
Algorithmic definability of algebraic structures

Igor Ashaev
Omsk State University

Generalized algorithms

Let $\mathcal{A} = \langle A; \sigma \rangle$ be an arbitrary algebraic structure. A notion of generalized algorithm:

- ▶ works with elements of the structure \mathcal{A} ,
- ▶ uses operations, predicates and constants from the signature σ as primitives,
- ▶ extends the classical notion of algorithm, in particular, for the structure $\langle \omega, +, \times, 0, 1 \rangle$ defines usual computable functions,
- ▶ be algebraic (does not use topology, limits etc).

Different approaches to define computability

It is known two approaches:

- ▶ with the help of definability by Σ -formulas of some logical language
- ▶ with the help of abstract computing devices

Σ -definability

J. Barwise and Yu. Ershov investigated definability in admissible structures.

- ▶ J. Barwise. *Admissible sets and structures*. – Springer-Verlag, Berlin-Heidelberg-New York, 1975.
- ▶ Yu.L. Ershov. *Definability and computability*. – New York: Plenum, 1996 (Siberian School of Algebra and Logic)

The simplest admissible structure can be constructed as follows. Let $\mathcal{A} = \langle A; \sigma \rangle$ be a structure. Define $HF(A)$ as a set which contains all elements of A , the empty set, all finite sets of elements of A , finite sets of sets and so on. A structure $\langle HF(A), \sigma, \in, \emptyset \rangle$ is a hereditarily finite sets superstructure.

Computability in a structure $\mathcal{A} \iff$ definability by Σ -formulas in $HF(\mathcal{A})$.

Machine computability

Example: generalized random access machine or equivalent constructions (many authors uses different modifications: H. Friedman, Tucker & Zucker, Blum & Shub & Smale, Ashaev & Belyaev & Myasnikov).

- ▶ H. Friedman. *Algorithmic procedures, generalized Turing algorithms, and elementary recursion theory* // Logic Colloquium'69 (R.O. Gandy and C.E.M. Yates eds), – North Holland, 1971, P. 361–390.
- ▶ H. Friedman, R. Mansfield. *Algorithmic procedures* // Transactions of the AMS. V.332. N.1. July 1992. P.297–312.
- ▶ J.V. Tucker, J.I. Zucker. *Projections of semicomputable relations on abstract data types* // International Journal of Foundations of Computer Science. – 2, 1991. – P. 267-296.

- ▶ L. Blum, M. Shub, S. Smale. *On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines* // Bull. Amer. Math. Soc. 1989. V.21. N1. P.1–46.
- ▶ L. Blum, F. Cucker, M. Shub, S. Smale. *Complexity and real computations*. – Springer Verlag: New York et al, 1998.
- ▶ I.V. Ashaev, V.Ya. Belyaev, A.G. Myasnikov. *Approaches to the theory of generalized computability* // Algebra i logika. 32. N 4 (1993). P. 349–386.
- ▶ A. Hemmerling. *Computability of string functions over algebraic structures* // Math. Logic Quarterly. – 44, 1998, nr.1. – P. 1-44.
- ▶ A. Hemmerling. *On approximate and algebraic computability over the real numbers* // Theoretical Computer Science. – 219 (1999). – P.185-223.

Machines for algebraic structures

Random access machine (RAM) for a structure \mathcal{A} consists of

- ▶ a finite set of registers, each of which can contain an element of the structure \mathcal{A} (denote them by variables),
- ▶ a finite sequence of commands as a program. Possible types of commands are:
 - ▶ Assumption of a constant: $y := c$, $c \in \sigma$ is a constant symbol,
 - ▶ Copy assumption: $y := x$,
 - ▶ Calculation of operation: $y := f(\bar{x})$, $\bar{x} = (x_1, \dots, x_n)$, $f^{(n)} \in \sigma$ is a functional symbol,
 - ▶ Conditional jump: *IF* $P(\bar{x})$ *GOTO* k , $P^{(n)} \in \sigma$ is a predicate symbol or equality relation,
 - ▶ Halting command: *HALT*.

Some obstacles

There are examples of structures and intuitively computable functions which are not RAM-computable.

Reasons:

- ▶ an arbitrary structure has no natural numbers, so we cannot to count or enumerate elements and so on,
- ▶ RAM has bounded memory while we need to store fine sets and sequences of arbitrary length.

So we need to extend our definition of the machine. There are different but equivalent approaches.

- ▶ string computability (Blum & Shub & Smale for rings and fields, Hemmerling general case),
- ▶ machines with stacks and counters,
- ▶ effective definitional schemes (Friedman),
- ▶ machines over the list superstructure (Ashaev & Belyaev & Myasnikov).

List superstructure

For a set A construct a set $HL(A)$ which consists of all elements of A , all lists (finite sequences) of elements, lists of lists and so on.

Examples:

$$a, b, (a, b \in A), \langle a, b \rangle, \langle \langle a \rangle \rangle, \langle a, \langle a, b \rangle, a \rangle, \langle \rangle.$$

Extend the signature σ with new constant nil (empty list) and operations $head(x)$ (get the first element of a list), $tail(x)$ (remove the first element of x), $cons(x, y)$ (include y as the new last elements to a list x).

The structure

$$HL(\mathcal{A}) = \langle HL(A); \sigma, nil, head, tail, cons \rangle$$

is the *list superstructure* over \mathcal{A} (Goncharov, Sviridenko).

Computability over the list superstructure

A function $f : A^n \rightarrow A$ is computable over \mathcal{A} iff it can be computed by a RAM over the superstructure $HL(\mathcal{A})$ [Ashaev, Belyaev, Myasnikov].

Now we have unbounded memory and natural numbers, since we can interpret

$$0 = \langle \rangle, n + 1 = \langle 0, \dots, n \rangle = \text{cons}(n, n).$$

A set $X \subseteq A^n$ is computable (recursive) iff its characteristic function $\chi(\bar{x}) = \begin{cases} 1, & \bar{x} \in X, \\ 0, & \bar{x} \notin X \end{cases}$ is computable.

Example: a set $\{(a_1, \dots, a_n) \mid t(\bar{a}) = s(\bar{a})\}$, t, s are fixed terms is computable

Two analogs of a notion of computably enumerable sets

- ▶ A set $X \subseteq A$ is halting iff there exists a computable function $f : A \rightarrow A$ such that $X = \text{dom } f$.
- ▶ A set $X \subseteq A$ is output iff there exists a computable function $f : A^n \rightarrow A$ such that $X = \text{rng } f$.

It is easy to construct many examples of structures which have different classes halting and output sets.

Theorem [ABM]. For the following types of structures both these classes coincides:

- 1) structures with quantifier elimination,
- 2) f.g. structures which signature contains generators.

Results

The following analogs of the classical results can be proved:

- ▶ Post theorem
- ▶ Graph theorem
- ▶ Projection theorem
- ▶ Universal function (for finite signature)
- ▶ Fixed-point theorem
- ▶ m -, 1- and T -reducibilities, Post problem solution
- ▶ Analogs of arithmetical hierarchy

Review:

I.V. Ashaev. *Computability over algebraic structures* //
Communications of OmSU. Special issue (2008). – P.107-129.

Logical description

We use formulas with infinite (countable) disjunctions.

A class of Σ^0 -formulas is a set constructed as follows:

- ▶ any quantifier-free first order logic formula belongs to Σ^0 ,
- ▶ if φ and ψ are Σ^0 -formulas then $\varphi \wedge \psi$ and $\varphi \vee \psi$ are too,
- ▶ if $\{\varphi_i \mid i \in I\}$ is a computably enumerable set of Σ^0 -formulas, then its disjunction $\bigvee_{i \in I} \varphi_i$ is a Σ^0 -formula.

Theorem [Ashaev, Belyaev, Myasnikov].

A set X is halting in a structure \mathcal{A} iff it is Σ^0 -definable in \mathcal{A} .

Graphs of computable functions

Graph theorem [ABM]. Let $f : A^n \rightarrow A$ be a computable function, $\Gamma_f = \{(a_1, \dots, a_n, b) \mid b = f(a_1, \dots, a_n)\}$ be its graph. Then Γ_f is a halting set. This statement cannot be inverted for arbitrary algebraic structure.

A Σ^0 -formula is said to be functional iff it has a form $\bigvee_{i \in I} \varphi_i$,

$$\varphi_i = \varphi_{i,1}(\bar{x}) \wedge \dots \wedge \varphi_{i,n} \wedge y = t_i(\bar{x})$$

where $\varphi_{i,j}$ are atomic or negations of atomic, t_i are terms. Different disjunction members φ_i, φ_k must contain contrary pair of formulas $\varphi_{i,j}, \varphi_{k,l} = \neg\varphi_{i,j}$.

Theorem [Ashaev]. A function f is computable iff it is definable by a functional Σ^0 -formula.

Σ^0 -definability of algebraic structures

Yu. Ershov: Σ -definability (with the help of Σ -formulas in admissible structures). It does not preserve machine computability.

I. Ashaev, G. Tjumentceva: Σ^0 -definability (a code consists of Σ^0 -formulas).

A code of a signature σ in a signature τ consists of the following Σ^0 -formulas and terms of the signature τ :

- ▶ a pair of formulas $\Phi_U(\bar{x}), \bar{\Phi}_U(\bar{x}), \bar{x} = (x_1, \dots, x_n)$,
- ▶ a pair of formulas $\Phi_E(\bar{x}, \bar{y}), \bar{\Phi}_E(\bar{x}, \bar{y})$,
- ▶ for any $P^{(k)} \in \sigma$ a pair of formulas $\Phi_P(\bar{x}_1, \dots, \bar{x}_k), \bar{\Phi}_P(\bar{x}_1, \dots, \bar{x}_k)$,
- ▶ for any $f^{(k)} \in \sigma$ a functional disjunction $\Phi_f(\bar{x}_1, \dots, \bar{x}_k, \bar{y})$,
- ▶ for any $c \in \sigma$ a tuple of closed terms (t_1, \dots, t_n) .

A code C can be applied to a structure $\mathcal{B} = \langle B, \tau \rangle$ iff

- ▶ Φ_U defines non-empty set $U \subseteq B^n$, $\bar{\Phi}_U$ defines its complement $B^n \setminus U$,
- ▶ Φ_E defines an equivalence relation E on U , $\bar{\Phi}_E$ defines its complement,
- ▶ Φ_P ($\bar{\Phi}_P$) defines a relation P_C (its complement) on U ,
- ▶ Φ_f defines an operation f_C on U ,
- ▶ (t_1, \dots, t_n) defines an element $d_C \in U$,
- ▶ the equivalence E is a congruence relative to all P_C and f_C .

A structure $\mathcal{A} = \langle A, \sigma \rangle$ is Σ^0 -definable in $\mathcal{B} = \langle B, \tau \rangle$ iff there exists a code C of the signature σ in τ which can be applied to \mathcal{B} and \mathcal{A} is isomorphic to $\langle U; P_C, f_C, d_C \rangle / E$.

Translations from \mathcal{A} to \mathcal{B} :

- ▶ computable functions,
- ▶ computable, halting and output sets,
- ▶ Turing degrees,
- ▶ classes of arithmetical hierarchy.

Thanks