Abstract computation over first-order structures

From configurations to first-order formulas

Christine Gaßner

University of Greifswald

Würzburg 2026

Updated on October 12, 2026

(Three lines were changed.)

Some Essential Notions

In our talk about a generalization of computational models



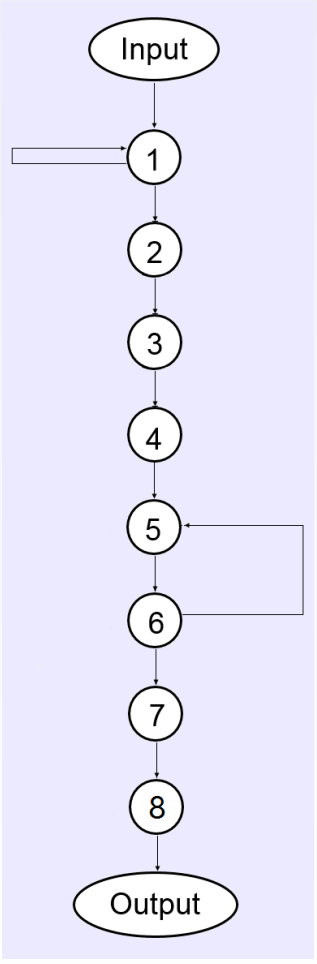
Unification (for a systematization)
Uniform (treatment of all inputs)
Unit costs (for executing an instruction)

Fixed time **Units**

A decision problem:
$P = \{x_1 \in \mathbb{R} \mid (\exists y \in \mathbb{N})(x_1^2 = y)\}$

The input, $\mathcal{P}$, and the output:

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 $c(Z_{c(I_1)})$.



A graph

Algorithms

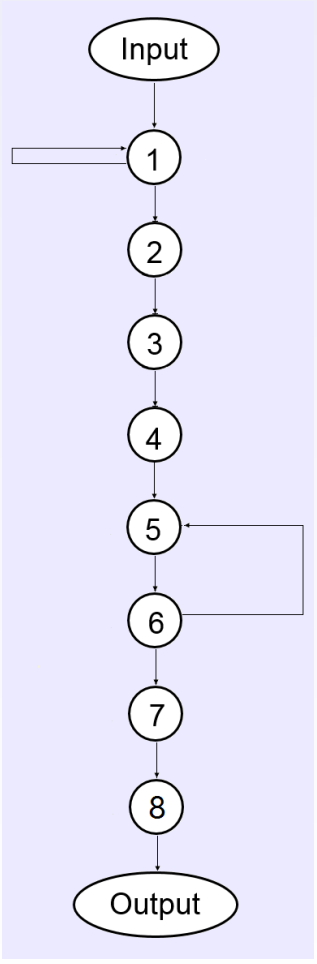
A program $\mathcal{P}$ and the flowchart $\mathfrak{Fl}_{\mathcal{P}}^{\text{label}}$

A decision problem:
$P = \{x_1 \in \mathbb{R} \mid (\exists y \in \mathbb{N})(x_1^2 = y)\}$

A $(2;2,2;2)$*-program* $\mathcal{P}$ *for computing* $\bar{\chi}_P$ *over* $(\mathbb{R};1,-1;+,\cdot\,;=)$

The input, $\mathcal{P}$*, and the output:*

Input $(x_1,\ldots,x_n) \in \mathbb{R}^\infty$.
1: if $I_1 = I_2$ then goto 2 else goto 1;
2: $Z_2 := c_2^0$;
3: $Z_3 := c_1^0$;
4: $Z_1 := f_2^2(Z_1, Z_1)$;
5: $Z_2 := f_1^2(Z_2, Z_3)$;
6: if $r_1^2(Z_1, Z_2)$ then goto 7 else goto 5;
7: $Z_1 := c_1^0$;
8: stop.
Output $c(Z_{c(I_1)})$.



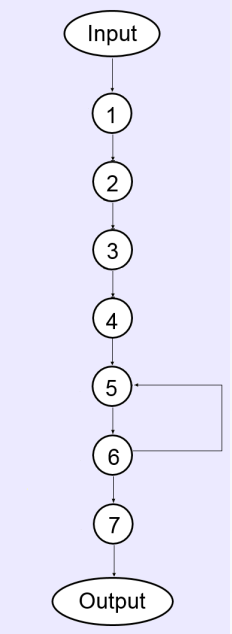
A graph

The decision problem $P = \{\vec{x} \in \mathbb{N}^\infty \mid x_1 \in 2\mathbb{N}+1\}$.

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

$\mathcal{P}$
1: $I_1 := 1$;
2: $Z_2 := c_3^0$;
3: $Z_1 := f_1^2(Z_1, Z_2)$;
4: $Z_3 := c_1^0$;
5: $Z_1 := f_2^2(Z_1, Z_2)$;
6: if $r_1^2(Z_1, Z_3)$ then goto 7 else goto 5;
7: stop.

Output $c(Z_{c(I_1)})$.



$\sigma = (3; 2, 2; 2)$, $\mathcal{P} \in \mathsf{P}_\sigma$, symbols: $c_1^0, c_2^0, c_3^0; f_1^2, f_2^2; r_1^2$.

Executing $\mathcal{P}$ over $\overbrace{(\mathbb{N}; 1, 0, 2; +, \dot{-}; \leq)}^{(U_\mathcal{A};\ c_1,c_2,c_3;\ f_1,\ f_2;\ r_1)}$ $\Rightarrow \chi_P$,

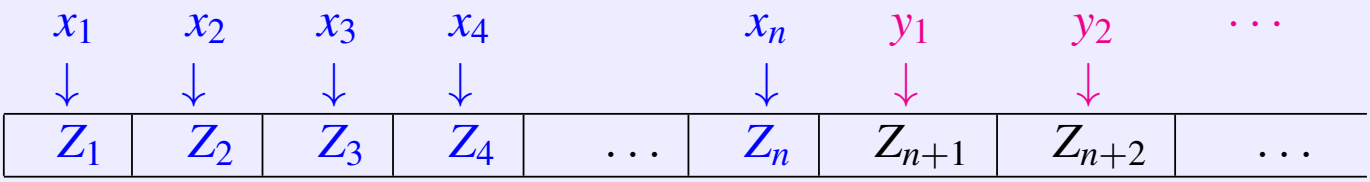
$(\mathbb{N}; 1, 0, 2; +, \dot{-}; =)$ $\Rightarrow \bar{\chi}_P$

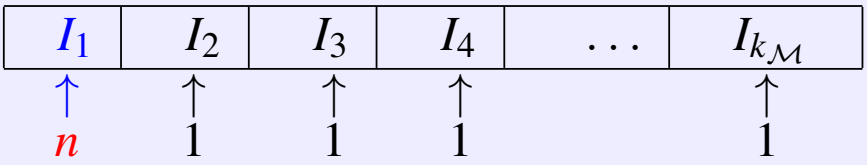
where $x \dot{-} y = \max\{0, x - y\}$ holds for all $x, y \in \mathbb{N}$.

Uniform Computation by BSS RAMs

Input procedures privide the length of the input

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 $y_1 = y_2 = \cdots = x_n$

↖ Any machine can recognize the end of an input.

Useful for

– Computing $\sum_{i=1}^{n} x_i$

– non-deterministically semi-deciding

$$\{\vec{x} \in \mathbb{R}^{\infty} \mid (\exists (y_1, \ldots, y_n) \in \{0,1\}^n)(\textstyle\sum_{i=1}^{n} y_i x_i = 0)\}$$

Label paths given by Configurations and Walks

Computation paths and program paths

Let $\ell_1 = 1, \quad \vec{\nu}^{(1)} = (n, 1, \ldots, 1), \quad \bar{u}^{(1)} = (x_1, \ldots, x_n, \ldots).$

Any sequence of configurations $(\ell_t \,.\, \vec{\nu}^{(t)} \,.\, \bar{u}^{(t)})_{t \geq 1}$ generated by $\rightarrow_{\mathcal{M}}$
determines
a computation path $(\ell_1, \ell_2, \ldots [, \ell_s])$
of a machine $\mathcal{M}$.

Each computation path of $\mathcal{M}$ corresponds to a walk in $\mathfrak{Fl}^{\text{label}}_{\mathcal{P}_{\mathcal{M}}}$.

Any walk $(\ell_1, (\ell_1, \ell_2), \ell_2, \ldots [, (\ell_{s-1}, \ell_s), \ell_s])$ in flowchart $\mathfrak{Fl}^{\text{label}}_{\mathcal{P}}$
determines
a program path $(\ell_1, \ell_2, \ldots [, \ell_s])$
of a program $\mathcal{P}$.

Any computation path traversed by $\mathcal{M}$ is a program path of $\mathcal{P}_{\mathcal{M}}$.

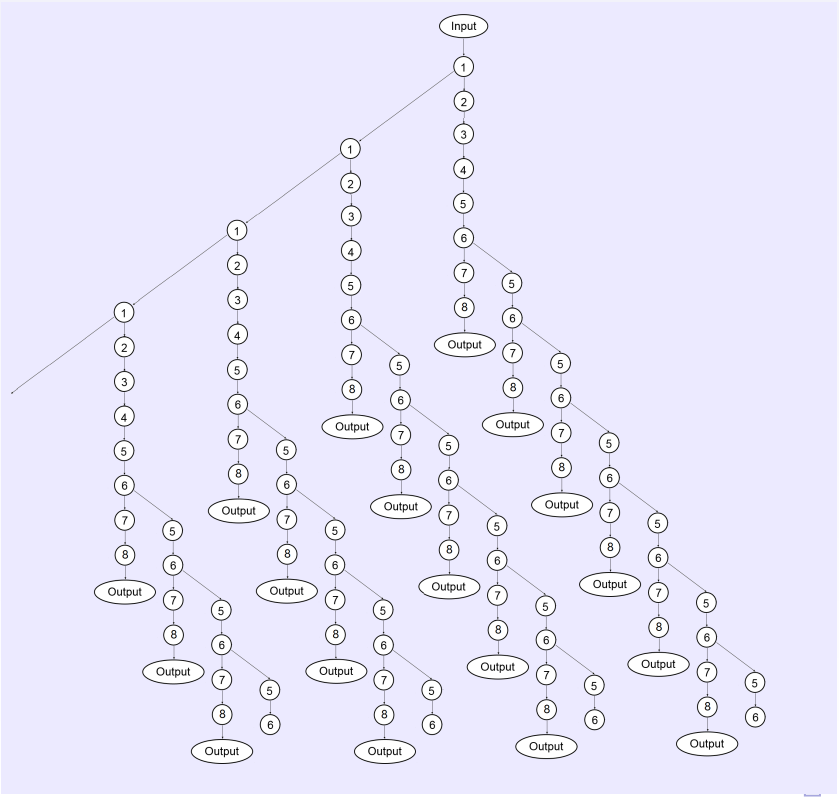
Which program paths are computation paths?

A Decision Tree and Program Paths

Background: Labeled graphs determined by walks (independent of $\mathcal{A}$)

For inputs of fixed length n, we have summaries of

- program paths with labels ℓ_t,
- sequences of partial configurations $(\ell_t \,.\, \vec{\nu}^{(t)})$,
- sequences of term-configurations $(\ell_t \,.\, \vec{\nu}^{(t)} \,.\, \bar{\tau}^{(t)})$.



Generating $(\mathcal{P}, \kappa, n)$-term-paths

The change of labels — conditions and σ-literals

$$\boxed{\ell\colon \text{ if } I_j = I_k \text{ then goto } l_1 \text{ else goto } l_2} \tag{7}$$

$$\begin{aligned} (\ell\,.\,\vec{\nu}\,.\,\bar{\tau}) &\to_{\mathcal{P},\kappa} (l_1\,.\,\vec{\nu}\,.\,\bar{\tau}) && \text{if } \nu_j = \nu_k \\ (\ell\,.\,\vec{\nu}\,.\,\bar{\tau}) &\to_{\mathcal{P},\kappa} (l_2\,.\,\vec{\nu}\,.\,\bar{\tau}) && \text{if } \nu_j \neq \nu_k \end{aligned}$$

Thus, we have **one** case (for a fixed n): either $\ell_{t+1} = l_1$ or $\ell_{t+1} = l_2$ if $\ell_t = \ell$.

$$\boxed{\ell\colon \text{ if } r_i^{k_i}(Z_{j_1}, \ldots, Z_{j_{k_i}}) \text{ then goto } l_1 \text{ else goto } l_2} \tag{4}$$

$$\begin{aligned} (\ell\,.\,\vec{\nu}\,.\,\bar{\tau}) &\to_{\mathcal{P},\kappa} (l_1\,.\,\vec{\nu}\,.\,\bar{\tau}) && \text{for } \overbrace{(\tau_{j_1}, \ldots, \tau_{j_{k_i}}) \in r_i}^{\sigma\text{-literals for branching}} \\ (\ell\,.\,\vec{\nu}\,.\,\bar{\tau}) &\to_{\mathcal{P},\kappa} (l_2\,.\,\vec{\nu}\,.\,\bar{\tau}) && \text{for } (\tau_{j_1}, \ldots, \tau_{j_{k_i}}) \notin r_i \end{aligned}$$

Thus, **two** cases (i.e. branching):

1) $\ell_{t+1} = l_1$ and
2) $\ell_{t+1} = l_2$ if $\ell_t = \ell$.

Systems of σ-literals

The evaluation of label paths

A program path B^* of a σ-program $\mathcal{P}$

- is syntactically derived from $\mathcal{P}$,
- **can be described by systems $\mathsf{sy}^{\mathsf{decision}}_{\mathcal{P},n}(B^*)$ of σ-literals,**
- is a computation path if $\mathsf{sy}^{\mathsf{decision}}_{\mathcal{P},n}(B^*)$ is satisfiable for some $n \geq 1$.

A computation path B^* of an $\mathcal{A}$-machine $\mathcal{M}$

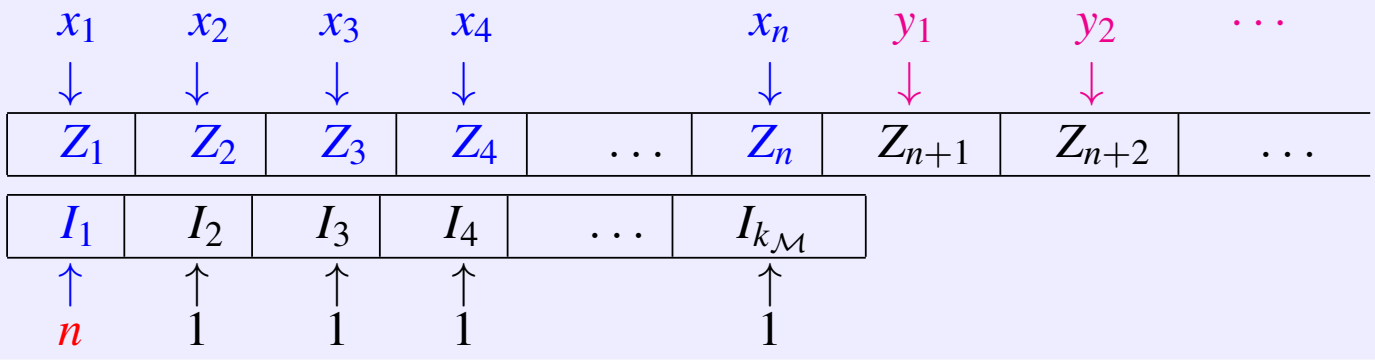
- is semantically determined by a sequence of configurations of $\mathcal{M}$,
- belongs to the set of program paths of $\mathcal{P}_{\mathcal{M}}$,
- **implies that $\mathsf{sy}^{\mathsf{decision}}_{\mathcal{P}_{\mathcal{M}},n}(B^*)$ is satisfiable over $\mathcal{A}$ for some $n \geq 1$.**

Program paths of $\mathcal{P}_{\mathcal{M}}$ are evaluated to recognize whether a considered program path is a computation path of $\mathcal{M}$.

Necessary: Separation of Indices and Z-registers

BSS machines (the main idea: 1989, pp. 12, 25) vs. BSS RAMs over $\mathbb{R}$

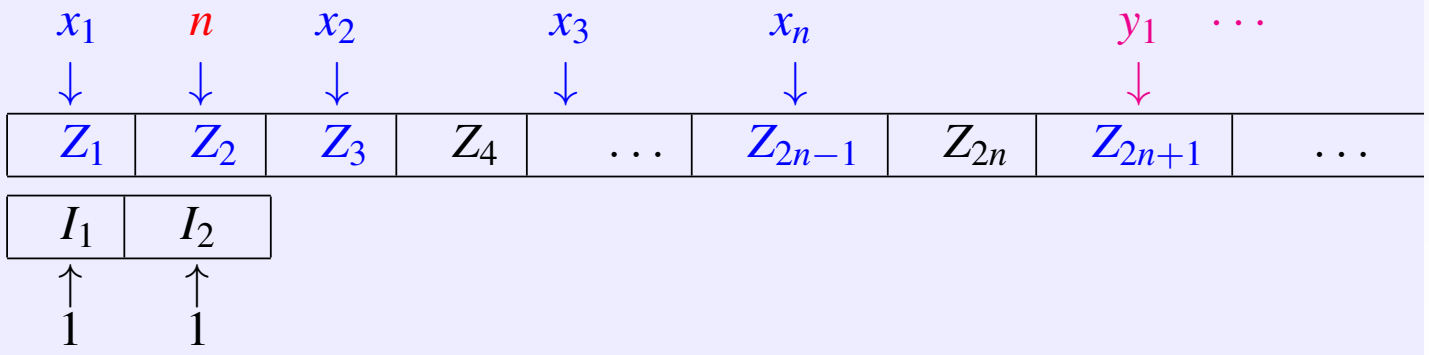
BSS RAM and the input of $(x_1, \ldots, x_n) \in \mathbb{R}^\infty$ — det. or non-det.



Real RAM works in general on inputs $\vec{x} \in \mathbb{R}^n$ — i.g. det.

(of fixed length n).

BSS machine and an input of $(x_1, \ldots, x_n, y_1, \ldots, y_m) \in \mathbb{R}^\infty$ — det.



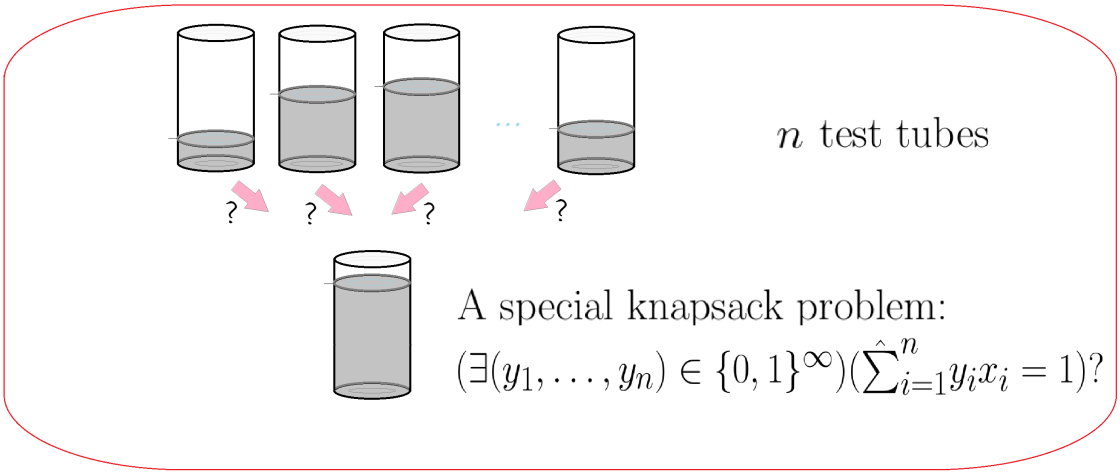
Reasons for our Generalization

Our motivation

1. A question:
   - Are there structures $\mathcal{A}$ with $\mathrm{P}_{\mathcal{A}} = \mathrm{NP}_{\mathcal{A}}$? (Poizat 1995)
2. The model useful for better understanding
   - the classical P-NP problem,
   - notions such as the *semi-decidability*,
   - the notion of *algorithm*.
3. We need
   - more systematization,
   - a classification of the models used in computer science.
4. We should
   - extend our consideration beyond the today's computer,
   - take into account the meaning of the term *algorithm* in the 20th century.
5. Benefits for the practical computer science:
   - First-order structures can represent classes in object-oriented programming (without current implementations),
   - BSS RAMs can used for describing the logical programming, simulating neural networks, and so on.

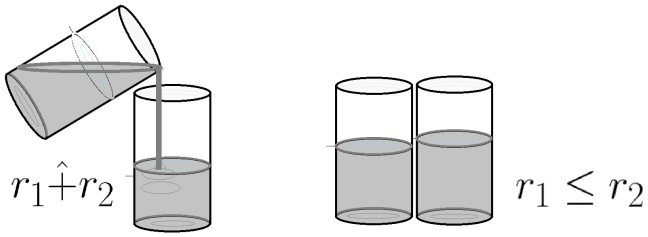
Reasons for our Generalization

Our motivation: the original meaning of the term algorithm (in the 20th century)



A $([0,1];0,1;\hat{+};\leq)$-machine realized by water and test tubes.

$r_1 \hat{+} r_2 =_{\mathrm{df}} \min\{r_1 + r_2, 1\}$



A further example:

T. Zerjatke: Möglichkeiten und Grenzen des DNA-Computings im Hinblick auf PSPACE

https://math-inf.uni-greifswald.de/storages/uni-greifswald/fakultaet/mnf/mathinf/gassner/ZerjatkeDiplomarbeit.pdf

Thanks

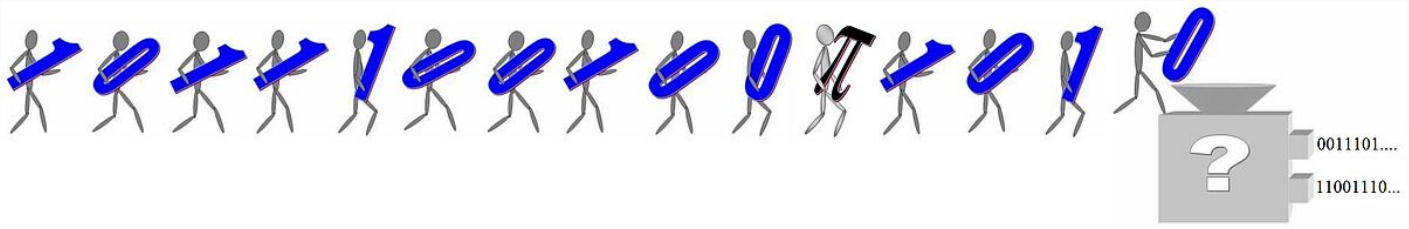
Thank you for your attention!

My thanks also go to the organizers.

I also thank
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

For more references see:


C. G. *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.


C. G. *Abstract computation over first-order structures. Part I: Deterministic and non-deterministic BSS RAMs*, arXiv:2502.17539.


C. G. *Abstract computation over first-order structures. Extras: From programs to decision trees I*, arXiv:2608.20424.

C · H · T