#### Undecidability in Logic

# <sup>1</sup>Lane Department of Computer Science and Electrical Engineering West Virginia University

# Incompleteness of Number Theory

Subramani

# Undecidability in Logic -Part II

K. Subramani<sup>1</sup>

# Outline

# Number-theoretic encoding of computation

# 2 Undecidability

- Sentence Classification
- Recursive Inseparability

# Incompleteness

Gödel's Incompleteness Theorem

# Outline



# 2 Undecidability

- Sentence Classification
- Recursive Inseparability



Gödel's Incompleteness Theorem

< 🗇 🕨 < 🖃 >

# Outline



# 2 Undecidability

- Sentence Classification
- Recursive Inseparability



# Incompleteness

### Turing Machine encodings

- Every Turing Machine M = (K, Σ, δ, s)) can be represented as a number in *b*-ary notation, where b = |K| + |Σ|.
- (ii) Therefore, configurations can be encoded as *sequences* of integers in *b*-ary representation.
- (iii) The "yields in one step" function over configurations of a Turing Machine, defines a relation  $Y_M\subseteq \mathcal{N}^2$

### Goal

To formulate a first-order expression yields  $_{M}(x, y)$  in number theory, over the free variables x and y, such that

 $\mathbf{N}_{x=m,y=n} \models yields_M(x, y) \text{ iff } Y_M(m, n).$ 

## **Turing Machine encodings**

- (i) Every Turing Machine M = (K, Σ, δ, s) can be represented as a number in *b*-ary notation, where b = |K| + |Σ|.
- (ii) Therefore, configurations can be encoded as sequences of integers in b-ary representation.
- (iii) The "yields in one step" function over configurations of a Turing Machine, defines a relation  $Y_M\subseteq \mathcal{N}^2$

### Goal

To formulate a first-order expression yields $_{M}(x, y)$  in number theory, over the free variables x and y, such that

 $\mathbf{N}_{x=m,y=n} \models yields_M(x, y) \text{ iff } Y_M(m, n).$ 

## Turing Machine encodings

- (i) Every Turing Machine M = (K, Σ, δ, s)) can be represented as a number in *b*-ary notation, where b = |K| + |Σ|.
- (ii) Therefore, configurations can be encoded as *sequences* of integers in *b*-ary representation.

(iii) The "yields in one step" function over configurations of a Turing Machine, defines a relation  $Y_M\subseteq \mathcal{N}^2$ 

### Goal

To formulate a first-order expression yields  $_{\rm M}(x,y)$  in number theory, over the free variables x and y, such that

 $\mathbf{N}_{x=m,y=n} \models yields_M(x, y) \text{ iff } Y_M(m, n).$ 

# Turing Machine encodings

- (i) Every Turing Machine M = (K, Σ, δ, s)) can be represented as a number in *b*-ary notation, where b = |K| + |Σ|.
- (ii) Therefore, configurations can be encoded as *sequences* of integers in *b*-ary representation.
- (iii) The "yields in one step" function over configurations of a Turing Machine, defines a relation  $Y_M\subseteq \mathcal{N}^2$

### Goal

To formulate a first-order expression yields  $_{\rm M}(x,y)$  in number theory, over the free variables x and y, such that

 $\mathbf{N}_{x=m,y=n} \models yields_M(x, y) \text{ iff } Y_M(m, n).$ 

## Turing Machine encodings

- (i) Every Turing Machine M = (K, Σ, δ, s)) can be represented as a number in *b*-ary notation, where b = |K| + |Σ|.
- (ii) Therefore, configurations can be encoded as *sequences* of integers in *b*-ary representation.

(iii) The "yields in one step" function over configurations of a Turing Machine, defines a relation  $Y_M\subseteq \mathcal{N}^2$ 

## Goal

To formulate a first-order expression yields  $_{M}(x,y)$  in number theory, over the free variables x and y , such that

 $N_{x=m,y=n} \models yields_M(x, y) \text{ iff } Y_M(m, n).$ 

## Turing Machine encodings

- (i) Every Turing Machine M = (K, Σ, δ, s) can be represented as a number in *b*-ary notation, where b = |K| + |Σ|.
- (ii) Therefore, configurations can be encoded as *sequences* of integers in *b*-ary representation.

(iii) The "yields in one step" function over configurations of a Turing Machine, defines a relation  $Y_M\subseteq \mathcal{N}^2$ 

## Goal

To formulate a first-order expression yields  $_{M}(x,y)$  in number theory, over the free variables x and y, such that

 $\mathbf{N}_{x=m,y=n} \models yields_M(x, y) \text{ iff } Y_M(m, n).$ 

$p \in K$ ,	$\sigma \in \Sigma$	$\delta(p, \sigma)$
S	а	(s, a, →)
S	b	$(s, b, \rightarrow)$
S		$(q, \sqcup, \leftarrow)$
S	⊳	$(q, \triangleright, \rightarrow)$
q	а	$(q,\sqcup,\leftarrow)$
q	b	("no", b, -)
q	⊳	$("yes", \triangleright, \rightarrow)$

Table: A Turing Machine that accepts a\*

### Example

Consider the  $C_1 = (q, \triangleright aa, \sqcup \sqcup)$  and the configuration that follows  $C_2 = (q, \triangleright a, \sqcup \sqcup \sqcup)$ . The corresponding encodings  $m = 022711_8$  and  $027111_8$  are related under Y<sub>M</sub>.

### Observation

- (i) *m* and *n* are identical, except for the replacement of  $271_8$  in m, by  $711_8$  in n.
- But this corresponds to Rule 5 in the table! Thus, every move is a local replacement of digits.

$p \in K$ ,	$\sigma \in \Sigma$	$\delta(p, \sigma)$
S	а	(s, a, →)
S	b	$(s, b, \rightarrow)$
S		$(q, \sqcup, \leftarrow)$
S	⊳	$(q, \triangleright, \rightarrow)$
q	а	$(q,\sqcup,\leftarrow)$
q	b	("no", b, -)
q	⊳	$("yes", \triangleright, \rightarrow)$

Table: A Turing Machine that accepts a\*

# Example

Consider the  $C_1 = (q, \triangleright aa, \sqcup \sqcup)$  and the

### configuration that follows

 $C_2 = (q, ▷a, ⊔ ⊔ ⊔)$ . The corresponding encodings  $m = 022711_8$  and  $027111_8$  are related under  $Y_M$ .

### Observation

- (i) *m* and *n* are identical, except for the replacement of 271<sub>8</sub> in m, by 711<sub>8</sub> in n.
- (ii) But this corresponds to Rule 5 in the table! Thus, every move is a local replacement of digits.

$p \in K$ ,	$\sigma \in \Sigma$	$\delta(p, \sigma)$
S	а	$(s, a, \rightarrow)$
S	b	$(s, b, \rightarrow)$
S		$(q, \sqcup, \leftarrow)$
S	⊳	$(q, \triangleright, \rightarrow)$
q	а	$(q,\sqcup,\leftarrow)$
q	b	("no", b, -)
q	⊳	$("yes", \triangleright, \rightarrow)$

Table: A Turing Machine that accepts a\*

### Example

Consider the  $C_1 = (q, \triangleright aa, \sqcup \sqcup)$  and the configuration that follows  $C_2 = (q, \triangleright a, \sqcup \sqcup \sqcup)$ . The corresponding encodings  $m = 022711_8$  and  $027111_8$  are related under  $Y_M$ .

### Observation

- (i) *m* and *n* are identical, except for the replacement of 271<sub>8</sub> in m, by 711<sub>8</sub> in n.
- But this corresponds to Rule 5 in the table! Thus, every move is a local replacement of digits.

$p \in K$ ,	$\sigma \in \Sigma$	$\delta(p, \sigma)$
S	а	(s, a, →)
S	b	$(s, b, \rightarrow)$
S		$(q, \sqcup, \leftarrow)$
S	⊳	$(q, \triangleright, \rightarrow)$
q	а	$(q,\sqcup,\leftarrow)$
q	b	("no", b, -)
q	⊳	$("yes", \triangleright, \rightarrow)$

Table: A Turing Machine that accepts a\*

# Example

Consider the  $C_1 = (q, \triangleright aa, \sqcup \sqcup)$  and the configuration that follows  $C_2 = (q, \triangleright a, \sqcup \sqcup \sqcup)$ . The corresponding encodings  $m = 022711_8$  and  $027111_8$  are related under  $Y_M$ .

## Observation

 (i) m and n are identical, except for the replacement of 271<sub>8</sub> in m, by 711<sub>8</sub> in n.

<ロ> <同> <同> < 同> < 同> < 同> <

3

But this corresponds to Rule 5 in the table! Thus, every move is a local replacement of digits.

$p \in K$ ,	$\sigma \in \Sigma$	$\delta(p, \sigma)$
S	а	(s, a, →)
S	b	$(s, b, \rightarrow)$
S		$(q, \sqcup, \leftarrow)$
S	⊳	$(q, \triangleright, \rightarrow)$
q	а	$(q,\sqcup,\leftarrow)$
q	b	("no", b, -)
q	⊳	$("yes", \triangleright, \rightarrow)$

Table: A Turing Machine that accepts a\*

### Example

Consider the  $C_1 = (q, \triangleright aa, \sqcup \sqcup)$  and the configuration that follows  $C_2 = (q, \triangleright a, \sqcup \sqcup \sqcup)$ . The corresponding encodings  $m = 022711_8$  and  $027111_8$  are related under Y<sub>M</sub>.

# Observation

- (i) *m* and *n* are identical, except for the replacement of 271<sub>8</sub> in *m*, by 711<sub>8</sub> in *n*.
- But this corresponds to Rule 5 in the table! Thus, every move is a local replacement of digits.

$p \in K$ ,	$\sigma \in \Sigma$	$\delta(p, \sigma)$
S	а	(s, a, →)
S	b	$(s, b, \rightarrow)$
S		$(q, \sqcup, \leftarrow)$
S	⊳	$(q, \triangleright, \rightarrow)$
q	а	$(q,\sqcup,\leftarrow)$
q	b	("no", b, -)
q	⊳	$("yes", \triangleright, \rightarrow)$

Table: A Turing Machine that accepts a\*

### Example

Consider the  $C_1 = (q, \triangleright aa, \sqcup \sqcup)$  and the configuration that follows  $C_2 = (q, \triangleright a, \sqcup \sqcup \sqcup)$ . The corresponding encodings  $m = 022711_8$  and  $027111_8$  are related under  $Y_M$ .

# Observation

- (i) *m* and *n* are identical, except for the replacement of 271<sub>8</sub> in *m*, by 711<sub>8</sub> in *n*.
- But this corresponds to Rule 5 in the table! Thus, every move is a local replacement of digits.

$p \in K$ ,	$\sigma \in \Sigma$	$\delta(p, \sigma)$
S	а	(s, a, →)
S	b	$(s, b, \rightarrow)$
S		$(q, \sqcup, \leftarrow)$
S	⊳	$(q, \triangleright, \rightarrow)$
q	а	$(q,\sqcup,\leftarrow)$
q	b	("no", b, -)
q	⊳	$("yes", \triangleright, \rightarrow)$

Table: A Turing Machine that accepts a\*

### Example

Consider the  $C_1 = (q, \triangleright aa, \sqcup \sqcup)$  and the configuration that follows  $C_2 = (q, \triangleright a, \sqcup \sqcup \sqcup)$ . The corresponding encodings  $m = 022711_8$  and  $027111_8$  are related under Y<sub>M</sub>.

# Observation

- (i) *m* and *n* are identical, except for the replacement of 271<sub>8</sub> in *m*, by 711<sub>8</sub> in *n*.
- But this corresponds to Rule 5 in the table! Thus, every move is a local replacement of digits.

# Triplet changes

Capture each rule change as a triplet transformation!

Table: Encoding the triplet changes

 $table_M(x, y) = ((x = 042_8 \land y = 024_8) \lor \dots (x = 371_8 \land y = 361_8))$ 

<ロ> <同> <同> < 同> < 同> < 同> 、

# **Triplet changes**

Capture each rule change as a triplet transformation!

Table: Encoding the triplet changes

 $table_M(x, y) = ((x = 042_8 \land y = 024_8) \lor \dots (x = 371_8 \land y = 361_8))$ 

<ロ> <同> <同> < 回> < 回> < 回> < 回> <

# **Triplet changes**

Capture each rule change as a triplet transformation!

$042_8 \rightarrow$	024 <sub>8</sub>
$043_8 \rightarrow$	0348
$041_8 \rightarrow$	014 <sub>8</sub>
$242_8 \rightarrow$	224 <sub>8</sub>
$243_8 \rightarrow$	234 <sub>8</sub>
$241_8 \rightarrow$	214 <sub>8</sub>
$242_8 \rightarrow$	224 <sub>8</sub>
$242_8\rightarrow$	224 <sub>8</sub>
	:
$371_8 \rightarrow$	361 <sub>8</sub>

Table: Encoding the triplet changes

 $table_M(x, y) = ((x = 042_8 \land y = 024_8) \lor \dots (x = 371_8 \land y = 361_8))$ 

<ロ> <同> <同> < 回> < 回> < 回> < 回> <

# **Triplet changes**

Capture each rule change as a triplet transformation!

$042_8 \rightarrow$	024 <sub>8</sub>
$043_8 \rightarrow$	0348
$041_8 \rightarrow$	014 <sub>8</sub>
$242_8 \rightarrow$	224 <sub>8</sub>
$243_8 \rightarrow$	234 <sub>8</sub>
$241_8 \rightarrow$	214 <sub>8</sub>
$242_8 \rightarrow$	224 <sub>8</sub>
$242_8\rightarrow$	224 <sub>8</sub>
:	:
$371_8 \rightarrow$	3618

Table: Encoding the triplet changes

 $table_M(x, y) = ((x = 042_8 \land y = 024_8) \lor \dots (x = 371_8 \land y = 361_8))$ 

Formulating the encoding technique (contd.)

# Padding

The computation of *M* on input aa:

**0422**<sub>8</sub>, **0242**<sub>8</sub>, 0224<sub>8</sub>, 02241<sub>8</sub>, .

<ロ> <同> < 回> < 回> < 回> < => < => <</p>

Formulating the encoding technique (contd.)

# Padding

The computation of *M* on input aa:

0422<sub>8</sub>, 0242<sub>8</sub>, 0224<sub>8</sub>, 02241<sub>8</sub>, ....

<ロ> <同> < 回> < 回> < 回> < => < => <</p>

Formulating the encoding technique (contd.)

# Padding

The computation of *M* on input aa:

0422<sub>8</sub>, 0242<sub>8</sub>, 0224<sub>8</sub>, 02241<sub>8</sub>, ....

<ロ> <同> < 回> < 回> < 回> < => < => <</p>

Formulating the encoding technique (contd.)

# Padding

The computation of *M* on input aa:

0422<sub>8</sub>, 0242<sub>8</sub>, 0224<sub>8</sub>, 02241<sub>8</sub>, ....

<ロ> <同> < 回> < 回> < 回> < => < => <</p>

Formulating the encoding technique (contd.)

# Padding

The computation of *M* on input aa:

 $0422_8, 0242_8, 0224_8, 02241_8, \ldots$ 

<ロ> <同> < 回> < 回> < 回> < => < => <</p>

A first-order Number-Theoretic expression for Computation

# The Actual Formula

 $\begin{aligned} yields_M(x,x') &= pads_M(x,x') \lor \\ (\exists y < x)(\exists z_1 < x)(\exists z_2 < x)(\exists z_2' < x)(\exists z_3 < x)(\exists z_3' < x)(\exists z_4 < x) \\ (conf_M(x) \land conf_M(x') \land \\ mod (x, b \uparrow y, z_1) \land div(x, b \uparrow y, z_2) \land mod (x', b \uparrow y, z_1) \land div(x', b \uparrow y, z_2') \land \\ mod (z_2, b \uparrow 3, z_3) \land div(z_2, b \uparrow 3, z_4) \land mod (z_2', b \uparrow 3, z_3') \land div(z_2', b \uparrow 3, z_4) \land \\ table_M(z_3, z_3') \end{aligned}$ 

### Auxiliary expressions

Similar expressions can be written for  $pads_M(x, x')$  (x' is obtained from x by adding a  $\sqcup$ ) and  $conf_M(x)$  (the *b*-ary representation of x correctly encodes a configuration of M).

A first-order Number-Theoretic expression for Computation

# The Actual Formula

 $yields_M(x, x') = pads_M(x, x') \lor$ 

 $(\exists y < x)(\exists z_1 < x)(\exists z_2 < x)(\exists z_2' < x)(\exists z_3 < x)(\exists z_3' < x)(\exists z_4 < x)(x)(z z_4 < x)(z = x)($ 

 $(conf_M(x) \land conf_M(x') \land$ 

 $\mathsf{mod} \ (\mathbf{x}, b \uparrow \mathbf{y}, \mathbf{z}_1) \land \mathsf{div}(\mathbf{x}, b \uparrow \mathbf{y}, \mathbf{z}_2) \land \mathsf{mod} \ (\mathbf{x}', b \uparrow \mathbf{y}, \mathbf{z}_1) \land \mathsf{div}(\mathbf{x}', b \uparrow \mathbf{y}, \mathbf{z}_2') \land$ 

 $\mathsf{mod} \ (z_2, b \uparrow 3, z_3) \land \mathsf{div}(z_2, b \uparrow 3, z_4) \land \mathsf{mod} \ (z'_2, b \uparrow 3, z'_3) \land \mathsf{div}(z'_2, b \uparrow 3, z_4) \land$ 

 $able_M(z_3, z_3')$ 

< ロ > < 同 > < 三 > < 三 > -

#### Auxiliary expressions

Similar expressions can be written for  $pads_M(x, x')$  (x' is obtained from x by adding a  $\sqcup$ ) and  $conf_M(x)$  (the *b*-ary representation of x correctly encodes a configuration of M).

A first-order Number-Theoretic expression for Computation

### The Actual Formula

 $yields_M(x, x') = pads_M(x, x') \lor$ 

 $(\exists y < x)(\exists z_1 < x)(\exists z_2 < x)(\exists z'_2 < x)(\exists z_3 < x)(\exists z'_3 < x)(\exists z_4 < x)$ 

 $(conf_M(x) \land conf_M(x') \land$ 

 $\mod (x, b \uparrow y, z_1) \land \operatorname{div}(x, b \uparrow y, z_2) \land \mod (x', b \uparrow y, z_1) \land \operatorname{div}(x', b \uparrow y, z_2') \land$ 

 $\mathsf{mod} \ (z_2, b \uparrow 3, z_3) \land \mathsf{div}(z_2, b \uparrow 3, z_4) \land \mathsf{mod} \ (z_2', b \uparrow 3, z_3') \land \mathsf{div}(z_2', b \uparrow 3, z_4) \land$ 

 $table_M(z_3, z'_3)$ 

#### Auxiliary expressions

Similar expressions can be written for  $pads_M(x, x')$  (x' is obtained from x by adding a  $\sqcup$ ) and  $conf_M(x)$  (the *b*-ary representation of x correctly encodes a configuration of M).

A first-order Number-Theoretic expression for Computation

### The Actual Formula

 $yields_{M}(x, x') = pads_{M}(x, x') \lor \\ (\exists y < x)(\exists z_{1} < x)(\exists z_{2} < x)(\exists z_{2}' < x)(\exists z_{3} < x)(\exists z_{3}' < x)(\exists z_{4} < x) \\ (conf_{M}(x) \land conf_{M}(x') \land \\ d(x, b \uparrow y, z_{1}) \land div(x, b \uparrow y, z_{2}) \land mod(x', b \uparrow y, z_{1}) \land div(x', b \uparrow y, z_{2}') \land \\ (z_{2}, b \uparrow \exists, z_{3}) \land div(z_{2}, b \uparrow \exists, z_{4}) \land mod(z_{2}', b \uparrow \exists, z_{4}) \land \\ (b) = (z_{2}, z_{1}') \land div(z_{2}, b \uparrow \exists, z_{4}) \land mod(z_{2}', b \uparrow \exists, z_{4}) \land \\ (b) = (z_{2}, z_{1}') \land div(z_{2}', b \uparrow \exists, z_{4}) \land mod(z_{2}', b \uparrow \exists, z_{4}) \land \\ (b) = (z_{2}, z_{1}') \land div(z_{2}', b \uparrow \exists, z_{4}) \land \\ (b) = (z_{2}, z_{1}') \land div(z_{2}', b \uparrow \exists, z_{4}) \land \\ (b) = (z_{2}, z_{1}') \land div(z_{2}', b \uparrow \exists, z_{4}) \land \\ (b) = (z_{1}, z_{2}', b \uparrow div(z_{2}', b \land div(z_{2}', b \uparrow div(z_{2}', b \land div(z_{2}$ 

### Auxiliary expressions

Similar expressions can be written for  $pads_M(x, x')$  (x' is obtained from x by adding a  $\sqcup$ ) and  $conf_M(x)$  (the *b*-ary representation of x correctly encodes a configuration of M).

A first-order Number-Theoretic expression for Computation

### The Actual Formula

 $\begin{aligned} yields_M(x,x') &= pads_M(x,x') \lor \\ (\exists y < x)(\exists z_1 < x)(\exists z_2 < x)(\exists z_2' < x)(\exists z_3 < x)(\exists z_3' < x)(\exists z_4 < x) \\ (conf_M(x) \land conf_M(x') \land \\ (x, b \uparrow y, z_1) \land div(x, b \uparrow y, z_2) \land \mod (x', b \uparrow y, z_1) \land div(x', b \uparrow y, z_2') \land \\ (z_2, b \uparrow 3, z_3) \land div(z_2, b \uparrow 3, z_4) \land \mod (z_2', b \uparrow 3, z_3') \land div(z_2', b \uparrow 3, z_4) \land \end{aligned}$ 

#### Auxiliary expressions

Similar expressions can be written for  $pads_M(x, x')$  (x' is obtained from x by adding a  $\sqcup$ ) and  $conf_M(x)$  (the *b*-ary representation of x correctly encodes a configuration of M).

A first-order Number-Theoretic expression for Computation

# The Actual Formula

m

$$yields_M(x, x') = pads_M(x, x') \lor$$

$$(\exists y < x)(\exists z_1 < x)(\exists z_2 < x)(\exists z'_2 < x)(\exists z'_3 < x)(\exists z'_3 < x)(\exists z'_4 < x)$$

$$(conf_M(x) \land conf_M(x') \land$$

$$d(x, b \uparrow y, z_1) \land \operatorname{div}(x, b \uparrow y, z_2) \land \mod (x', b \uparrow y, z_1) \land \operatorname{div}(x', b \uparrow y, z'_2) \land$$

$$d(z_2, b \uparrow 3, z_3) \land \operatorname{div}(z_2, b \uparrow 3, z_4) \land \mod (z'_2, b \uparrow 3, z'_3) \land \operatorname{div}(z'_2, b \uparrow 3, z_4) \land$$

$$(z'_2, b \uparrow 3, z'_3) \land \operatorname{div}(z'_2, b \uparrow 3, z'_4) \land$$

#### Auxiliary expressions

Similar expressions can be written for  $pads_M(x, x')$  (x' is obtained from x by adding a  $\sqcup$ ) and  $conf_M(x)$  (the *b*-ary representation of x correctly encodes a configuration of M).

< ロ > < 同 > < 三 > < 三 > 、

E

A first-order Number-Theoretic expression for Computation

# The Actual Formula

$$\begin{aligned} yields_M(x,x') &= pads_M(x,x') \lor \\ (\exists y < x)(\exists z_1 < x)(\exists z_2 < x)(\exists z_2' < x)(\exists z_3 < x)(\exists z_3' < x)(\exists z_4 < x) \\ (conf_M(x) \land conf_M(x') \land \\ mod \ (x,b \uparrow y,z_1) \land \operatorname{div}(x,b \uparrow y,z_2) \land \mod (x',b \uparrow y,z_1) \land \operatorname{div}(x',b \uparrow y,z_2') \land \\ mod \ (z_2,b \uparrow 3,z_3) \land \operatorname{div}(z_2,b \uparrow 3,z_4) \land \mod (z_2',b \uparrow 3,z_3') \land \operatorname{div}(z_2',b \uparrow 3,z_4) \land \\ table_M(z_3,z_3') \end{aligned}$$

### Auxiliary expressions

Similar expressions can be written for  $pads_M(x, x')$  (x' is obtained from x by adding a  $\sqcup$ ) and  $conf_M(x)$  (the *b*-ary representation of x correctly encodes a configuration of M).

A first-order Number-Theoretic expression for Computation

# The Actual Formula

$$\begin{aligned} yields_M(x,x') &= pads_M(x,x') \lor \\ (\exists y < x)(\exists z_1 < x)(\exists z_2 < x)(\exists z_2' < x)(\exists z_3 < x)(\exists z_3' < x)(\exists z_4 < x) \\ (conf_M(x) \land conf_M(x') \land \\ mod \ (x,b \uparrow y,z_1) \land \operatorname{div}(x,b \uparrow y,z_2) \land \mod (x',b \uparrow y,z_1) \land \operatorname{div}(x',b \uparrow y,z_2') \land \\ mod \ (z_2,b \uparrow 3,z_3) \land \operatorname{div}(z_2,b \uparrow 3,z_4) \land \mod (z_2',b \uparrow 3,z_3') \land \operatorname{div}(z_2',b \uparrow 3,z_4) \land \\ table_M(z_3,z_3') \end{aligned}$$

### Auxiliary expressions

Similar expressions can be written for  $pads_M(x, x')$  (x' is obtained from x by adding a  $\sqcup$ ) and  $conf_M(x)$  (the *b*-ary representation of x correctly encodes a configuration of M).

(二)、(同)、(三)、(三)、

A first-order Number-Theoretic expression for Computation

# The Actual Formula

$$\begin{aligned} yields_M(x,x') &= pads_M(x,x') \lor \\ (\exists y < x)(\exists z_1 < x)(\exists z_2 < x)(\exists z_2' < x)(\exists z_3 < x)(\exists z_3' < x)(\exists z_4 < x) \\ (conf_M(x) \land conf_M(x') \land \\ mod \ (x,b \uparrow y,z_1) \land \operatorname{div}(x,b \uparrow y,z_2) \land \mod (x',b \uparrow y,z_1) \land \operatorname{div}(x',b \uparrow y,z_2') \land \\ mod \ (z_2,b \uparrow 3,z_3) \land \operatorname{div}(z_2,b \uparrow 3,z_4) \land \mod (z_2',b \uparrow 3,z_3') \land \operatorname{div}(z_2',b \uparrow 3,z_4) \land \\ table_M(z_3,z_3') \end{aligned}$$

# Auxiliary expressions

Similar expressions can be written for  $pads_M(x, x')$  (x' is obtained from x by adding a  $\square$ ) and  $conf_M(x)$  (the *b*-ary representation of *x* correctly encodes a configuration of *M*).

### A first-order Number-Theoretic expression for Computation (contd.)

### Observation

Whole computations of M can be encoded!

#### \_emma

For each Turing machine M, we can construct a bounded expression  $comp_M(x)$  in number theory such that:  $\forall n \in \mathcal{N}, \ N_{x=n} \models comp_M(x) \leftrightarrow$  the b-ary representation of n is the juxtaposition of consecutive configurations of a halting computation of M, starting from the empty string.

<ロ> <同> <同> < 同> < 同> < 同>
#### A first-order Number-Theoretic expression for Computation (contd.)

### Observation

Whole computations of M can be encoded!

#### \_emma

For each Turing machine M, we can construct a bounded expression  $comp_M(x)$  in number theory such that:  $\forall n \in \mathcal{N}, \ N_{x=n} \models comp_M(x) \leftrightarrow$  the b-ary representation of n is the juxtaposition of consecutive configurations of a halting computation of M, starting from the empty string.

#### A first-order Number-Theoretic expression for Computation (contd.)

### Observation

Whole computations of M can be encoded!

#### emma

For each Turing machine M, we can construct a bounded expression  $comp_M(x)$  in number theory such that:  $\forall n \in \mathcal{N}, \ N_{x=n} \models comp_M(x) \leftrightarrow$  the b-ary representation of n is the juxtaposition of consecutive configurations of a halting computation of M, starting from the empty string.

< ロ > < 同 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ >

A first-order Number-Theoretic expression for Computation (contd.)

### Observation

Whole computations of M can be encoded!

#### Lemma

For each Turing machine *M*, we can construct a bounded expression  $comp_M(x)$  in number theory such that:  $\forall n \in \mathcal{N}, \ \mathbf{N}_{x=n} \models comp_M(x) \leftrightarrow$  the b-ary representation of *n* is the juxtaposition of consecutive configurations of a halting computation of *M*, starting from the empty string.

Number-theoretic encoding of computation Undecidability Incompleteness

Sentence Classification Recursive Inseparability

# Outline

# Number-theoretic encoding of computation

# 2 Undecidability

- Sentence Classification
- Recursive Inseparability

# 3 Incompleteness

- (i)  $\phi$  is valid, i.e.,  $\models \phi(L_v)$ .
- (ii)  $\phi$  is provable from **NT**, i.e., **NT**  $\vdash \phi$  (*L*<sub>*p*</sub>).
- (iii) **N** is a model for  $\phi$ , i.e., **N**  $\models \phi$  (*L*<sub>m</sub>).
- (iv) N is a model for  $\neg \phi$ , i.e., N  $\models \neg \phi$  ( $L_{nm}$ ).
- (v)  $\neg \phi$  is provable from **NT**, i.e., **NT**  $\vdash \neg \phi$  (*L*<sub>*np*</sub>).
- (vi)  $\neg \phi$  is valid, i.e.,  $\models \neg \phi$  (*L*<sub>us</sub>).

<ロ> <同> < 回> < 回> < 回> < => < => <</p>

æ.

- (i)  $\phi$  is valid, i.e.,  $\models \phi(L_v)$ .
- (ii)  $\phi$  is provable from **NT**, i.e., **NT**  $\vdash \phi$  ( $L_p$ ).
- (iii) **N** is a model for  $\phi$ , i.e., **N**  $\models \phi$  (*L<sub>m</sub>*).
- (iv) N is a model for  $\neg \phi$ , i.e., N  $\models \neg \phi$  ( $L_{nm}$ ).
- (v)  $\neg \phi$  is provable from **NT**, i.e., **NT**  $\vdash \neg \phi$  ( $L_{np}$ ).
- (vi)  $\neg \phi$  is valid, i.e.,  $\models \neg \phi$  (*L*<sub>us</sub>).

<ロ> <同> < 回> < 回> < 回> < => < => <</p>

æ.

Sentence Classification Recursive Inseparability

#### Sentence Classification

- (i)  $\phi$  is valid, i.e.,  $\models \phi(L_v)$ .
- (ii)  $\phi$  is provable from **NT**, i.e., **NT**  $\vdash \phi$  ( $L_{\rho}$ ).
- (iii) **N** is a model for  $\phi$ , i.e., **N**  $\models \phi$  (*L*<sub>m</sub>).
- (iv) **N** is a model for  $\neg \phi$ , i.e., **N**  $\models \neg \phi$  (*L*<sub>nm</sub>).
- (v)  $\neg \phi$  is provable from **NT**, i.e., **NT**  $\vdash \neg \phi$  (*L<sub>np</sub>*).
- (vi)  $\neg \phi$  is valid, i.e.,  $\models \neg \phi$  (*L*<sub>us</sub>).

< ロ > < 回 > < 回 > < 回 > < 回 > <</p>

E

- (i)  $\phi$  is valid, i.e.,  $\models \phi(L_v)$ .
- (ii)  $\phi$  is provable from **NT**, i.e., **NT**  $\vdash \phi$  ( $L_{\rho}$ ).
- (iii) **N** is a model for  $\phi$ , i.e., **N**  $\models \phi$  (*L<sub>m</sub>*).
- (iv) **N** is a model for  $\neg \phi$ , i.e., **N**  $\models \neg \phi$  ( $L_{nm}$ ).
- (v)  $\neg \phi$  is provable from **NT**, i.e., **NT**  $\vdash \neg \phi$  (*L*<sub>*np*</sub>).
- (vi)  $\neg \phi$  is valid, i.e.,  $\models \neg \phi$  (*L*<sub>us</sub>).

<ロ> <同> <同> < 同> < 同> < 同> <

- (i)  $\phi$  is valid, i.e.,  $\models \phi(L_v)$ .
- (ii)  $\phi$  is provable from **NT**, i.e., **NT**  $\vdash \phi$  ( $L_{\rho}$ ).
- (iii) **N** is a model for  $\phi$ , i.e., **N**  $\models \phi$  (*L<sub>m</sub>*).
- (iv) **N** is a model for  $\neg \phi$ , i.e., **N**  $\models \neg \phi$  (*L*<sub>nm</sub>).
- (v)  $\neg \phi$  is provable from **NT**, i.e., **NT**  $\vdash \neg \phi$  (*L<sub>np</sub>*).
- (vi)  $\neg \phi$  is valid, i.e.,  $\models \neg \phi$  (*L*<sub>us</sub>).

<ロ> <同> <同> < 同> < 同> < 同> <

- (i)  $\phi$  is valid, i.e.,  $\models \phi(L_v)$ .
- (ii)  $\phi$  is provable from **NT**, i.e., **NT**  $\vdash \phi$  ( $L_{\rho}$ ).
- (iii) **N** is a model for  $\phi$ , i.e., **N**  $\models \phi$  (*L<sub>m</sub>*).
- (iv) **N** is a model for  $\neg \phi$ , i.e., **N**  $\models \neg \phi$  (*L*<sub>nm</sub>).
- (v)  $\neg \phi$  is provable from **NT**, i.e., **NT**  $\vdash \neg \phi$  (*L*<sub>*np*</sub>).

(vi)  $\neg \phi$  is valid, i.e.,  $\models \neg \phi$  (*L*<sub>us</sub>).

<ロ> <同> <同> < 同> < 同> < 同> <

- (i)  $\phi$  is valid, i.e.,  $\models \phi(L_v)$ .
- (ii)  $\phi$  is provable from **NT**, i.e., **NT**  $\vdash \phi$  ( $L_{\rho}$ ).
- (iii) **N** is a model for  $\phi$ , i.e., **N**  $\models \phi$  (*L<sub>m</sub>*).
- (iv) **N** is a model for  $\neg \phi$ , i.e., **N**  $\models \neg \phi$  (*L*<sub>nm</sub>).
- (v)  $\neg \phi$  is provable from **NT**, i.e., **NT**  $\vdash \neg \phi$  (*L*<sub>*np*</sub>).
- (vi)  $\neg \phi$  is valid, i.e.,  $\models \neg \phi$  (*L*<sub>us</sub>).

<ロ> <同> <同> < 同> < 同> < 同> <

Number-theoretic encoding of computation Undecidability Incompleteness

Sentence Classification Recursive Inseparability

# Outline

# Number-theoretic encoding of computation

# 2 Undecidability

- Sentence Classification
- Recursive Inseparability

# Incompleteness

#### **Recursive Inseparability**

### Definition

Two languages  $L_1$  and  $L_2$  are said to be recursively inseparable, if there does not exist a recursive language R such that  $L_1 \cap R = \emptyset$  and  $L_2 \subset R$ .

#### Theorem

Let  $L_1 = \{M : M(M) = "yes''\}$  and  $L_2 = \{M : M(M) = "no''\}$ .  $L_1$  and  $L_2$  are recursively incomprehended.

### Corollary (Inseparability of halting on empty string)

Let  $L_y = \{M : M(\epsilon) = "yes''\}$  and  $L_n = \{M : M(\epsilon) = "no''\}$ .  $L_y$  and  $L_n$  are recursively inseparable.

イロト イ団ト イヨト イヨト

#### **Recursive Inseparability**

### Definition

Two languages  $L_1$  and  $L_2$  are said to be recursively inseparable, if there does not exist a recursive language R such that  $L_1 \cap R = \emptyset$  and  $L_2 \subset R$ .

### Theorem

Let  $L_1 = \{M : M(M) = "yes''\}$  and  $L_2 = \{M : M(M) = "no''\}$ .  $L_1$  and  $L_2$  are recursively inseparable.

### Corollary (Inseparability of halting on empty string)

Let  $L_y = \{M : M(\epsilon) = "yes''\}$  and  $L_n = \{M : M(\epsilon) = "no''\}$ .  $L_y$  and  $L_n$  are recursively inseparable.

イロン イ部 とくほど くほとう ほう

#### **Recursive Inseparability**

### Definition

Two languages  $L_1$  and  $L_2$  are said to be recursively inseparable, if there does not exist a recursive language R such that  $L_1 \cap R = \emptyset$  and  $L_2 \subset R$ .

#### Theorem

Let  $L_1 = \{M : M(M) = "yes''\}$  and  $L_2 = \{M : M(M) = "no''\}$ .  $L_1$  and  $L_2$  are recursively inseparable.

### Corollary (Inseparability of halting on empty string)

Let  $L_y = \{M : M(\epsilon) = "yes''\}$  and  $L_n = \{M : M(\epsilon) = "no''\}$ .  $L_y$  and  $L_n$  are recursively inseparable.

(日)

### Theorem

L<sub>p</sub> and L<sub>us</sub> are recursively inseparable.

### Proof.

**Main idea:** Given an Turing Machine *M*, we construct an expression  $\phi_M$  such that if  $M(\epsilon) = "yes"$ , then NT  $\vdash \phi_M$  and if  $M(\epsilon) = "no"$ , them  $\phi_M$  is unsatisfiable. Assume that there exists an algorithm  $\mathcal{A}$  to separate  $L_p$  from  $L_{us}$ , i.e.,  $\mathcal{A}$  separates the true properties of integers from the unsatisfiable sentences. But now we can separate  $L_p$  and  $L_n$ ! Given an arbitrary Turing machine *M*, construct  $\phi_M$  and then provide it to  $\mathcal{A}$ ! What is  $\phi_M$ ?  $\phi_M = \mathsf{NT} \land \psi$ , where,

 $\psi = (\exists x)(comp_M(x) \land ((\forall y < x) \neg comp_M(y)) \land \mod (x, b \uparrow 2, b \cdot (|\Sigma| + 1)))$ 

 $\psi$  states that there exists a smallest integer, which encodes an accepting computation of M.

### Theorem

L<sub>p</sub> and L<sub>us</sub> are recursively inseparable.

### Proof.

Main idea: Given an Turing Machine *M*, we construct an expression  $\phi_M$  such that if  $M(\epsilon) = "yes"$ , then **NT**  $\vdash \phi_M$  and if  $M(\epsilon) = "no"$ , them  $\phi_M$  is unsatisfiable. Assume that there exists an algorithm  $\mathcal{A}$  to separate  $L_0$  from  $L_{ie}$ , i.e.,  $\mathcal{A}$  separates the true properties of integers from the unsatisfiable sentences. But now we can separate  $L_2$  and  $L_0$ .

 $\phi_M = \mathsf{NT} \wedge \psi$ , where,

 $\psi = (\exists x)(comp_M(x) \land ((\forall y < x) \neg comp_M(y)) \land \mod (x, b \uparrow 2, b \cdot (|\Sigma| + 1)))$ 

 $\psi$  states that there exists a smallest integer, which encodes an accepting computation of M.

### Theorem

L<sub>p</sub> and L<sub>us</sub> are recursively inseparable.

### Proof.

Main idea: Given an Turing Machine *M*, we construct an expression  $\phi_M$  such that if  $M(\epsilon) = "yes"$ , then **NT**  $\vdash \phi_M$  and if  $M(\epsilon) = "no"$ , them  $\phi_M$  is unsatisfiable. Assume that there exists an algorithm  $\mathcal{A}$  to separate  $L_p$  from  $L_{us}$ , i.e.,  $\mathcal{A}$  separates the true properties of integers from the unsatisfiable sentences. But now we can separate  $L_p$  and  $L_1$ . Given an arbitrary Turing machine *M*, construct  $\phi_M$  and then provide it to  $\mathcal{A}$ ! What is  $\phi_M$ ?

### Theorem

L<sub>p</sub> and L<sub>us</sub> are recursively inseparable.

### Proof.

Main idea: Given an Turing Machine *M*, we construct an expression  $\phi_M$  such that if  $M(\epsilon) = "yes"$ , then **NT**  $\vdash \phi_M$  and if  $M(\epsilon) = "no"$ , them  $\phi_M$  is unsatisfiable. Assume that there exists an algorithm  $\mathcal{A}$  to separate  $L_p$  from  $L_{us}$ , i.e.,  $\mathcal{A}$  separates the true properties of integers from the unsatisfiable sentences. But now we can separate  $L_y$  and  $L_n!$ Given an arbitrary turing machine *M*, construct  $\phi_M$  and then provide it to  $\mathcal{A}!$  What is  $\phi_M$ ?

### Theorem

L<sub>p</sub> and L<sub>us</sub> are recursively inseparable.

### Proof.

Main idea: Given an Turing Machine *M*, we construct an expression  $\phi_M$  such that if  $M(\epsilon) = "yes"$ , then **NT**  $\vdash \phi_M$  and if  $M(\epsilon) = "no"$ , them  $\phi_M$  is unsatisfiable. Assume that there exists an algorithm  $\mathcal{A}$  to separate  $L_p$  from  $L_{us}$ , i.e.,  $\mathcal{A}$  separates the true properties of integers from the unsatisfiable sentences. But now we can separate  $L_y$  and  $L_n$ ! Given an arbitrary Turing machine *M*, construct  $\phi_M$  and then provide it to  $\mathcal{A}$ ! What is  $\phi_M$ ?  $\phi_M = NT \land \psi$ , where.

 $\psi = (\exists x)(comp_M(x) \land ((\forall y < x) \neg comp_M(y)) \land \mod (x, b \uparrow 2, b \cdot (|\Sigma| + 1)))$ 

 $\psi$  states that there exists a smallest integer, which encodes an accepting computation of M.

### Theorem

L<sub>p</sub> and L<sub>us</sub> are recursively inseparable.

### Proof.

Main idea: Given an Turing Machine *M*, we construct an expression  $\phi_M$  such that if  $M(\epsilon) = "yes''$ , then **NT**  $\vdash \phi_M$  and if  $M(\epsilon) = "no''$ , then  $\phi_M$  is unsatisfiable. Assume that there exists an algorithm  $\mathcal{A}$  to separate  $L_p$  from  $L_{us}$ , i.e.,  $\mathcal{A}$  separates the true properties of integers from the unsatisfiable sentences. But now we can separate  $L_y$  and  $L_n$ ! Given an arbitrary Turing machine *M*, construct  $\phi_M$  and then provide it to  $\mathcal{A}$ ! What is  $\phi_M$ ?  $\phi_M = NT \land \phi$ , where.

 $\psi = (\exists x)(comp_M(x) \land ((\forall y < x) \neg comp_M(y)) \land \mod (x, b \uparrow 2, b \cdot (|\Sigma| + 1)))$ 

 $\psi$  states that there exists a smallest integer, which encodes an accepting computation of M.

### Theorem

L<sub>p</sub> and L<sub>us</sub> are recursively inseparable.

### Proof.

Main idea: Given an Turing Machine *M*, we construct an expression  $\phi_M$  such that if  $M(\epsilon) = "yes''$ , then **NT**  $\vdash \phi_M$  and if  $M(\epsilon) = "no''$ , them  $\phi_M$  is unsatisfiable. Assume that there exists an algorithm  $\mathcal{A}$  to separate  $L_p$  from  $L_{us}$ , i.e.,  $\mathcal{A}$  separates the true properties of integers from the unsatisfiable sentences. But now we can separate  $L_y$  and  $L_n$ ! Given an arbitrary Turing machine *M*, construct  $\phi_M$  and then provide it to  $\mathcal{A}$ ! What is  $\phi_M$ ?  $\phi_M = \mathbf{NT} \land \psi$ , where,

 $\psi = (\exists x)(comp_{M}(x) \land ((\forall y < x) \neg comp_{M}(y)) \land \mod (x, b \uparrow 2, b \cdot (|\Sigma| + 1))).$ 

 $\psi$  states that there exists a smallest integer, which encodes an accepting computation of M.

### Theorem

L<sub>p</sub> and L<sub>us</sub> are recursively inseparable.

### Proof.

Main idea: Given an Turing Machine *M*, we construct an expression  $\phi_M$  such that if  $M(\epsilon) = "yes''$ , then **NT**  $\vdash \phi_M$  and if  $M(\epsilon) = "no''$ , them  $\phi_M$  is unsatisfiable. Assume that there exists an algorithm  $\mathcal{A}$  to separate  $L_\rho$  from  $L_{us}$ , i.e.,  $\mathcal{A}$  separates the true properties of integers from the unsatisfiable sentences. But now we can separate  $L_\gamma$  and  $L_n$ ! Given an arbitrary Turing machine *M*, construct  $\phi_M$  and then provide it to  $\mathcal{A}$ ! What is  $\phi_M$ ?  $\phi_M = \mathbf{NT} \land \psi$ , where,

 $\psi = (\exists x)(comp_{M}(x) \land ((\forall y < x) \neg comp_{M}(y)) \land \mod (x, b \uparrow 2, b \cdot (|\Sigma| + 1))).$ 

 $\psi$  states that there exists a smallest integer, which encodes an accepting computation of M.

<ロ> <同> <同> < 同> < 三> < 三> <

### Proof (contd.)

(i) Assume *M*(ε) = "yes". There exists a unique computation of *M* that starts with ε and halts in the "yes" state. Thus, there exists a unique integer *n*, such that N ⊨ comp<sub>M</sub>[x ← n]. Therefore, N ⊨ (∃x)(comp<sub>M</sub>(x) ∧ ((∀y < x)¬comp<sub>M</sub>(y))). Since the last two digits of the *b*-ary expansion of *n* are |Σ| + 1 and 0, we have N ⊨ ψ. Observe that ψ can be written as a bounded sentence in prenex form. Thus, NT ⊢ ψ and hence NT ⊢ φ<sub>M</sub>. In other words, *M*(ε) = "yes" implies NT ⊢ φ<sub>M</sub>.

(ii) Assume that  $M(\epsilon) = "no"$ . Using the above argument, we can show that  ${f N} \models \phi_M'$ , where

 $\phi'_{M} = (\exists x')(comp_{M}(x') \land ((\forall y < x) \neg comp_{M}(y)) \land \mod (x', b \uparrow 2, b \cdot (|\Sigma| + 2))).$ 

Since  $\phi'_M$  can be written as a bounded sentence, NT  $\vdash \phi'_M$ . We need to show that  $\phi_M$  and  $\phi'_M$  are inconsistent. But this is obvious!

### Proof (contd.)

- (i) Assume M(ε) = "yes". There exists a unique computation of M that starts with ε and halts in the "yes" state. Thus, there exists a unique integer n, such that N ⊨ comp<sub>M</sub>[x ← n]. Therefore, N ⊨ (∃x)(comp<sub>M</sub>(x)) and since n is unique. N ⊨ (∃x)(comp<sub>M</sub>(x)) ∧ ((∀y < x) ¬ comp<sub>M</sub>(y))). Since the last two digits of the b-ary expansion of n are |Σ|+1 and 0, we have N ⊨ ψ. Observe that ψ can be written as a bounded sentence in prenex form. Thus, NT ⊢ ψ and hence NT ⊢ φ<sub>M</sub>. In other words, M(ε) = "yes" implies NT ⊢ φ<sub>M</sub>.
- (ii) Assume that  $M(\epsilon) = "no''$ . Using the above argument, we can show that  $N \models \phi'_M$ , where

 $\phi'_{M} = (\exists x')(comp_{M}(x') \land ((\forall y < x) \neg comp_{M}(y)) \land \mod (x', b \uparrow 2, b \cdot (|\Sigma| + 2)))$ 

Since  $\phi'_M$  can be written as a bounded sentence, NT  $\vdash \phi'_M$ . We need to show that  $\phi_M$  and  $\phi'_M$  are inconsistent. But this is obvious!

### Proof (contd.)

(i) Assume M(ε) = "yes". There exists a unique computation of M that starts with ε and halts in the "yes" state. Thus, there exists a unique integer n, such that N ⊨ comp<sub>M</sub>[x ← n]. Therefore, N ⊨ (∃x) comp<sub>M</sub>(x) and since n is unique. N ⊨ (∃x) comp<sub>M</sub>(x) and since n is unique. N ⊨ (∃x) comp<sub>M</sub>(x) = (1) and (x) we have N ⊨ since the last two digits of the b-ary expansion of n are [X ← 1] and 0, we have N ⊨ so the present two digits of the b-ary expansion of n are [X ← 1] and 0, we have N ⊨ so the present two digits of the b-ary expansion of n are [X ← 1] and 0, we have N ⊨ so that N ⊨ so the present two digits of the b-ary expansion of n are [X ← 1] and 0, we have N ⊨ so that N ⊨ so that

(ii) Assume that  $M(\epsilon) = "no"$ . Using the above argument, we can show that  $\mathbf{N} \models \phi'_M$ , where

 $\phi'_{M} = (\exists x')(comp_{M}(x') \land ((\forall y < x) \neg comp_{M}(y)) \land \mod (x', b \uparrow 2, b \cdot (|\Sigma| + 2))).$ 

Since  $\phi'_M$  can be written as a bounded sentence, NT  $\vdash \phi'_M$ . We need to show that  $\phi_M$  and  $\phi'_M$  are inconsistent. But this is obvious!

### Proof (contd.)

(i) Assume M(ε) = "yes". There exists a unique computation of M that starts with ε and halts in the "yes" state. Thus, there exists a unique integer n, such that N ⊨ comp<sub>M</sub>[x ← n]. Therefore, N ⊨ (∃x) comp<sub>M</sub>(x) and since n is unique.

 $\mathbb{N} \models (\exists x)(comp_M(x) \land ((\forall y < x) \neg comp_M(y))))$ . Since the last two digits of the *b*-ary expansion of *n* are  $|\Sigma| + 1$  and 0, we have  $\mathbb{N} \models \psi$ .

Observe that  $\psi$  can be written as a bounded sentence in prenex form. Thus,  $NT \vdash \psi$  and hence  $NT \vdash \phi_M$ . In other words,  $M(\epsilon) = "yes''$  implies  $NT \vdash \phi_M$ .

(ii) Assume that  $M(\epsilon)=$  "no". Using the above argument, we can show that  ${\sf N}\models\phi_M'$ , where

 $\phi'_{M} = (\exists x')(comp_{M}(x') \land ((\forall y < x) \neg comp_{M}(y)) \land \mod (x', b \uparrow 2, b \cdot (|\Sigma| + 2))).$ 

Since  $\phi'_M$  can be written as a bounded sentence, **NT**  $\vdash \phi'_M$ . We need to show that  $\phi_M$  and  $\phi'_M$  are inconsistent. But this is obvious!

### Proof (contd.)

- (i) Assume M(ε) = "yes". There exists a unique computation of M that starts with ε and halts in the "yes" state. Thus, there exists a unique integer n, such that N ⊨ comp<sub>M</sub>[x ← n]. Therefore, N ⊨ (∃x)comp<sub>M</sub>(x) and since n is unique. N ⊨ (∃x)(comp<sub>M</sub>(x) ∧ ((y < x) → comp<sub>M</sub>(y))). Since the last two digits of the *b*-ary expansion of n are |Σ| + 1 and 0, we have N ⊨ ∞.
- (ii) Assume that M(c) = "no". Using the above argument, we can show that  $\mathbf{N} \vdash \phi'$

 $\phi'_{M} = (\exists x')(comp_{M}(x') \land ((\forall y < x) \neg comp_{M}(y)) \land mod (x', b \uparrow 2, b \cdot (|\Sigma| + 2))).$ 

Since  $\phi'_M$  can be written as a bounded sentence, **NT**  $\vdash \phi'_M$ . We need to show that  $\phi_M$  and  $\phi'_M$  are inconsistent. But this is obvious!

### Proof (contd.)

- (i) Assume M(ε) = "yes". There exists a unique computation of M that starts with ε and halts in the "yes" state. Thus, there exists a unique integer n, such that N ⊨ comp<sub>M</sub>[x ← n]. Therefore, N ⊨ (∃x)comp<sub>M</sub>(x) and since n is unique,
   N ⊨ (∃x)(comp<sub>M</sub>(x) ∧ ((∀y < x)¬comp<sub>M</sub>(y))). Since the last two digits of the *b*-ary expansion of n are [z] + 1 and 0, we have N ⊨ ∞.
   Observe that e can be written as a bounded sentence in prenex form. Thus, NT he and hence NT = out in other words. M(z) = yes implies NT = out.
- (ii) Assume that  $M(\epsilon) = "no''$ . Using the above argument, we can show that  ${\sf N} \models \phi_M'$ , where

 $\phi_M' = (\exists x')(comp_M(x') \land ((\forall y < x) \neg comp_M(y)) \land \mod (x', b \uparrow 2, b \cdot (|\Sigma| + 2))).$ 

Since  $\phi'_M$  can be written as a bounded sentence, **NT**  $\vdash \phi'_M$ . We need to show that  $\phi_M$  and  $\phi'_M$  are inconsistent. But this is obvious!

### Proof (contd.)

- (i) Assume M(ε) = "yes". There exists a unique computation of *M* that starts with ε and halts in the "yes" state. Thus, there exists a unique integer *n*, such that N ⊨ comp<sub>M</sub>[x ← n]. Therefore, N ⊨ (∃x)comp<sub>M</sub>(x) and since *n* is unique, N ⊨ (∃x)(comp<sub>M</sub>(x) ∧ ((∀y < x)¬comp<sub>M</sub>(y))). Since the last two digits of the *b*-ary expansion of *n* are |Σ| + 1 and 0, we have N ⊨ ψ.
  Observe that ψ can be written as a bounded sentence in prenex form. Thus, NT ⊨ ψ and hence NT ⊨ φ<sub>M</sub>. In other words, M(ε) = predimples NT = q<sub>M</sub>.
- (ii) Assume that  $M(\epsilon) = "no"$ . Using the above argument, we can show that  $N \models \phi'_M$ , where

 $\phi'_{M} = (\exists x')(comp_{M}(x') \land ((\forall y < x) \neg comp_{M}(y)) \land \mod (x', b \uparrow 2, b \cdot (|\Sigma| + 2))).$ 

Since  $\phi'_M$  can be written as a bounded sentence, **NT**  $\vdash \phi'_M$ . We need to show that  $\phi_M$  and  $\phi'_M$  are inconsistent. But this is obvious!

(a)

### Proof (contd.)

(i) Assume M(ε) = "yes". There exists a unique computation of *M* that starts with ε and halts in the "yes" state. Thus, there exists a unique integer *n*, such that N ⊨ comp<sub>M</sub>[x ← n]. Therefore, N ⊨ (∃x)comp<sub>M</sub>(x) and since *n* is unique,
N ⊨ (∃x)(comp<sub>M</sub>(x) ∧ ((∀y < x)¬comp<sub>M</sub>(y))). Since the last two digits of the *b*-ary expansion of *n* are |Σ| + 1 and 0, we have N ⊨ ψ.
Observe that ψ can be written as a bounded sentence in prenex form. Thus, NT ⊨ we and hence NT ⊨ we are preserved and the words. M(c) = yes implies NT ⊨ we are preserved.

 $\phi'_{M} = (\exists x')(comp_{M}(x') \land ((\forall y < x) \neg comp_{M}(y)) \land \mod (x', b \uparrow 2, b \cdot (|\Sigma| + 2)))$ 

Since  $\phi'_M$  can be written as a bounded sentence, NT  $\vdash \phi'_M$ . We need to show that  $\phi_M$  and  $\phi'_M$  are inconsistent. But this is obvious!

### Proof (contd.)

(i) Assume M(ε) = "yes". There exists a unique computation of *M* that starts with ε and halts in the "yes" state. Thus, there exists a unique integer *n*, such that N ⊨ comp<sub>M</sub>[x ← n]. Therefore, N ⊨ (∃x)comp<sub>M</sub>(x) and since *n* is unique, N ⊨ (∃x)(comp<sub>M</sub>(x) ∧ ((∀y < x)¬comp<sub>M</sub>(y))). Since the last two digits of the *b*-ary expansion of *n* are |Σ| + 1 and 0, we have N ⊨ ψ.
Observe that ψ can be written as a bounded sentence in prenex form. Thus, NT ⊢ ψ and hence NT ⊢ φ<sub>M</sub>. In other words, M(ε) = "yes" implies NT ⊨ φ<sub>M</sub>.
(ii) Assume that M(ε) = "no". Using the above argument, we can show that N ⊨ equivalent.

 $\phi'_{M} = (\exists x')(comp_{M}(x') \land ((\forall y < x) \neg comp_{M}(y)) \land \mod (x', b \uparrow 2, b \cdot (|\Sigma| + 2)))$ 

Since  $\phi'_M$  can be written as a bounded sentence, NT  $\vdash \phi'_M$ . We need to show that  $\phi_M$  and  $\phi'_M$  are inconsistent. But this is obvious!

### Proof (contd.)

(i) Assume M(ε) = "yes". There exists a unique computation of M that starts with ε and halts in the "yes" state. Thus, there exists a unique integer n, such that N ⊨ comp<sub>M</sub>[x ← n]. Therefore, N ⊨ (∃x)comp<sub>M</sub>(x) and since n is unique, N ⊨ (∃x)(comp<sub>M</sub>(x) ∧ ((∀y < x)¬comp<sub>M</sub>(y))). Since the last two digits of the b-ary expansion of n are |Σ| + 1 and 0, we have N ⊨ ψ. Observe that ψ can be written as a bounded sentence in prenex form. Thus, NT ⊢ ψ and hence NT ⊢ φ<sub>M</sub>. In other words, M(ε) = "yes" implies NT ⊢ φ<sub>M</sub>.

(ii) Assume that  $M(\epsilon) = "no''$ . Using the above argument, we can show that  $\mathbb{N} \models \phi'_M$ , where

 $\phi_M' = (\exists x')(comp_M(x') \land ((\forall y < x) \neg comp_M(y)) \land \mod (x', b \uparrow 2, b \cdot (|\Sigma| + 2))).$ 

Since  $\phi'_M$  can be written as a bounded sentence, NT  $\vdash \phi'_M$ . We need to show that  $\phi_M$  and  $\phi'_M$  are inconsistent. But this is obvious!

< ロ > < 同 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ >

### Proof (contd.)

- (i) Assume M(ε) = "yes". There exists a unique computation of M that starts with ε and halts in the "yes" state. Thus, there exists a unique integer n, such that N ⊨ comp<sub>M</sub>[x ← n]. Therefore, N ⊨ (∃x)comp<sub>M</sub>(x) and since n is unique, N ⊨ (∃x)(comp<sub>M</sub>(x) ∧ ((∀y < x)¬comp<sub>M</sub>(y))). Since the last two digits of the b-ary expansion of n are |Σ| + 1 and 0, we have N ⊨ ψ. Observe that ψ can be written as a bounded sentence in prenex form. Thus, NT ⊢ ψ and hence NT ⊢ φ<sub>M</sub>. In other words, M(ε) = "yes" implies NT ⊢ φ<sub>M</sub>.
- (ii) Assume that  $M(\epsilon) = "no''$ . Using the above argument, we can show that  $\mathbb{N} \models \phi'_M$ , where

 $\phi_M' = (\exists x')(comp_M(x') \land ((\forall y < x) \neg comp_M(y)) \land \mod (x', b \uparrow 2, b \cdot (|\Sigma| + 2))).$ 

Since  $\phi'_M$  can be written as a bounded sentence,  $NT \vdash \phi'_M$ . We need to show that  $\phi_M$  and  $\phi'_M$  are inconsistent. But this is obvious!

< ロ > < 同 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ >

## Proof (contd.)

- (i) Assume M(ε) = "yes". There exists a unique computation of *M* that starts with ε and halts in the "yes" state. Thus, there exists a unique integer *n*, such that N ⊨ comp<sub>M</sub>[x ← n]. Therefore, N ⊨ (∃x)comp<sub>M</sub>(x) and since *n* is unique, N ⊨ (∃x)(comp<sub>M</sub>(x) ∧ ((∀y < x)¬comp<sub>M</sub>(y))). Since the last two digits of the *b*-ary expansion of *n* are |Σ| + 1 and 0, we have N ⊨ ψ. Observe that ψ can be written as a bounded sentence in prenex form. Thus, NT ⊢ ψ and hence NT ⊢ φ<sub>M</sub>. In other words, M(ε) = "yes" implies NT ⊢ φ<sub>M</sub>.
- (ii) Assume that  $M(\epsilon) = "no"$ . Using the above argument, we can show that  $\mathbf{N} \models \phi'_M$ , where

$$\phi_M' = (\exists x')(comp_M(x') \land ((\forall y < x) \neg comp_M(y)) \land \mod (x', b \uparrow 2, b \cdot (|\Sigma| + 2))).$$

Since  $\phi'_M$  can be written as a bounded sentence, **NT**  $\vdash \phi'_M$ . We need to show that  $\phi_M$  and  $\phi'_M$  are inconsistent. But this is obvious!

## Proof (contd.)

- (i) Assume M(ε) = "yes". There exists a unique computation of *M* that starts with ε and halts in the "yes" state. Thus, there exists a unique integer *n*, such that N ⊨ comp<sub>M</sub>[x ← n]. Therefore, N ⊨ (∃x)comp<sub>M</sub>(x) and since *n* is unique, N ⊨ (∃x)(comp<sub>M</sub>(x) ∧ ((∀y < x)¬comp<sub>M</sub>(y))). Since the last two digits of the *b*-ary expansion of *n* are |Σ| + 1 and 0, we have N ⊨ ψ.
  Observe that ψ can be written as a bounded sentence in prenex form. Thus, NT ⊢ ψ and hence NT ⊢ φ<sub>M</sub>. In other words, M(ε) = "yes" implies NT ⊢ φ<sub>M</sub>.
- (ii) Assume that  $M(\epsilon) = "no"$ . Using the above argument, we can show that  $\mathbf{N} \models \phi'_M$ , where

$$\phi_M' = (\exists x')(comp_M(x') \land ((\forall y < x) \neg comp_M(y)) \land \mod (x', b \uparrow 2, b \cdot (|\Sigma| + 2))).$$

Since  $\phi'_M$  can be written as a bounded sentence, **NT**  $\vdash \phi'_M$ . We need to show that  $\phi_M$  and  $\phi'_M$  are inconsistent. But this is obvious!

<ロ> <同> <同> < 回> < 回> < => < => <
#### Recursive Inseparability (contd.)

## Proof (contd.)

- (i) Assume M(ε) = "yes". There exists a unique computation of M that starts with ε and halts in the "yes" state. Thus, there exists a unique integer n, such that N ⊨ comp<sub>M</sub>[x ← n]. Therefore, N ⊨ (∃x)comp<sub>M</sub>(x) and since n is unique, N ⊨ (∃x)(comp<sub>M</sub>(x) ∧ ((∀y < x)¬comp<sub>M</sub>(y))). Since the last two digits of the b-ary expansion of n are |Σ| + 1 and 0, we have N ⊨ ψ.
  Observe that ψ can be written as a bounded sentence in prenex form. Thus, NT ⊢ ψ and hence NT ⊢ φ<sub>M</sub>. In other words, M(ε) = "yes" implies NT ⊢ φ<sub>M</sub>.
- (ii) Assume that  $M(\epsilon) = "no"$ . Using the above argument, we can show that  $\mathbf{N} \models \phi'_M$ , where

$$\phi_M' = (\exists x')(comp_M(x') \land ((\forall y < x) \neg comp_M(y)) \land \mod (x', b \uparrow 2, b \cdot (|\Sigma| + 2)))$$

Since  $\phi'_M$  can be written as a bounded sentence, **NT**  $\vdash \phi'_M$ . We need to show that  $\phi_M$  and  $\phi'_M$  are inconsistent. But this is obvious!

#### Recursive Inseparability (contd.)

## Proof (contd.)

- (i) Assume M(ε) = "yes". There exists a unique computation of *M* that starts with ε and halts in the "yes" state. Thus, there exists a unique integer *n*, such that N ⊨ comp<sub>M</sub>[x ← n]. Therefore, N ⊨ (∃x)comp<sub>M</sub>(x) and since *n* is unique, N ⊨ (∃x)(comp<sub>M</sub>(x) ∧ ((∀y < x)¬comp<sub>M</sub>(y))). Since the last two digits of the *b*-ary expansion of *n* are |Σ| + 1 and 0, we have N ⊨ ψ.
  Observe that ψ can be written as a bounded sentence in prenex form. Thus, NT ⊢ ψ and hence NT ⊢ φ<sub>M</sub>. In other words, M(ε) = "yes" implies NT ⊢ φ<sub>M</sub>.
- (ii) Assume that  $M(\epsilon) = "no"$ . Using the above argument, we can show that  $\mathbf{N} \models \phi'_M$ , where

$$\phi_M' = (\exists x')(comp_M(x') \land ((\forall y < x) \neg comp_M(y)) \land \mod (x', b \uparrow 2, b \cdot (|\Sigma| + 2))).$$

Since  $\phi'_M$  can be written as a bounded sentence, **NT**  $\vdash \phi'_M$ . We need to show that  $\phi_M$  and  $\phi'_M$  are inconsistent. But this is obvious!

### Theorem

The following questions, regarding a given sentence  $\phi$ , are undecidable:



æ.

### Theorem

The following questions, regarding a given sentence  $\phi$ , are undecidable:



<ロ> <同> < 回> < 回> < 回> < => < => <</p>

æ.

### Theorem

The following questions, regarding a given sentence  $\phi$ , are undecidable:

(i) 
$$Is \models \phi$$
?  
(ii)  $Is \vdash \phi$ ?  
(iii)  $Does N \models$   
(iv)  $Does N \vdash$ 

E.

### Theorem

The following questions, regarding a given sentence  $\phi$ , are undecidable:



<ロ> <同> < 回> < 回> < 回> < => < => <</p>

æ.

### Theorem

The following questions, regarding a given sentence  $\phi$ , are undecidable:

- (i)  $Is \models \phi$ ?
- (ii)  $Is \vdash \phi$ ?
- (iii) Does  $\mathbf{N} \models \phi$ ?
- (iv) Does **NT**  $\vdash \phi$ ?

<ロ> <同> < 回> < 回> < 回> < => < => <</p>

æ.

# Outline

# Number-theoretic encoding of computation

# 2 Undecidability

- Sentence Classification
- Recursive Inseparability



### Incompleteness

### Theorem

There does not exist a recursively enumerable set of axioms  $\Xi$ , such that for all sentences  $\phi$ ,  $\Xi \vdash \phi$  if and only if  $\mathbf{N} \models \phi$ .

### Proof.

# Let $L_{pr}$ denote the set of all proofs from $\Xi$ . Since $\Xi$ is recursively enumerable, so is $L_{pr}$ : For each expression in the sequence, check whether is

- (i) a logical axiom,
- (ii) it follows by modus ponens,
- (iii) it is in  $\Xi$ .

Since  $L_{pr}$  is recursively enumerable, there exists a Turing machine that enumerates it. It follows that there exists a Turing Machine that enumerates  $\{\phi : \Xi \vdash \phi\}$ . By the hypothesis, there exists a Turing machine that enumerates  $L_{e} = \{\phi : \mathbb{N} \models \phi\}$ . Hence,  $L_{e}$  is recursively enumerable. Arguing in identical fashion, the language  $L_{ne} = \{\phi : \mathbb{N} \models \neg\phi\}$  is recursively enumerable. This means that  $L_{e}$  and  $L_{ne}$  are recursive!

### Theorem

There does not exist a recursively enumerable set of axioms  $\Xi$ , such that for all sentences  $\phi$ ,  $\Xi \vdash \phi$  if and only if  $\mathbf{N} \models \phi$ .

### Proof.

Let  $L_{\rho r}$  denote the set of all proofs from  $\Xi$ . Since  $\Xi$  is recursively enumerable, so is  $L_{\rho r}$ : For each expression in the sequence, check whether it is

- (i) a logical axiom,
- (ii) it follows by modus ponens,
- (iii) it is in  $\Xi$ .

Since  $L_{pr}$  is recursively enumerable, there exists a Turing machine that enumerates it. It follows that there exists a Turing Machine that enumerates  $\{\phi : \Xi \vdash \phi\}$ . By the hypothesis, there exists a Turing machine that enumerates  $L_e = \{\phi : \mathbb{N} \models \phi\}$ . Hence,  $L_e$  is recursively enumerable. Arguing in identical fashion, the language  $L_{ne} = \{\phi : \mathbb{N} \models \neg \phi\}$  is recursively enumerable. This means that  $L_e$  and  $L_{ne}$  are recursive!

### Theorem

There does not exist a recursively enumerable set of axioms  $\Xi$ , such that for all sentences  $\phi$ ,  $\Xi \vdash \phi$ 

 $\textit{if and only if } \mathbf{N} \models \phi.$ 

### Proof.

Let  $L_{\rho r}$  denote the set of all proofs from  $\Xi$ . Since  $\Xi$  is recursively enumerable, so is  $L_{\rho r}$ : For each expression in the sequence, check whether it is

- (i) a logical axiom,
- (ii) it follows by modus ponens,
- (iii) it is in  $\Xi$ .

Since  $L_{pr}$  is recursively enumerable, there exists a Turing machine that enumerates it. It follows that

there exists a Turing Machine that enumerates  $\{\phi : \Xi \vdash \phi\}$ . By the hypothesis, there exists a Turing machine that enumerates  $L_e = \{\phi : \mathbb{N} \models \phi\}$ . Hence,  $L_e$  is recursively enumerable. Arguing in identical fashion, the language  $L_{ne} = \{\phi : \mathbb{N} \models \neg\phi\}$  is recursively enumerable. This means that  $L_e$  and  $L_{ne}$  are recursive!

### Theorem

There does not exist a recursively enumerable set of axioms  $\Xi$ , such that for all sentences  $\phi$ ,  $\Xi \vdash \phi$  if and only if  $\mathbf{N} \models \phi$ .

# Proof.

Let  $L_{\rho r}$  denote the set of all proofs from  $\Xi$ . Since  $\Xi$  is recursively enumerable, so is  $L_{\rho r}$ : For each expression in the sequence, check whether it is

- (i) a logical axiom,
- (ii) it follows by modus ponens,
- (iii) it is in Ξ.

Since  $L_{pr}$  is recursively enumerable, there exists a Turing machine that enumerates it. It follows that there exists a Turing Machine that enumerates  $\{\phi : \Xi \vdash \phi\}$ . By the hypothesis, there exists a Turing machine that enumerates  $L_{e} = \{\phi : \mathbb{N} \models \phi\}$ . Hence,  $L_{e}$  is recursively enumerable. Arguing in identical fashion, the language  $L_{pe} = \{\phi : \mathbb{N} \models \phi\}$  is recursively enumerable. This means that  $L_{e}$  and  $L_{pe}$  are recursivel

### Theorem

There does not exist a recursively enumerable set of axioms  $\Xi$ , such that for all sentences  $\phi$ ,  $\Xi \vdash \phi$  if and only if  $\mathbf{N} \models \phi$ .

### Proof.

Let  $L_{\rho r}$  denote the set of all proofs from  $\Xi$ . Since  $\Xi$  is recursively enumerable, so is  $L_{\rho r}$ : For each expression in the sequence, check whether it is

- (i) a logical axiom,
- (ii) it follows by modus ponens,
- (iii) it is in Ξ.

Since  $L_{pr}$  is recursively enumerable, there exists a Turing machine that enumerates it. It follows that there exists a Turing Machine that enumerates  $\{\phi : \Xi \vdash \phi\}$ . By the hypothesis, there exists a Turing machine that enumerates  $L_e = \{\phi : \mathbf{N} \models \phi\}$ . Hence,  $L_e$  is recursively enumerable. Arguing in identical fashion, the language  $L_{pe} = \{\phi : \mathbf{N} \models \neg \phi\}$  is recursively enumerable. This means that  $L_e$  and  $L_{pr}$  are recursivel

### Theorem

There does not exist a recursively enumerable set of axioms  $\Xi$ , such that for all sentences  $\phi$ ,  $\Xi \vdash \phi$  if and only if  $\mathbf{N} \models \phi$ .

### Proof.

Let  $L_{\rho r}$  denote the set of all proofs from  $\Xi$ . Since  $\Xi$  is recursively enumerable, so is  $L_{\rho r}$ : For each expression in the sequence, check whether it is

- (i) a logical axiom,
- (ii) it follows by modus ponens,
- (iii) it is in  $\Xi$ .

Since  $L_{pr}$  is recursively enumerable, there exists a Turing machine that enumerates it. It follows that there exists a Turing Machine that enumerates  $\{\phi : \Xi \vdash \phi\}$ . By the hypothesis, there exists a Turing machine that enumerates  $L_e = \{\phi : \mathbf{N} \models \phi\}$ . Hence,  $L_e$  is recursively enumerable. Arguing in identical fashion, the language  $L_{ne} = \{\phi : \mathbf{N} \models \neg \phi\}$  is recursively enumerable. This means that  $L_e$  and  $L_{ne}$  are recursive!

### Theorem

There does not exist a recursively enumerable set of axioms  $\Xi$ , such that for all sentences  $\phi$ ,  $\Xi \vdash \phi$  if and only if  $\mathbf{N} \models \phi$ .

### Proof.

Let  $L_{\rho r}$  denote the set of all proofs from  $\Xi$ . Since  $\Xi$  is recursively enumerable, so is  $L_{\rho r}$ : For each expression in the sequence, check whether it is

- (i) a logical axiom,
- (ii) it follows by modus ponens,
- (iii) it is in Ξ.

Since  $L_{pr}$  is recursively enumerable, there exists a Turing machine that enumerates it. It follows that there exists a Turing Machine that enumerates  $\{\phi : \Xi \vdash \phi\}$ . By the hypothesis, there exists a Turing machine that enumerates  $L_e = \{\phi : \mathbf{N} \models \phi\}$ . Hence,  $L_e$  is recursively enumerable. Arguing in identical fashion, the language  $L_{ne} = \{\phi : \mathbf{N} \models \neg\phi\}$  is recursively enumerable. This means that

*L<sub>e</sub>* and *L<sub>ne</sub>* are recursive!

### Theorem

There does not exist a recursively enumerable set of axioms  $\Xi$ , such that for all sentences  $\phi$ ,  $\Xi \vdash \phi$  if and only if  $\mathbf{N} \models \phi$ .

### Proof.

Let  $L_{\rho r}$  denote the set of all proofs from  $\Xi$ . Since  $\Xi$  is recursively enumerable, so is  $L_{\rho r}$ : For each expression in the sequence, check whether it is

- (i) a logical axiom,
- (ii) it follows by modus ponens,
- (iii) it is in Ξ.

Since  $L_{pr}$  is recursively enumerable, there exists a Turing machine that enumerates it. It follows that there exists a Turing Machine that enumerates  $\{\phi : \Xi \vdash \phi\}$ . By the hypothesis, there exists a Turing machine that enumerates  $L_e = \{\phi : \mathbf{N} \models \phi\}$ . Hence,  $L_e$  is recursively enumerable. Arguing in identical fashion, the language  $L_{ne} = \{\phi : \mathbf{N} \models \neg\phi\}$  is recursively enumerable. This means that  $L_e$  and  $L_{ne}$  are recursive!

#### Non-existence

There **cannot** exist a recursively enumerable (much less recursive) set of axioms that captures all and only the true properties of integers. Any sound system **must** be incomplete, i.e., there must exist a true property of integers that cannot be proved by it.

### Categorization

The languages  $L = \{\phi : \mathbb{N} \models \phi\}$  and  $L^{\circ} = \{\phi : \mathbb{N} \models \neg\phi\}$  are not recursively enumerable. Thus L and  $L^{\circ}$  are neither RE nor coRE!

< ロ > < 同 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ >

### Non-existence

There cannot exist a recursively enumerable (much less recursive) set of axioms that captures all

and only the true properties of integers. Any sound system must be incomplete, i.e., there must

exist a true property of integers that cannot be proved by it.

### Categorization

The languages  $L = \{\phi : \mathbb{N} \models \phi\}$  and  $L^c = \{\phi : \mathbb{N} \models \neg\phi\}$  are not recursively enumerable. Thus L and  $L^c$  are neither **RE** nor **coRE**!

### Non-existence

There **cannot** exist a recursively enumerable (much less recursive) set of axioms that captures all and only the true properties of integers. Any sound system **must** be incomplete, i.e., there must exist a true property of integers that cannot be proved by it.

### Categorization

The languages  $L = \{\phi : \mathbb{N} \models \phi\}$  and  $L^c = \{\phi : \mathbb{N} \models \neg\phi\}$  are not recursively enumerable. Thus L and  $L^c$  are neither **RE** nor **coRE**!

< ロ > < 同 > < 三 > < 三 > -

### Non-existence

There **cannot** exist a recursively enumerable (much less recursive) set of axioms that captures all and only the true properties of integers. Any sound system **must** be incomplete, i.e., there must exist a true property of integers that cannot be proved by it.

## Categorization

The languages  $L = \{\phi : \mathbf{N} \models \phi\}$  and  $L^c = \{\phi : \mathbf{N} \models \neg \phi\}$  are not recursively enumerable. Thus L

and *L<sup>c</sup>* are neither **RE** nor **coRE**!

<ロ> <同> <同> < 回> < 回> < 回> < 回> <

### Non-existence

There **cannot** exist a recursively enumerable (much less recursive) set of axioms that captures all and only the true properties of integers. Any sound system **must** be incomplete, i.e., there must exist a true property of integers that cannot be proved by it.

### Categorization

The languages  $L = \{\phi : \mathbf{N} \models \phi\}$  and  $L^c = \{\phi : \mathbf{N} \models \neg\phi\}$  are not recursively enumerable. Thus L and  $L^c$  are neither **RE** nor **coRE**!