The metalogical use of Markov-algorithms
The quantification calculus (QC)

András Máté

11.10.2024

Definite classes

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:

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

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 $K \mapsto x\mu y$.

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 $K \mapsto x\mu y$.

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.

Proof(continuation2)

The calculus K_0:

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

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.

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.

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 $\mathcal{A}$-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 (27th September 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).

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).

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

First-order languages

First-order languages

All the symbols are strings of some given alphabet $\mathcal{A}$.

First-order languages

All the symbols are strings of some given alphabet $\mathcal{A}$.

The class of arities $A = \{\varnothing,\ o,\ oo, \ldots\}$ was defined inductively earlier.

First-order languages

All the symbols are strings of some given alphabet $\mathcal{A}$.

The class of arities $A = \{\varnothing,\ o,\ oo, \ldots\}$ was defined inductively earlier.

A first-order language $\mathcal{L}^1$ is a quintuple

$$< Log, Var, Con, Term, Form >$$

where $Log = \{(,\ ),\ \neg,\ \supset,\ \forall,\ =\}$ is the class of logical constants, Var is the infinite class of variables defined inductively, and $Con = N \cup P = \bigcup_{a \in A} P_a \cup \bigcup_{a \in A} N_a$ is the class of non-logical constants containing all the classes P_a of a-ary predicates and N_a of a-ary name functors.

First-order languages

All the symbols are strings of some given alphabet $\mathcal{A}$.

The class of arities $A = \{\varnothing,\ o,\ oo, \ldots\}$ was defined inductively earlier.

A first-order language $\mathcal{L}^1$ is a quintuple

$$< Log, Var, Con, Term, Form >$$

where $Log = \{(,\ ),\ \neg,\ \supset,\ \forall,\ =\}$ is the class of logical constants, Var is the infinite class of variables defined inductively, and $Con = N \cup P = \bigcup_{a \in A} P_a \cup \bigcup_{a \in A} N_a$ is the class of non-logical constants containing all the classes P_a of a-ary predicates and N_a of a-ary name functors.

It is assumed that for $a_i \neq a_j \in A$, $N_{a_i} \cap N_{a_j} = P_{a_i} \cap P_{a_j} = \emptyset$ and $N \cap P = \emptyset$.

Terms and a-tuples of terms

Terms and a-tuples of terms

The class of a-tuples of terms $a \in A$ is $T(a)$.

The class of a-tuples of terms $a \in A$ is $T(a)$.

The simultaneous inductive definition of the classes $Term$ and $T(a)$:

Terms and a-tuples of terms

The class of a-tuples of terms $a \in A$ is $T(a)$.

The simultaneous inductive definition of the classes $Term$ and $T(a)$:

1. $Var \subseteq Term$
2. $T(\varnothing) = \{\varnothing\}$
3. $(s \in T(a)\ \&\ t \in Term) \Rightarrow \ulcorner s(t) \urcorner \in T(ao)$
4. $(\varphi \in N_a\ \&\ s \in T(a)) \Rightarrow \ulcorner \varphi s \urcorner \in Term$

Formulas

Formulas

1. $\pi \in P_a \;\&\; s \in T(a) \Rightarrow \ulcorner \pi s \urcorner \in Form$
2. $s, t \in Term \Rightarrow \ulcorner s = t \urcorner \in Form$
3. $A \in Form \Rightarrow \ulcorner \neg A \urcorner \in Form$
4. $A, B \in Form \Rightarrow \ulcorner A \supset B \urcorner \in Form$
5. $A \in Form \;\&\; x \in Var \Rightarrow \ulcorner \forall x A \urcorner \in Form$

Formulas

1. $\pi \in P_a \; \& \; s \in T(a) \Rightarrow \ulcorner \pi s \urcorner \in Form$
2. $s, t \in Term \Rightarrow \ulcorner s = t \urcorner \in Form$
3. $A \in Form \Rightarrow \ulcorner \neg A \urcorner \in Form$
4. $A, B \in Form \Rightarrow \ulcorner A \supset B \urcorner \in Form$
5. $A \in Form \; \& \; x \in Var \Rightarrow \ulcorner \forall x A \urcorner \in Form$

Atomic formulas are the formulas generated by the rules 1. and 2.

Formulas

1. $\pi \in P_a \;\&\; s \in T(a) \Rightarrow \ulcorner \pi s \urcorner \in Form$
2. $s, t \in Term \Rightarrow \ulcorner s = t \urcorner \in Form$
3. $A \in Form \Rightarrow \ulcorner \neg A \urcorner \in Form$
4. $A, B \in Form \Rightarrow \ulcorner A \supset B \urcorner \in Form$
5. $A \in Form \;\&\; x \in Var \Rightarrow \ulcorner \forall x A \urcorner \in Form$

Atomic formulas are the formulas generated by the rules 1. and 2.

Other logical constants ($\vee$, $\wedge$, $\equiv$, $\exists$) are introduced by abbreviation conventions.

Formulas

1. $\pi \in P_a \ \& \ s \in T(a) \Rightarrow \ulcorner \pi s \urcorner \in Form$
2. $s, t \in Term \Rightarrow \ulcorner s = t \urcorner \in Form$
3. $A \in Form \Rightarrow \ulcorner \neg A \urcorner \in Form$
4. $A, B \in Form \Rightarrow \ulcorner A \supset B \urcorner \in Form$
5. $A \in Form \ \& \ x \in Var \Rightarrow \ulcorner \forall x A \urcorner \in Form$

Atomic formulas are the formulas generated by the rules 1. and 2.

Other logical constants ($\vee$, $\wedge$, $\equiv$, $\exists$) are introduced by abbreviation conventions.

Be $A, B \in Form$. B is a subformula of A iff A is of the form uBv $(u, v \in \mathcal{A}^\circ)$.

1. $\pi \in P_a \; \& \; s \in T(a) \Rightarrow \ulcorner \pi s \urcorner \in Form$
2. $s, t \in Term \Rightarrow \ulcorner s = t \urcorner \in Form$
3. $A \in Form \Rightarrow \ulcorner \neg A \urcorner \in Form$
4. $A, B \in Form \Rightarrow \ulcorner A \supset B \urcorner \in Form$
5. $A \in Form \; \& \; x \in Var \Rightarrow \ulcorner \forall x A \urcorner \in Form$

Atomic formulas are the formulas generated by the rules 1. and 2.

Other logical constants ($\vee$, $\wedge$, $\equiv$, $\exists$) are introduced by abbreviation conventions.

Be $A, B \in Form$. B is a subformula of A iff A is of the form uBv ($u, v \in \mathcal{A}^\circ$).

If $x \in Var$ and $A \in Form$, an occurrence of x in A is a bound occurrence of x in A iff it lies in a subformula of A of the form $\forall x B$. Other occurrences are called free occurrences.

Some further auxiliary notions

Some further auxiliary notions

A term is open iff at least one variable is a substring of it; otherways it is closed.

Some further auxiliary notions

A term is open iff at least one variable is a substring of it; otherways it is closed.

A formula is open if it contains at least one free occurrence of a variable; otherwise it is closed. Closed formulas are called sentences.

Some further auxiliary notions

A term is <u>open</u> iff at least one variable is a substring of it; otherways it is <u>closed</u>.

A formula is <u>open</u> if it contains at least one free occurrence of a variable; otherwise it is <u>closed</u>. Closed formulas are called <u>sentences</u>.

A formula A is <u>free from</u> the variable x iff x has no free occurrences in A. $\Gamma \subseteq Form$ is <u>free from</u> x if each member of it is.

Some further auxiliary notions

A term is open iff at least one variable is a substring of it; otherways it is closed.

A formula is open if it contains at least one free occurrence of a variable; otherwise it is closed. Closed formulas are called sentences.

A formula A is free from the variable x iff x has no free occurrences in A. $\Gamma \subseteq Form$ is free from x if each member of it is.

Be $A \in Form$, $x, y \in Var$. y is substitutable for x in A iff for every subformula of A of the form $\forall y B$, B is free from x.

$t \in Term$ is substitutable for x in A iff every variable occurring in t is substitutable. If t is substitutable for x in A, then $A^{t/x}$ denotes (in the metalanguage) the formula obtained from A substituting t for every free occurrence of x in A.

The quantification calculus (**QC**) 1: the axioms

The quantification calculus (**QC**) 1: the axioms

Given a first-order language $\mathcal{L}^1$, the logical axioms (basic formulas) are defined by the help of the following schemes:

The quantification calculus (**QC**) 1: the axioms

Given a first-order language $\mathcal{L}^1$, the logical axioms (basic formulas) are defined by the help of the following schemes:

(B1) $(A \supset (B \supset A))$

(B2) $((A \supset (B \supset C)) \supset ((A \supset B) \supset (A \supset C)))$

(B3) $((\neg B \supset \neg A) \supset (A \supset B))$

Given a first-order language $\mathcal{L}^1$, the logical axioms (basic formulas) are defined by the help of the following schemes:

(B1) $(A \supset (B \supset A))$

(B2) $((A \supset (B \supset C) \supset ((A \supset B) \supset (A \supset C)))$

(B3) $((\neg B \supset \neg A) \supset (A \supset B))$

(B4) $(\forall x A \supset A^{t/x})$

(B5) $(\forall x(A \supset B) \supset (\forall x A \supset \forall x B))$

(B6) $(A \supset \forall x A)$ provided that A is free from x

The quantification calculus (**QC**) 1: the axioms

Given a first-order language $\mathcal{L}^1$, the logical axioms (basic formulas) are defined by the help of the following schemes:

(B1) $(A \supset (B \supset A))$

(B2) $((A \supset (B \supset C) \supset ((A \supset B) \supset (A \supset C)))$

(B3) $((\neg B \supset \neg A) \supset (A \supset B))$

(B4) $(\forall x A \supset A^{t/x})$

(B5) $(\forall x(A \supset B) \supset (\forall x A \supset \forall x B))$

(B6) $(A \supset \forall x A)$ provided that A is free from x

(B7) $(\mathfrak{x} = \mathfrak{x})$

(B8) $((x = y) \supset (A^{x/z} \supset A^{y/z}))$

The quantification calculus (QC) 1: the axioms

Given a first-order language $\mathcal{L}^1$, the logical axioms (basic formulas) are defined by the help of the following schemes:

(B1) $(A \supset (B \supset A))$

(B2) $((A \supset (B \supset C) \supset ((A \supset B) \supset (A \supset C)))$

(B3) $((\neg B \supset \neg A) \supset (A \supset B))$

(B4) $(\forall x A \supset A^{t/x})$

(B5) $(\forall x(A \supset B) \supset (\forall x A \supset \forall x B))$

(B6) $(A \supset \forall x A)$ provided that A is free from x

(B7) $(\mathfrak{x} = \mathfrak{x})$

(B8) $((x = y) \supset (A^{x/z} \supset A^{y/z}))$

The class BF of logical axioms is defined inductively:

The quantification calculus (**QC**) 1: the axioms

Given a first-order language $\mathcal{L}^1$, the logical axioms (basic formulas) are defined by the help of the following schemes:

(B1) $(A \supset (B \supset A))$

(B2) $((A \supset (B \supset C) \supset ((A \supset B) \supset (A \supset C)))$

(B3) $((\neg B \supset \neg A) \supset (A \supset B))$

(B4) $(\forall x A \supset A^{t/x})$

(B5) $(\forall x(A \supset B) \supset (\forall x A \supset \forall x B))$

(B6) $(A \supset \forall x A)$ provided that A is free from x

(B7) $(\mathfrak{x} = \mathfrak{x})$

(B8) $((x = y) \supset (A^{x/z} \supset A^{y/z}))$

The class BF of logical axioms is defined inductively:

i If we substitute formulas for A, B, C , variables for x, y, z and terms for t of $\mathcal{L}^1$ in the above schemes, we get members of BF.

Given a first-order language $\mathcal{L}^1$, the logical axioms (basic formulas) are defined by the help of the following schemes:

(B1) $(A \supset (B \supset A))$

(B2) $((A \supset (B \supset C) \supset ((A \supset B) \supset (A \supset C)))$

(B3) $((\neg B \supset \neg A) \supset (A \supset B))$

(B4) $(\forall x A \supset A^{t/x})$

(B5) $(\forall x(A \supset B) \supset (\forall x A \supset \forall x B))$

(B6) $(A \supset \forall x A)$ provided that A is free from x

(B7) $(\mathfrak{x} = \mathfrak{x})$

(B8) $((x = y) \supset (A^{x/z} \supset A^{y/z}))$

The class BF of logical axioms is defined inductively:

i If we substitute formulas for A, B, C , variables for x, y, z and terms for t of $\mathcal{L}^1$ in the above schemes, we get members of BF.

ii If $A \in BF$ and $x \in Var$, then $\ulcorner \forall x A \urcorner \in BF$.

QC 2: deducibility and some metatheorems

Base for the inductive definition of $\Gamma \vdash A$: if $A \in \Gamma \cup BF$, then $\Gamma \vdash A$. Inductive rule is detachment: if $\Gamma \vdash A$ and $\Gamma \vdash A \supset B$, then $\Gamma \vdash B$.

QC 2: deducibility and some metatheorems

Base for the inductive definition of $\Gamma \vdash A$: if $A \in \Gamma \cup BF$, then $\Gamma \vdash A$. Inductive rule is detachment: if $\Gamma \vdash A$ and $\Gamma \vdash A \supset B$, then $\Gamma \vdash B$.

- Deduction Theorem: If $\Gamma \cup \{A\} \vdash C$, then $\Gamma \vdash A \supset C$.

QC 2: deducibility and some metatheorems

Base for the inductive definition of $\Gamma \vdash A$: if $A \in \Gamma \cup BF$, then $\Gamma \vdash A$. Inductive rule is detachment: if $\Gamma \vdash A$ and $\Gamma \vdash A \supset B$, then $\Gamma \vdash B$.

- Deduction Theorem: If $\Gamma \cup \{A\} \vdash C$, then $\Gamma \vdash A \supset C$.
- Cut: If $\Gamma \vdash A$ and $\Gamma' \cup \{A\} \vdash B$ then $\Gamma \cup \Gamma' \vdash B$.

Base for the inductive definition of $\Gamma \vdash A$: if $A \in \Gamma \cup BF$, then $\Gamma \vdash A$. Inductive rule is detachment: if $\Gamma \vdash A$ and $\Gamma \vdash A \supset B$, then $\Gamma \vdash B$.

- Deduction Theorem: If $\Gamma \cup \{A\} \vdash C$, then $\Gamma \vdash A \supset C$.
- Cut: If $\Gamma \vdash A$ and $\Gamma' \cup \{A\} \vdash B$ then $\Gamma \cup \Gamma' \vdash B$.
- Universal generalization: If $\Gamma \vdash A$ and Γ is free from x, then $\Gamma \vdash \forall x A$.

QC 2: deducibility and some metatheorems

Base for the inductive definition of $\Gamma \vdash A$: if $A \in \Gamma \cup BF$, then $\Gamma \vdash A$. Inductive rule is detachment: if $\Gamma \vdash A$ and $\Gamma \vdash A \supset B$, then $\Gamma \vdash B$.

- Deduction Theorem: If $\Gamma \cup \{A\} \vdash C$, then $\Gamma \vdash A \supset C$.
- Cut: If $\Gamma \vdash A$ and $\Gamma' \cup \{A\} \vdash B$ then $\Gamma \cup \Gamma' \vdash B$.
- Universal generalization: If $\Gamma \vdash A$ and Γ is free from x, then $\Gamma \vdash \forall x A$.
- Universal generalization 2.: If $t \in T(\varnothing)$ s.t. it occurs neither in A nor in the members of Γ and $\Gamma \vdash A^{t/x}$ then $\Gamma \vdash \forall x A$.

Base for the inductive definition of $\Gamma \vdash A$: if $A \in \Gamma \cup BF$, then $\Gamma \vdash A$. Inductive rule is detachment: if $\Gamma \vdash A$ and $\Gamma \vdash A \supset B$, then $\Gamma \vdash B$.

- Deduction Theorem: If $\Gamma \cup \{A\} \vdash C$, then $\Gamma \vdash A \supset C$.
- Cut: If $\Gamma \vdash A$ and $\Gamma' \cup \{A\} \vdash B$ then $\Gamma \cup \Gamma' \vdash B$.
- Universal generalization: If $\Gamma \vdash A$ and Γ is free from x, then $\Gamma \vdash \forall x A$.
- Universal generalization 2.: If $t \in T(\varnothing)$ s.t. it occurs neither in A nor in the members of Γ and $\Gamma \vdash A^{t/x}$ then $\Gamma \vdash \forall x A$.

A definition: If $A \in Form$ and the variables having free occurrences in A are $x_1, x_2, \ldots x_n$, then the universal closure of A is the formula $\forall x_1 \forall x_2 \ldots \forall x_n A$.

Consequences, consistency, first-order theories

Given any logical calculus Σ in a language $\mathcal{L}$ and a class Γ of formulas of $\mathcal{L}$, the class of the consequences of Γ is the class

$$Cns(\Gamma) = \{A \in Form : \Gamma \vdash_\Sigma A\}$$

Given any logical calculus Σ in a language $\mathcal{L}$ and a class Γ of formulas of $\mathcal{L}$, the class of the consequences of Γ is the class

$$Cns(\Gamma) = \{A \in Form : \Gamma \vdash_{\Sigma} A\}$$

Γ is inconsistent if $Cns(\Gamma) = Form$, consistent in the other case.

Consequences, consistency, first-order theories

Given any logical calculus Σ in a language $\mathcal{L}$ and a class Γ of formulas of $\mathcal{L}$, the class of the consequences of Γ is the class

$$Cns(\Gamma) = \{A \in Form : \Gamma \vdash_{\Sigma} A\}$$

Γ is inconsistent if $Cns(\Gamma) = Form$, consistent in the other case.

In first-order logic, Γ is consistent iff there is no $A \in Form$ s. t. both $\Gamma \vdash A$ and $\Gamma \vdash \neg A$.

Given any logical calculus Σ in a language $\mathcal{L}$ and a class Γ of formulas of $\mathcal{L}$, the class of the consequences of Γ is the class

$$Cns(\Gamma) = \{A \in Form : \Gamma \vdash_{\Sigma} A\}$$

Γ is <u>inconsistent</u> if $Cns(\Gamma) = Form$, <u>consistent</u> in the other case.

In first-order logic, Γ is consistent iff there is no $A \in Form$ s. t. both $\Gamma \vdash A$ and $\Gamma \vdash \neg A$.

The pair $T =< \mathcal{L}^1, \Gamma >$ is a <u>first-order theory</u> if $\mathcal{L}^1$ is a first-order language and Γ is a class of its *closed* formulas (called *axioms* of T).

Consequences, consistency, first-order theories

Given any logical calculus Σ in a language $\mathcal{L}$ and a class Γ of formulas of $\mathcal{L}$, the class of the consequences of Γ is the class

$$Cns(\Gamma) = \{A \in Form : \Gamma \vdash_{\Sigma} A\}$$

Γ is inconsistent if $Cns(\Gamma) = Form$, consistent in the other case.

In first-order logic, Γ is consistent iff there is no $A \in Form$ s. t. both $\Gamma \vdash A$ and $\Gamma \vdash \neg A$.

The pair $T =< \mathcal{L}^1, \Gamma >$ is a first-order theory if $\mathcal{L}^1$ is a first-order language and Γ is a class of its *closed* formulas (called *axioms* of T).

The theorems of T are the members of $Cns(\Gamma)$. T is said consistent resp. inconsistent if Γ is consistent resp. inconsistent.