The object language
Inductive definitions

András Máté

10.03.2023

On alphabets

On alphabets

If our alphabet consists of the first two numerals, then we should write $\mathcal{A} = \{‘0’, ‘1’\}$. We may have two „ideologies” for omitting the quotation marks in such cases:

On alphabets

If our alphabet consists of the first two numerals, then we should write $\mathcal{A} = \{\text{‘0’}, \text{‘1’}\}$. We may have two „ideologies” for omitting the quotation marks in such cases:

- We use letters *autonymously*, i. e. as their own names.

On alphabets

If our alphabet consists of the first two numerals, then we should write $\mathcal{A} = \{\text{‘0’}, \text{‘1’}\}$. We may have two „ideologies” for omitting the quotation marks in such cases:

- We use letters *autonymously*, i. e. as their own names.
- We don’t need to know what the letters are; it is enough that we have names for them.

On alphabets

If our alphabet consists of the first two numerals, then we should write $\mathcal{A} = \{$‘0’, ‘1’$\}$. We may have two „ideologies” for omitting the quotation marks in such cases:

- We use letters *autonymously*, i. e. as their own names.
- We don’t need to know what the letters are; it is enough that we have names for them.

Consider first the one-letter alphabet

$$\mathcal{A}_0 = \{\alpha\}$$

This is sufficient to name the natural numbers.

On alphabets

If our alphabet consists of the first two numerals, then we should write $\mathcal{A} = \{\text{`0'}, \text{`1'}\}$. We may have two „ideologies" for omitting the quotation marks in such cases:

- We use letters *autonymously*, i. e. as their own names.
- We don't need to know what the letters are; it is enough that we have names for them.

Consider first the one-letter alphabet

$$\mathcal{A}_0 = \{\alpha\}$$

This is sufficient to name the natural numbers.
The two-letter alphabet

$$\mathcal{A}_1 = \{\alpha, \beta\}$$

is sufficient for anything.

On alphabets

If our alphabet consists of the first two numerals, then we should write $\mathcal{A} = \{\text{‘0’}, \text{‘1’}\}$. We may have two „ideologies" for omitting the quotation marks in such cases:

- We use letters *autonymously*, i. e. as their own names.
- We don't need to know what the letters are; it is enough that we have names for them.

Consider first the one-letter alphabet

$$\mathcal{A}_0 = \{\alpha\}$$

This is sufficient to name the natural numbers.
The two-letter alphabet

$$\mathcal{A}_1 = \{\alpha, \beta\}$$

is sufficient for anything.
I. e., any language over some finite alphabet can be simulated by $\mathcal{A}_1$.

Inductive definitions of string classes

Inductive definitions of string classes

Let $\mathcal{C}$ be a finite alphabet and $\mathcal{C}^\circ$ the class of the strings over it. The inductive definition of an F subclass of $\mathcal{C}^\circ$ consists of the following three components:

Inductive definitions of string classes

Let $\mathcal{C}$ be a finite alphabet and $\mathcal{C}^\circ$ the class of the strings over it. The inductive definition of an F subclass of $\mathcal{C}^\circ$ consists of the following three components:

- Base of the induction: a class $B \subseteq \mathcal{C}^\circ$ given by some definition. We stipulate that $B \subseteq F$.

Inductive definitions of string classes

Let $\mathcal{C}$ be a finite alphabet and $\mathcal{C}^\circ$ the class of the strings over it. The inductive definition of an F subclass of $\mathcal{C}^\circ$ consists of the following three components:

- Base of the induction: a class $B \subseteq \mathcal{C}^\circ$ given by some definition. We stipulate that $B \subseteq F$.
- Inductive rules: a finite collection of stipulations of the form

$$\ulcorner a_1, a_2, \ldots, a_n \in F \Rightarrow b \in F \urcorner$$

where $a_1, \ldots a_n, b$ are strings over an alphabet $\mathcal{C} \cup \mathcal{V}$. (The members of $\mathcal{V}$ are understood as variables over $\mathcal{C}^\circ$.)

Inductive definitions of string classes

Let $\mathcal{C}$ be a finite alphabet and $\mathcal{C}^\circ$ the class of the strings over it. The inductive definition of an F subclass of $\mathcal{C}^\circ$ consists of the following three components:

- Base of the induction: a class $B \subseteq \mathcal{C}^\circ$ given by some definition. We stipulate that $B \subseteq F$.
- Inductive rules: a finite collection of stipulations of the form

$$\ulcorner a_1, a_2, \ldots, a_n \in F \Rightarrow b \in F \urcorner$$

  where $a_1, \ldots a_n, b$ are strings over an alphabet $\mathcal{C} \cup \mathcal{V}$. (The members of $\mathcal{V}$ are understood as variables over $\mathcal{C}^\circ$.)
- Closure: the members of F are just the strings produced from the basis by finitely many applications of the inductive rules.

Inductive definitions of string classes

Let $\mathcal{C}$ be a finite alphabet and $\mathcal{C}^\circ$ the class of the strings over it. The inductive definition of an F subclass of $\mathcal{C}^\circ$ consists of the following three components:

- Base of the induction: a class $B \subseteq \mathcal{C}^\circ$ given by some definition. We stipulate that $B \subseteq F$.
- Inductive rules: a finite collection of stipulations of the form

  $$\ulcorner a_1, a_2, \ldots, a_n \in F \Rightarrow b \in F \urcorner$$

  where $a_1, \ldots a_n, b$ are strings over an alphabet $\mathcal{C} \cup \mathcal{V}$. (The members of $\mathcal{V}$ are understood as variables over $\mathcal{C}^\circ$.)
- Closure: the members of F are just the strings produced from the basis by finitely many applications of the inductive rules.

We assume that the closure condition works (and we don't mention it any more).

Inductive classes and some notation

Inductive classes and some notation

Inductive classes (of strings over an alphabet $\mathcal{C}$) are those subclasses of $\mathcal{C}^{\circ}$ which have an inductive definition.

Inductive classes and some notation

Inductive classes (of strings over an alphabet $\mathcal{C}$) are those subclasses of $\mathcal{C}^\circ$ which have an inductive definition.
$\mathcal{C}^\circ$ is an inductive class itself (trivial).

Inductive classes and some notation

Inductive classes (of strings over an alphabet $\mathcal{C}$) are those subclasses of $\mathcal{C}^\circ$ which have an inductive definition.

$\mathcal{C}^\circ$ is an inductive class itself (trivial).

Our rules have the form

$$\ulcorner a_1, a_2, \ldots, a_n \in F \Rightarrow b \in F \urcorner$$

We can write them equivalently as

$$\ulcorner a_1 \in F \Rightarrow a_2 \in F \Rightarrow \ldots \Rightarrow a_n \in F \Rightarrow b \in F \urcorner$$

Inductive classes and some notation

Inductive classes (of strings over an alphabet $\mathcal{C}$) are those subclasses of $\mathcal{C}^\circ$ which have an inductive definition.
$\mathcal{C}^\circ$ is an inductive class itself (trivial).
Our rules have the form

$$\ulcorner a_1, a_2, \ldots, a_n \in F \Rightarrow b \in F \urcorner$$

We can write them equivalently as

$$\ulcorner a_1 \in F \Rightarrow a_2 \in F \Rightarrow \ldots \Rightarrow a_n \in F \Rightarrow b \in F \urcorner$$

Let us conventionally omit the reference to the class to be defined $\ulcorner \in F \urcorner$ and remember to this by using $\rightarrow$ instead of $\Rightarrow$.
So our rules have now the form

$$\ulcorner a_1 \rightarrow a_2 \rightarrow \ldots \rightarrow a_n \rightarrow b \urcorner$$

An example: numbers divisible by 3, in dyadic notation

An example: numbers divisible by 3, in dyadic notation

Let us use the alphabet $\mathcal{A}_d = \{0,\ 1\}$. The strings are 0-1 sequences including the dyadic numerals.

An example: numbers divisible by 3, in dyadic notation

Let us use the alphabet $\mathcal{A}_d = \{0,\ 1\}$. The strings are 0-1 sequences including the dyadic numerals.
The first numbers divisible by 3 are $0,\ 11$ and 110. They will give the base for our definition. We can put them into the definition as input-free rules.

An example: numbers divisible by 3, in dyadic notation

Let us use the alphabet $\mathcal{A}_d = \{0,\ 1\}$. The strings are 0-1 sequences including the dyadic numerals.

The first numbers divisible by 3 are $0,\ 11$ and 110. They will give the base for our definition. We can put them into the definition as input-free rules.

If a number is divisible by 3 and its numeral ends with 00, (so the numeral is of the form $x00$), then the next number divisible by 3 will be $x11$. As a formal rule,

$$x00 \to x11$$

An example: numbers divisible by 3, in dyadic notation

Let us use the alphabet $\mathcal{A}_d = \{0,\ 1\}$. The strings are 0-1 sequences including the dyadic numerals.
The first numbers divisible by 3 are $0,\ 11$ and 110. They will give the base for our definition. We can put them into the definition as input-free rules.
If a number is divisible by 3 and its numeral ends with 00, (so the numeral is of the form $x00$), then the next number divisible by 3 will be $x11$. As a formal rule,

$$x00 \to x11$$

If our number is $x01$, then the next number divisible by 3 will be $y00$, where y is the *follower* of x. We most now encode the relation of following in the rule. We use an auxiliary letter F to do this:

$$x01 \to xFy \to y00$$

Numbers divisible by 3, continuation

Numbers divisible by 3, continuation

Similarly, we need the rules $x10 \to xFy \to y01$ and $x11 \to xFy \to y10$.

Numbers divisible by 3, continuation

Similarly, we need the rules $x10 \to xFy \to y01$ and $x11 \to xFy \to y10$.
Let us define the relation F inductively, too. Base: $x0Fx1$, rule: $xFy \to x1Fy0$. For technical reasons, we need to add $1F10$ to the base.

Numbers divisible by 3, continuation

Similarly, we need the rules $x10 \to xFy \to y01$ and $x11 \to xFy \to y10$.
Let us define the relation F inductively, too. Base: $x0Fx1$, rule: $xFy \to x1Fy0$. For technical reasons, we need to add $1F10$ to the base.
Our definition has now the following form: (see next slide)

Numbers divisible by 3, continuation2

0

11

110

$x0Fx1$

$1F10$

$xFy \to x1Fy0$

$x00 \to x11$

$x01 \to xFy \to y00$

$x10 \to xFy \to y01$

$x11 \to xFy \to y10$

Numbers divisible by 3, continuation2

$$0$$

$$11$$

$$110$$

$$x0Fx1$$

$$1F10$$

$$xFy \to x1Fy0$$

$$x00 \to x11$$

$$x01 \to xFy \to y00$$

$$x10 \to xFy \to y01$$

$$x11 \to xFy \to y10$$

This is now of the sort (form) of inductive definitions we call canonical calculus. Formal definition on the next slide.

Canonical calculus (formal definition)

Let $\mathcal{C}$ be a (finite) alphabet and '$\rightarrow$'$\notin \mathcal{C}$. $\underline{\mathcal{C}\text{-rule}}$s are defined inductively as follows:

Canonical calculus (formal definition)

Let $\mathcal{C}$ be a (finite) alphabet and '$\rightarrow$'$\notin \mathcal{C}$. $\underline{\mathcal{C}\text{-rule}}$s are defined inductively as follows:

(i) If $f \in \mathcal{C}^{\circ}$, then f is a $\mathcal{C}$-rule.

(ii) If r is a $\mathcal{C}$-rule and $f \in \mathcal{C}^{\circ}$, then $\ulcorner f \rightarrow r \urcorner$ is a $\mathcal{C}$-rule.

Canonical calculus (formal definition)

Let $\mathcal{C}$ be a (finite) alphabet and '$\rightarrow$'$\notin \mathcal{C}$. $\mathcal{C}$-rules are defined inductively as follows:

(i) If $f \in \mathcal{C}^{\circ}$, then f is a $\mathcal{C}$-rule.

(ii) If r is a $\mathcal{C}$-rule and $f \in \mathcal{C}^{\circ}$, then $\ulcorner f \rightarrow r \urcorner$ is a $\mathcal{C}$-rule.

Let $\mathcal{C}$ and $\mathcal{V}$ alphabets s.t. '$\rightarrow$' $\notin \mathcal{C} \cup \mathcal{V}$. A finite class K of $\mathcal{C} \cup \mathcal{V}$-rules is called a canonical calculus over $\mathcal{C}$. The members of K are the rules of K and the members of $\mathcal{V}$ (if any) are the variables of K.

Strings derivable in a canonical calculus

Let $\mathcal{C}$ be an alphabet and K a canonical calculus over $\mathcal{C}$. The relation $K \mapsto f$ (read: „K derives f" or „f is derivable in K") is defined by induction:

Strings derivable in a canonical calculus

Let $\mathcal{C}$ be an alphabet and K a canonical calculus over $\mathcal{C}$. The relation $K \mapsto f$ (read: „K derives f" or „f is derivable in K") is defined by induction:

(i) $f \in K \Rightarrow K \mapsto f$

Strings derivable in a canonical calculus

Let $\mathcal{C}$ be an alphabet and K a canonical calculus over $\mathcal{C}$. The relation $K \mapsto f$ (read: „K derives f" or „f is derivable in K") is defined by induction:

(i) $f \in K \Rightarrow K \mapsto f$

(ii) If $K \mapsto f$ and f' is the result of substituting a $\mathcal{C}$-string for all occurrences of a variable in f, then $K \mapsto f'$ (Substitution)

Let $\mathcal{C}$ be an alphabet and K a canonical calculus over $\mathcal{C}$. The relation $K \mapsto f$ (read: „K derives f" or „f is derivable in K") is defined by induction:

(i) $f \in K \Rightarrow K \mapsto f$

(ii) If $K \mapsto f$ and f' is the result of substituting a $\mathcal{C}$-string for all occurrences of a variable in f, then $K \mapsto f'$ (Substitution)

(iii) If $K \mapsto f$, $K \mapsto f \to g$ and '$\to$' does not occur in f, then $K \mapsto g$ (Detachment)

Inductive (sub)classes

Inductive (sub)classes

Let $\mathcal{A}$ be an alphabet. The class of strings F is an <u>inductive subclass</u> of $\mathcal{A}^\circ$ iff there exist $\mathcal{C}$ and K s.t.

- $\mathcal{C}$ is an alphabet and $\mathcal{A} \subseteq \mathcal{C}$;
- K is a canonical calculus over $\mathcal{C}$;
- $F = \{x : \ x \in \mathcal{A}^\circ \wedge K \mapsto x\}$.

Inductive (sub)classes

Let $\mathcal{A}$ be an alphabet. The class of strings F is an <u>inductive subclass</u> of $\mathcal{A}^\circ$ iff there exist $\mathcal{C}$ and K s.t.

- $\mathcal{C}$ is an alphabet and $\mathcal{A} \subseteq \mathcal{C}$;
- K is a canonical calculus over $\mathcal{C}$;
- $F = \{x : \; x \in \mathcal{A}^\circ \wedge K \mapsto x\}$.

The members of the class $\mathcal{B} = \mathcal{C} - \mathcal{A}$ are the auxiliary letters.

Inductive (sub)classes

Let $\mathcal{A}$ be an alphabet. The class of strings F is an <u>inductive subclass</u> of $\mathcal{A}^\circ$ iff there exist $\mathcal{C}$ and K s.t.

- $\mathcal{C}$ is an alphabet and $\mathcal{A} \subseteq \mathcal{C}$;
- K is a canonical calculus over $\mathcal{C}$;
- $F = \{x : \; x \in \mathcal{A}^\circ \wedge K \mapsto x\}$.

The members of the class $\mathcal{B} = \mathcal{C} - \mathcal{A}$ are the auxiliary letters.
Homework: If F and G are inductive subclasses of some string class $\mathcal{A}^\circ$, then $F \cup G$ and $F \cap G$ are inductive subclasses of it, too.

Inductive (sub)classes

Let $\mathcal{A}$ be an alphabet. The class of strings F is an <u>inductive subclass</u> of $\mathcal{A}^\circ$ iff there exist $\mathcal{C}$ and K s.t.

- $\mathcal{C}$ is an alphabet and $\mathcal{A} \subseteq \mathcal{C}$;
- K is a canonical calculus over $\mathcal{C}$;
- $F = \{x : \ x \in \mathcal{A}^\circ \wedge K \mapsto x\}$.

The members of the class $\mathcal{B} = \mathcal{C} - \mathcal{A}$ are the auxiliary letters.
Homework: If F and G are inductive subclasses of some string class $\mathcal{A}^\circ$, then $F \cup G$ and $F \cap G$ are inductive subclasses of it, too.
A convention about the use of auxiliary letters: We use them to express predicates of strings. If we want to use P to express a monadic predicate, we write it as a prefix: Px. If it is an n-adic predicate ($n \geq 2$), we write it infix, on the following way: $x_1 P x_2 P \ldots P x_n$.

Some additional remarks

Some additional remarks

Let $\mathcal{A}^{\circ}$ be the class of all strings over an alphabet $\mathcal{A}$.

Some additional remarks

Let $\mathcal{A}^\circ$ be the class of all strings over an alphabet $\mathcal{A}$.

- The empty class $\emptyset$ is an inductive subclass of $\mathcal{A}^\circ$.

Let $\mathcal{A}^\circ$ be the class of all strings over an alphabet $\mathcal{A}$.

- The empty class $\emptyset$ is an inductive subclass of $\mathcal{A}^\circ$.
- $\mathcal{A}^\circ$ is an inductive subclass of itself.

Some additional remarks

Let $\mathcal{A}^\circ$ be the class of all strings over an alphabet $\mathcal{A}$.

- The empty class $\emptyset$ is an inductive subclass of $\mathcal{A}^\circ$.
- $\mathcal{A}^\circ$ is an inductive subclass of itself.
- Any class $\{a_1, a_2, \ldots a_n\}$ (i.e., any string class defined by finite enumeration) is an inductive subclass of $\mathcal{A}^\circ$.

Some additional remarks

Let $\mathcal{A}^\circ$ be the class of all strings over an alphabet $\mathcal{A}$.

- The empty class $\emptyset$ is an inductive subclass of $\mathcal{A}^\circ$.
- $\mathcal{A}^\circ$ is an inductive subclass of itself.
- Any class $\{a_1, a_2, \ldots a_n\}$ (i.e., any string class defined by finite enumeration) is an inductive subclass of $\mathcal{A}^\circ$.
- Inductive classes are not closed for difference. Even the complement $\bar{B} = \mathcal{A}^\circ - B$ of an inductive class B is not necessarily inductive.

Let $\mathcal{A}^\circ$ be the class of all strings over an alphabet $\mathcal{A}$.

- The empty class $\emptyset$ is an inductive subclass of $\mathcal{A}^\circ$.
- $\mathcal{A}^\circ$ is an inductive subclass of itself.
- Any class $\{a_1, a_2, \ldots a_n\}$ (i.e., any string class defined by finite enumeration) is an inductive subclass of $\mathcal{A}^\circ$.
- Inductive classes are not closed for difference. Even the complement $\bar{B} = \mathcal{A}^\circ - B$ of an inductive class B is not necessarily inductive.

The first three remarks are trivial. The fourth one is extremely important for metalogic and will be proved (by examples) later.

Smullyan's mysterious automaton

Smullyan's mysterious automaton

We have are given a machine that works like a canonical calculus. We know only that it prints strings of an alphabet $\mathcal{S} = \{\neg,\ P,\ N,\ (,\ )\}$.

We have are given a machine that works like a canonical calculus. We know only that it prints strings of an alphabet $\mathcal{S} = \{\neg,\ P,\ N,\ (,\ )\}$.

We equip the strings with the following grammar and semantics:

We have are given a machine that works like a canonical calculus. We know only that it prints strings of an alphabet $\mathcal{S} = \{\neg,\ P,\ N,\ (,\ )\}$.

We equip the strings with the following grammar and semantics:

- Sentences are strings of the form $P(X),\ \neg P(X),\ PN(X), \neg PN(X)$, where X is any member of $\mathcal{S}^\circ$.

Smullyan's mysterious automaton

We have are given a machine that works like a canonical calculus. We know only that it prints strings of an alphabet $\mathcal{S} = \{\neg,\ P,\ N,\ (,\ )\}$.

We equip the strings with the following grammar and semantics:

- Sentences are strings of the form $P(X),\ \neg P(X),\ PN(X), \neg PN(X)$, where X is any member of $\mathcal{S}^\circ$.
- Norm of the string X is the string $\ulcorner X(X) \urcorner$.

Smullyan's mysterious automaton

We have are given a machine that works like a canonical calculus. We know only that it prints strings of an alphabet $\mathcal{S} = \{\neg,\ P,\ N,\ (,\ )\}$.

We equip the strings with the following grammar and semantics:

- Sentences are strings of the form $P(X),\ \neg P(X),\ PN(X), \neg PN(X)$, where X is any member of $\mathcal{S}^\circ$.
- Norm of the string X is the string $\ulcorner X(X) \urcorner$.
- The sentence $P(X)$ is true iff the string X gets (sometimes) printed by our machine; $PN(X)$ is true iff the norm of X, i.e. $X(X)$ will be printed sometimes.

We have are given a machine that works like a canonical calculus. We know only that it prints strings of an alphabet $\mathcal{S} = \{\neg,\ P,\ N,\ (,\ )\}$.

We equip the strings with the following grammar and semantics:

- Sentences are strings of the form $P(X),\ \neg P(X),\ PN(X), \neg PN(X)$, where X is any member of $\mathcal{S}^\circ$.
- Norm of the string X is the string $\ulcorner X(X) \urcorner$.
- The sentence $P(X)$ is true iff the string X gets (sometimes) printed by our machine; $PN(X)$ is true iff the norm of X, i.e. $X(X)$ will be printed sometimes.
- '$\neg$' means negation.

Prove that the machine cannot print all and only the true sentences. (Maybe it prints strings that are not sentences, but we speak this time only about sentences the machine can print.) I.e., if it prints only true sentences, then there is at least one true sentence which will be never printed.

Bonus: If you proved this proposition, you may propose a name for it.