# X

Tenth Lecture of

COMPUTABILITY, DECIDABILITY, INCOMPLETENESS

CDI

20 JUNE 2023

**GOAL:**

(from Lecture IX:)

Reduce $\mathbb{K}$ to $\{\varphi_j \vdash \varphi\}$ by a reduction function $w \longmapsto \varphi_w$.

**Proof idea**

We reduce the undecidability of $\{\varphi_j \vdash \varphi\}$ to the undecidability of $\mathbb{K}$ by providing a reduction function witnessing

$$\mathbb{K} \leq_m \{\varphi_j \vdash \varphi\}$$

[i.e., $h : \mathbb{W} \longrightarrow \mathbb{W}$
$x \in \mathbb{K} \longleftrightarrow h(x) \in \{\varphi_i \vdash \varphi\}$,

i.e. $w \longmapsto \varphi_w$ s.t.

$$f_{w,1}(w)\downarrow \iff \vdash \varphi_w.$$

Idea:

Formula $\varphi_w$ expressed "The machine coded by $w$ halts on input $w$."

So: For fixed $w$, build formula $\varphi_w$ expressing "The machine coded by $w$ halts on input $w$".

# WHAT WE HAVE DONE SO FAR

Our logical language:

| | | INTERPRETED AS |
|---|---|---|
| NAT | unary predicate | $\mathbb{N}$ |
| S | unary function | $x \mapsto x+1$ |
| Z | constant | $0$ |

1. NAT(Z)
2. $\forall x$ (NAT(x) $\longrightarrow$ NAT(S(x)))

| | | |
|---|---|---|
| SYMBOL | unary predicate | $\Sigma$ |
| WORD | unary predicate | $W$ |
| SEQWORDS | unary predicate | $(W)^*$ |
| LENGTH | unary function | $\lvert\cdot\rvert$ |
| EMPTY | constant | $\varepsilon$ |
| LAST | unary function | last symbol in w |
| ADD | binary function | $W \times \Sigma \to W$ |
| REMOVE | unary function | $W \setminus \{\varepsilon\} \to W$ |
| PROJ | binary function | $(w, n) \mapsto w_n$ |

3. $\forall x \forall y$ SEQWORDS(x) $\wedge$ NAT(y) $\longrightarrow$ WORD(PROJ(x,y))
4. $\forall x$ WORD(x) $\longrightarrow$ NAT(LENGTH(x))
5. WORD(EMPTY) $\wedge \neg$ SYMBOL(EMPTY)
6. $\forall x$ WORD(x) $\longrightarrow$ SYMBOL(LAST(x)) $\vee$ LAST(x) = EMPTY
7. $\forall x \forall y$ WORD(x) $\wedge$ SYMBOL(y) $\longrightarrow$ WORD(ADD(x,y))
8. $\forall x$ WORD(x) $\longrightarrow$ WORD(REMOVE(
9. $\forall x \forall y$ WORD(x) $\wedge$ SYMBOL(y) $\longrightarrow$ LAST(ADD(x,y)) = y
10. $\forall x \forall y$ WORD(x) $\wedge$ SYMBOL(y) LAST(x) = y $\longrightarrow$ ADD(REMOVE(x),y) = x

FROM LECTURE IX

Mostly the syntax of talking about configurations & machines.

| | | |
|---|---|---|
| STATE | unary relations | states |
| INST | | instructions |
| CONFIG | | configurations |
| START | constants | q$_S$ |
| HALT | | q$_H$ |
| +, ?1, ?2, − | unary relations | type of instruction |
| IREG | unary functions | information given to instruction |
| ISYM | | |
| IST1 | | |
| IST2 | | |
| CSTATE | unary functions | state/content of config. |
| CCONTENT | | |

11. $\forall x$ INST(x) $\longleftrightarrow$ +(x) $\vee$ ?1(x) $\vee$ ?2(x) $\vee$ −(x)
12-15. $\forall x$ +(x) $\longrightarrow \neg$?1(x) $\wedge \neg$?2(x) $\wedge \neg -(x)$ [and similar for ?1, ?2, −]
16. $\forall x$ INST(x) $\longrightarrow$ NAT(IREG(x)) $\wedge$ SYMBOL(ISYM(x)) $\wedge$ STATE(IST1(x)) $\wedge$ STATE(IST2(x))
17. $\forall x$ CONFIG(x) $\longrightarrow$ STATE(CSTATE(x))
18. $\forall x$ CONFIG(x) $\longrightarrow$ SEQWORDS(CCONTENT(x))
19. STATE(START) $\wedge$ STATE(HALT)

| | | |
|---|---|---|
| MSTATES | unary function | Q |
| MPROG | unary function | P |
| IN | binary relation | e |
| ASS | binary function | function application |

20. $\forall x \forall y$ IN(y, MSTATES(x)) $\longrightarrow$ STATE(y)
21. $\forall x \forall y \forall z$ y = MPROG(x) $\wedge$ IN(z, MSTATES(x)) $\longrightarrow$ INST(ASS(y,z))

# ENCODING MACHINES $\mathbb{M} = (\Sigma, Q, P)$ IN A FORMOLA.

Reminder (Lecture V) pages 7-9

We already encoded machines as words.

$$\mathbb{M} \longmapsto \text{code}(\mathbb{M}) \in \mathbb{W}$$

For example, our favourite non-halting machine

$$\mathbb{M} = (\{a\}, \{q_S, q_H\}, q_S \longmapsto +(0, a, q_S), \; q_H \longmapsto +(0, a, q_S))$$

was encode by encoding $q_S$ as $\mathbb{0}$
$q_H$ as $\mathbb{1}$

and

$$\text{code}(\mathbb{M}) = \mathbb{0} \mapsto \#(\mathbb{0}, a, \mathbb{0}), \mathbb{1} \mapsto \#(\mathbb{0}, a, \mathbb{0})$$

is the code for $\mathbb{M}$.

Similarly, had coding of sequences of words

$$\vec{w} \longmapsto \text{code}(\vec{w}) \in \mathbb{W}$$

E.g., $(ab, aaa, bbb) \longmapsto ab \square aaa \square bbb \square$

Use this idea to write down a formula $\varphi_w$ that describes that an object is the machine coded by $w$, so

$$\varphi_w(M) \iff \text{“}\, w = code(M) \,\text{”}.$$

Describe that $M$ has only the states $q_S, q_H$:

$$\forall x \Big( IN(x, MSTATES(M)) \longrightarrow (x = START \vee x = HALT) \Big)$$

Similarly, describe that $M$ has three states:

$$\exists s \Big( s \neq START \wedge s \neq HALT \wedge START \neq HALT \wedge \forall x \Big( IN(x, MSTATES(M)) \longrightarrow x = START \vee x = HALT \vee x = s \Big) \Big)$$

Obviously, you can continue and describe that $M$ has $n$ states.

Let $\varphi_n(M)$ be the formula describing that $M$ has $n$ states.

How do we describe the program of M?

Recall the ex. from page 3:

$$q_S \longmapsto +(0,a,q_S) \qquad q_H \longmapsto +(0,a,q_S)$$

$$\Phi(M) \left\{ \begin{array}{l} \left( \forall x \left( x = ASS(MPROG(M), START) \right. \right. \\ \qquad \longrightarrow +(x) \wedge IREG(x) = Z \\ \qquad \left. \left. \wedge\ ISYM(x) = a \wedge IST1(x) = START \right) \right. \\ \wedge \left( x = ASS(MPROG(M), HALT) \right. \\ \qquad \longrightarrow +(x) \wedge IREG(x) = Z \\ \qquad \left. \wedge\ ISYM(x) = a \wedge IST1(x) = START \right) \end{array} \right.$$

Then $\varphi_2(M) \wedge \Phi(M)$ describes that M is the machine from page 3.

Similarly, for any $w$ s.t. $w = code(M)$, I can find a formula $\varphi_w(M)$ s.t. $\varphi_w(M)$ states that M is described by $code(M)$.

I.e., M represents M in our logical languages.

## IMPORTANT REMARK

> Because of the nature of the encoding code(M) and the given formula $\varphi_w$, the map
>
> $$w \longmapsto \varphi_w$$
>
> can be performed by a TM.

Very similarly, we get assignments of formulas for describing sequences of words or configurations:

① If $w = code(\vec{w})$, then $\psi_w(x)$ describes that $x$ represents the seq. $\vec{w}$

② If $C$ is a configuration and $w = code(C)$ then $\xi_w(x)$ describes that $x$ represents the configuration $C$.

Now let's encode being the computation sequence in our language.

Add two new symbols:

TRANS — transformation — binary function

22. $\forall M \forall x \; CONFIG(x) \longrightarrow CONFIG(TRANS(M,x))$

COMP — computation sequence — ternary function

23. $\forall M \forall x \forall n \; NAT(u) \longrightarrow CONFIG(COMP(M,x,u))$.

Let's describe the definition of transformation in our language:

$$\forall u \Big[ u \neq IREG(ASS(MPROG(M), CSTATE(x))) \longrightarrow [PRO](CCONTENT(TRANS(M,x)), u) = PRO(CCONTENT(x), u) \Big]$$

$$\wedge \; u = IREG(ASS(MPROG(M), CSTATE(x))) \longrightarrow + (ASS(MPROG(M), CSTATE(x)))$$

$$\longrightarrow \Big[ \; [\text{NEXT PAGE}] \; \Big]$$

PROJ (CCONTENT (TRANS (M, x)), u)
= ADD ( PROJ (CCONTENT (x), u),
ISYM (ASS (MPROG (M),
CSTATE (x))))

∧
CSTATE (TRANS (M, x)) =
IST1 (ASS (MPROG (M), CSTATE (x)))

∧ three more partial formulas of this type describing the behaviour of $\Sigma(u, a, q, q')$
$?(u, \varepsilon, q, q')$
$-(u, q, q')$

$\Xi(x, M)$

This very long formula describes that TRANS(M,x) is the transformation of a configuration x by a machine M.

24. $\forall x \forall M \; \Xi(x, M)$.

Back to the computation sequence :

25. $\forall x \forall M$ CSTATE (COMP (M, x, Z)) = START
26. $\forall x \forall M$ CCONTENT (COMP (M, x, Z)) = x
27. $\forall x \forall M \forall u$ COMP (M, x, S(u)) = TRANS (COMP (M, x, u))

This now allows us to formulate that M halts at input x:

$$\exists n \; NAT(n) \wedge CSTATE(COMP(\dot{M}, x, n)) = HALT$$

$$\| \; !!$$

$$\psi(\dot{M}, x)$$

$\psi$ describes "M halts at input x".

Remember we had $\psi_w(x)$ describing "x is the seq. $\vec{w}$ whose code$(\vec{w}) = w$".

Now specialize to

$\psi^*_w(x)$ describing "x is the seq. of length 1 with unique element w".

Let $\Phi$ be the conjunction of all of our axioms 1. to 27.

$$\sigma_w := \forall M \forall x \; \overline{\Phi} \longrightarrow [\varphi_w(M) \wedge \psi^*_w(x) \wedge \psi(M, x)]$$

**Theorem** The map $w \longmapsto \sigma_w$ is a reduction of $K$ to $\{\varphi : \vdash \varphi\}$.

**Corollary** This implies that the Entscheidungsproblem is unsolvable.

**Proof of Thm.** By earlier observations, the map is computable, so we only need to show

$$w \in K \iff \vdash \sigma_w .$$

**Remarks of Gödel's completeness thm**

For all $\sigma$, $\vdash \sigma \iff \models \sigma$
$\iff$ for all $S$, we have $S \models \sigma$.

In particular, if $\vdash \sigma_w$ is true, there is the structure containing real computations, configurations etc where we interpret all symbols intended in $\mathcal{L}$ by their real meanings, $\sigma_w$ must be true. But that means that the computation of $f_{w,1}(w)$ halts.

So $w \in K$.

**Remark** The other direction must be slightly harder since $wEF$ expresses that $\sigma w$ is true, but by Gödel's incompleteness theorem (which we haven't proved yet) this is in general not equivalent to $\vdash \sigma w$.

Let's work slightly harder. Assume $wEF$.
So this means that

$$f_{w,1}(w)\downarrow,$$

i.e. there is a natural number $n$ s.t. the computation halts after $n$ steps.
In other words, we have

$$C_0, C_1, \ldots, C_n \quad \text{s.t.} \quad (*)$$

$C_0$ is the config. $(q_S, w)$
$C_n$ is a config $(q_H, \vec{v})$

(*) follows from axioms 25. & 26.

But then $C_1$ must be the result of applying the transformation operation of $M$ to $C_0$

Let $w_i$ be words s.t.

$$w_i = \text{code}(C_i).$$

Then $\xi_{w_i}(x)$ means that $x$ represents $C_i$ and therefore 25. & 26. imply

$$\forall M \forall x \; \xi_{w_0}(y) \wedge \varphi_w(M) \wedge \psi_w^*(x) \longrightarrow \text{COMP}(M, x, z) = y$$

Similary, 24. and the above imply

$$\forall M \forall x \forall y \; \xi_{w_1}(y) \wedge \varphi_w(M) \wedge \psi_w^*(x) \longrightarrow \text{COMP}(M, x, S(z)) = y$$

So formulate the formula $\eta_i$

$$\eta_i := \forall M \forall x \forall y \; \xi_{w_i}(y) \wedge \varphi_w(M) \wedge \psi_w^*(x) \longrightarrow \text{COMP}(M, x, S^i(z)) = y$$

Since $\eta_0, \eta_1$ have already been proved, we can prove by induction that for all $i$

$$\vdash \eta_i .$$

In particular, $\vdash \eta_u$ where $u$ is the true halting time

But $\gamma_n \longrightarrow \sigma_\omega$.

So $\vdash \sigma_\omega$.

q.e.d.

Remark. (1) Observe that the second implication requires an external proof using induction on $\mathbb{N}$ since internally we do not have enough axioms to make sure that NAT behaves like $\mathbb{N}$.

(2) We shall not be able to properly describe $\mathbb{N}$ since there would require infinitely many axioms & we need that $\sigma_\omega$ is a single formula.

Next time: INCOMPLETENESS

We'll show that

$$\{\varphi ; T \vdash \varphi\} \neq \{\varphi ; \varphi \text{ is true in a fixed structure}\}$$

for appropriate languages and T.