} & \in \mathrm{DEC}_{\mathcal{A}}. \\ (e) & \Leftrightarrow & \{c_1, c_2\} & \in \mathrm{DEC}_{\mathcal{A}}. \end{array}$$

(cf. Part IIb) (cf. Part IIb) (cf. Part IIb)

Thus:

$$\boxed{(b) \Leftrightarrow (c)}$$
$$\Leftrightarrow \quad \chi_{\mathrm{id}_{U_{\mathcal{A}}}} \in \{\mathrm{Res}_{\mathcal{M}} \mid \mathcal{M} \in \mathsf{M}_{\mathcal{A}}\}.$$

(see below)

More: Decidability of P and Computability of χ_P

Over structures with two constants and $P \subseteq U_{\mathcal{A}}^{\infty}$

Definition (The characteristic function)

$$\chi_P : U_{\mathcal{A}}^{\infty} \to \{c_1, c_2\} \quad \textit{and} \quad \chi_P(\vec{x}) = \begin{cases} c_1 & \textit{if } \vec{x} \in P, \\ c_2 & \textit{if } \vec{x} \in U_{\mathcal{A}}^{\infty} \setminus P \end{cases}.$$

$$P \in \mathrm{DEC}_{\mathcal{A}} \quad \Rightarrow \quad \chi_P \in \{\mathrm{Res}_{\mathcal{M}} \mid \mathcal{M} \in \mathsf{M}_{\mathcal{A}}\}. \qquad \text{(cf. Part IIb)}$$

The computation of χ_P by an $\mathcal{M} \in \mathsf{M}_{\mathcal{A}}$ should mean that

- $\chi_P(\vec{x}) = c_1$ means $\vec{x} \in P$ and $\vec{x}$ is accepted by $\mathcal{M}$,
- $\chi_P(\vec{x}) = c_2$ means $\vec{x} \in U_{\mathcal{A}}^{\infty} \setminus P$ and $\vec{x}$ is rejected by $\mathcal{M}$.

More: Decidability of P and Computability of χ_P

Over structures with two constants and $P \subseteq U_{\mathcal{A}}^{\infty}$

Definition (The characteristic function)

$$\chi_P : U_{\mathcal{A}}^{\infty} \to \{c_1, c_2\} \quad \textit{and} \quad \chi_P(\vec{x}) = \begin{cases} c_1 & \textit{if } \vec{x} \in P, \\ c_2 & \textit{if } \vec{x} \in U_{\mathcal{A}}^{\infty} \setminus P \end{cases}.$$

$$P \in \mathrm{DEC}_{\mathcal{A}} \quad \Rightarrow \quad \chi_P \in \{\mathrm{Res}_{\mathcal{M}} \mid \mathcal{M} \in \mathsf{M}_{\mathcal{A}}\}. \qquad \text{(cf. Part IIb)}$$

What about with "$\Leftarrow$"?

$$\chi_P \in \{\mathrm{Res}_{\mathcal{M}} \mid \mathcal{M} \in \mathsf{M}_{\mathcal{A}}\} \text{ and } (d) \quad \Rightarrow \quad P \in \mathrm{DEC}_{\mathcal{A}}.$$

(for several proofs see Part IIb)

Recall: $(a) \Rightarrow (b) \Leftrightarrow (c) \Rightarrow (d)$

where

(a)	$\mathcal{A}$ contains	$\mathrm{id}_{U_{\mathcal{A}}}$,
(b)	$\mathrm{id}_{U_{\mathcal{A}}}$	$\in \mathrm{DEC}_{\mathcal{A}}$,
(c)	$\mathrm{id}_{U_{\mathcal{A}}}$	$\in \mathrm{SDEC}_{\mathcal{A}}$,
(d)	$\{c_1\}, \{c_2\}$	$\in \mathrm{SDEC}_{\mathcal{A}}$.
(e)	$\{c_1, c_2\}$	$\in \mathrm{SDEC}_{\mathcal{A}}$ is not sufficient.

Thanks

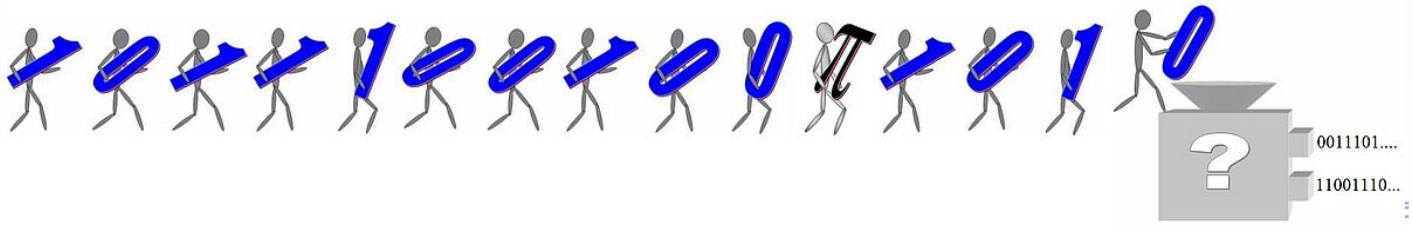
Thank you for your attention!

My thanks also go to the organizers.

I also thank
Patrick Steinbrunner and Sebastian Bierbaß,
Arno Pauly, Florian Steinberg,
Vasco Brattka, Philipp Schlicht, and Rupert Hölzl
for discussions.

My research was supported by many colleagues and
the International Office of the University Greifswald and the DVMLG.

I thank
Michael Schürmann, Volkmar Liebscher, Rainer Schimming,
Michael Rathjen, and Peter Schuster.



References
Incl. further references

- Gaßner *An introduction to a model of abstract computation: The BSS-RAM model*, Adrian Rezus (ed.): Contemporary Logic and Computing, College Publications [Landscapes in Logic 1], London 2020, pp. 574–603.
- Gaßner*Abstract computation over first-order structures. Part I: Deterministic and non-deterministic BSS RAMs*, arXiv:2502.17539.
- Gaßner*Abstract computation over first-order structures. .Part IIb: Moschovakis' operator and other non-determinisms*, arXiv:2507.03827.
- Lenore Blum, Michael Shub, and Steve Smale *On a theory of computation and complexity over the real numbers; NP-completeness, recursive functions and universal machines*, Bulletin of the American Mathematical Society 21 1989, pp. 1–46.
- Armin Hemmerling *Computability of string functions over algebraic structures*, Mathematical Logic Quarterly 44, 1998, pp. 1–44.
- Alan M. Turing *On computable numbers, with an application to the Entscheidungsproblem*, Proceedings of the London Mathematical Society 42 (1), 1937, pp. 230–265.