The metalogical use of Markov-algorithms
Logical calculi

András Máté

17.10.2025

Definite classes

Definite classes

A class of strings of an alphabet is <u>decidable</u> if there is some effective procedure that decides about any string of the alphabet whether it is a member of the class or not (informal notion). This is the corresponding formal notion:

Definite classes

A class of strings of an alphabet is decidable if there is some effective procedure that decides about any string of the alphabet whether it is a member of the class or not (informal notion).
This is the corresponding formal notion:

Be $\mathcal{A}$ an alphabet. F is a definite subclass of $\mathcal{A}^\circ$ iff there is a Markov algorithm N over some alphabet $\mathcal{B} \supseteq \mathcal{A}$ and a w $\mathcal{B}$-string s. t. N is applicable to every f $\mathcal{A}$-string and $f \in F$ iff $N(f) = w$.

Definite classes

A class of strings of an alphabet is decidable if there is some effective procedure that decides about any string of the alphabet whether it is a member of the class or not (informal notion). This is the corresponding formal notion:

Be $\mathcal{A}$ an alphabet. F is a definite subclass of $\mathcal{A}^\circ$ iff there is a Markov algorithm N over some alphabet $\mathcal{B} \supseteq \mathcal{A}$ and a w $\mathcal{B}$-string s. t. N is applicable to every f $\mathcal{A}$-string and $f \in F$ iff $N(f) = w$.
Markov thesis: Every effective procedure can be simulated by a Markov algorithm and every Markov algorithm is an effective procedure. Therefore, 'definite' and 'decidable' is the same. This is an *empirical* proposition that can be reinforced (although not proved) or refuted by examples.

Definite and inductive classes

Earlier, informal argument: a class of strings is decidable iff both the class itself and its complement is inductive. We want to prove the formal counterpart of it, with ‘definite’ instead of ‘decidable’. First step: we show that Markov-algorithms can be represented by canonical calculi.

Definite and inductive classes

Earlier, informal argument: a class of strings is decidable iff both the class itself and its complement is inductive. We want to prove the formal counterpart of it, with ‘definite’ instead of ‘decidable’. First step: we show that Markov-algorithms can be represented by canonical calculi.

Theorem 1: Let us have an algorithm N over some alphabet $\mathcal{B} \supseteq \mathcal{A}$ that is applicable for every $\mathcal{A}$-string. Then we can construct a calculus K over some $\mathcal{C} \supseteq \mathcal{B}$ using a code letter $\mu \in \mathcal{C} - \mathcal{B}$ such that for all x $\mathcal{A}$-string and y $\mathcal{B}$-string, $N(x) = y$ iff $K \mapsto x\mu y$.

Earlier, informal argument: a class of strings is decidable iff both the class itself and its complement is inductive. We want to prove the formal counterpart of it, with 'definite' instead of 'decidable'. First step: we show that Markov-algorithms can be represented by canonical calculi.

Theorem 1: Let us have an algorithm N over some alphabet $\mathcal{B} \supseteq \mathcal{A}$ that is applicable for every $\mathcal{A}$-string. Then we can construct a calculus K over some $\mathcal{C} \supseteq \mathcal{B}$ using a code letter $\mu \in \mathcal{C} - \mathcal{B}$ such that for all x $\mathcal{A}$-string and y $\mathcal{B}$-string, $N(x) = y$ iff $K \mapsto x\mu y$.

Proof: Be $N = \langle C_1,\ C_2, \ldots C_n \rangle$. The calculus K will be the union of the calculi $K_1,\ K_2, \ldots K_n$ associated to the commands of N plus a calculus K_0.

Proof(continuation)

Proof(continuation)

If the command C_i is of the form $\varnothing \to v_i$ or $\varnothing \to .v_i$, then the calculus K_i consists of the single rule

$$x\Delta^i v_i x$$

(Δ^i is an auxiliary letter.)

Proof(continuation)

If the command C_i is of the form $\varnothing \to v_i$ or $\varnothing \to .v_i$, then the calculus K_i consists of the single rule

$$x\Delta^i v_i x$$

(Δ^i is an auxiliary letter.)

If C_i is of the form $u_i \to v_i$ or $u_i \to .v_i$, where $u_i = b_1 b_2 \dots b_k$, then the calculus K_i will be this:

$$\begin{array}{lll}
i1. & \Delta_{i1}x & \\
i2. & x\Delta_{i1}by \to xb\Delta_{i1}y & b \in \mathcal{B} - \{b_1\} \\
i3. & x\Delta_{ij}by \to x\Delta_{i1}by & b \in \mathcal{B} - \{b_j\}, 1 \leq j \leq k \\
i4. & x\Delta_{ij}b_jy \to xb_j\Delta_{i,j+1}y & 1 \leq j \leq k \\
i5. & x\Delta_{ij} \to \Delta_{i0}x & 1 \leq j \leq k \\
i6. & xu_i\Delta_{i,k+1}y \to xu_iy\Delta^i xv_iy &
\end{array}$$

(Δ^i, Δ_{i0}, $\Delta_{i1}, \dots \Delta_{ik}$, $\Delta_{i,k+1}$ are auxiliary letters.)

Proof(continuation2)

The calculus K_0:

$$
\begin{array}{rl}
1. & x\Delta^1 y \to xZy \\
2. & \Delta_{10}x \to x\Delta^2 y \to xZy \\
3. & \Delta_{10}x \to \Delta_{20}x \to x\Delta^3 y \to xZy \\
\ldots & \\
i+1. & \Delta_{10}x \to \ldots \to \Delta_{i0}x \to x\Delta^{i+1} y \to xZy \\
\ldots & \\
n. & \Delta_{10}x \to \ldots \to \Delta_{n-1,0}x \to x\Delta^n y \to xZy \\
n+1. & xMy \to yMz \to xMz \\
n+2. & xMy \to y\mu z \to x\mu z
\end{array}
$$

where in the ith rule $(1 \leq i \leq n)$ Z stands for μ if C_i is a stop command and for M if it is not.

The calculus K_0:

$$
\begin{array}{rl}
1. & x\Delta^1 y \to xZy \\
2. & \Delta_{10}x \to x\Delta^2 y \to xZy \\
3. & \Delta_{10}x \to \Delta_{20}x \to x\Delta^3 y \to xZy \\
& \ldots \\
i+1. & \Delta_{10}x \to \ldots \to \Delta_{i0}x \to x\Delta^{i+1} y \to xZy \\
& \ldots \\
n. & \Delta_{10}x \to \ldots \to \Delta_{n-1,0}x \to x\Delta^n y \to xZy \\
n+1. & xMy \to yMz \to xMz \\
n+2. & xMy \to y\mu z \to x\mu z
\end{array}
$$

where in the ith rule $(1 \leq i \leq n)$ Z stands for μ if C_i is a stop command and for M if it is not.
Now the calculus K is ready.

Definite classes are inductive classes

Definite classes are inductive classes

Theorem 2. If $\mathcal{A}$ is an alphabet and F is a definite subclass of $\mathcal{A}^\circ$, then F is an inductive subclass of it.

Definite classes are inductive classes

Theorem 2. If $\mathcal{A}$ is an alphabet and F is a definite subclass of $\mathcal{A}^\circ$, then F is an inductive subclass of it.

Proof: Let the deciding algorithm for F be N over $\mathcal{B} \supseteq \mathcal{A}$, $w \in \mathcal{B}^\circ$ such that

$$f \in F \Leftrightarrow N(f) = w.$$

Definite classes are inductive classes

Theorem 2. If $\mathcal{A}$ is an alphabet and F is a definite subclass of $\mathcal{A}^\circ$, then F is an inductive subclass of it.

Proof: Let the deciding algorithm for F be N over $\mathcal{B} \supseteq \mathcal{A}$, $w \in \mathcal{B}^\circ$ such that

$$f \in F \Leftrightarrow N(f) = w.$$

Be K the calculus representing N according to the the previous theorem ($\mathcal{C}$, μ like in the previous theorem, too.) Then for any $f \in \mathcal{A}^\circ$, $N(f) = g \Leftrightarrow K \mapsto f\mu g$.

Then $N(f) = w$ iff $K \mapsto x\mu w$. Let us add the rule $x\mu w \to x$ to K to get the calculus K'. From the proof of the previous theorem you can see that K derives no $\mathcal{A}$-string, therefore K' derives $\mathcal{A}$-strings by using this last rule only.

Theorem 2. If $\mathcal{A}$ is an alphabet and F is a definite subclass of $\mathcal{A}^\circ$, then F is an inductive subclass of it.

Proof: Let the deciding algorithm for F be N over $\mathcal{B} \supseteq \mathcal{A}$, $w \in \mathcal{B}^\circ$ such that

$$f \in F \Leftrightarrow N(f) = w.$$

Be K the calculus representing N according to the the previous theorem ($\mathcal{C}$, μ like in the previous theorem, too.) Then for any $f \in \mathcal{A}^\circ$, $N(f) = g \Leftrightarrow K \mapsto f\mu g$.

Then $N(f) = w$ iff $K \mapsto x\mu w$. Let us add the rule $x\mu w \to x$ to K to get the calculus $K^{'}$. From the proof of the previous theorem you can see that K derives no $\mathcal{A}$-string, therefore $K^{'}$ derives $\mathcal{A}$-strings by using this last rule only.

Therefore, for any $\mathcal{A}$-string f,

$$f \in F \Leftrightarrow N(f) = w \Leftrightarrow K \mapsto f\mu w \Leftrightarrow K^{'} \mapsto f.$$

I.e., $K^{'}$ defines inductively F.

Decidable and inductive classes

Decidable and inductive classes

A decision algorithm for some string class $\mathcal{A}$ can be modified to an algorithm that decides its complement class (for the class of $\overline{\mathcal{A}}$-strings). (See the identifying algorithm.) Therefore, if a string class is definite, then both the class itself and its complement are inductive ones.

Decidable and inductive classes

A decision algorithm for some string class $\mathcal{A}$ can be modified to an algorithm that decides its complement class (for the class of $\mathcal{A}$-strings). (See the identifying algorithm.) Therefore, if a string class is definite, then both the class itself and its complement are inductive ones.

According to the Markov thesis, decidable classes are the same as definite classes. Therefore, if a class is decidable, then both the class and its complement are inductive classes. We have seen earlier the converse of this claim. Hence, *a string class F is decidable if and only if both F and its complement are inductive classes.*

Decidable and inductive classes

A decision algorithm for some string class $\mathcal{A}$ can be modified to an algorithm that decides its complement class (for the class of $\mathcal{A}$-strings). (See the identifying algorithm.) Therefore, if a string class is definite, then both the class itself and its complement are inductive ones.

According to the Markov thesis, decidable classes are the same as definite classes. Therefore, if a class is decidable, then both the class and its complement are inductive classes. We have seen earlier the converse of this claim. Hence, *a string class F is decidable if and only if both F and its complement are inductive classes.*

We have proven (3rd October presentation) that the class of autonomous numerals Aut is inductive, but its complement for the class of all numerals, i. e. the class of non-autonomous numerals is not inductive. Therefore, it is not decidable.

Logical calculi

Next goal: the first-order theory of canonical calculi. In our metalogic, this theory will be the basic example for incompleteness.

Next goal: the first-order theory of canonical calculi. In our metalogic, this theory will be the basic example for incompleteness.

Logical calculus:

Logical calculi

Next goal: the first-order theory of canonical calculi. In our metalogic, this theory will be the basic example for incompleteness.

Logical calculus:

- An L family of languages with a distinguished category $Form_L$;

Logical calculi

Next goal: the first-order theory of canonical calculi. In our metalogic, this theory will be the basic example for incompleteness.

Logical calculus:

- An L family of languages with a distinguished category $Form_L$;
- Inductive definition of the *syntactic* consequence (deducibility) relation $\Gamma \vdash_L A$, where $\Gamma \subseteq Form_L$ (premises) and $A \in Form_L$ (conclusion).

Logical calculi

Next goal: the first-order theory of canonical calculi. In our metalogic, this theory will be the basic example for incompleteness.
Logical calculus:

- An L family of languages with a distinguished category $Form_L$;
- Inductive definition of the *syntactic* consequence (deducibility) relation $\Gamma \vdash_L A$, where $\Gamma \subseteq Form_L$ (premises) and $A \in Form_L$ (conclusion).

Base of the inductive definition: a class of formulas deducible from the empty class of premises (*basic formulas* or *logical axioms*).

Logical calculi

Next goal: the first-order theory of canonical calculi. In our metalogic, this theory will be the basic example for incompleteness.

Logical calculus:

- An L family of languages with a distinguished category $Form_L$;
- Inductive definition of the *syntactic* consequence (deducibility) relation $\Gamma \vdash_L A$, where $\Gamma \subseteq Form_L$ (premises) and $A \in Form_L$ (conclusion).

Base of the inductive definition: a class of formulas deducible from the empty class of premises (*basic formulas* or *logical axioms*).

Inductive rules (rules of deduction, proof rules) prescribe how you can arrive from some given relations $\Gamma \vdash A_1, \Gamma \vdash A_2, \ldots$ to some new relation $\Gamma \vdash A$.

Logical calculi (continuation)

Logical calculi (continuation)

Different ways to define the deducibility relation: many axioms and only one or two rules of deduction (Frege-Hilbert style of calculus) versus no axioms at all, only rules (Gentzen-style or natural deduction systems).

Logical calculi (continuation)

Different ways to define the deducibility relation: many axioms and only one or two rules of deduction (Frege-Hilbert style of calculus) versus no axioms at all, only rules (Gentzen-style or natural deduction systems).

Equivalence of different calculi (for the same family of languages): on the natural way (the extension of the relation $\vdash$ is the same).

Different ways to define the deducibility relation: many axioms and only one or two rules of deduction (Frege-Hilbert style of calculus) versus no axioms at all, only rules (Gentzen-style or natural deduction systems).

Equivalence of different calculi (for the same family of languages): on the natural way (the extension of the relation $\vdash$ is the same).

A natural demand for the class of logical axioms and the rules of deduction: they should be decidable.