大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

SOKENDAI

JST

ERATO MMSD

理論計算機科学入門
― 有限と無限のあいだ

数学的理論から，AI・自動運転

蓮尾 一郎

国立情報学研究所 システム設計数理国際研究センター センター長
同 アーキテクチャ研究系 准教授
総合研究大学院大学 准教授
JST ERATO 蓮尾メタ数理システムデザインプロジェクト 研究総括

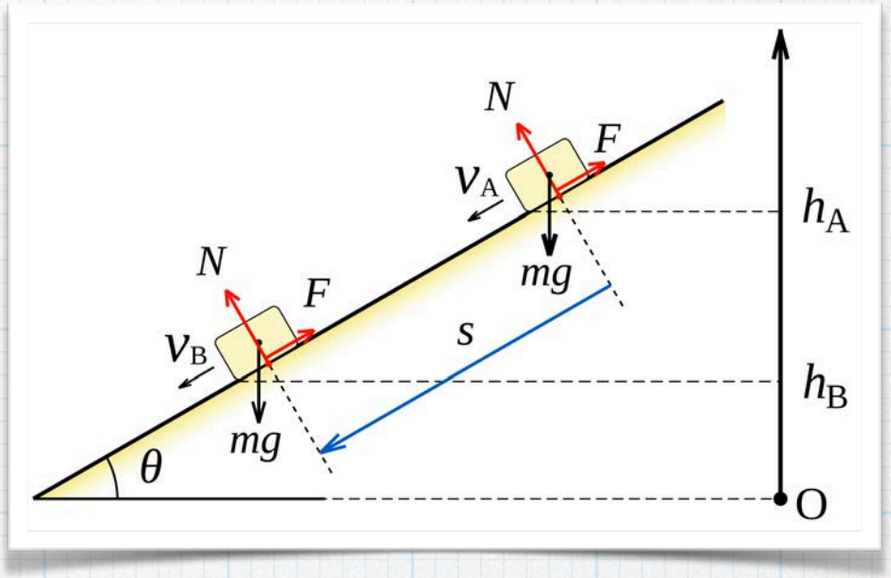
理論計算機科学入門 — 有限と無限のあいだ

アウトライン

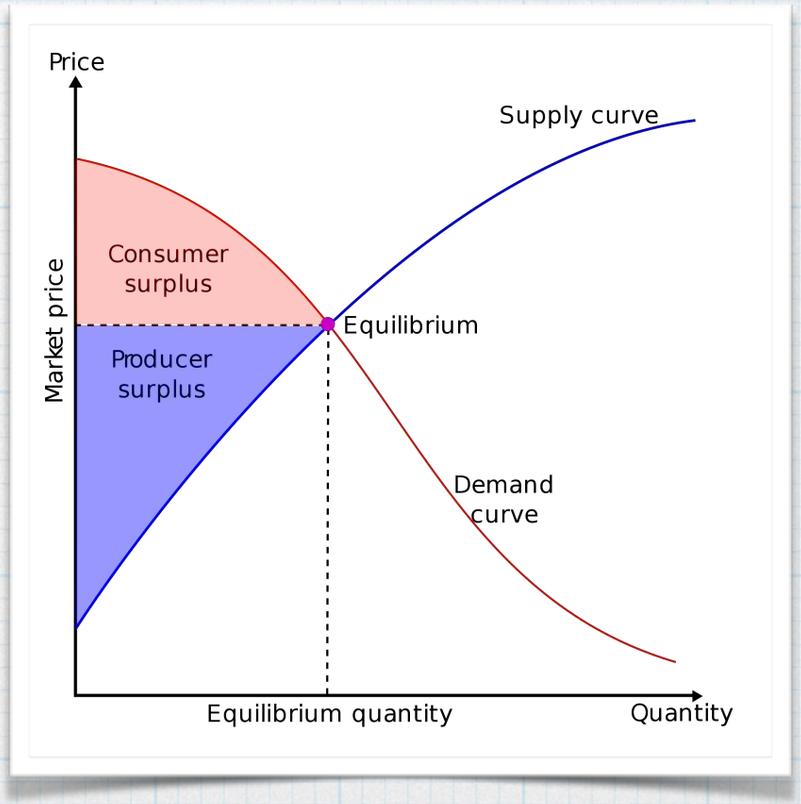
* 理論計算機科学とは — 有限と無限のせめぎあい
* いろいろな無限 — 対角線論法
* オートマトン理論入門
  * 包含問題を解くアルゴリズム — 有限の方法で無限の対象を操作
  * オートマトンの表現能力 — 有限性の限界
* 形式検証 — オートマトンを用いたシステム品質保証
* AI 時代の形式検証 — 研究紹介

理論計算機科学とは？

* 応用数学の一分野です



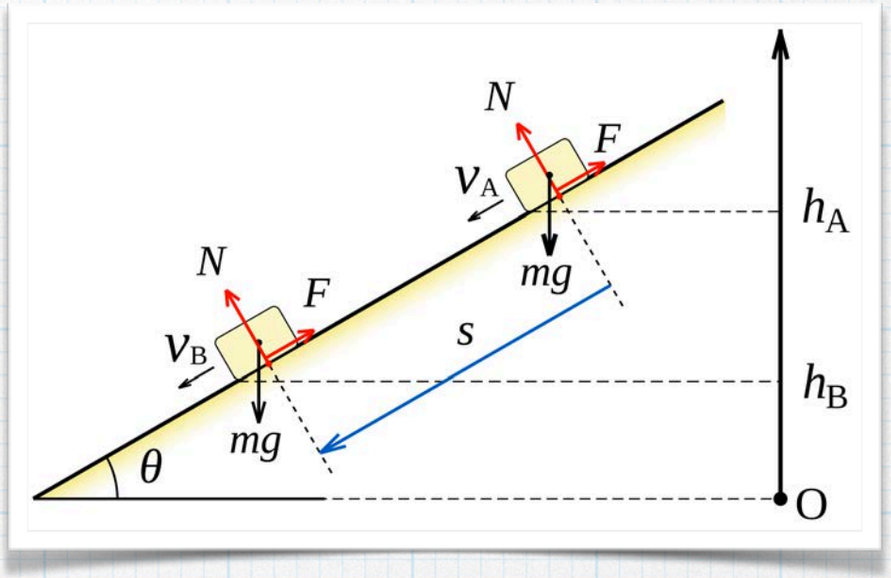
https://commons.wikimedia.org/wiki/File:%E5%8B%95%E6%91%A9%E6%93%A6%E3%81%A8%E5%8A%9B%E5%AD%A6%E7%9A%84%E3%82%A8%E3%83%8D%E3%83%AB%E3%82%AE%E3%83%BC.svg



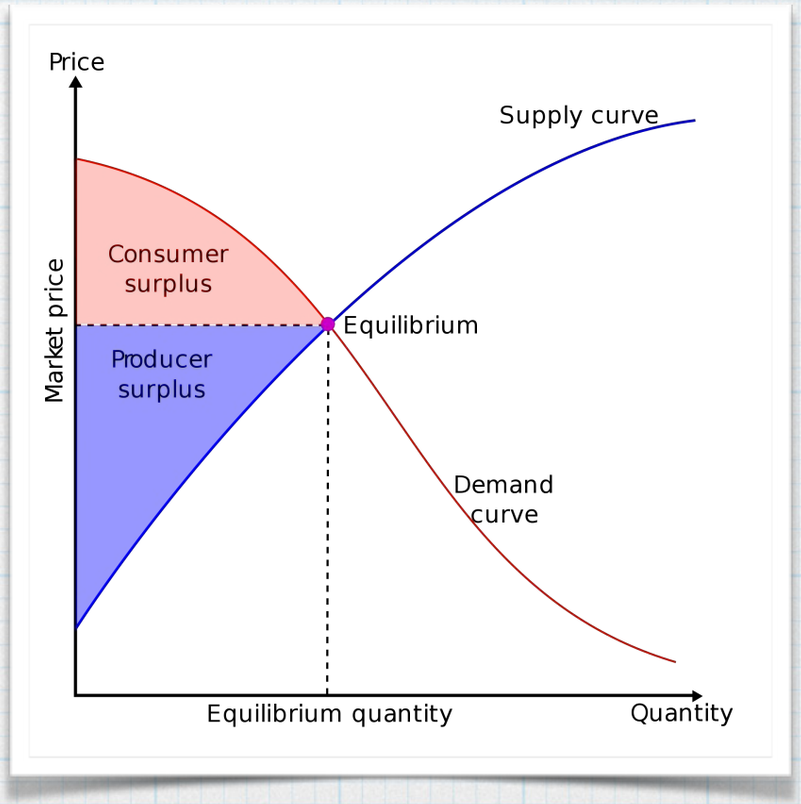
By User:SilverStar - Own work, CC BY 2.5, https://commons.wikimedia.org/w/index.php?curid=1450405

理論計算機科学とは？

* 応用数学の一分野です
  * 理論物理学：
    物理現象の数学的研究



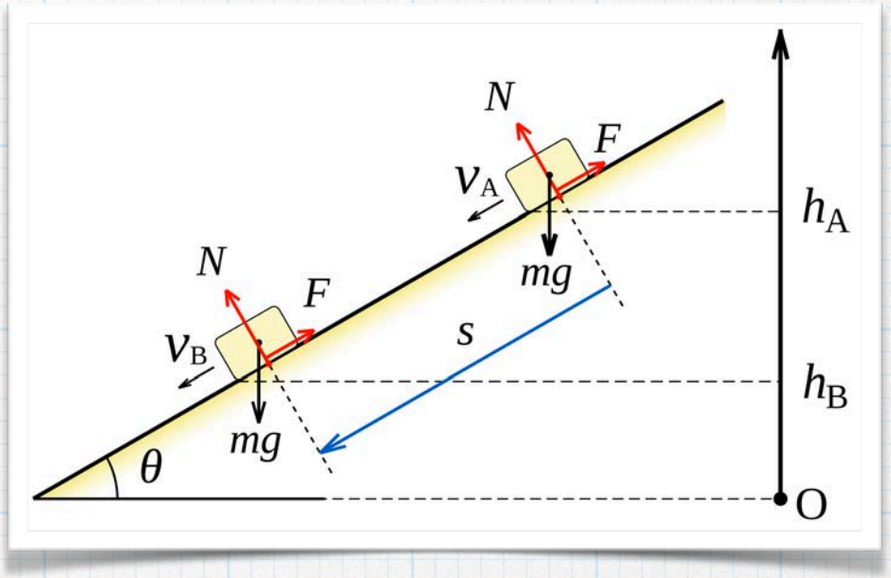
https://commons.wikimedia.org/wiki/File:%E5%8B%95%E6%91%A9%E6%93%A6%E3%81%A8%E5%8A%9B%E5%AD%A6%E7%9A%84%E3%82%A8%E3%83%8D%E3%83%AB%E3%82%AE%E3%83%BC.svg



By User:SilverStar - Own work, CC BY 2.5, https://commons.wikimedia.org/w/index.php?curid=1450405

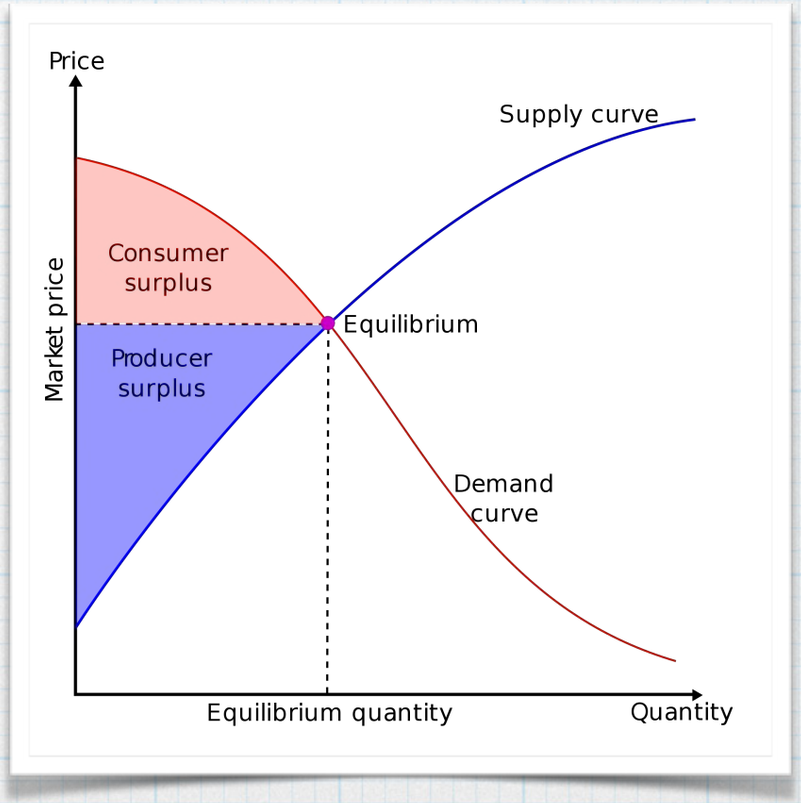
理論計算機科学とは？

* 応用数学の一分野です
  * 理論物理学：
    物理現象の数学的研究



https://commons.wikimedia.org/wiki/File:%E5%8B%95%E6%91%A9%E6%93%A6%E3%81%A8%E5%8A%9B%E5%AD%A6%E7%9A%84%E3%82%A8%E3%83%8D%E3%83%AB%E3%82%AE%E3%83%BC.svg

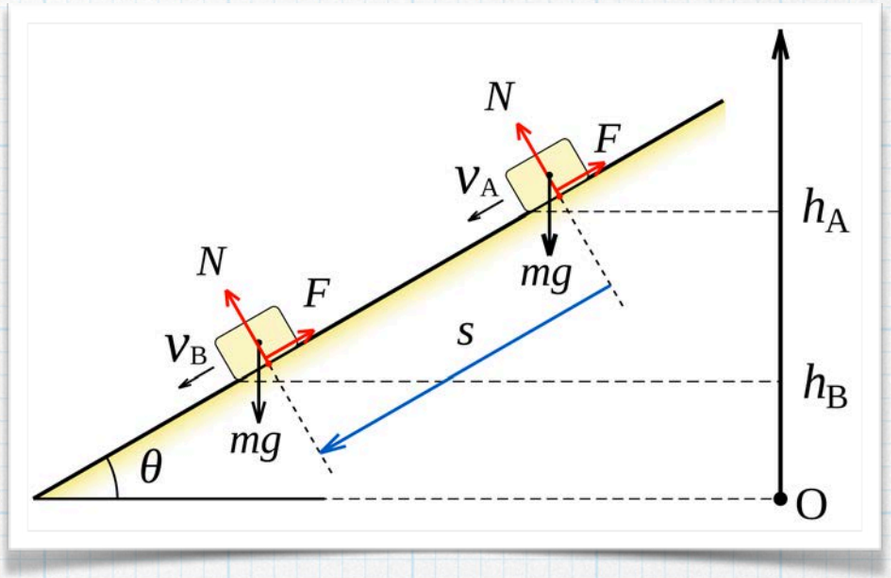
* 理論経済学：
  経済現象の数学的研究



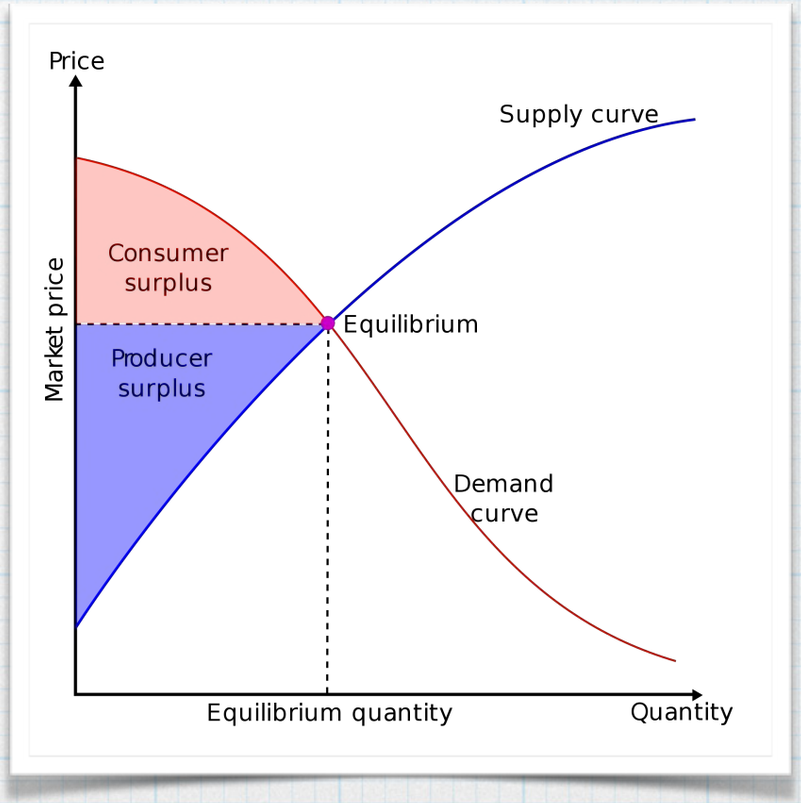
By User:SilverStar - Own work, CC BY 2.5, https://commons.wikimedia.org/w/index.php?curid=1450405

理論計算機科学とは？

* 応用数学の一分野です
  * 理論物理学：
    物理現象の数学的研究
  * 理論経済学：
    経済現象の数学的研究
  * 理論計算機科学：
    情報現象の数学的研究



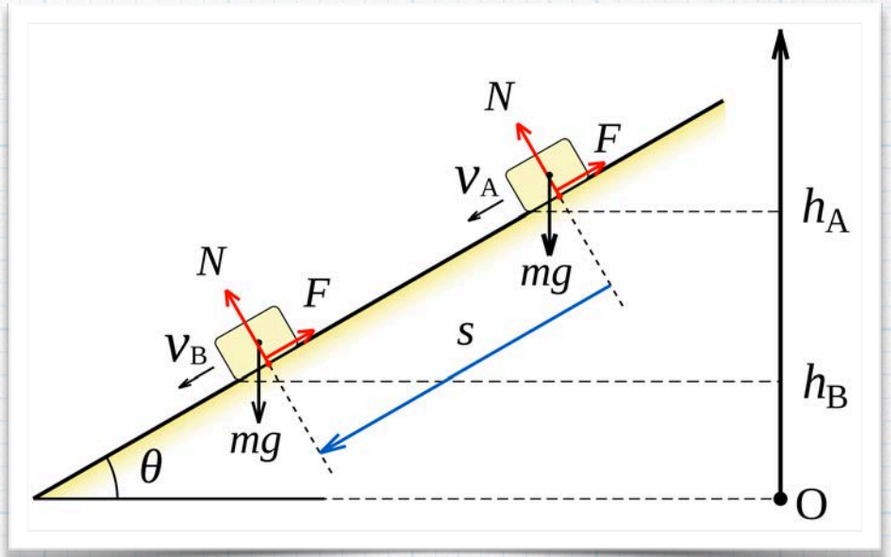
https://commons.wikimedia.org/wiki/File:%E5%8B%95%E6%91%A9%E6%93%A6%E3%81%A8%E5%8A%9B%E5%AD%A6%E7%9A%84%E3%82%A8%E3%83%8D%E3%83%AB%E3%82%AE%E3%83%BC.svg



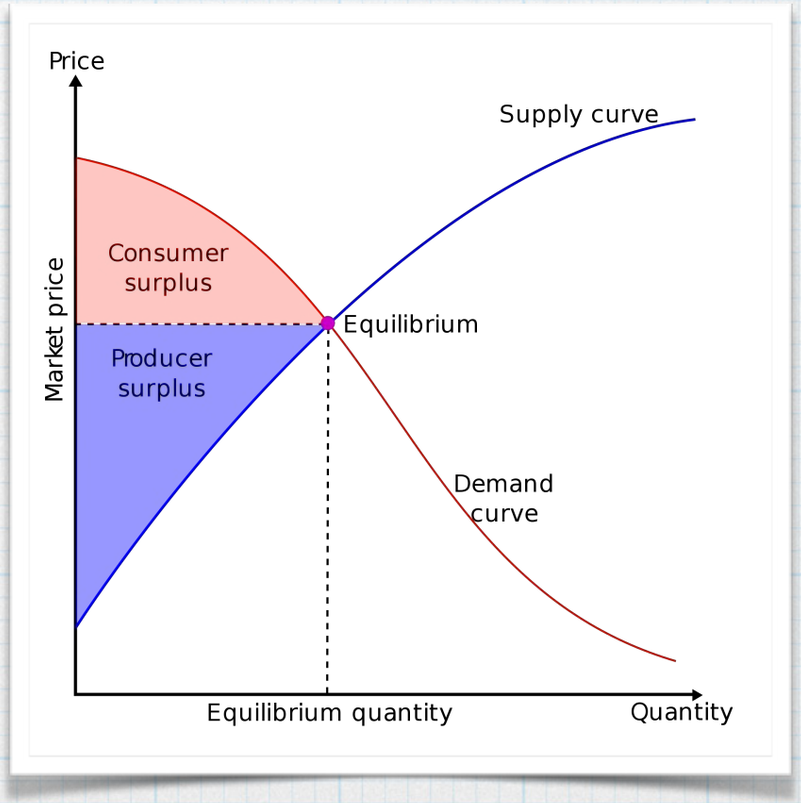
By User:SilverStar - Own work, CC BY 2.5, https://commons.wikimedia.org/w/index.php?curid=1450405

理論計算機科学とは？

* 応用数学の一分野です
  * 理論物理学：
    物理現象の数学的研究
  * 理論経済学：
    経済現象の数学的研究
  * 理論計算機科学：
    情報現象の数学的研究
  * 数学基礎論：
    数学者の営為の数学的研究



https://commons.wikimedia.org/wiki/File:%E5%8B%95%E6%91%A9%E6%93%A6%E3%81%A8%E5%8A%9B%E5%AD%A6%E7%9A%84%E3%82%A8%E3%83%8D%E3%83%AB%E3%82%AE%E3%83%BC.svg



By User:SilverStar - Own work, CC BY 2.5, https://commons.wikimedia.org/w/index.php?curid=1450405

さっそく脱線：
数学基礎論，メタ数学

Hilbert, Gödel, Church, Gentzen, Cohen, Turing, ...
→ computers!

Definition. A *proof* is a finite tree subject to the derivation rules:

$$\cdots \qquad \frac{\Gamma \Rightarrow \Delta, A \quad B, \Pi \Rightarrow \Sigma}{A \supset B, \Gamma, \Pi \Rightarrow \Delta, \Sigma}\ (\supset\text{-L}) \qquad \frac{A, \Gamma \Rightarrow \Delta, B}{\Gamma \Rightarrow \Delta, A \supset B}\ (\supset\text{-R}) \qquad \cdots$$

$$\frac{\Gamma \Rightarrow \Delta, A \quad A, \Pi \Rightarrow \Sigma}{\Gamma, \Pi \Rightarrow \Delta, \Sigma}\ (\textsc{Cut})$$

$$\vdots$$

$$\Rightarrow A$$

A *theorem* is a formula A with a proof $\dfrac{}{\Rightarrow A}$.

Theorem (cut elimination).

$$\vdots$$

$$\Rightarrow A$$

Any theorem A has a proof $\dfrac{}{\Rightarrow A}$ where the $(\textsc{Cut})$ rule is never used.

Proof. Let Π be a proof of a theorem A. We turn Π into a cut-free proof Π' by induction. . .

* 数学者の営為の数学的研究
* 定義，定理，証明などの概念が数学的に定義される

さっそく脱線：
数学基礎論，メタ数学

Hilbert, Gödel, Church, Gentzen, Cohen, Turing, ...
→ computers!

Definition. A *proof* is a finite tree subject to the derivation rules:

$$\frac{\Gamma \Rightarrow \Delta, A \quad B, \Pi \Rightarrow \Sigma}{A \supset B, \Gamma, \Pi \Rightarrow \Delta, \Sigma}\ (\supset\text{-L}) \qquad \frac{A, \Gamma \Rightarrow \Delta, B}{\Gamma \Rightarrow \Delta, A \supset B}\ (\supset\text{-R})$$

$$\frac{\Gamma \Rightarrow \Delta, A \quad A, \Pi \Rightarrow \Sigma}{\Gamma, \Pi \Rightarrow \Delta, \Sigma}\ (\textsc{Cut})$$

A *theorem* is a formula A with a proof $\dfrac{\vdots}{\Rightarrow A}$.

Theorem (cut elimination). Any theorem A has a proof $\dfrac{\vdots}{\Rightarrow A}$ where the (Cut) rule is never used.

Proof. Let Π be a proof of a theorem A. We turn Π into a cut-free proof Π' by induction. . .

やっぱり後で
（時間があれば）

* 数学者の営為の数学的研究
* 定義，定理，証明などの概念が数学的に定義される

さっそく脱線：
数学基礎論，メタ数学

Hilbert, Gödel, Church, Gentzen, Cohen, Turing, ...
→ computers!

Definition. A *proof* is a finite tree subject to the derivation rules:

$$\frac{\Gamma \Rightarrow \Delta, A \quad B, \Pi \Rightarrow \Sigma}{A \supset B, \Gamma, \Pi \Rightarrow \Delta, \Sigma}\ (\supset\text{-L}) \qquad \frac{A, \Gamma \Rightarrow \Delta, B}{\Gamma \Rightarrow \Delta, A \supset B}\ (\supset\text{-R})$$

$$\frac{\Gamma \Rightarrow \Delta, A \quad A, \Pi \Rightarrow \Sigma}{\Gamma, \Pi \Rightarrow \Delta, \Sigma}\ (\text{Cut})$$

A *theorem* is a formula A with a proof $\dfrac{\vdots}{\Rightarrow A}$.

Theorem (cut elimination). Any theorem A has a proof $\dfrac{\vdots}{\Rightarrow A}$ where the (Cut) rule is never used.

Proof. Let Π be a proof of a theorem A. We turn Π into a cut-free proof Π' by induction. . .

やっぱり後で
（時間があれば）

Image by Robin Higgins from Pixabay

数学者

* 数学者の営為の数学的研究
* 定義，定理，証明などの概念が数学的に定義される

さっそく脱線：
数学基礎論，メタ数学

Hilbert, Gödel, Church, Gentzen, Cohen, Turing, ...
→ computers!

Definition. A *proof* is a finite tree subject to the derivation rules:

$$\frac{\Gamma \Rightarrow \Delta, A \quad B, \Pi \Rightarrow \Sigma}{A \supset B, \Gamma, \Pi \Rightarrow \Delta, \Sigma}\ (\supset\text{-L}) \qquad \frac{A, \Gamma \Rightarrow \Delta, B}{\Gamma \Rightarrow \Delta, A \supset B}\ (\supset\text{-R})$$

$$\frac{\Gamma \Rightarrow \Delta, A \quad A, \Pi \Rightarrow \Sigma}{\Gamma, \Pi \Rightarrow \Delta, \Sigma}\ (\text{Cut})$$

A *theorem* is a formula A with a proof $\dfrac{\vdots}{\Rightarrow A}$.

Theorem (cut elimination). Any theorem A has a proof $\dfrac{\vdots}{\Rightarrow A}$ where the (Cut) rule is never used.

Proof. Let Π be a proof of a theorem A. We turn Π into a cut-free proof Π' by induction. . .

やっぱり後で
（時間があれば）

メタ数学者

Image by Robin Higgins from Pixabay

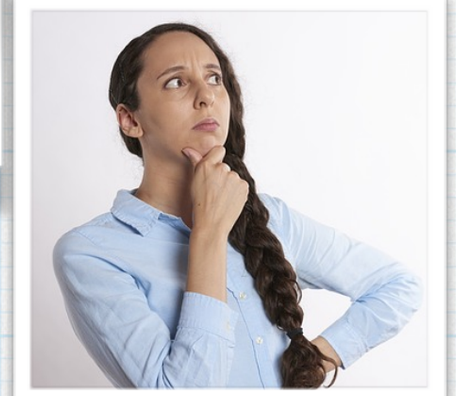


Image by Robin Higgins from Pixabay

数学者

* 数学者の営為の数学的研究
* 定義，定理，証明などの概念が数学的に定義される

さっそく脱線：
数学基礎論，メタ数学

Hilbert, Gödel, Church, Gentzen, Cohen, Turing, ...
→ computers!

Definition. A *proof* is a finite tree subject to th

$$\frac{\Gamma \Rightarrow \Delta, A \quad B, \Pi \Rightarrow \Sigma}{A \supset B, \Gamma, \Pi \Rightarrow \Delta, \Sigma}\ (\supset\text{-L}) \qquad \frac{A, \Gamma \Rightarrow \Delta, B}{\Gamma \Rightarrow \Delta, A \supset B}$$

$$\frac{\Gamma \Rightarrow \Delta, A \quad A, \Pi \Rightarrow \Sigma}{\Gamma, \Pi \Rightarrow \Delta, \Sigma}\ (\text{Cut})$$

A *theorem* is a formula A with a proof $\dfrac{\cdot}{\cdot}\ \Rightarrow A$.

Theorem (cut elimination). $\dfrac{\vdots}{\Rightarrow A}$
Any theorem A has a proof $\Rightarrow A$ where the (Cut) rule is never used.

Proof. Let Π be a proof of a theorem A. We turn Π into a cut-free proof Π' by induction. . .

やっぱり後で
（時間があれば）

どんなにがんばっても
それは証明できないよ…

メタ数学者

Image by Robin Higgins from Pixabay

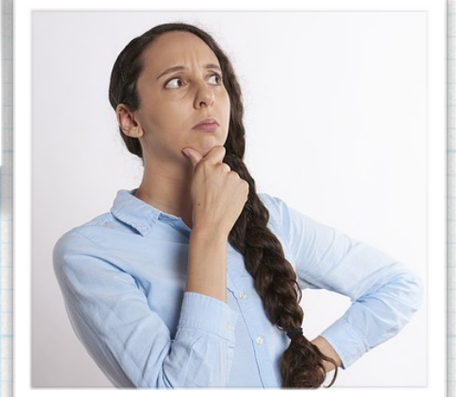


Image by Robin Higgins from Pixabay

数学者

* 数学者の営為の数学的研究
* 定義，定理，証明などの概念が数学的に定義される

さっそく脱線：
数学基礎論，メタ数学

Hilbert, Gödel, Church, Gentzen, Cohen, Turing, ...
→ computers!

Definition. A *proof* is a finite tree subject to th

$$\frac{\Gamma \Rightarrow \Delta, A \quad B, \Pi \Rightarrow \Sigma}{A \supset B, \Gamma, \Pi \Rightarrow \Delta, \Sigma}\ (\supset\text{-L}) \qquad \frac{A, \Gamma \Rightarrow \Delta, B}{\Gamma \Rightarrow \Delta, A \supset B}$$

$$\frac{\Gamma \Rightarrow \Delta, A \quad A, \Pi \Rightarrow \Sigma}{\Gamma, \Pi \Rightarrow \Delta, \Sigma}\ (\text{Cut})$$

A *theorem* is a formula A with a proof $\dfrac{\vdots}{\Rightarrow A}$.

Theorem (cut elimination). Any theorem A has a proof $\dfrac{\vdots}{\Rightarrow A}$ where the (Cut) rule is never used.

Proof. Let Π be a proof of a theorem A. We turn Π into a cut-free proof Π' by induction. . .

どんなにがんばっても
それは証明できないよ…

…ということを
私は証明できます！

メタ数学者

Image by Robin Higgins from Pixabay

やっぱり後で
（時間があれば）

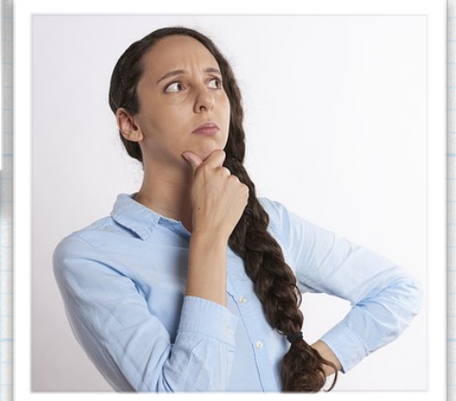


Image by Robin Higgins from Pixabay

数学者

* 数学者の営為の数学的研究
* 定義，定理，証明などの概念が数学的に定義される

理論計算機科学とは？

* 応用数学の一分野です
  * ↔ 純粋数学
    * 「応用する気のない数学」
    * 「応用できない数学」「役に立たない数学」とは違う
* 本人にその気がなくても，
  時々応用で大ホームランを飛ばす
* あなどれない！！

理論計算機科学とは？

* 応用数学の一分野です
  * ↔ 純粋数学
    * 「応用する気がない数学」
    * 「応用できない数学」「役に立たない数学」とは違う
      * 本人にその気がなくても，
        時々応用で大ホームランを飛ばす
      * あなどれない！！

やっぱり後で（時間があれば）

いろいろな数学

2011 日本数学会

年会プログラム

期　　日　3 月 20 日 (日) ～ 3 月 23 日 (水)
会　　場　早稲田大学西早稲田キャンパス
連 絡 先　早稲田大学理工学術院
基幹理工学部数学科
〒169-8555 東京都新宿区大久保 3-4-1
Tel (03)5286-3015
Fax (03)5286-3483
但し会期中は　Tel (03)5286-0556
Fax (03)5286-0557
社団法人 日本数学会
Tel (03)-3835-3483

会場 日時	I 57 号館 201 教室	II 57 号館 202 教室	III 63 号館 201 教室	IV 63 号館 202 教室	V 52 号館 101 教室	VI 52 号館 102 教室	VII 52 号館 201 教室	VIII 52 号館 203 教室	IX 52 号館 301 教室	X 52 号館 303 教室
20 日 (日)	代数学 9:30～11:35 14:15～16:10	幾何学 9:45～12:00 14:15～15:30	函数方程式論 9:15～12:00 14:15～15:50	応用数学 9:30～11:20 14:15～15:15		数学基礎論 および歴史 9:00～11:40 14:15～16:50	統計数学 9:00～12:00 14:15～14:35	トポロジー 9:10～12:00 14:15～15:30	実函数論 8:30～12:00 14:15～16:10	
	企画特別講演　13:00～14:00									
	特別講演 16:30～17:30	特別講演 15:45～16:45	特別講演 16:00～17:00	特別講演 15:30～16:30			特別講演 14:45～15:45 15:55～16:55	特別講演 15:45～16:45	特別講演 16:15～17:15	
	代数学	幾何学	函数方程式論	応用数学	無限可積分系	数学基礎論	統計数学	トポロジー	実函数論	函数解析学

いろいろな数学

2011 日本数学会

年会プログラム

期　　日　3 月 20 日 (日) ～ 3 月 23 日 (水)
会　　場　早稲田大学西早稲田キャンパス
連 絡 先　早稲田大学理工学術院
基幹理工学部数学科
〒169-8555 東京都新宿区大久保 3-4-1
Tel (03)5286-3015
Fax (03)5286-3483
但し会期中は　Tel (03)5286-0556
Fax (03)5286-0557
社団法人 日本数学会
Tel (03)-3835-3483

会場 日時	I 57 号館 201 教室	II 57 号館 202 教室	III 63 号館 201 教室	IV 63 号館 202 教室	V 52 号館 101 教室	VI 52 号館 102 教室	VII 52 号館 201 教室	VIII 52 号館 203 教室	IX 52 号館 301 教室	X 52 号館 303 教室
20 日 (日)	代数学 9:30～ 14:15～16	幾何学 9:45～12:00 15:30	函数方程式論 9:15～12:00 14:15～15:50	応用数学 9:30～11:20 14:15～15:15		数学基礎論 および歴史 9:00～11:40 14:15～16:50	統計数学 9:00～12:00 14:15～14:35	トポロジー 9:10～12:00 14:15～15:30	実函数論 8:30～12:00 14:15～16:10	
	企画特別講演　13:00～14:00									
								特別講演 5:45～16:45	特別講演 16:15～17:15	
								トポロジー	実函数論	函数解析学

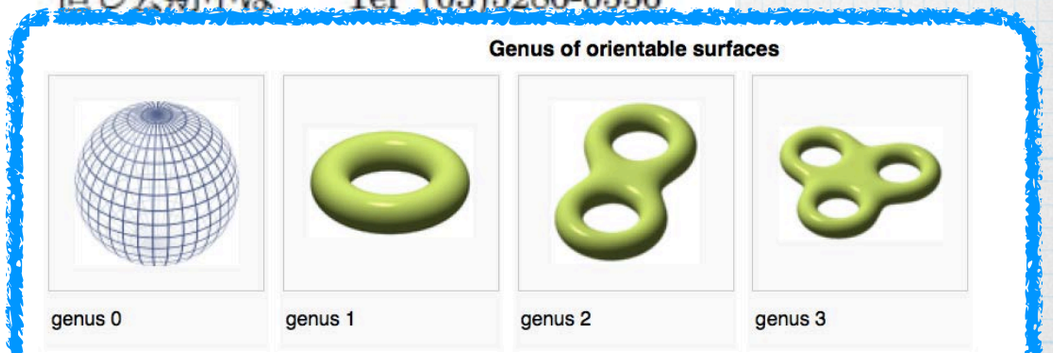
$$\cdots \xrightarrow{d_3} P_2 \xrightarrow{d_2} P_1 \xrightarrow{d_1} P_0 \xrightarrow{\varepsilon} V \longrightarrow 0$$

いろいろな数学

2011 日本数学会

年会プログラム

期　　日　3 月 20 日 (日) ～ 3 月 23 日 (水)
会　　場　早稲田大学西早稲田キャンパス
連 絡 先　早稲田大学理工学術院
基幹理工学部数学科
〒169-8555 東京都新宿区大久保 3-4-1
Tel (03)5286-3015
Fax (03)5286-3483
但し会期中は　Tel (03)5286-0556



会場 日時	I 57 号館 201 教室	II 57 号館 202 教室	III 63 号館 201 教室	IV 63 号館 202 教室	V 52 号館 101 教室	VI 52 号館 102 教室	VII 52 号館 201 教室	VIII 52 号館 203 教	301 教室	X 52 号館 303 教室
20 日 (日)	代 数 学 9:30～ 14:15～16	幾 何 学 9:45～12:00 15:30	函数方程式論 9:15～12:00 14:15～15:50	応用数学 9:30～11:20 14:15～15:15		数学基礎論 および歴史 9:00～11:40 14:15～16:50	統計数学 9:00～12:00 14:15～14:35	トポロジー 9:10～12:00 14:15～15:30	実函数論 8:30～12:00 14:15～16:10	
	企画特別講演　13:00～14:00									
								特別講演 5:45～16:45	特別講演 16:15～17:15	
								トポロジー	実函数論	函数解析学

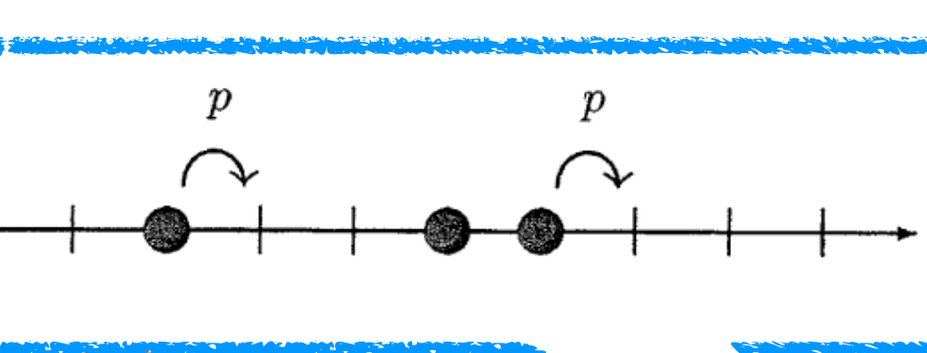
$$\cdots \xrightarrow{d_3} P_2 \xrightarrow{d_2} P_1 \xrightarrow{d_1} P_0 \xrightarrow{\varepsilon} V \longrightarrow 0$$

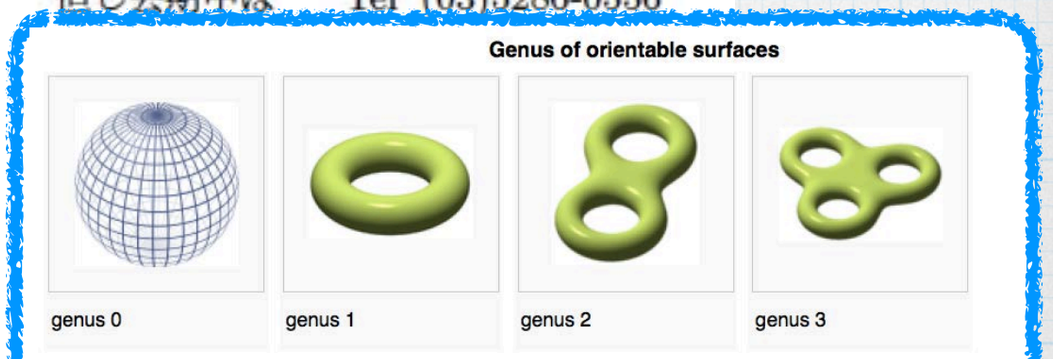
いろいろな数学

2011 日本数学会

年会プログラム

期　　日　3 月 20 日 (日) ～ 3 月 23 日 (水)
会　　場　早稲田大学西早稲田キャンパス
連 絡 先　早稲田大学理工学術院
基幹理工学部数学科
〒169-8555 東京都新宿区大久保 3-4-1
Tel (03)5286-3015
Fax (03)5286-3483
但し会期中は　Tel (03)5286-0556





日時	57 号館 201 教室	57 号館 202 教室	201 教	63 号館 202 教室	V 52 号館 101 教室	VI 52 号館 102 教室	VII 52 号館 201 教室	VIII 52 号館 203 教	301 教室	X 52 号館 303 教室
20 日 (日)	代 数 学 9:30～ 14:15～16	幾 何 学 9:45～12:00 15:30	函数方程式論 9:15～12:00 14:15～15:50	応用数学 9:30～11:20 14:15～15:15		数学基礎論および歴史 9:00～11:40 14:15～16:50	統計数学 9:00～12:00 14:15～14:35	トポロジー 9:10～12:00 14:15～15:30	実函数論 8:30～12:00 14:15～16:10	
	企画特別講演　13:00～14:00									
								特別講演 5:45～16:45	特別講演 16:15～17:15	
								トポロジー	実函数論	函数解析学

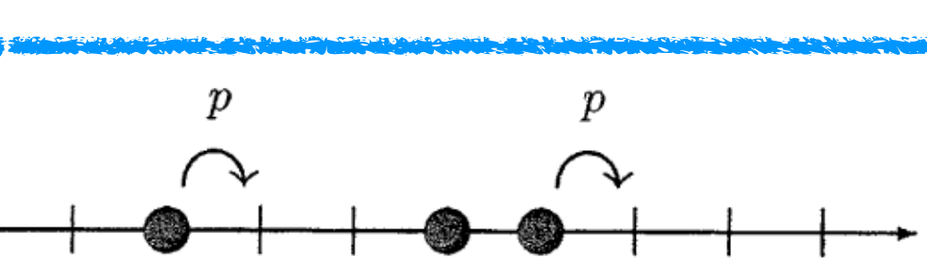
$$\cdots \xrightarrow{d_3} P_2 \xrightarrow{d_2} P_1 \xrightarrow{d_1} P_0 \xrightarrow{\varepsilon} V \longrightarrow 0$$

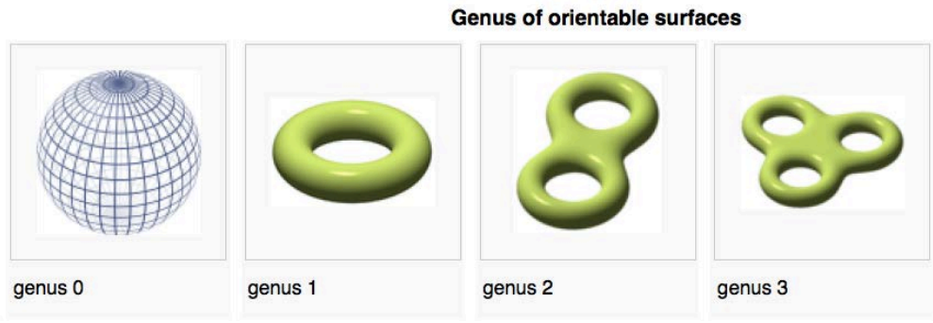
いろいろな数学

2011 日本数学会

年会プログラム

期　　日　3 月 20 日 (日) ～ 3 月 23 日 (水)
会　　場　早稲田大学西早稲田キャンパス
連 絡 先　早稲田大学理工学術院
基幹理工学部数学科
〒169-8555 東京都新宿区大久保 3-4-1
Tel (03)5286-3015
Fax (03)5286-3483
但し会期中は　Tel (03)5286-0556





日時	57 号館 201 教室	57 号館 202 教室	201 教	63 号館 202 教室	V 52 号館 101 教室	VI 52 号館 102 教室	VII 52 号館 201 教室	VIII 52 号館 203 教	301 教室	X 52 号館 303 教室
20 日 (日)	代 数 学 9:30～ 14:15～16	幾 何 学 9:45～12:00 15:30	函数方程式論 9:15～12:00 14:15～15:50	応用数学 9:30～11:20 14:15～15:15		数学基礎論および歴史 9:00～11:40 14:15～16:50	統計数学 9:00～ 14:15～14:3	トポロジー	実函数論	
	企画特別講演　13:00～14:00									

$$\cdots \xrightarrow{d_3} P_2 \xrightarrow{d_2} P_1 \xrightarrow{d_1} P_0 \xrightarrow{\varepsilon} V \longrightarrow 0$$

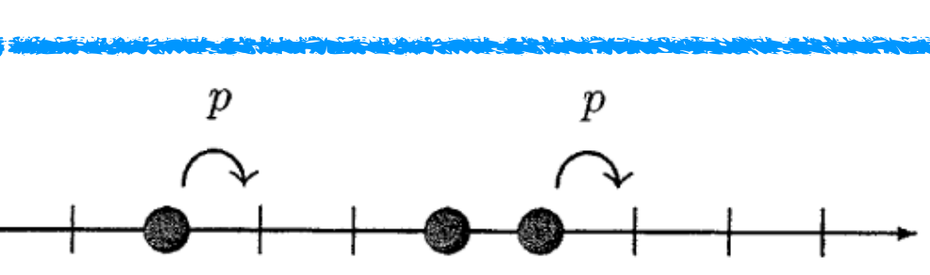
$$\dfrac{\dfrac{p \supset r \qquad \dfrac{p \wedge q}{p}}{r}}{\dfrac{(p \wedge q) \supset r}{(p \supset r) \supset \big((p \wedge q) \supset r\big)}}$$

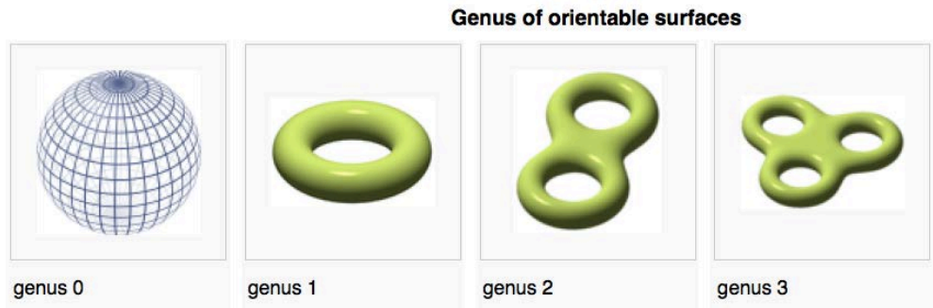
いろいろな数学

2011 日本数学会
年会プログラム

今日は理論計算機科学で使う
数学を見てみます

期　日　3 月 20 日 (日) ～ 3 月 23 日 (水)
会　場　早稲田大学西早稲田キャンパス
連絡先　早稲田大学理工学術院
基幹理工学部数学科
〒169-8555 東京都新宿区大久保 3-4-1
Tel (03)5286-3015
Fax (03)5286-3483
但し会期中は　Tel (03)5286-0556



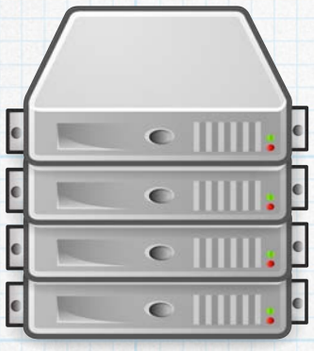


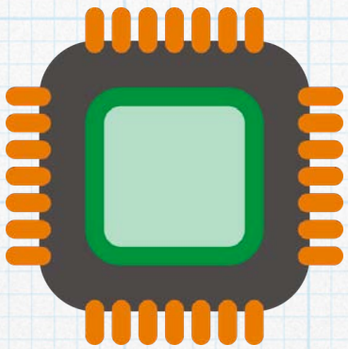
日時	57 号館 201 教室	57 号館 202 教室	201 教	63 号館 202 教室	V 52 号館 101 教室	VI 52 号館 102 教室	VII 52 号館 201 教室	VIII 52 号館 203 教	301 教室	X 52 号館 303 教室
20 日 (日)	代 数 学 9:30～ 14:15～16	幾 何 学 9:45～12:00 15:30	函数方程式論 9:15～12:00 14:15～15:50	応用数学 9:30～11:20 14:15～15:15		数学基礎論および歴史 9:00～11:40 14:15～16:50	統計数学 9:00～ 14:15～14:3	トポロジー	実函数論	
	企画特別講演　13:00～14:00									

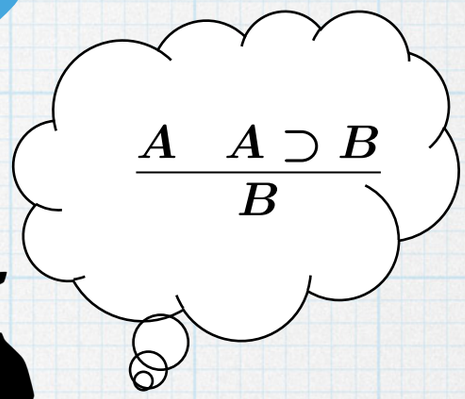
$$\cdots \xrightarrow{d_3} P_2 \xrightarrow{d_2} P_1 \xrightarrow{d_1} P_0 \xrightarrow{\varepsilon} V \longrightarrow 0$$

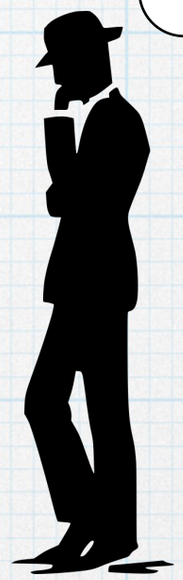
$$\dfrac{\dfrac{p \supset r \qquad \dfrac{p \wedge q}{p}}{r}}{\dfrac{(p \wedge q) \supset r}{(p \supset r) \supset \big((p \wedge q) \supset r\big)}}$$

理論計算機科学とは？







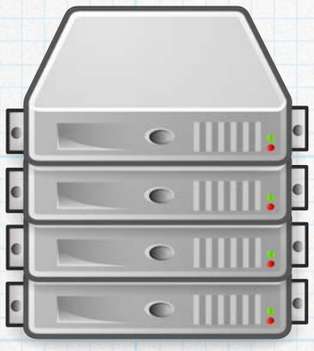


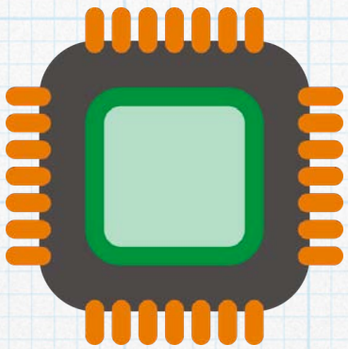
https://www.publicdomainpictures.net/en/view-image.php?image=42208&picture=computer-silhouette
RRZEicons, CC BY-SA 3.0
https://publicdomainvectors.org/en/free-clipart/Generic-computer-chip-vector-image/13455.html

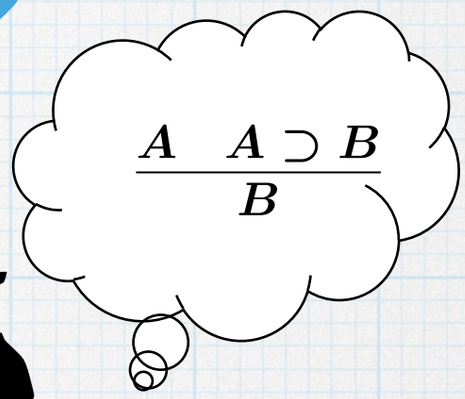
https://svgsilh.com/image/294276.html

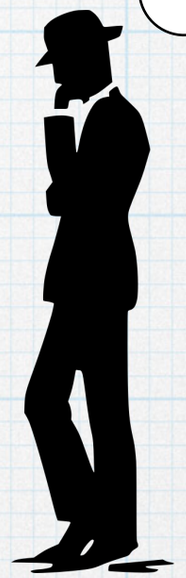
* 計算機，計算機システムの振る舞いを数学的に研究
  * 速さ（アルゴリズム，計算量理論）
  * 正しさ（「バグがないか？」．形式手法，プログラミング言語理論）
* 使う数学：
  * 論理学，代数学，グラフ理論など
  * 離散的（↔ 連続的）
  * 有限（記号の世界）と無限（イデアの世界）をはっきり重視

理論計算機科学とは？









https://www.publicdomainpictures.net/en/view-image.php?image=42208&picture=computer-silhouette
RRZEicons, CC BY-SA 3.0
https://publicdomainvectors.org/en/free-clipart/Generic-computer-chip-vector-image/13455.html

https://svgsilh.com/image/294276.html

* 計算機，計算機システムの振る舞いを数学的に研究
  * 速さ（アルゴリズム，計算量理論）
  * 正しさ（「バグがないか？」．形式手法，プログラミング言語理論）

* 使う数学：
  * 論理学，代数学，グラフ理論など
  * 離散的（↔ 連続的）
  * 有限（記号の世界）と無限（イデアの世界）をはっきり重視

人間の手の届かない無限を，
有限の記号列で表現

理論計算機科学入門 — 有限と無限のあいだ

アウトライン

* 理論計算機科学とは — 有限と無限のせめぎあい
* いろいろな無限 — 対角線論法
* オートマトン理論入門
  * 包含問題を解くアルゴリズム — 有限の方法で無限の対象を操作
  * オートマトンの表現能力 — 有限性の限界
* 形式検証 — オートマトンを用いたシステム品質保証
* AI 時代の形式検証 — 研究紹介

いろいろな「無限」

自然数	0, 1, 2, …
整数	…, -2, -1, 0, 1, 2, …
有理数	「分数」 $\frac{p}{q}$ ✱ 循環小数
実数	有理数 ＋ 無理数 ✱ $\pi = 3.141592\ldots$ とか

いろいろな「無限」

自然数	0, 1, 2, …
含まれる 整数	…, -2, -1, 0, 1, 2, …
含まれる 有理数	「分数」 $\frac{p}{q}$ ＊ 循環小数
含まれる 実数	有理数 ＋ 無理数 ＊ $\pi = 3.141592\ldots$ とか

いろいろな「無限」

自然数	0, 1, 2, …
含まれる 整数	…, -2, -1, 0, 1, 2, …
含まれる 有理数	「分数」 $\frac{p}{q}$ ＊ 循環小数
含まれる 実数	有理数 ＋ 無理数 ＊ $\pi = 3.141592\ldots$ とか

「個数」は本当に違うの？

いろいろな「無限」

自然数	0, 1, 2, …
整数	…, -2, -1, 0, 1, 2, …
有理数	「分数」 $\frac{p}{q}$ ＊ 循環小数
実数	有理数 ＋ 無理数 ＊ $\pi = 3.141592\ldots$ とか

「個数」は本当に違うの？

いろいろな「無限」

* 定義　1対1対応が付くとき，「個数が同じ」
（「濃度が同じ」という）

いろいろな「無限」

もれがないペアリング

* 定義　1対1対応が付くとき，「個数が同じ」
（「濃度が同じ」という）

いろいろな「無限」

もれがないペアリング

* 定義　1対1対応が付くとき，「個数が同じ」
（「濃度が同じ」という）

* 自然数と整数は，濃度が同じ

いろいろな「無限」

もれがないペアリング

* 定義　1対1対応が付くとき，「個数が同じ」
（「濃度が同じ」という）

* 自然数と整数は，濃度が同じ

… -2 -1 0 1 2 …

いろいろな「無限」

もれがないペアリング

* 定義　1対1対応が付くとき，「個数が同じ」
（「濃度が同じ」という）

* 自然数と整数は，濃度が同じ

… -2 -1 0 1 2 …

0

いろいろな「無限」

もれがないペアリング

* 定義　1対1対応が付くとき，「個数が同じ」
（「濃度が同じ」という）

* 自然数と整数は，濃度が同じ

… -2 -1 0 1 2 …

0 1

いろいろな「無限」

もれがないペアリング

* 定義　1対1対応が付くとき，「個数が同じ」
（「濃度が同じ」という）

* 自然数と整数は，濃度が同じ

… -2 -1 0 1 2 …

2 0 1

いろいろな「無限」

もれがないペアリング

* 定義　1対1対応が付くとき，「個数が同じ」
（「濃度が同じ」という）

* 自然数と整数は，濃度が同じ

…	-2	-1	0	1	2	…
		2	0	1	3	

いろいろな「無限」

もれがないペアリング

* 定義　1対1対応が付くとき，「個数が同じ」
（「濃度が同じ」という）

* 自然数と整数は，濃度が同じ

…	-2	-1	0	1	2	…
	4	2	0	1	3	

いろいろな「無限」

もれがないペアリング

* 定義　1対1対応が付くとき，「個数が同じ」
（「濃度が同じ」という）

* 自然数と整数は，濃度が同じ

…	-2	-1	0	1	2	…
…	4	2	0	1	3	…

いろいろな「無限」

* 有理数と自然数も，濃度が同じ

1/1	2/1	3/1	…
1/2	2/2	3/2	…
1/3	2/3	3/3	…
⋮	⋮	⋮	⋱

いろいろな「無限」

* 有理数と自然数も，濃度が同じ

1/1	2/1	3/1	…
1/2	~~2/2~~	3/2	…
1/3	2/3	3/3	…
⋮	⋮	⋮	⋱

いろいろな「無限」

* 有理数と自然数も，濃度が同じ

0	1/1	2/1	3/1 …
1/2	~~2/2~~	3/2	…
1/3	2/3	3/3	…
⋮	⋮	⋮	⋱

いろいろな「無限」

* 有理数と自然数も，濃度が同じ

0 1/1	**1** 2/1	3/1	…
1/2	~~2/2~~	3/2	…
1/3	2/3	3/3	…
⋮	⋮	⋮	⋱

いろいろな「無限」

* 有理数と自然数も，濃度が同じ

0	1/1	2/1	3/1 …
2	1/2	~~2/2~~	3/2 …
	1/3	2/3	3/3 …
	⋮	⋮	⋮ ⋱

いろいろな「無限」

* 有理数と自然数も，濃度が同じ

0 1/1	1 2/1	3 3/1	…
2 1/2	~~2/2~~	3/2	…
1/3	2/3	3/3	…
⋮	⋮	⋮	⋱

いろいろな「無限」

* 有理数と自然数も，濃度が同じ

0 1/1	1 2/1	3 3/1	…
2 1/2	~~2/2~~	3/2	…
4 1/3	2/3	3/3	…
⋮	⋮	⋮	⋱

いろいろな「無限」

* 有理数と自然数も，濃度が同じ

0 1/1	1 2/1	3 3/1	5 …
2 1/2	~~2/2~~	3/2	…
4 1/3	2/3	3/3	…
⋮	⋮	⋮	⋱

いろいろな「無限」

* 有理数と自然数も，濃度が同じ

0 1/1	1 2/1	3 3/1	5 …
2 1/2	~~2/2~~	6 3/2	…
4 1/3	2/3	3/3	…
⋮	⋮	⋮	⋱

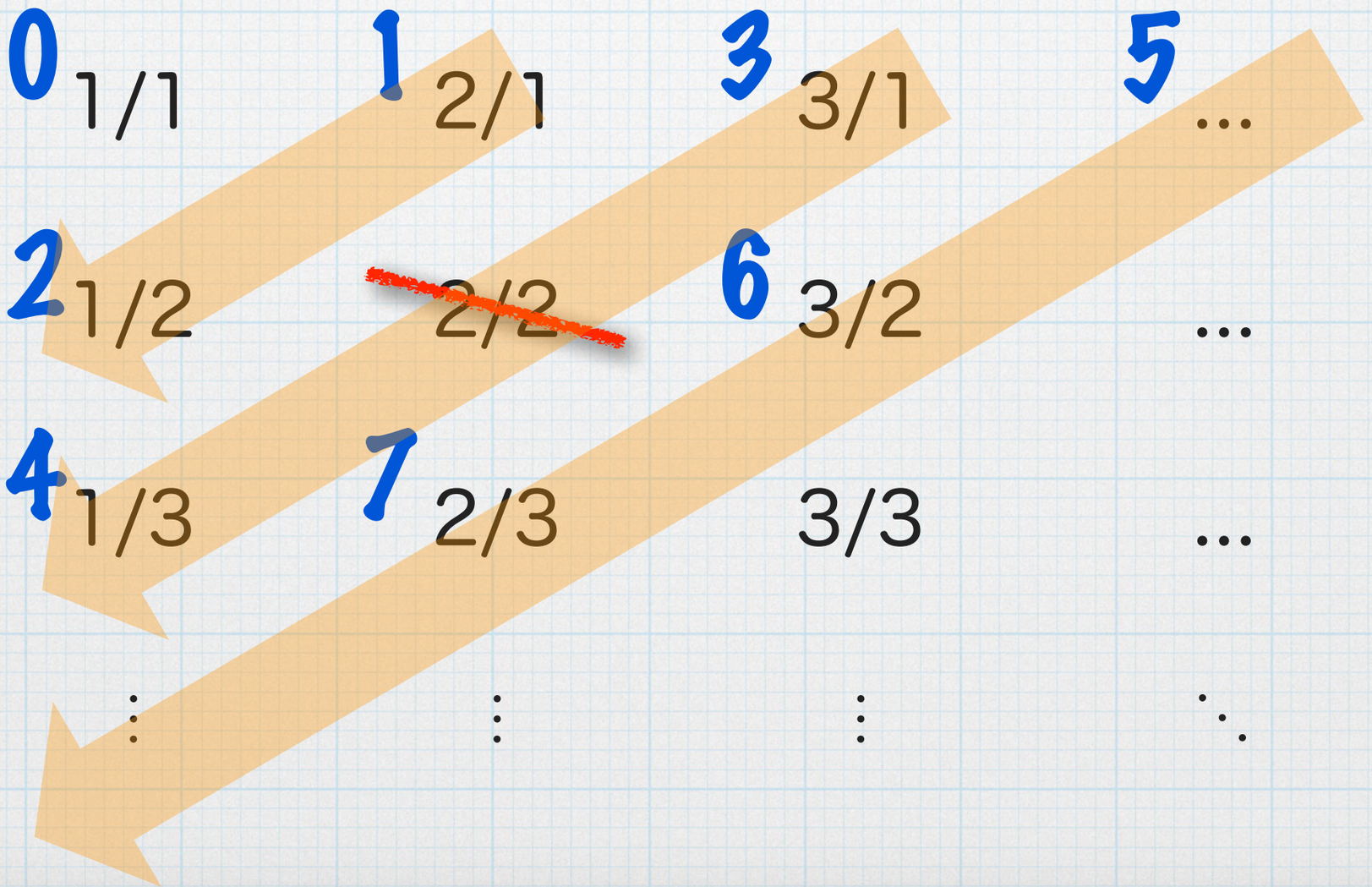
いろいろな「無限」

* 有理数と自然数も，濃度が同じ

0 1/1	1 2/1	3 3/1	5 …
2 1/2	~~2/2~~	6 3/2	…
4 1/3	7 2/3	3/3	…
⋮	⋮	⋮	⋱

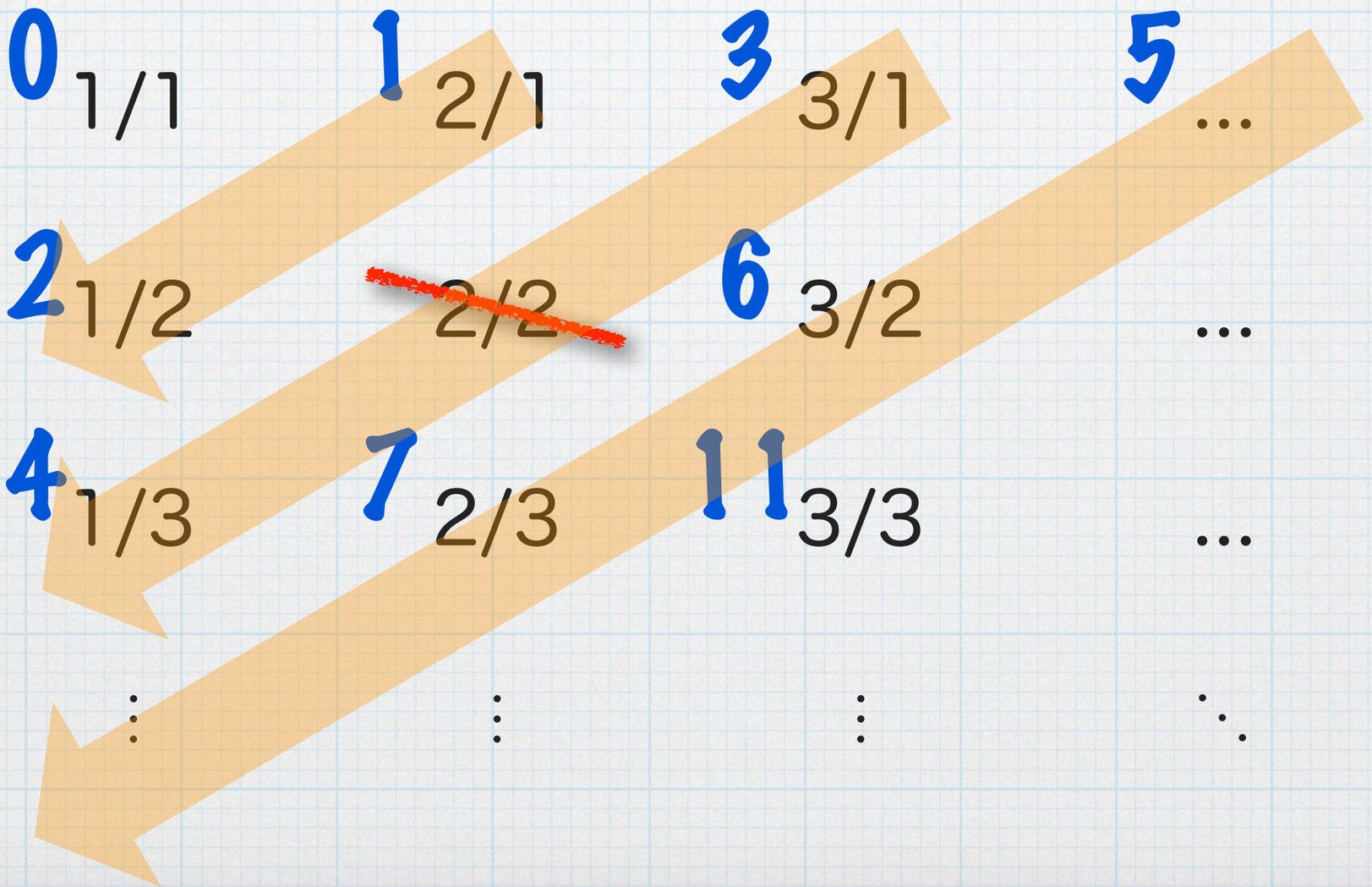
いろいろな「無限」

* 有理数と自然数も，濃度が同じ



いろいろな「無限」

* 有理数と自然数も，濃度が同じ



いろいろな「無限」

自然数	0, 1, 2, …
整数	…, -2, -1, 0, 1, 2, …
有理数	「分数」 $\frac{p}{q}$ ＊ 循環小数
実数	有理数 ＋ 無理数 ＊ $\pi = 3.141592\ldots$ とか

「個数」は本当に違うの？

いろいろな「無限」

自然数	0, 1, 2, …
整数	…, -2, -1, 0, 1, 2, …
有理数	「分数」 $\frac{p}{q}$ ＊ 循環小数
実数	有理数 ＋ 無理数 ＊ $\pi = 3.141592\ldots$ とか

「個数」は本当に違うの？

対角線論法

* 定理　実数と自然数は，濃度がちがう．
つまり，1対1対応がない．

対角線論法

* 定理　実数と自然数は，濃度がちがう．
  つまり，1対1対応がない．

* 証明（背理法）
  仮に，1対1対応があるとしよう．（矛盾を導く）

対角線論法

* 定理　実数と自然数は，濃度がちがう．
  つまり，1対1対応がない．

* 証明（背理法）
  仮に，1対1対応があるとしよう．（矛盾を導く）

0	0	.	1	8	2	6	…
1	0	.	4	0	3	7	…
2	0	.	2	1	5	6	…
3	0	.	3	1	4	1	…
⋮			⋮	⋮	⋮	⋮	⋱

対角線論法

* 定理　実数と自然数は，濃度がちがう．
  つまり，1対1対応がない．

* 証明（背理法）
  仮に，1対1対応があるとしよう．（矛盾を導く）

0	0	.	1	8	2	6	…
1	0	.	4	0	3	7	…
2	0	.	2	1	5	6	…
3	0	.	3	1	4	1	…
⋮			⋮	⋮	⋮	⋮	⋱

対角線論法

* 定理　実数と自然数は，濃度がちがう．
  つまり，1対1対応がない．

* 証明（背理法）
  仮に，1対1対応があるとしよう．（矛盾を導く）

0	0	.	1	8	2	6	…
1	0	.	4	0	3	7	…
2	0	.	2	1	5	6	…
3	0	.	3	1	4	1	…
⋮			⋮	⋮	⋮	⋮	⋱

対角線を見て，実数
0.1051… を作る

対角線論法

* 定理　実数と自然数は，濃度がちがう．
  つまり，1対1対応がない．

* 証明（背理法）
  仮に，1対1対応があるとしよう．（矛盾を導く）

0	0	.	1	8	2	6	…
1	0	.	4	0	3	7	…
2	0	.	2	1	5	6	…
3	0	.	3	1	4	1	…
⋮			⋮	⋮	⋮	⋮	⋱

対角線を見て，実数
0.1051… を作る

各桁の数字をずらして，
0.2162… を考えよう

対角線論法

* 定理　実数と自然数は，濃度がちがう．
  つまり，1対1対応がない．

* 証明（背理法）
  仮に，1対1対応があるとしよう．（矛盾を導く）

0	0	.	1	8	2	6	…
1	0	.	4	0	3	7	…
2	0	.	2	1	5	6	…
3	0	.	3	1	4	1	…
⋮			⋮	⋮	⋮	⋮	⋱

対角線を見て，実数
0.1051… を作る

各桁の数字をずらして，
0.2162… を考えよう

0.2162… は何番目の実数？

対角線論法

* 定理　実数と自然数は，濃度がちがう．
  つまり，1対1対応がない．

* 証明（背理法）
  仮に，1対1対応があるとしよう．（矛盾を導く）

0	0	.	1	8	2	6 …
1	0	.	4	0	3	7 …
2	0	.	2	1	5	6 …
3	0	.	3	1	4	1 …
⋮			⋮	⋮	⋮	⋮ ⋱
k	0	.	2	1	6	2 …

対角線を見て，実数
0.1051… を作る

各桁の数字をずらして，
0.2162… を考えよう

0.2162… は何番目の実数？

対角線論法

* 定理　実数と自然数は，濃度がちがう．
  つまり，1対1対応がない．

* 証明（背理法）
  仮に，1対1対応があるとしよう．（矛盾を導く）

0	0	.	1	8	2	6 …
1	0	.	4	0	3	7 …
2	0	.	2	1	5	6 …
3	0	.	3	1	4	1 …
⋮			⋮	⋮	⋮	⋮ ⋱
k	0	.	2	1	6	2 …

対角線を見て，実数
0.1051… を作る

各桁の数字をずらして，
0.2162… を考えよう

0.2162… は何番目の実数？

対角線論法

* 定理　実数と自然数は，濃度がちがう．
  つまり，1対1対応がない．

* 証明（背理法）
  仮に，1対1対応があるとしよう．（矛盾を導く）

0	0	.	1	8	2	6 …
1	0	.	4	0	3	7 …
2	0	.	2	1	5	6 …
3	0	.	3	1	4	1 …
⋮			⋮	⋮	⋮	⋮ ⋱
k	0	.	2	1	6	2 …

対角線を見て，実数
0.1051... を作る

各桁の数字をずらして，
0.2162... を考えよう

0.2162... は何番目の実数？

➜ ない！　（矛盾）

対角線論法
（予備スライド）

* 定理　実数と自然数は，濃度がちがう.
つまり，1対1対応がない.

* 証明（背理法）
仮に，1対1対応があるとしよう.　（矛盾を導く）

0	0	.	1	8	2	6	…
1	0	.	4	0	3	7	…
2	0	.	2	1	5	6	…
3	0	.	3	1	4	1	…
⋮			⋮	⋮	⋮	⋮	⋱

0.2162... は何番目？
➜ ない！　（矛盾）

* 対角線から作られる実数（の10進表記）を
0. n_1 n_2 n_3 … としよう.

* あらたな実数 0. m_1 m_2 m_3 … を，
$n_1 \neq m_1$, $n_2 \neq m_2$, …　　（＊）
となるように作る
（たとえば，数字をひとつずらそう）

* 0. m_1 m_2 m_3 … は左の対応表に現れるはず
k 番目としよう.

* しかし，すると0. m_1 m_2 m_3 … が
対角線とぶつかるところの数字 m_k は，
n_k と同じはず
➜ 上の（＊）に矛盾！ 証明終わり.

対角線論法
（予備スライド）

* 定理　実数と自然数は，濃度がちがう.
つまり，1対1対応がない.

* 証明（背理法）
仮に，1対1対応があるとしよう.　（矛盾を導く）

0	0 . 1 8 2 6 …
1	0 . 4 0 3 7 …
2	0 . 2 1 5 6 …
3	0 . 3 1 4 1 …
⋮	⋮ ⋮ ⋮ ⋮ ⋱
k	0 . m_1 m_2 m_3 ⋯

0.2162... は何番目？
➜ ない！　（矛盾）

* 対角線から作られる実数（の10進表記）を
0. n_1 n_2 n_3 ⋯
としよう.

* あらたな実数 0. m_1 m_2 m_3 ⋯ を，
$n_1 \neq m_1$, $n_2 \neq m_2$, ⋯　　（＊）
となるように作る
（たとえば，数字をひとつずらそう）

* 0. m_1 m_2 m_3 ⋯ は左の対応表に現れるはず
k 番目としよう.

* しかし，すると0. m_1 m_2 m_3 ⋯ が
対角線とぶつかるところの数字 m_k は，
n_k と同じはず
➜ 上の（＊）に矛盾！ 証明終わり.

対角線論法
（予備スライド）

* 定理　実数と自然数は，濃度がちがう.
つまり，1対1対応がない.

* 証明（背理法）
仮に，1対1対応があるとしよう.　（矛盾を導く）

1	0	.	4	0	3	7	…
2	0	.	2	1	5	6	…
3	0	.	3	1	4	1	…
⋮			⋮	⋮	⋮	⋮	⋱
k	0	.	m_1	m_2	m_3	⋯	

0　0 . 1 8 2 6 …

n_1 n_2 n_3 n_4

0.2162... は何番目？
➜ ない！　（矛盾）

* 対角線から作られる実数（の10進表記）を
0. n_1 n_2 n_3 ⋯
としよう.

* あらたな実数 0. m_1 m_2 m_3 ⋯ を，
$n_1 \neq m_1$, $n_2 \neq m_2$, ⋯　　（＊）
となるように作る
（たとえば，数字をひとつずらそう）

* 0. m_1 m_2 m_3 ⋯ は左の対応表に現れるはず
k 番目としよう.

* しかし，すると0. m_1 m_2 m_3 ⋯ が
対角線とぶつかるところの数字 m_k は，
n_k と同じはず
➜ 上の（＊）に矛盾！ 証明終わり.

対角線論法
（予備スライド）

* 定理　実数と自然数は，濃度がちがう.
つまり，1対1対応がない.

* 証明（背理法）
仮に，1対1対応があるとしよう.　（矛盾を導く）

1	0	.	4	0	3	7	…
2	0	.	2	1	5	6	…
3	0	.	3	1	4	1	…
⋮			⋮	⋮	⋮	⋮	⋱
k	0	.	m_1	m_2	m_3	…	

0　0 . 1 8 2 6 …

n_1 n_2 n_3 n_4

0.2162... は何番目？
➜ ない！　（矛盾）

* 対角線から作られる実数（の10進表記）を
0. n_1 n_2 n_3 … としよう.

* あらたな実数 0. m_1 m_2 m_3 … を，
$n_1 \neq m_1$, $n_2 \neq m_2$, …　（＊）
となるように作る
（たとえば，数字をひとつずらそう）

* 0. m_1 m_2 m_3 … は左の対応表に現れるはず
k 番目としよう.

* しかし，すると0. m_1 m_2 m_3 … が
対角線とぶつかるところの数字 m_k は，
n_k と同じはず
➜ 上の（＊）に矛盾！ 証明終わり.

対角線論法

* 定義
  集合 X の power set P(X) とは，X の部分集合全体の集合

* 例　X = {a, b, c} なら，
  P(X) = { {}，{a}，{b}，{c}，{b,c}，{c,a}，{a,b}，{a,b,c} }

* 練習問題
  有限集合 X （元は n 個）に対して，P(X) の元の個数は？

対角線論法

* 定義
  集合 X の power set P(X) とは，X の部分集合全体の集合

* 例　X = {a, b, c} なら，
  P(X) = { {}，{a}，{b}，{c}，{b,c}，{c,a}，{a,b}，{a,b,c} }

* 練習問題
  有限集合 X （元は n 個）に対して，P(X) の元の個数は？

* 答え： 2^n 個

対角線論法

* 定理　任意の集合 X に対して，X と P(X) の間に1対1対応はない

対角線論法

* 定理　任意の集合 X に対して，X と P(X) の間に1対1対応はない

* 証明（背理法）
  仮に，1対1対応 f: X → P(X) があるとしよう．（矛盾を導く）

（アイデア： ヤバそうな P(X) の元を持ってくる）

U = {x | x は f(x) に含まれない } と定義する．U は P(X) の元．

f: X → P(X) は1対1対応なので，X の元 u であって，f(u) = U
となるものが存在する．ここで，u が U に含まれるかを考える．

* u が U に含まれる場合： f(u) = U より，u は f(u) に含まれる．
  U の定義より，u は U に含まれない ➜ 矛盾！

* u が U に含まれない場合： f(u) = U より，u は f(u) に含まれない．
  U の定義より，u は U に含まれる ➜ 矛盾！

対角線論法

なんかモヤモヤする．．．
➜ 本質は
「嘘つきのパラドックス」と同じ

* 定理　任意の集合 X に対して，X と P(X) の間に1対1対応はない
* 証明（背理法）
  仮に，1対1対応 f: X → P(X) があるとしよう．（矛盾を導く）

（アイデア： ヤバそうな P(X) の元を持ってくる）

U = {x | x は f(x) に含まれない } と定義する．U は P(X) の元．

f: X → P(X) は1対1対応なので，X の元 u であって，f(u) = U
となるものが存在する．ここで，u が U に含まれるかを考える．

* u が U に含まれる場合： f(u) = U より，u は f(u) に含まれる．
  U の定義より，u は U に含まれない ➜ 矛盾！
* u が U に含まれない場合： f(u) = U より，u は f(u) に含まれない．
  U の定義より，u は U に含まれる ➜ 矛盾！

うそつきのパラドックス

うそつきのパラドックス

* 「私の言っていることはウソである」

うそつきのパラドックス

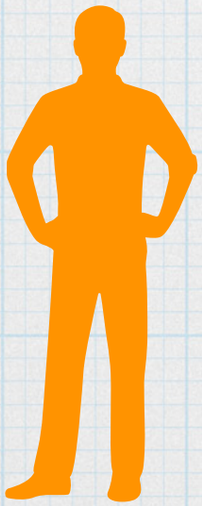
* 「私の言っていることはウソである」

* 「クレタ人はうそつきばっかりだ」
『ところであなた，ご出身は？』
「クレタです！」

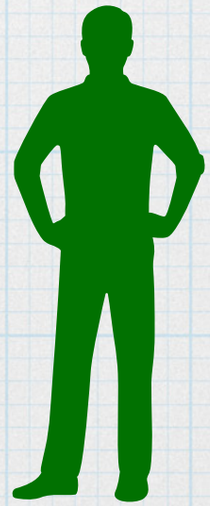
うそつきのパラドックス

Bさんの言ってること
とは本当です

Aさんの言ってること
とはウソです



A



B

うそつきのパラドックス

* 「私の言っていることはウソである」
* ➜ 否定的なメタ言及

* 対角線論法の本質.
  対角線論法はいろんなところで使われる
  * 自然数と実数の濃度の違いの証明
  * 停止問題の決定不可能性の証明
  * ゲーデルの不完全性定理の証明

理論計算機科学入門 — 有限と無限のあいだ

アウトライン

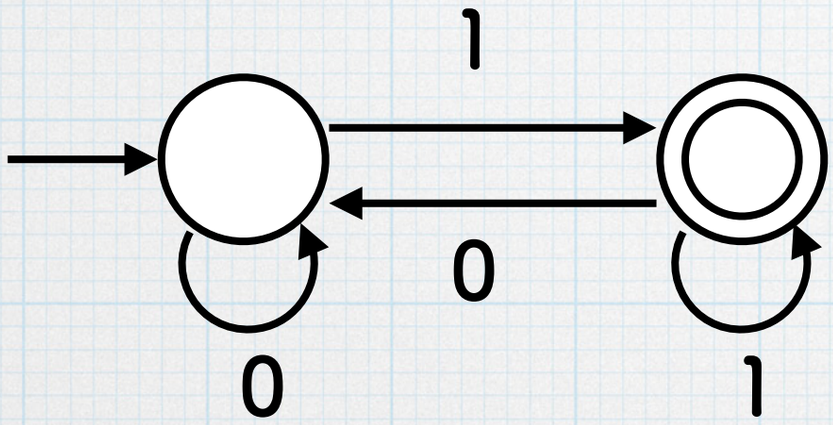
* 理論計算機科学とは — 有限と無限のせめぎあい
* いろいろな無限 — 対角線論法
* オートマトン理論入門
  * 包含問題を解くアルゴリズム — 有限の方法で無限の対象を操作
  * オートマトンの表現能力 — 有限性の限界
* 形式検証 — オートマトンを用いたシステム品質保証
* AI 時代の形式検証 — 研究紹介

オートマトン

https://ja.wikipedia.org/wiki/%E3%83%95%E3%82%A1%E3%82%A4%E3%83%AB:Pierrot_%C3%A9crivain.jpg

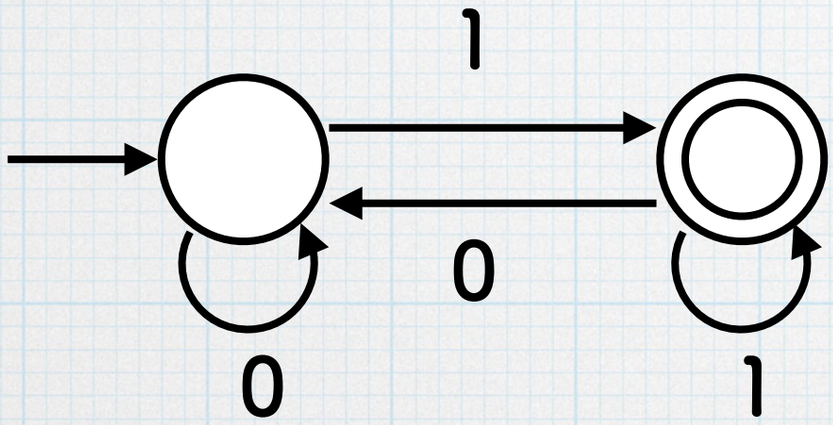
* 機械仕掛けの自動人形
* 左：
  野坂オートマタ美術館所蔵
  （静岡県伊東市）
  「手紙を書くピエロ」
  レオポール・ランベール
  作　1900年
* … この話ではありません

有限オートマトン



* 計算モデルのうち，単純なものの一つ
  * 計算モデル：
    「計算とは何か？」の数学的定義
  * さまざまな計算モデル
  * 有限オートマトン
  * < …
  * < プッシュダウン・オートマトン
  * < …
  * < チューリングマシン

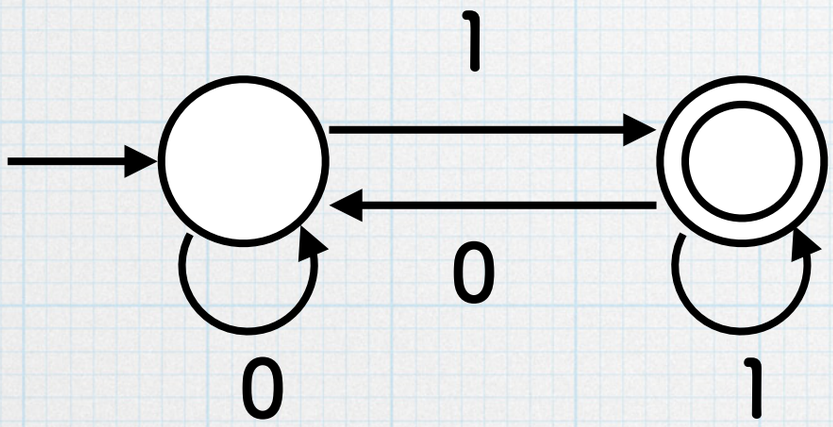
有限オートマトン



* 計算モデルのうち，単純なものの一つ
  * 計算モデル：
    「計算とは何か？」の数学的定義
  * さまざまな計算モデル
  * 有限オートマトン
  * < …
  * < プッシュダウン・オートマトン
  * < …
  * < チューリングマシン

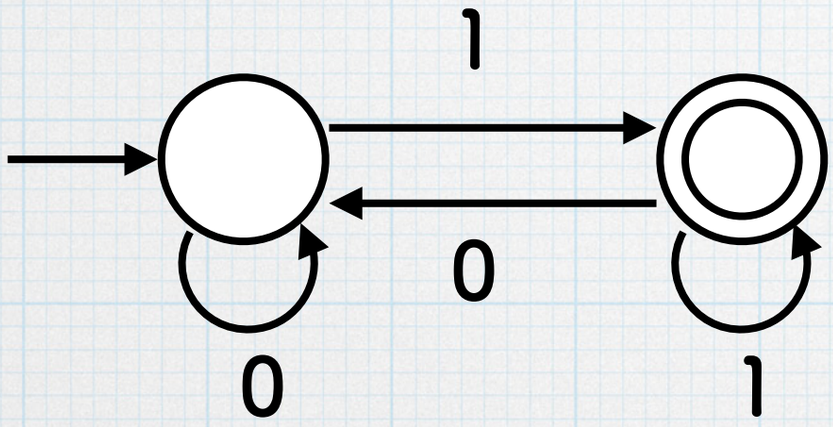
Church の提題：
「チューリングマシンでできることを『計算』と呼ぼう」

有限オートマトン



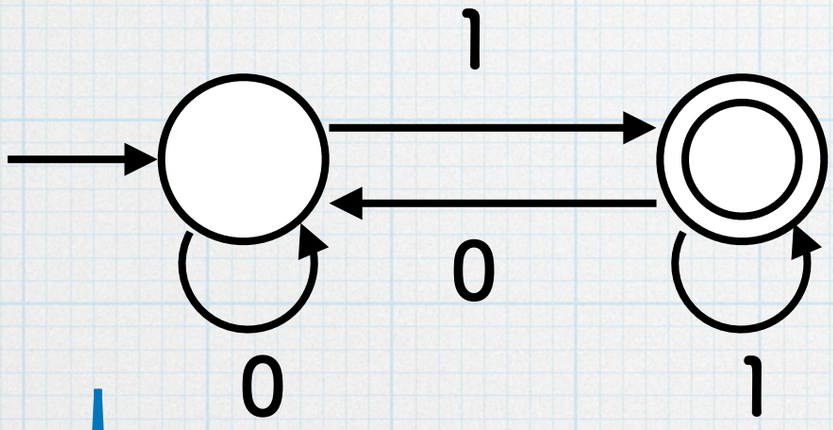
* 有限個の状態（ここでは2個）
* 状態間の遷移（矢印）
  * 各遷移は文字でラベル付け
    （ここでの文字は0,1）
* 初期状態
  （左の状態．矢印で表現）
* 状態の「色付け」．2色：
  * ○： 非受理状態
  * ◎： 受理状態

有限オートマトン



* 機能：
  * 有限長の文字列を読んで，
  * 「受理（OK!）」あるいは「非受理（NG!）」と言う
* 初期状態から，文字列に沿って遷移をたどっていって，最後にたどり着いた状態が
  * ◎ なら「受理」
  * ○ なら「非受理」

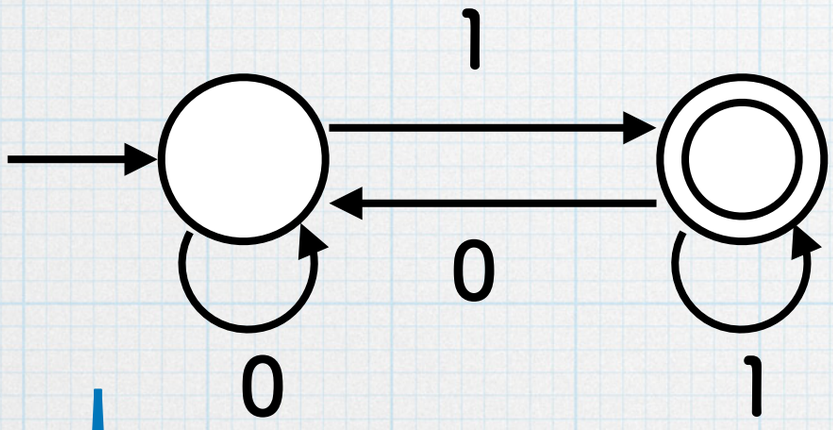
有限オートマトン



00 は非受理

* 機能：
  * 有限長の文字列を読んで，
  * 「受理 (OK!)」あるいは「非受理 (NG!)」と言う
* 初期状態から，文字列に沿って遷移をたどっていって，最後にたどり着いた状態が
  * ◎ なら「受理」
  * ○ なら「非受理」

有限オートマトン

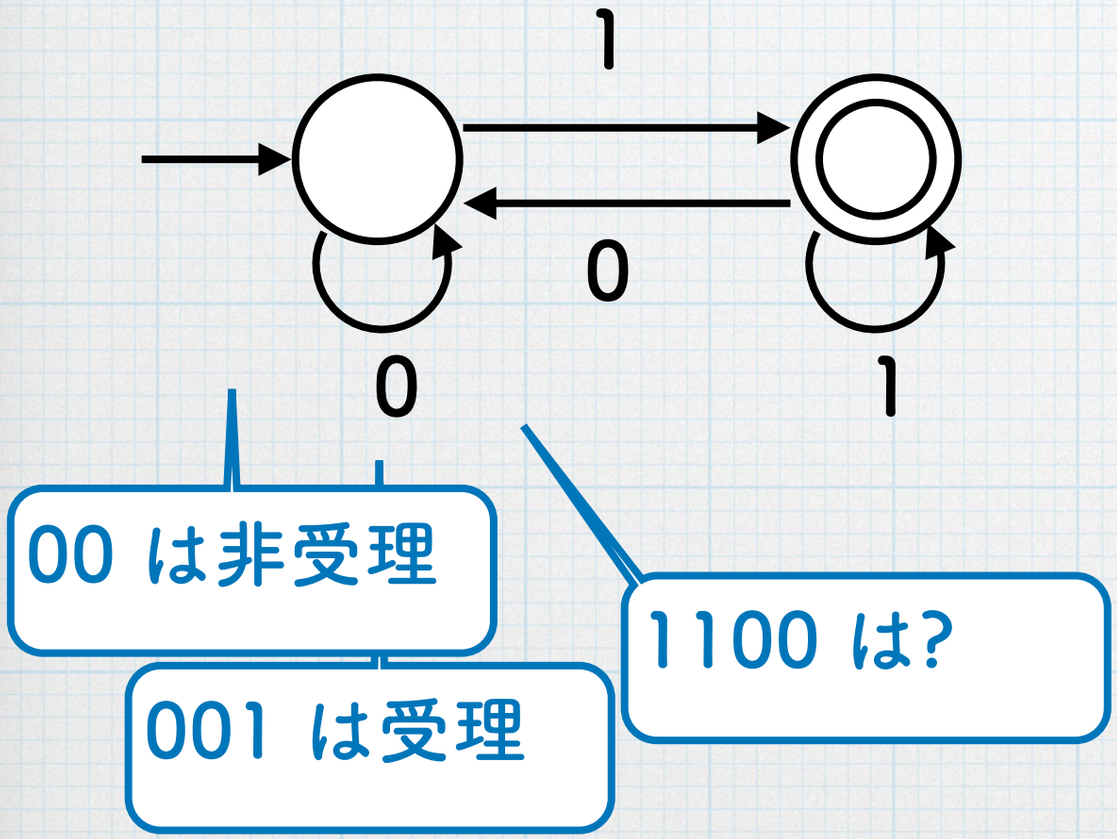


00 は非受理

001 は受理

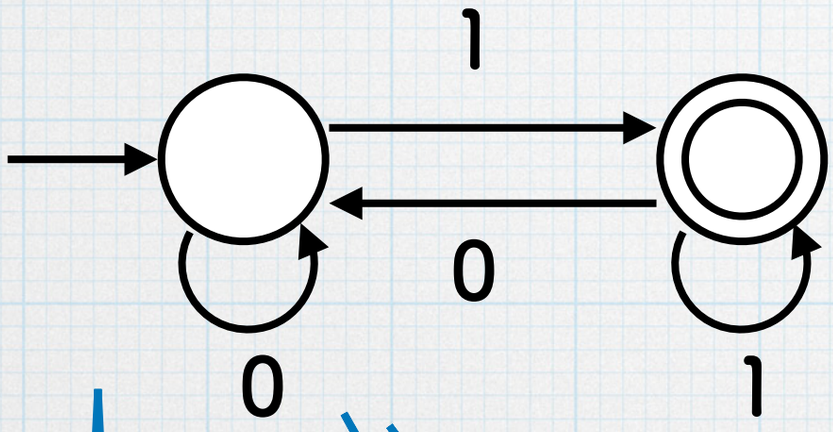
* 機能：
  * 有限長の文字列を読んで，
  * 「受理 (OK!)」あるいは「非受理 (NG!)」と言う
* 初期状態から，文字列に沿って遷移をたどっていって，最後にたどり着いた状態が
  * ◎ なら「受理」
  * ○ なら「非受理」

有限オートマトン



* 機能：
  * 有限長の文字列を読んで，
  * 「受理 (OK!)」あるいは「非受理 (NG!)」と言う
* 初期状態から，文字列に沿って遷移をたどっていって，最後にたどり着いた状態が
  * ◎ なら「受理」
  * ○ なら「非受理」

有限オートマトン



00 は非受理

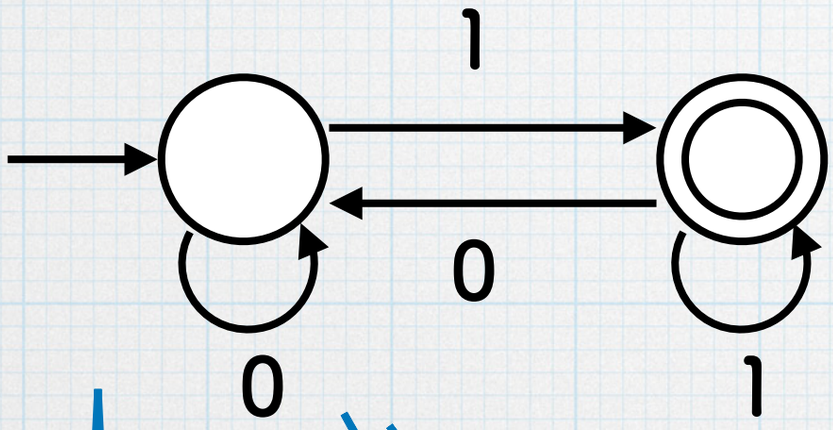
001 は受理

1100 は?

空列（長さ0の文字列）は？

* 機能：
  * 有限長の文字列を読んで，
  * 「受理 (OK!)」あるいは「非受理 (NG!)」と言う
* 初期状態から，文字列に沿って遷移をたどっていって，最後にたどり着いた状態が
  * ◎ なら「受理」
  * ○ なら「非受理」

有限オートマトン



00 は非受理

001 は受理

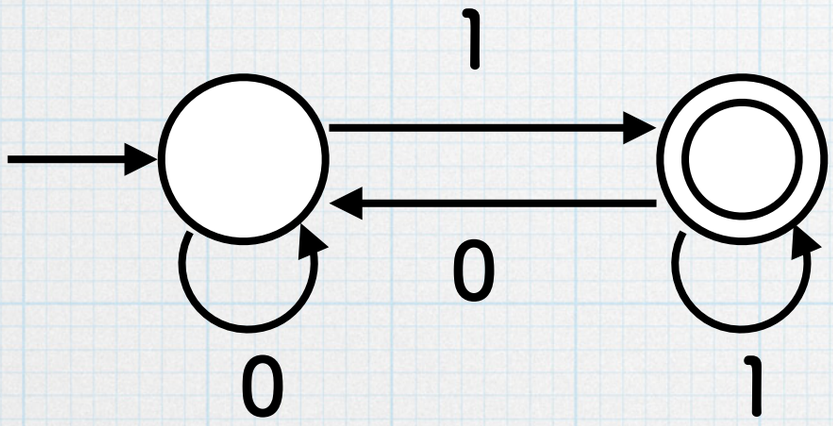
… 一般的な特徴づけができるか

1100 は?

空列（長さ0の文字列）は？

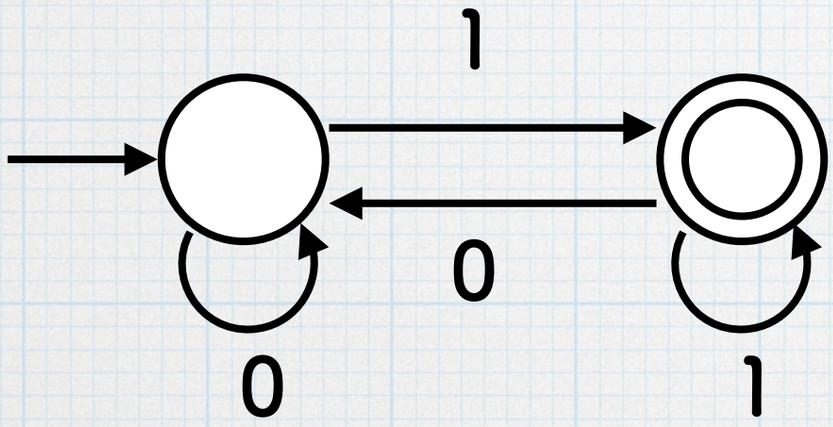
* 機能：
  * 有限長の文字列を読んで，
  * 「受理 (OK!)」あるいは「非受理 (NG!)」と言う
* 初期状態から，文字列に沿って遷移をたどっていって，最後にたどり着いた状態が
  * ◎ なら「受理」
  * ○ なら「非受理」

有限状態オートマトン



* 機能：
  * 有限の文字列を読んで，
  * 「受理（OK!）」あるいは「非受理（NG!）」と言う
* 初期状態から，文字列に沿って遷移をたどっていって，最後にたどり着いた状態が
  * ◎ なら「受理」
  * ○ なら「非受理」

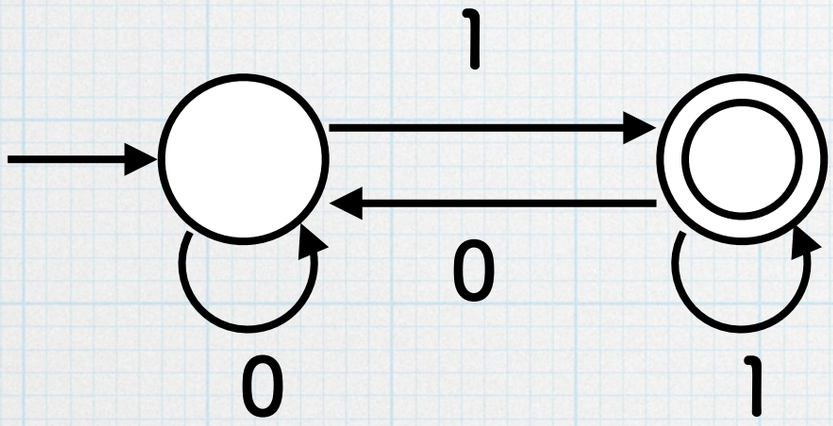
有限状態オートマトン



文字列 w が受理される
⇔
w の最後の文字が 1

* 機能：
  * 有限の文字列を読んで，
  * 「受理 (OK!)」あるいは「非受理 (NG!)」と言う
* 初期状態から，文字列に沿って遷移をたどっていって，最後にたどり着いた状態が
  * ◎ なら「受理」
  * ○ なら「非受理」

有限状態オートマトン



文字列 w が受理される

⇔

w の最後の文字が 1

* 機能：
  * 有限の文字列を読んで，
  * 「受理 (OK!)」あるいは
    「非受理 (NG!)」と言う

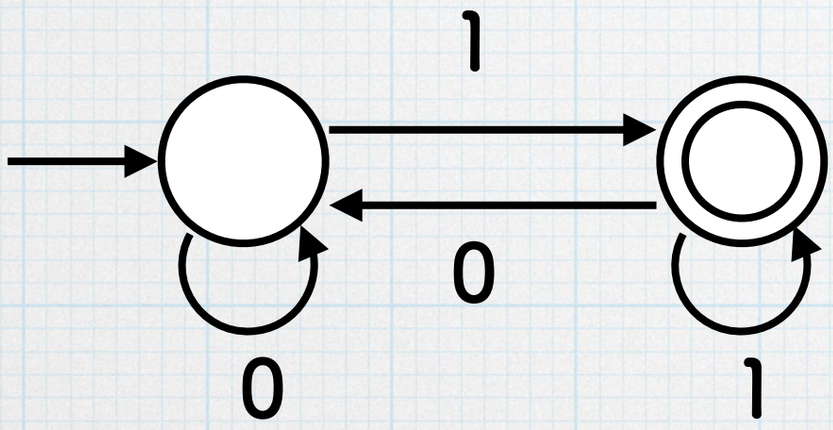
* すなわち，オートマトン M は，
  文字列の集合 L(M) を
  表現する（ひきおこす）．
  ここでは，

$$L(M) = \{ w1 : \ w \text{ は任意の文字列} \}$$
$$= \{1, 01, 11, 001, 011, 101, 111, \ldots\}$$

有限状態オートマトン

M は有限.
特に，計算機上で表現・処理可能

```
States: 2
Start: 0
Acceptance: 1
char: "0" "1"
--BODY--
State: 0
  [0] 0
  [1] 1
State: 1
  [0] 0
  [1] 1
--END--
```



文字列 w が受理される
⇔
w の最後の文字が 1

* 有限の文字列を読んで，
* 「受理 (OK!)」あるいは
  「非受理 (NG!)」と言う

* すなわち，オートマトン M は，
  文字列の集合 L(M) を
  表現する（ひきおこす）．
  ここでは，

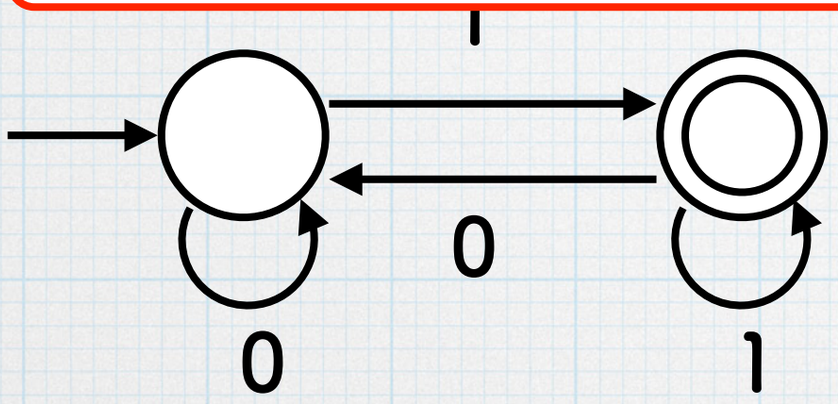
$$L(M) = \{ w \mid w \text{ は任意の文字列} \}$$
$$= \{1, 01, 11, 001, 011, 101, 111, \ldots\}$$

有限状態オ

M は有限.
特に，計算機上で表現・処理可能

```
States: 2
Start: 0
Acceptance: 1
char: "0" "1"
--BODY--
State: 0
  [0] 0
  [1] 1
State: 1
  [0] 0
  [1] 1
--END--
```

L(M) は（一般には）は無限集合.
計算機上で（そのままでは）表現・処理不可能

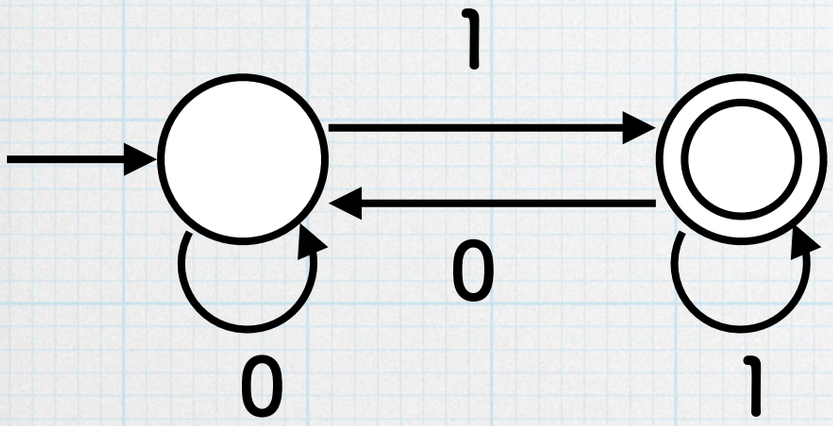


文字列 w が受理される
⇔
w の最後の文字が 1

…で，
…るいは
「非受理 (NG!)」と言う

* すなわち，オートマトン M は，
文字列の集合 L(M) を
表現する（ひきおこす）.
ここでは，
L(M) = { w1 ： w は任意の文字列}
= {1, 01, 11,
001, 011, 101, 111,
… }

有限オートマトン



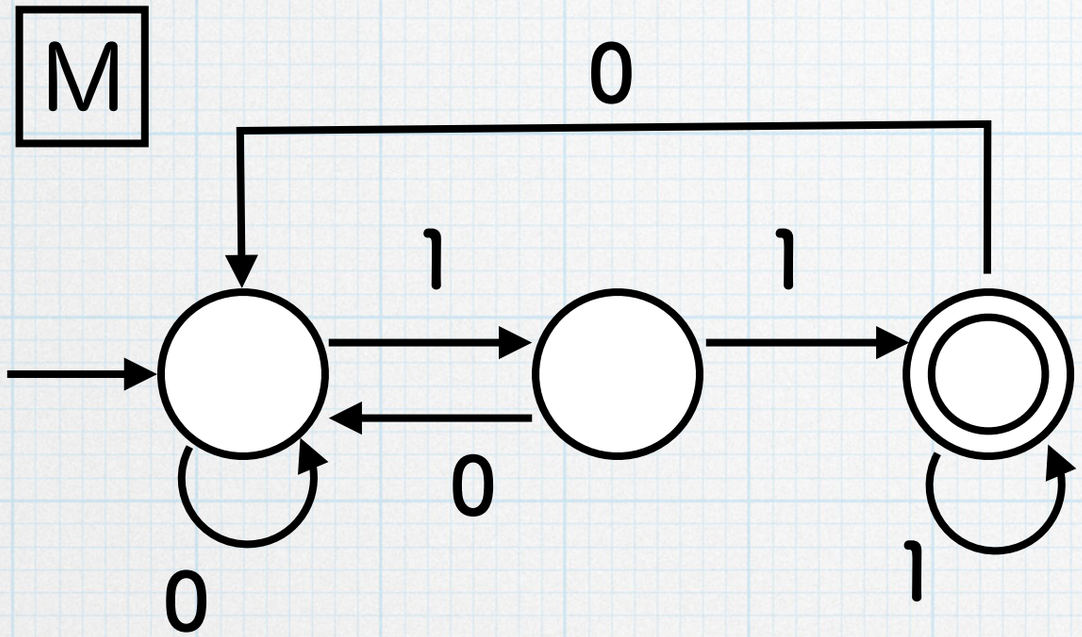
* つまり…
  * 無限の数学的実体 L(M) を
  * 有限のフォーマリズムであるオートマトンMが表現している
* 有限と無限のせめぎあい
* これからの話題
  * オートマトンのアルゴリズム処理
    （「有限で良かった！」）
  * オートマトンの表現能力の限界

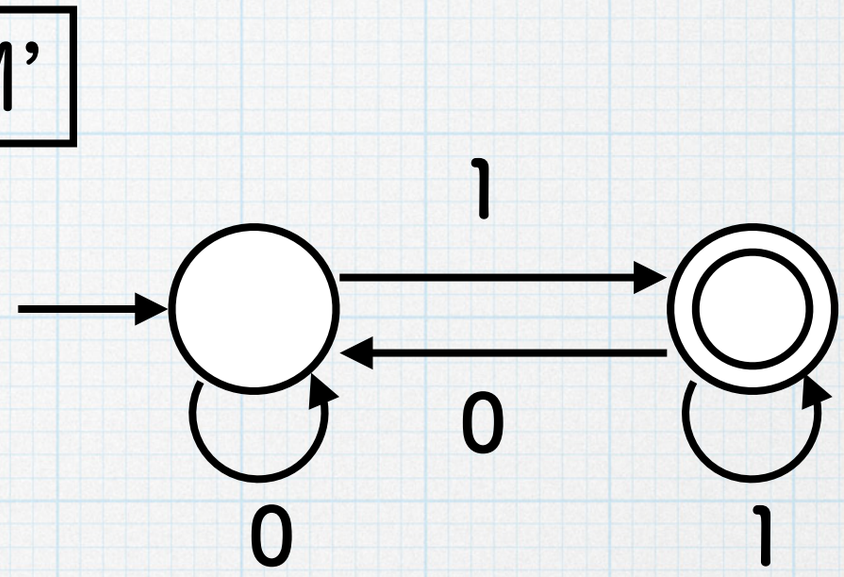
理論計算機科学入門 — 有限と無限のあいだ

アウトライン

* 理論計算機科学とは — 有限と無限のせめぎあい
* いろいろな無限 — 対角線論法
* オートマトン理論入門
  * 包含問題を解くアルゴリズム — 有限の方法で無限の対象を操作
  * オートマトンの表現能力 — 有限性の限界
* 形式検証 — オートマトンを用いたシステム品質保証
* AI 時代の形式検証 — 研究紹介

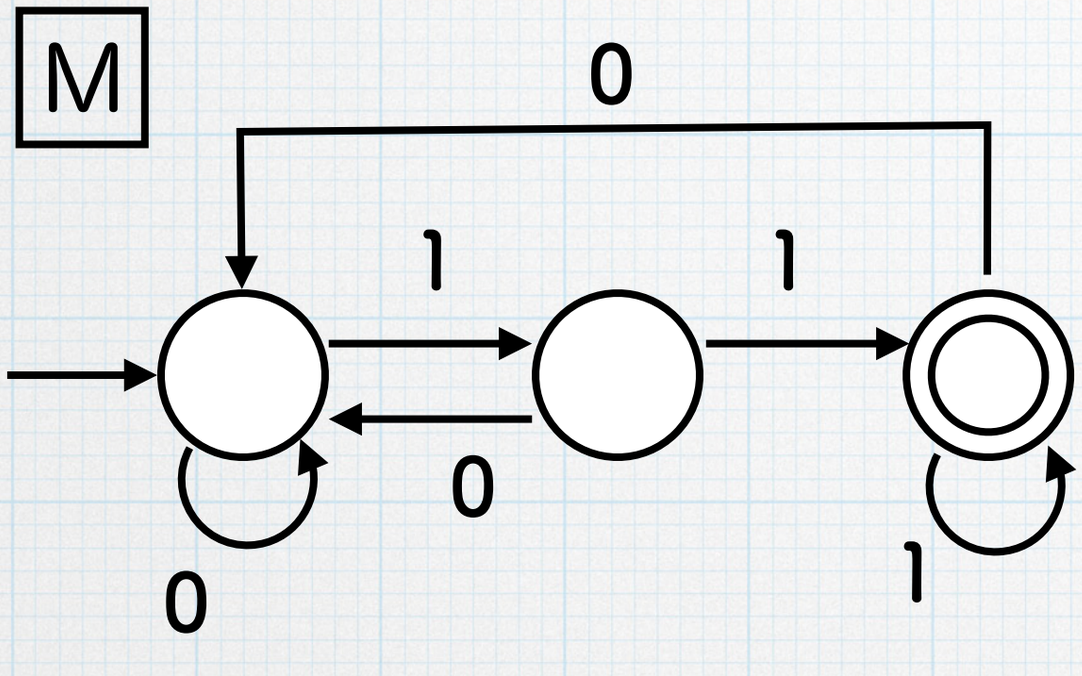
オートマトンの包含問題

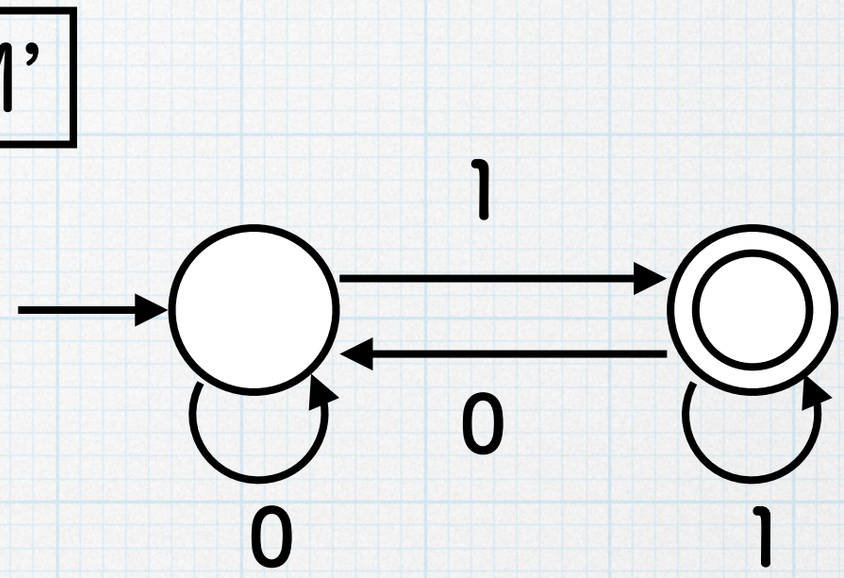




* <u>クイズ</u>　次は成り立つ？　$L(M) \subseteq L(M')$

オートマトンの包含問題



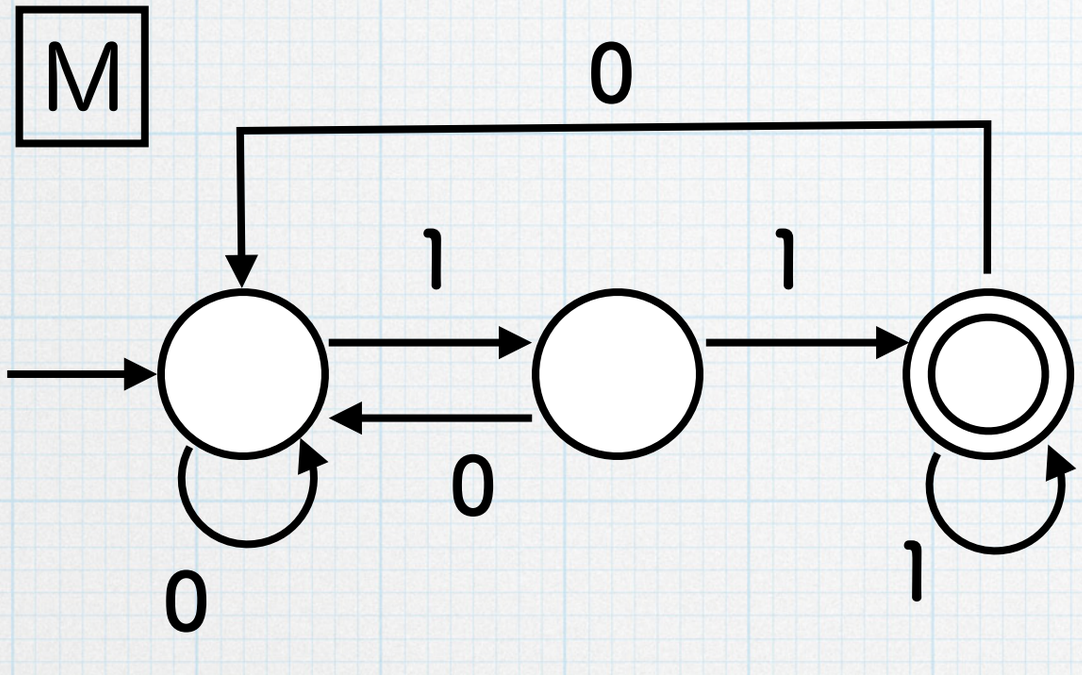


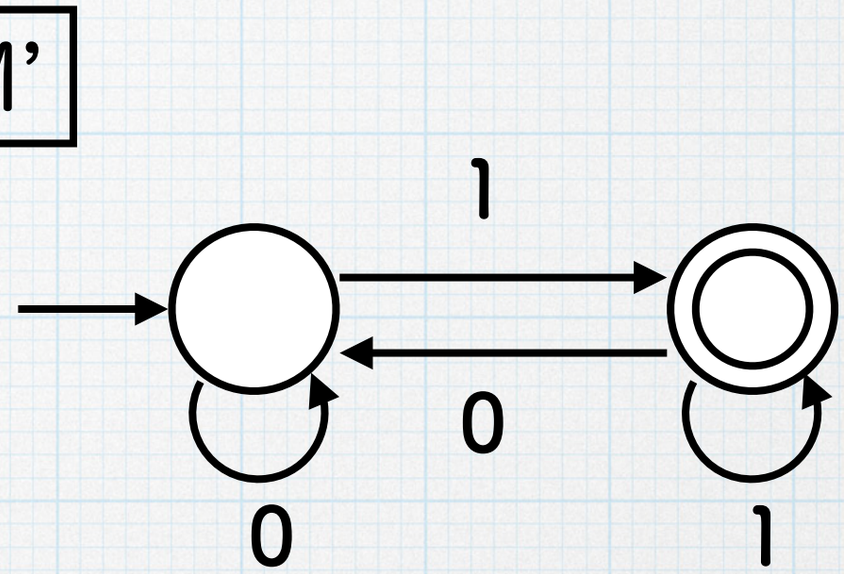
* クイズ　次は成り立つ？ $L(M) \subseteq L(M')$
* 答え Yes!

  L(M) = { 11 で終わる文字列全体 }

  L(M′) = { 1 で終わる文字列全体 }

オートマトンの包含問題





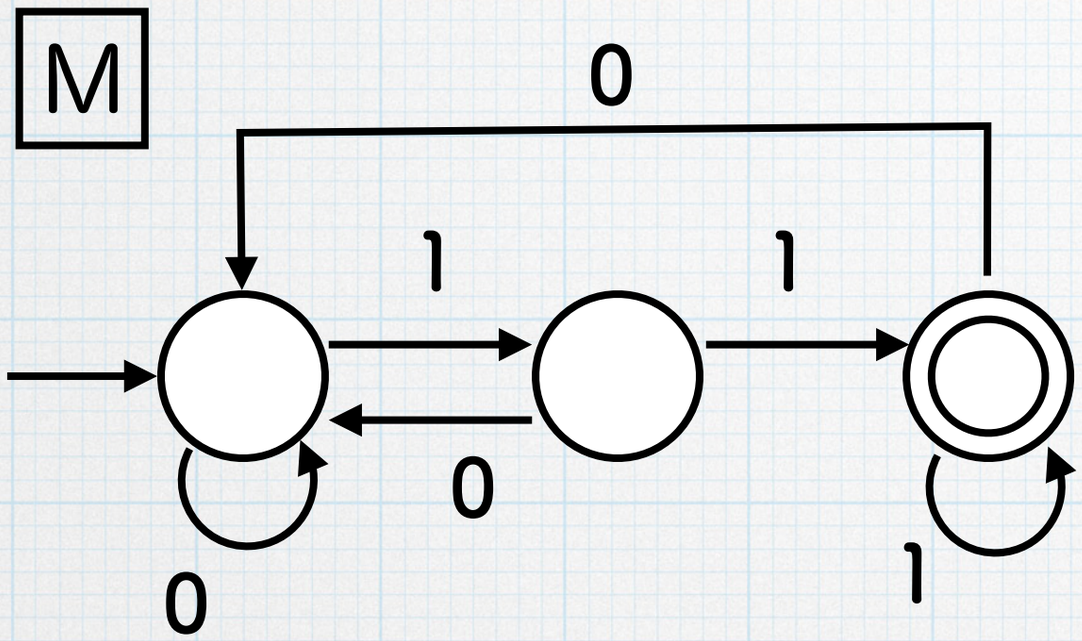
* クイズ　次は成り立つ？ $L(M) \subseteq L(M')$
* 答え Yes!

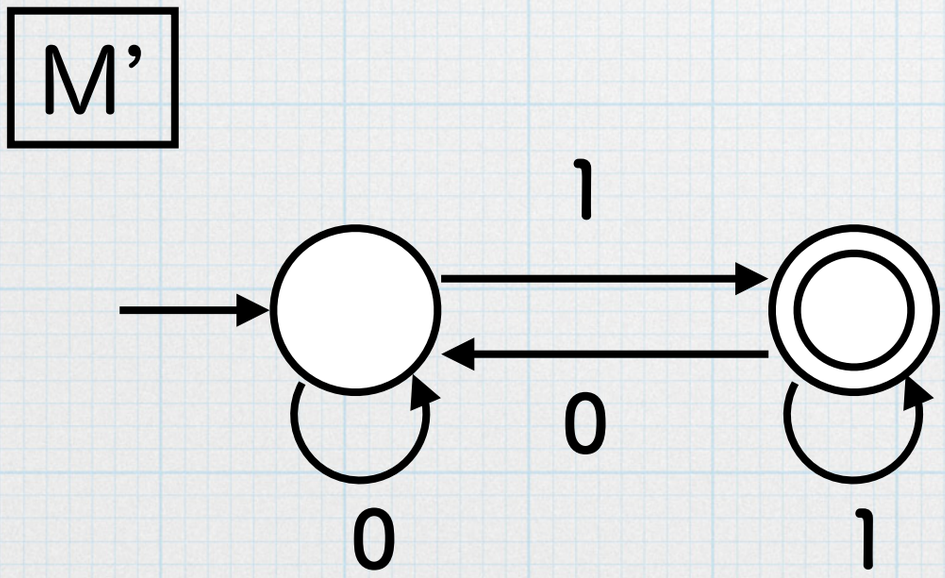
  L(M) = { 11 で終わる文字列全体 }

  L(M’) = { 1 で終わる文字列全体 }

人間の頭を
使った答え

包含問題を自動で解きたい



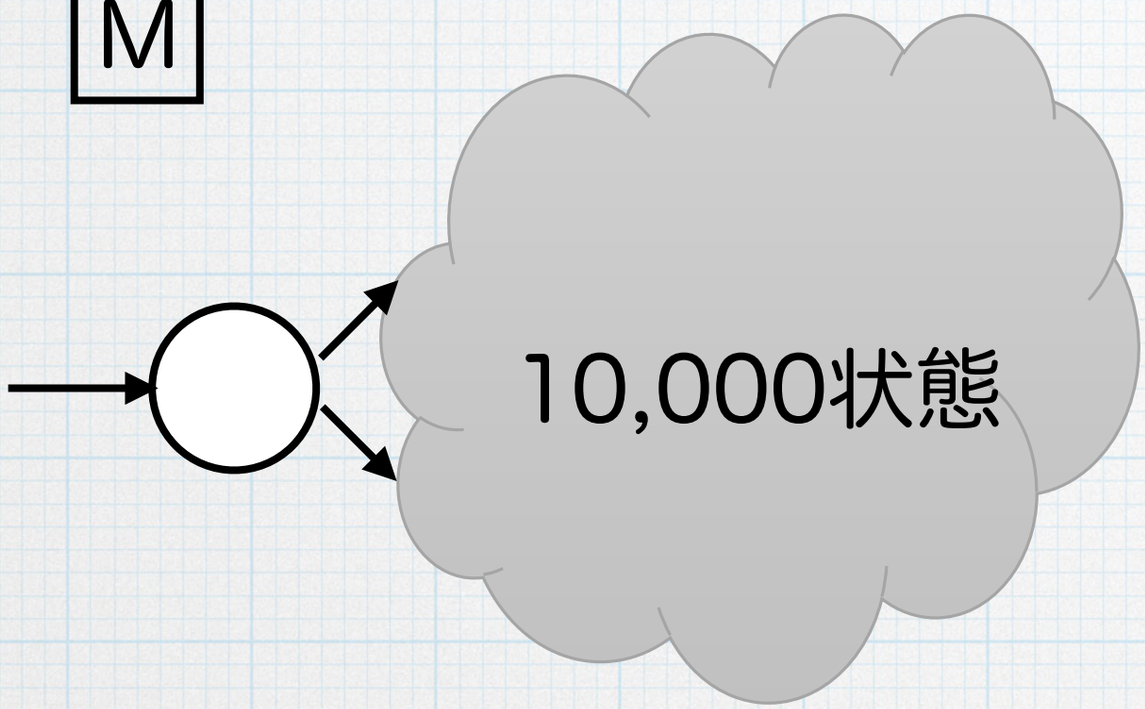


$$L(M) \subseteq L(M')$$

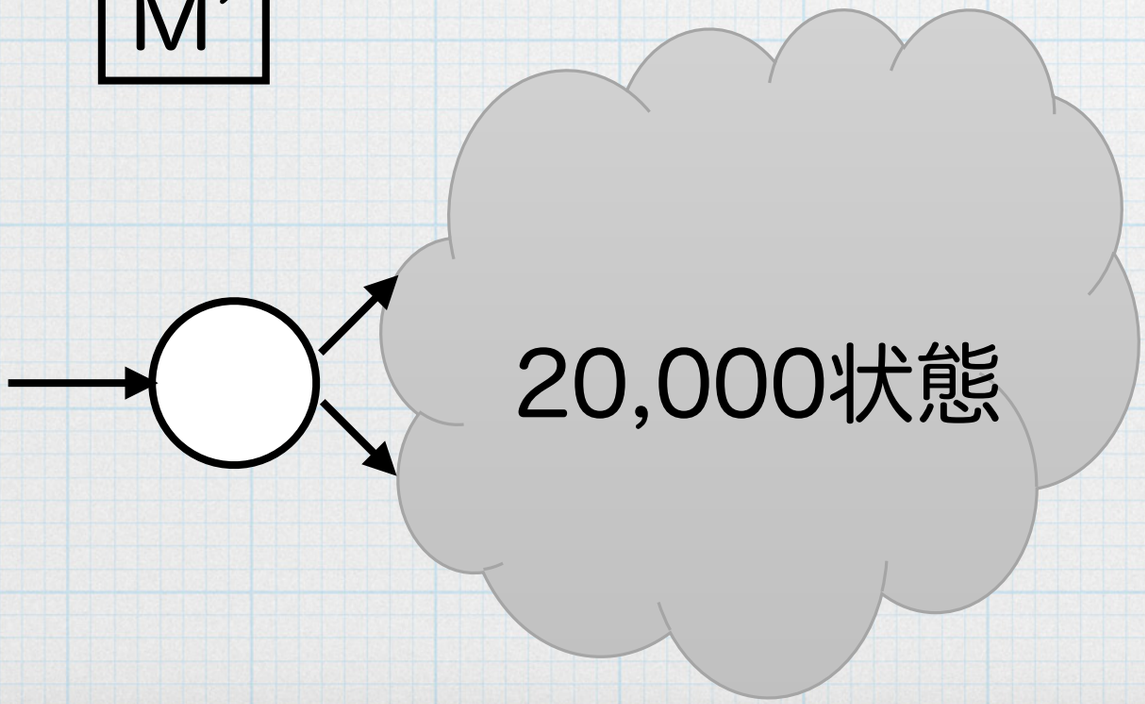
Yes/No

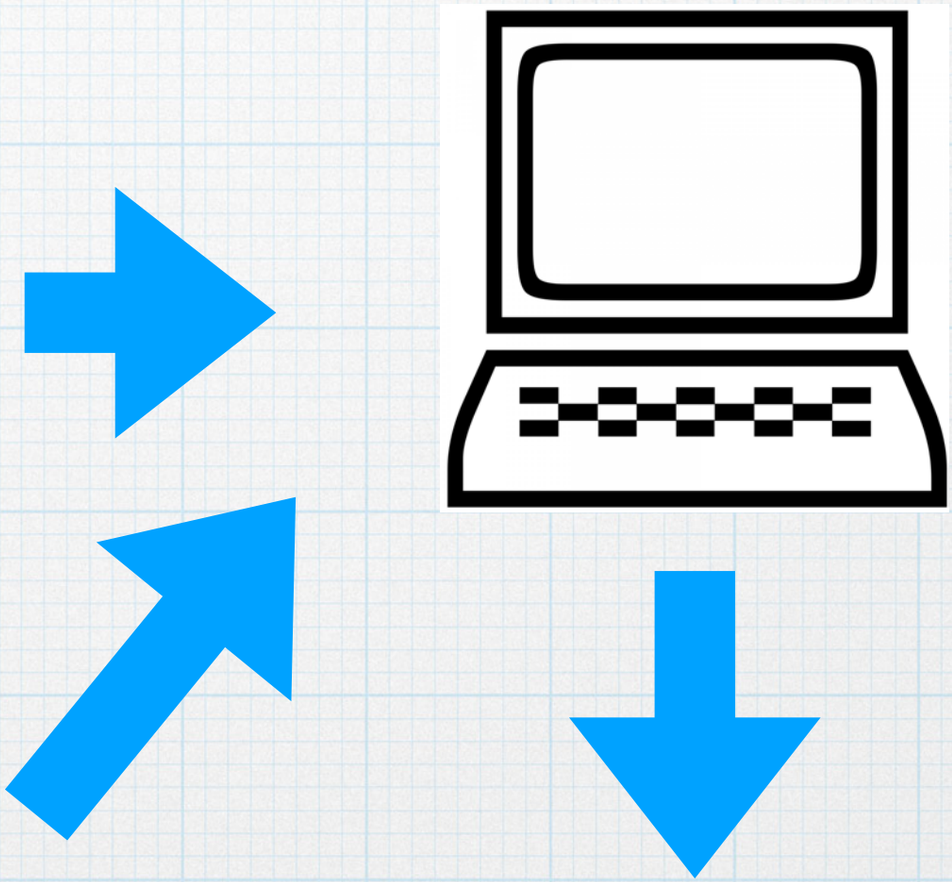
包含問題を自動で解きたい

M



M’





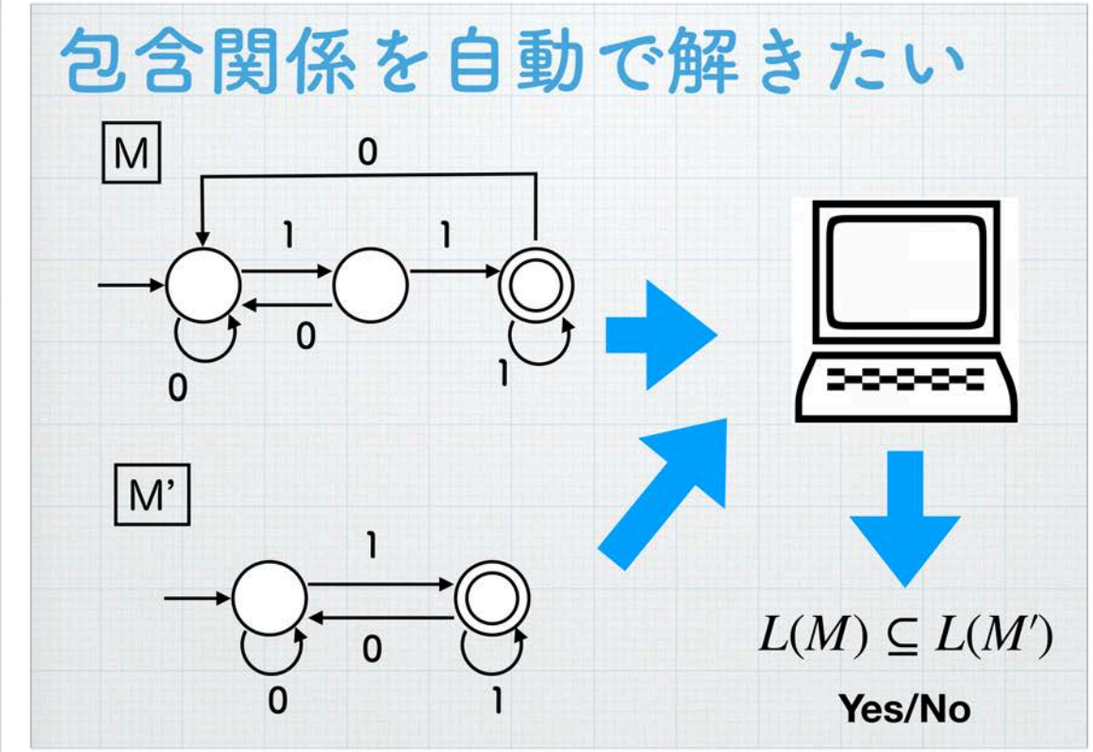
$$L(M) \subseteq L(M')$$

Yes/No

問題

* 入力：
  有限オートマトン M, M′
* 出力： 次が成り立つかどうか

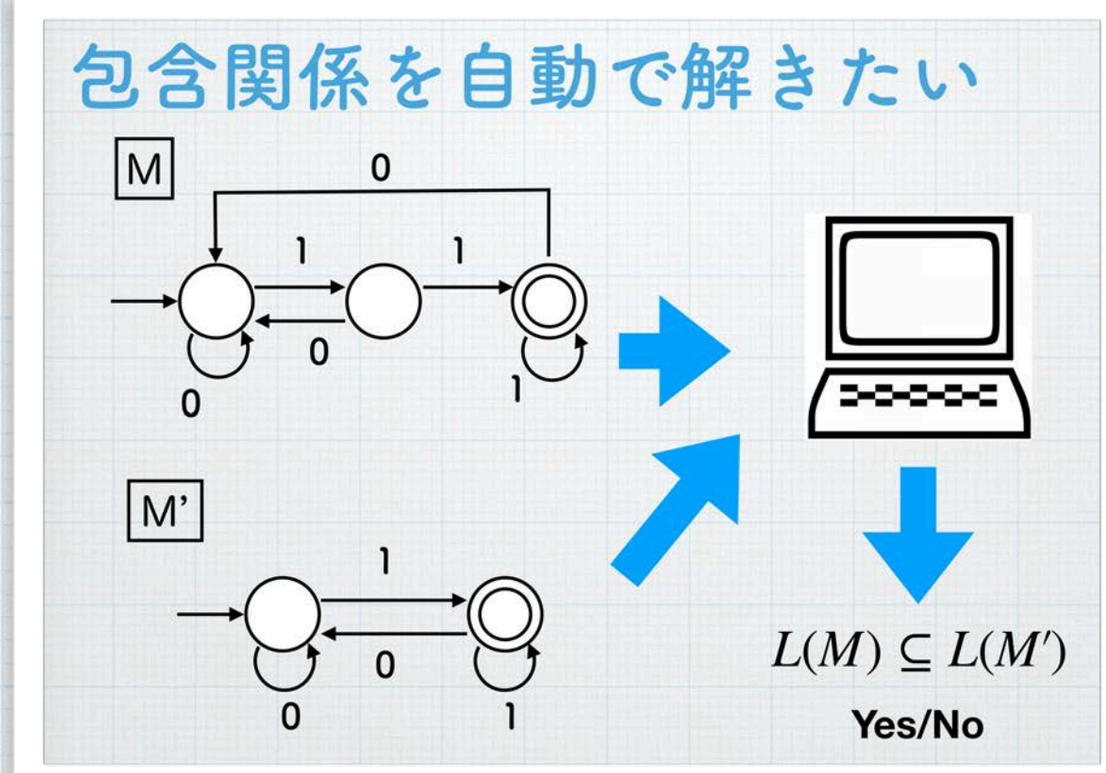
$$L(M) \subseteq L(M')$$



問題

* 入力：
  有限オートマトン M, M′
* 出力： 次が成り立つかどうか

$$L(M) \subseteq L(M')$$



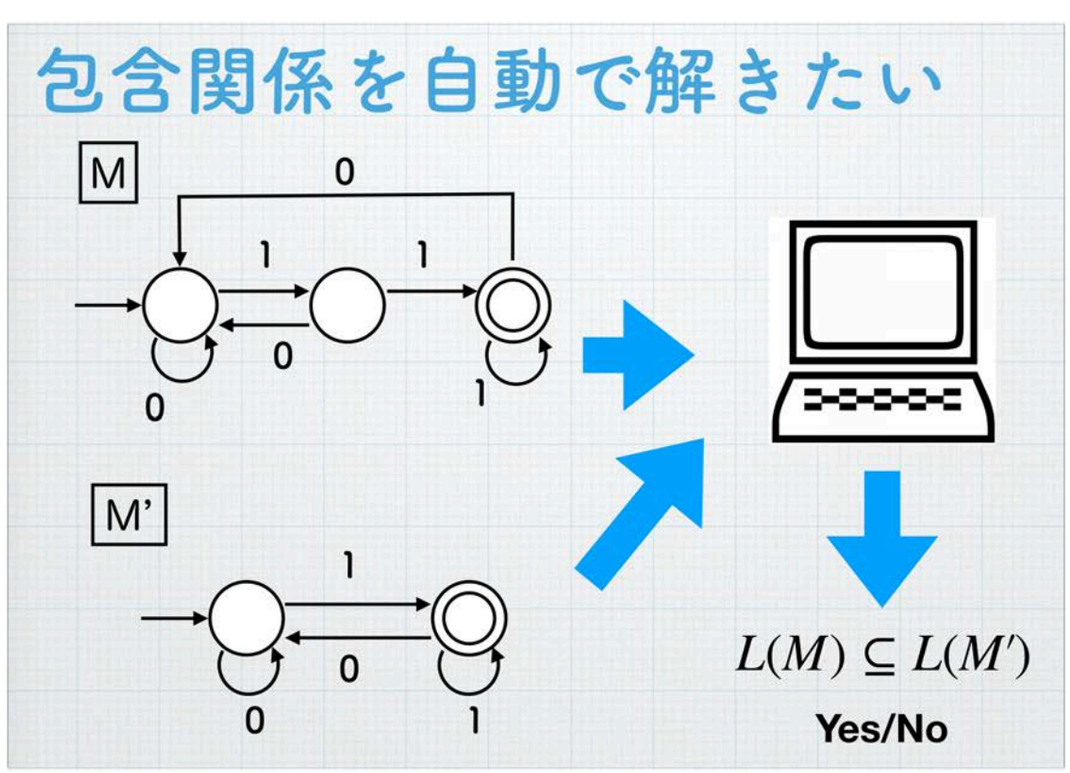
* （間違った答え）

文字列 w それぞれについて，
「w が M に受理されるならば，w は M′ にも受理される」
ことを確かめる

問題

* 入力：
  有限オートマトン M, M′
* 出力： 次が成り立つかどうか

$$L(M) \subseteq L(M')$$



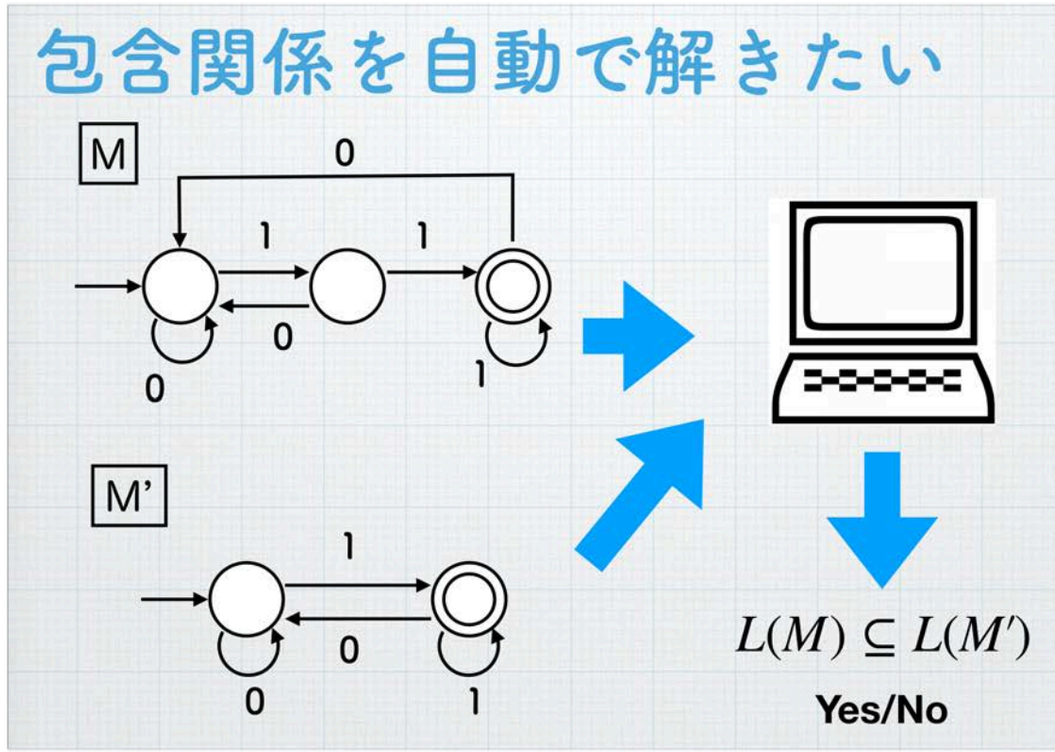
* （間違った答え）
  文字列 w それぞれについて，
  「w が M に受理されるならば，w は M′ にも受理される」
  ことを確かめる

w は無限個ある．一方，人生は有限
（cf.「数え上げおねえさん」で検索）

問題

* 入力：
  有限オートマトン M, M′
* 出力： 次が成り立つかどうか

$$L(M) \subseteq L(M')$$



* （間違った答え）
  文字列 w それぞれについて，
  「w が M に受理されるならば，w は M′ にも受理される」
  ことを確かめる

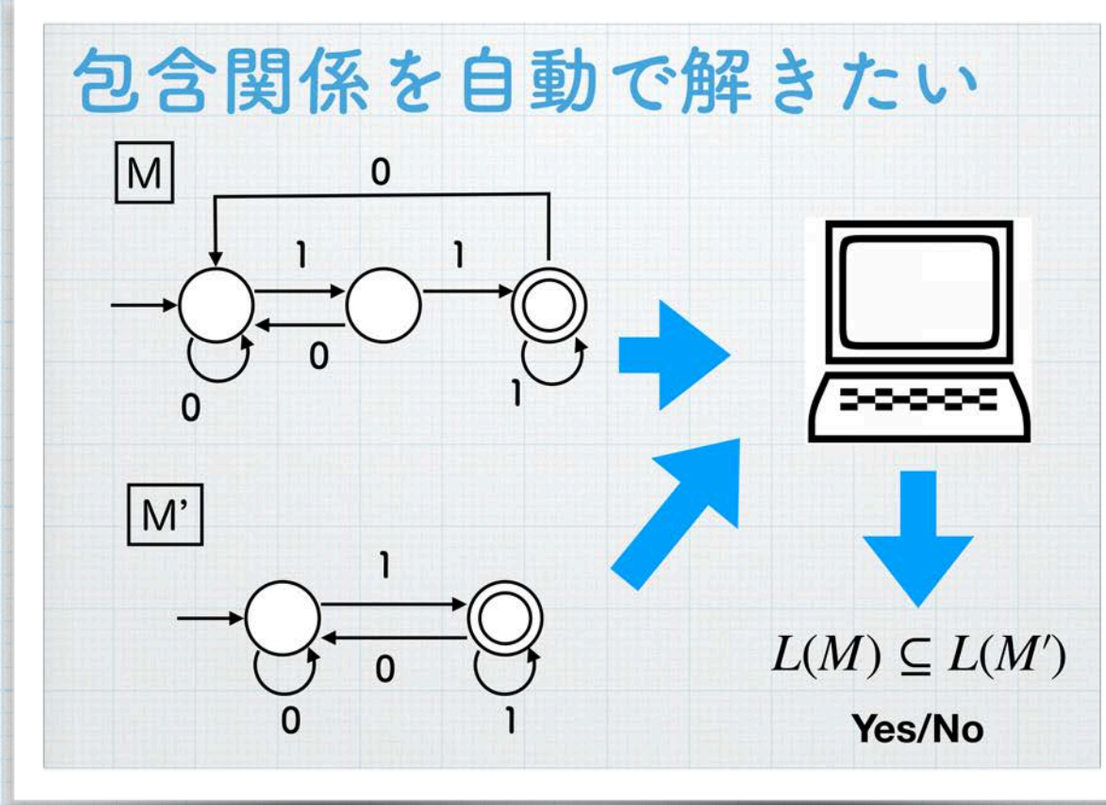
w は無限個ある．一方，人生は有限
（cf.「数え上げおねえさん」で検索）

* L(M), L(M′) は無限集合 ➜ 直接操作できない
* アイデア： 有限の表現たる M, M′ を使ってがんばる

問題

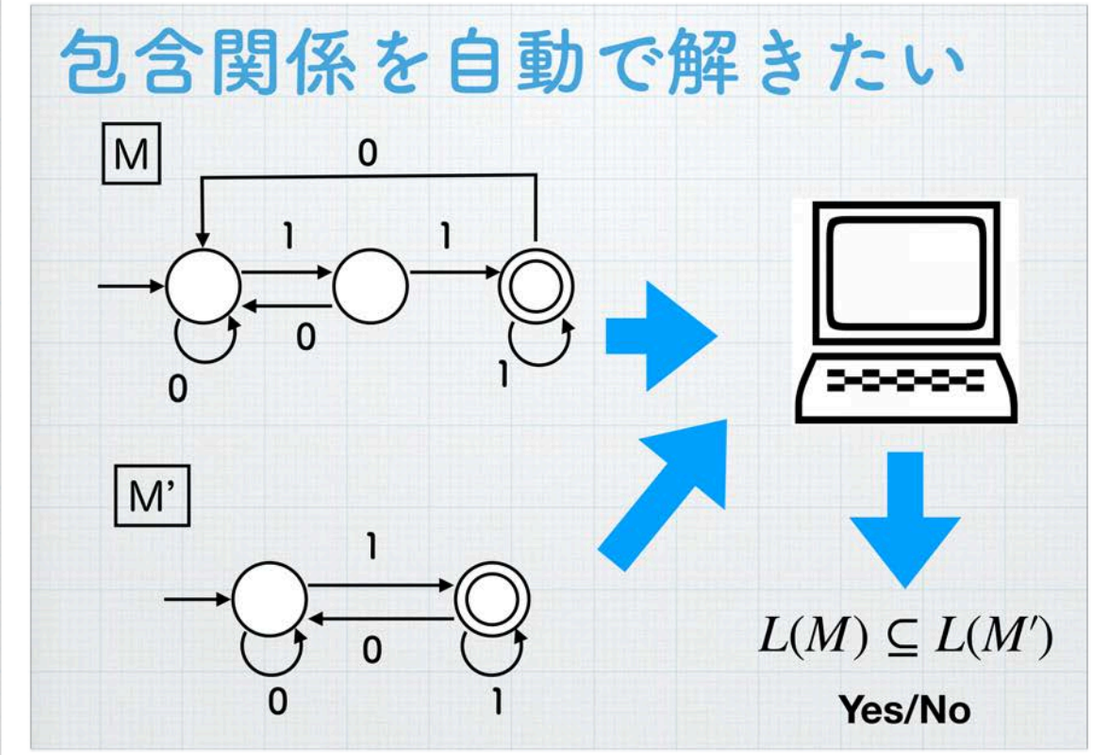
* 入力：
  有限オートマトン M, M′
* 出力： 次が成り立つかどうか

$$L(M) \subseteq L(M')$$



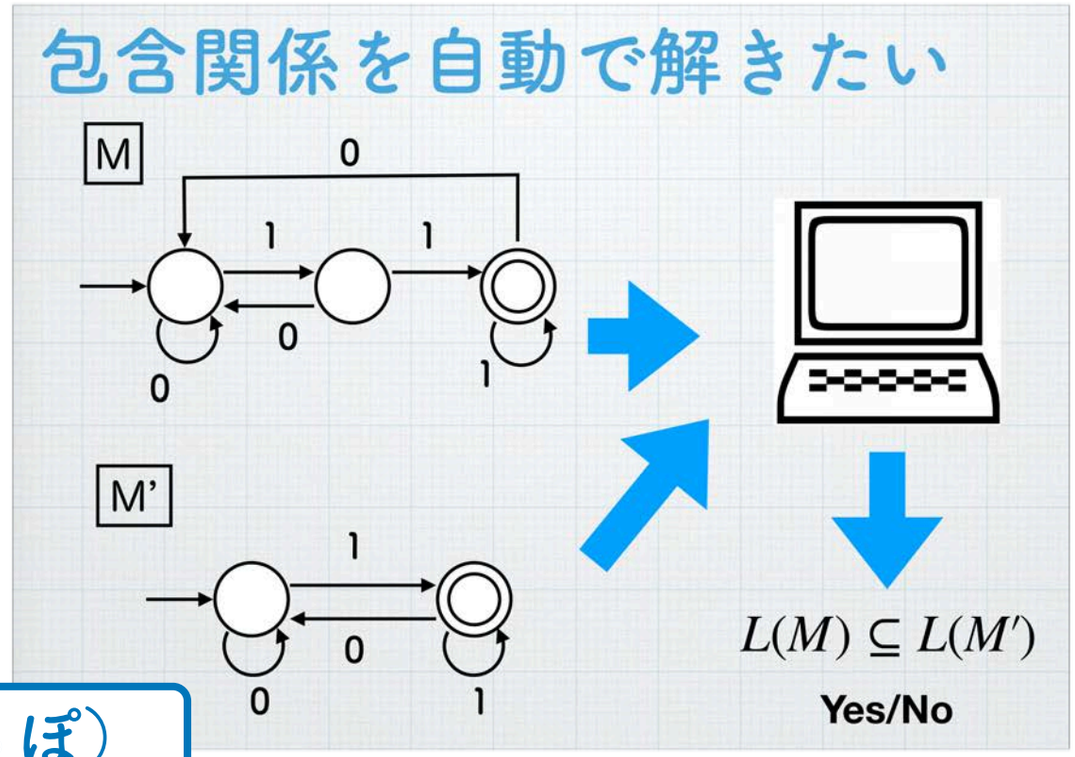
* 以下，アルゴリズムの詳細を説明します.
  * 材料1, 2, 3 の組み合わせ
    * 材料1： 空判定
    * 材料2： 同期積
    * 材料3： 言語反転
  * 数式多数 ➜ わかりにくいかも
  * 寝ちゃった方は起こしますので，ご安心を :)

材料1：空判定



* 入力：
  有限オートマトン M
* 出力：次が成り立つかどうか $L(M) = \emptyset$

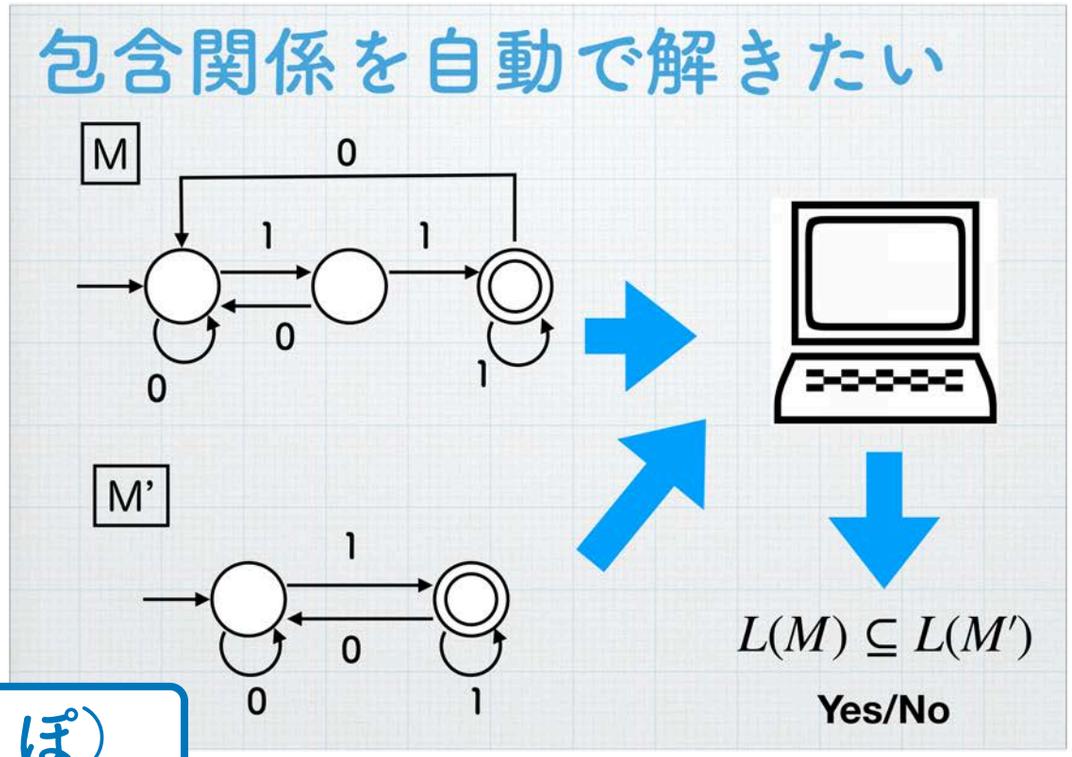
材料1：空判定



* 入力：
  有限オートマトン M
* 出力：次が成り立つかどうか $L(M) = \emptyset$

空集合（からっぽ）

材料1：空判定

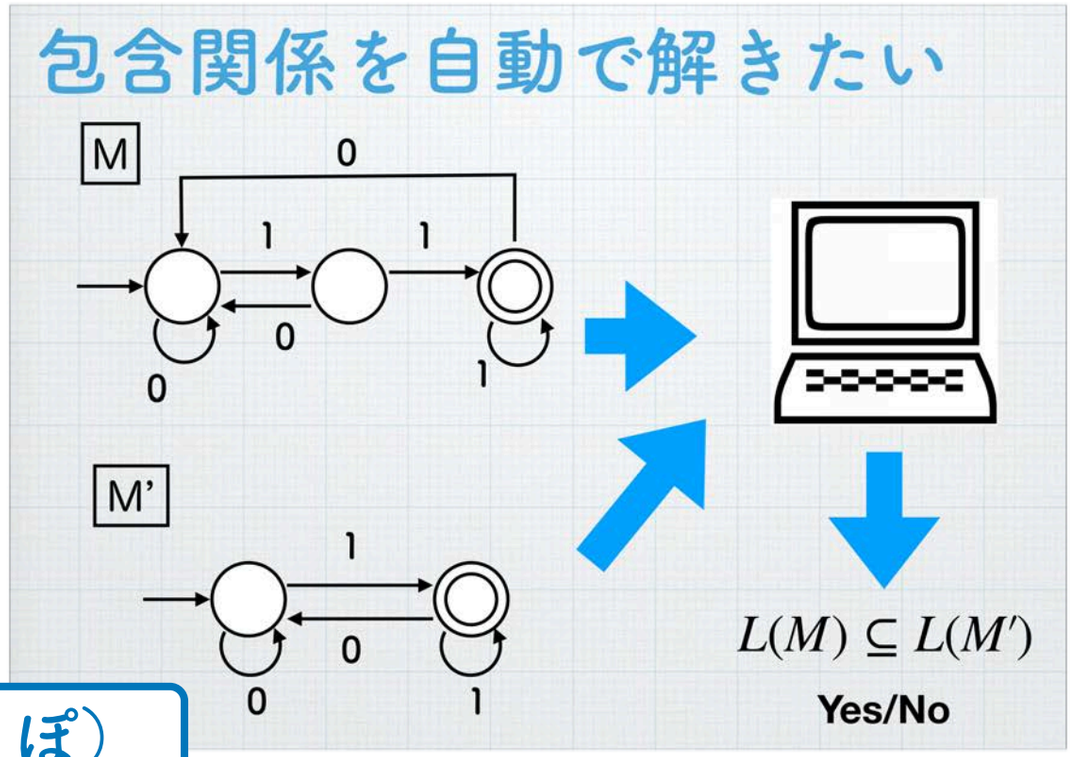


* 入力：
  有限オートマトン M
* 出力：次が成り立つかどうか $L(M) = \emptyset$

空集合（からっぽ）

オートマトン M は
どの文字列も受理しない

材料1：空判定



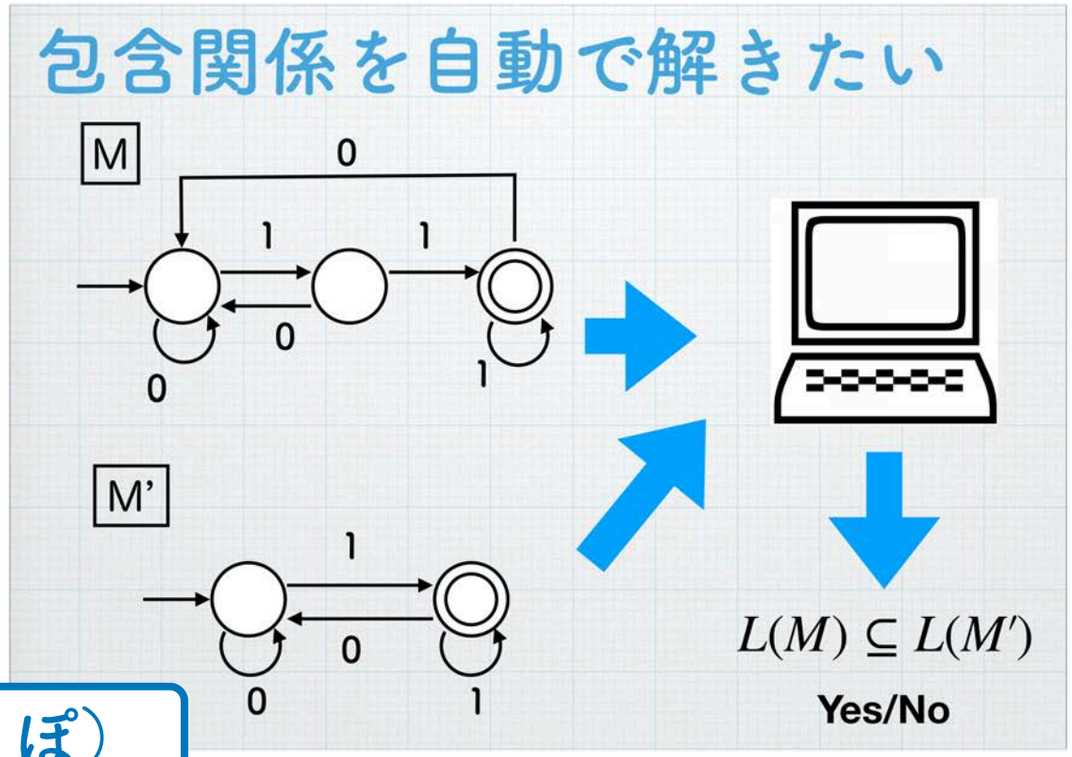
* 入力：
  有限オートマトン M
* 出力：次が成り立つかどうか $L(M) = \emptyset$

空集合（からっぽ）

オートマトン M は
どの文字列も受理しない

「初期状態（→○）から
受理状態（◎）に辿り着けない」と同じ

材料1：空判定



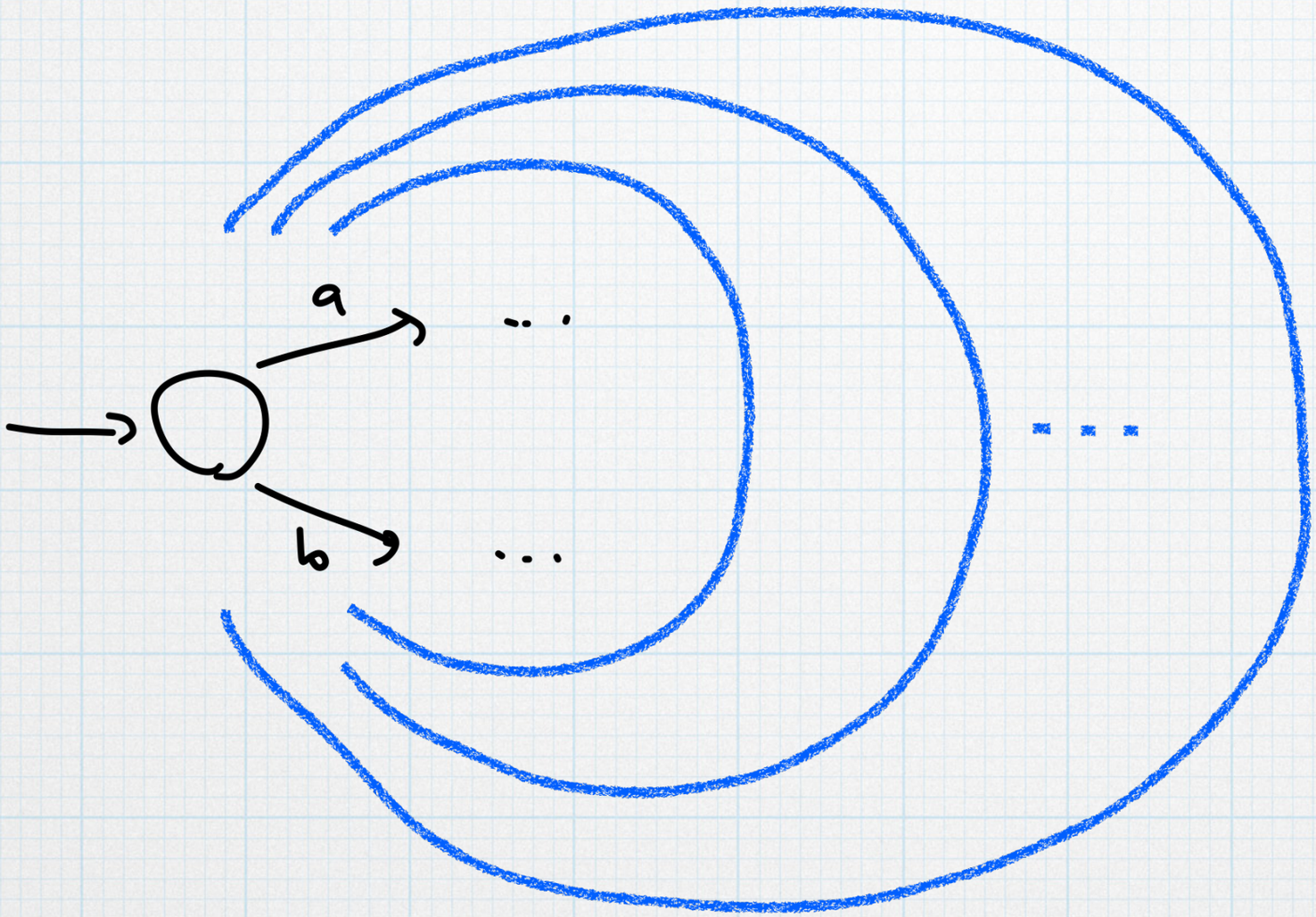
* 入力：
  有限オートマトン M
* 出力：次が成り立つかどうか $L(M) = \emptyset$

空集合（からっぽ）

オートマトン M は
どの文字列も受理しない

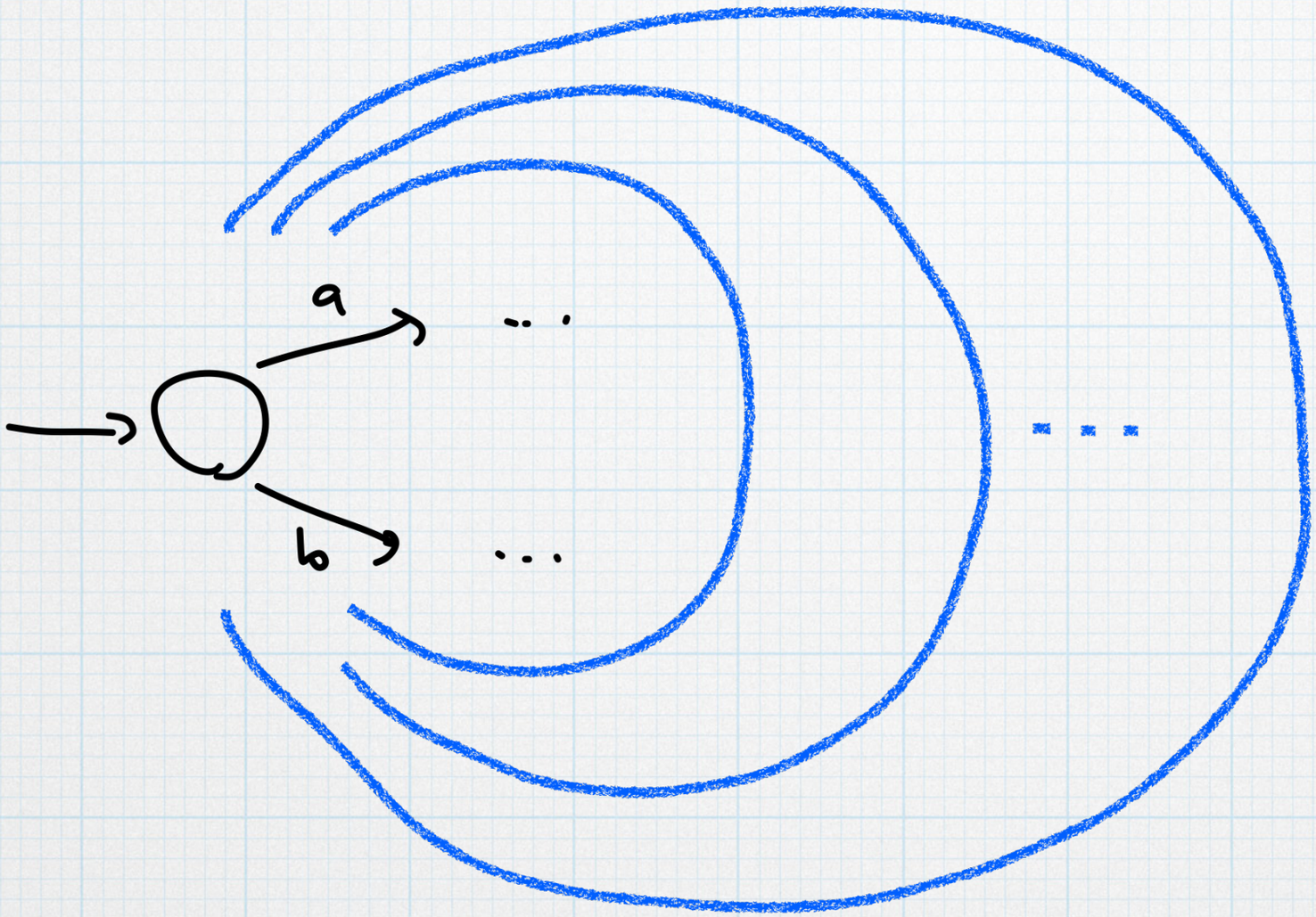
「初期状態（→○）から
受理状態（◎）に辿り着けない」と同じ

「初期状態（→○）から辿り着ける状態」を列挙して
（有限だからできる），◎が含まれないかどうかチェック


a
b
...
...

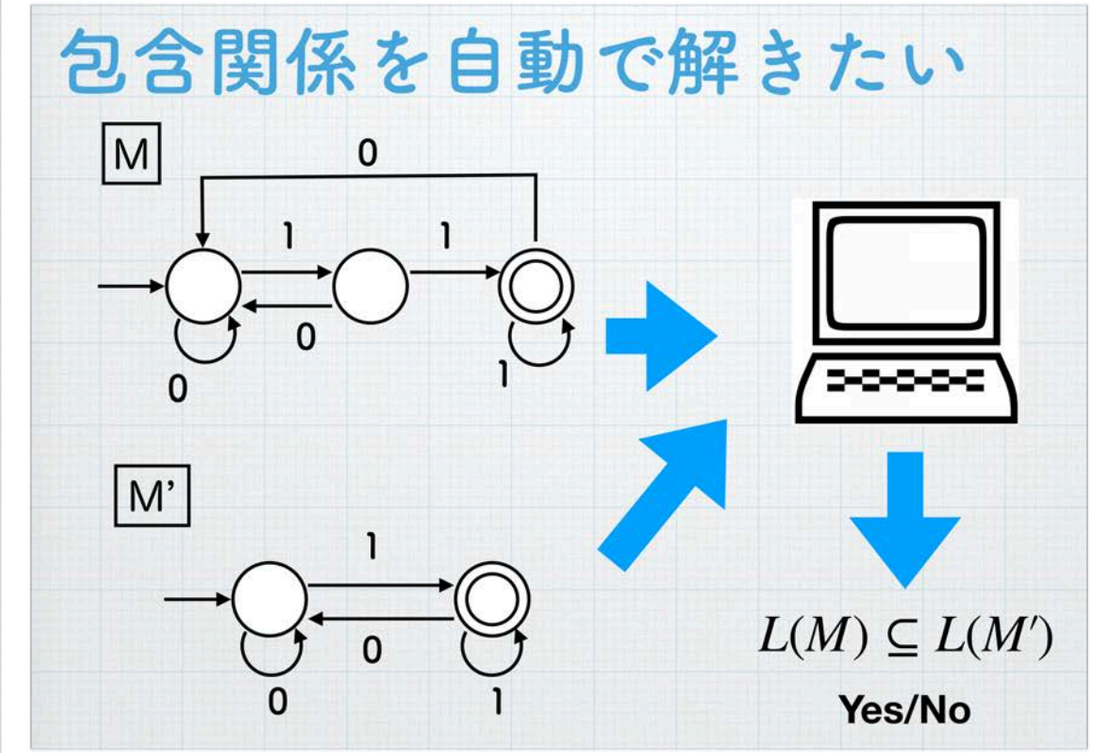
@

@



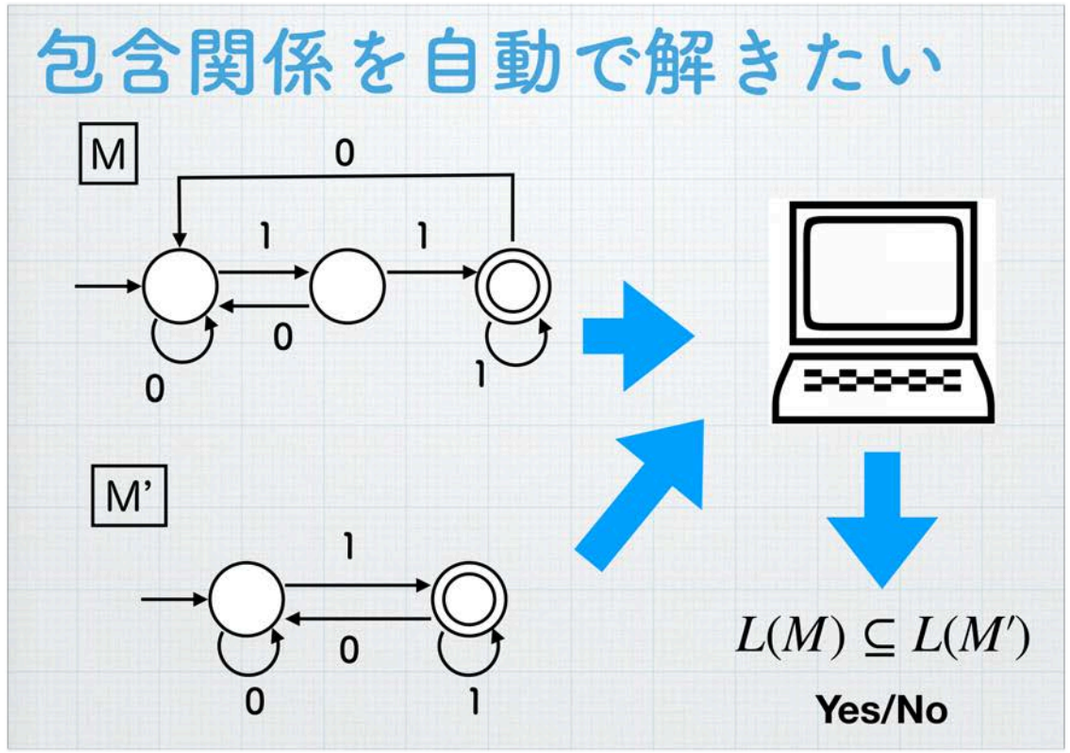
「初期状態（→○）から辿り着ける状態」を列挙して
（有限だからできる），◎が含まれないかどうかチェック

材料1：
空判定（詳細）



* 入力：
  有限オートマトン M
* 出力：次が成り立つかどうか $L(M) = \emptyset$

材料1：
空判定（詳細）

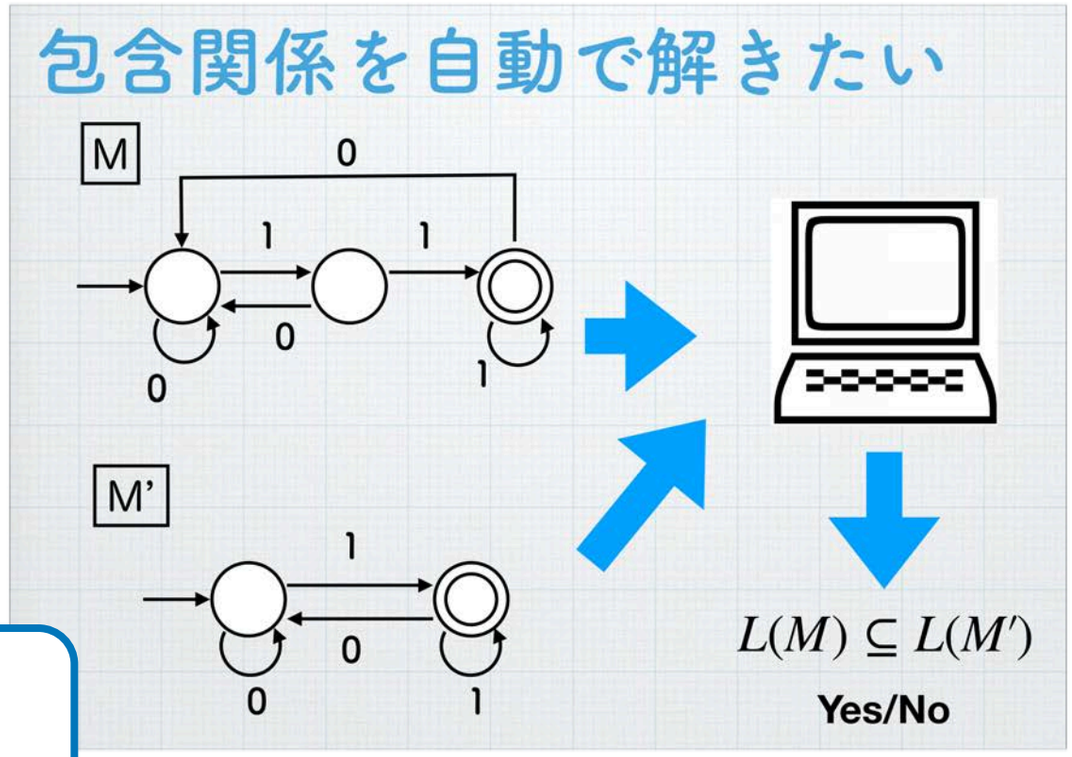


* 入力：
  有限オートマトン M
* 出力：次が成り立つかどうか $L(M) = \emptyset$

* グラフ探索アルゴリズムで解ける
  * S_0 = { 初期状態 } とする
  * n = 0, 1, 2, … に対して,
    $S_{n+1} = S_n \cup \{ S_n$ から一歩で到達できる状態全体 $\}$ とする
  * 増えなくなったら（$S_{n+1} = S_n$ になったら）終わり
    そのとき，S_n = { 初期状態から到達可能な状態全体 }
  * $S_n \cap$ { 受理状態 } = $\emptyset$ かどうかを調べればよい

S_n = { 初期状態から n 歩以内で到達できる状態全体 }

材料1：
空判定（詳細）



必ず終わる．なぜか？

… 終わらないとすると，状態集合

$S_0, S_1, S_2, \cdots$ はどんどん，際限なく大きくなる．

しかし状態は有限個しかないから，これは不可能

➜ 有限オートマトンでよかった！！

初期状態から n 歩以内で到達できる状態全体 }

* n = 0, 1, 2, … に対して，

  $S_{n+1} = S_n \cup \{ S_n$ から一歩で到達できる状態全体 $\}$　とする

* 増えなくなったら（$S_{n+1} = S_n$ になったら）終わり

  そのとき，$S_n = \{$ 初期状態から到達可能な状態全体 $\}$

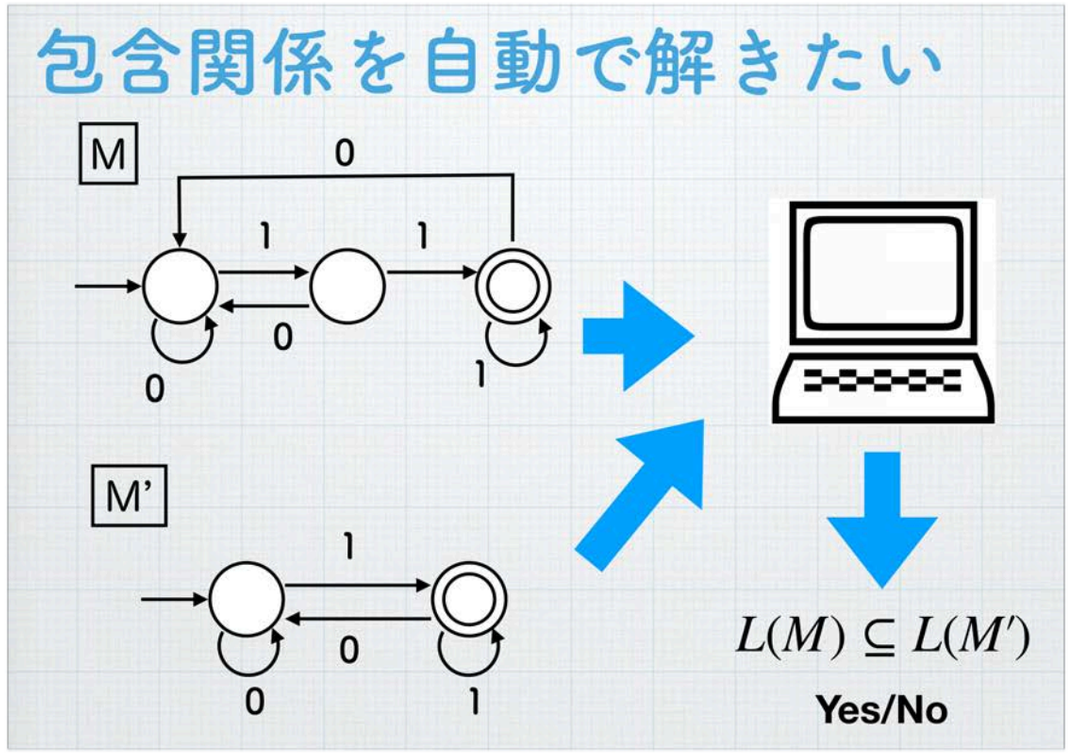
* $S_n \cap \{$ 受理状態 $\} = \varnothing$　かどうかを調べればよい

材料2：
オートマトンの同期積

* 入力： 有限オートマトン M, M′
* 出力：

$$L(M \otimes M') = L(M) \cap L(M')$$

となるオートマトン $M \otimes M'$

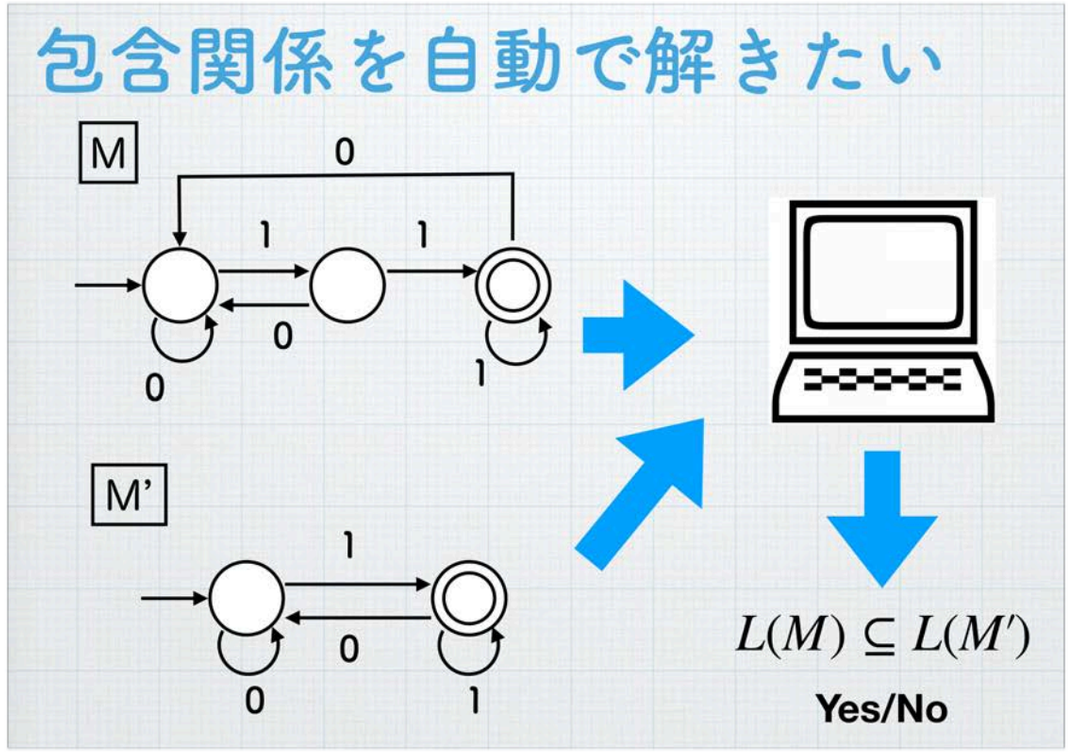


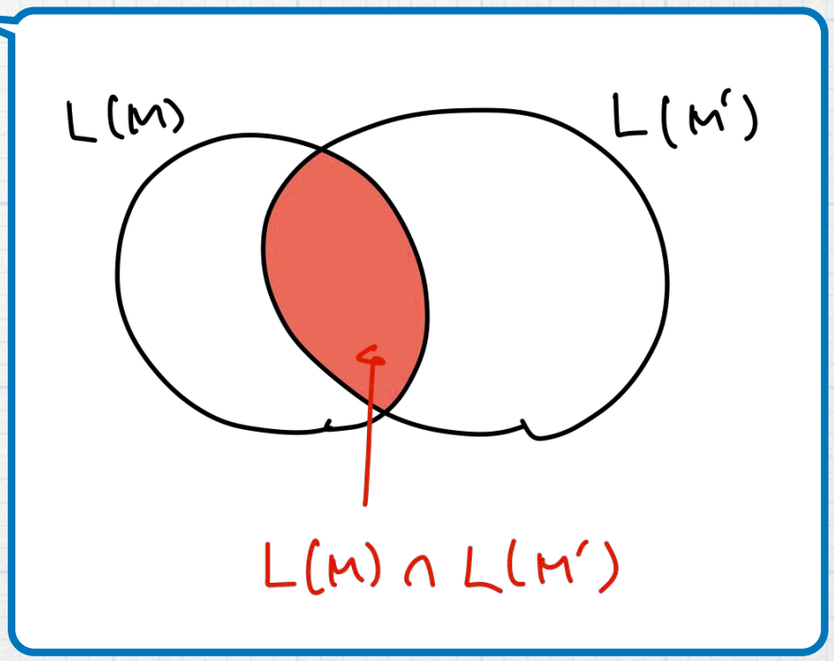
材料2：
オートマトンの同期積

* 入力： 有限オートマトン M, M′
* 出力：

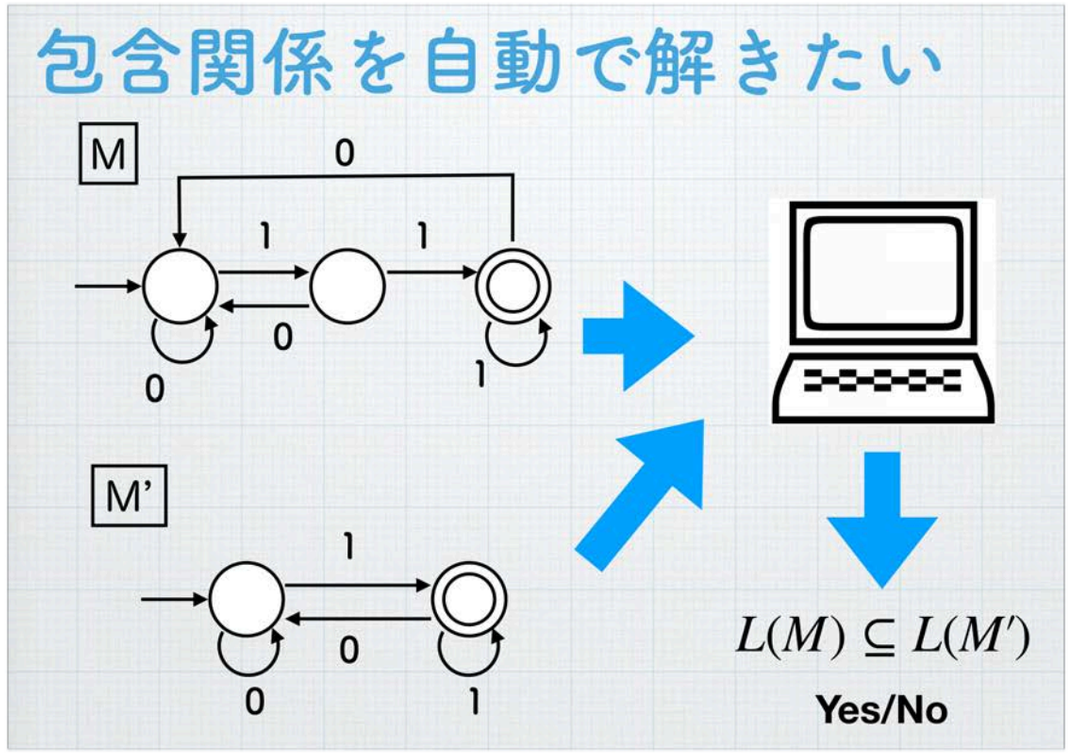
$$L(M \otimes M') = L(M) \cap L(M')$$

となるオートマトン $M \otimes M'$





材料2：
オートマトンの同期積

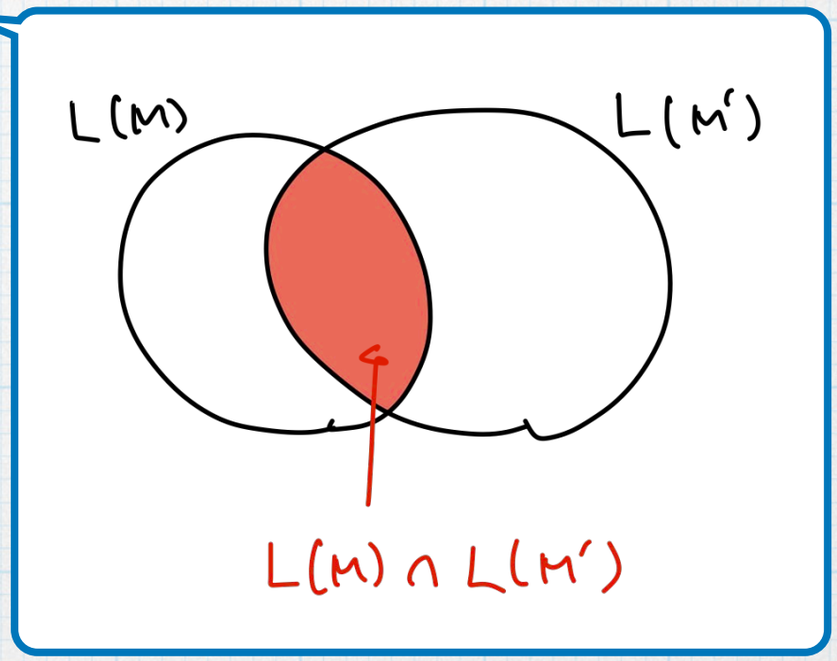


* 入力： 有限オートマトン M, M′
* 出力：

$$L(M \otimes M') = L(M) \cap L(M')$$

となるオートマトン $M \otimes M'$

$M \otimes M'$ は，MとM′の両方に受理される文字列（そしてそれらのみ）を受理

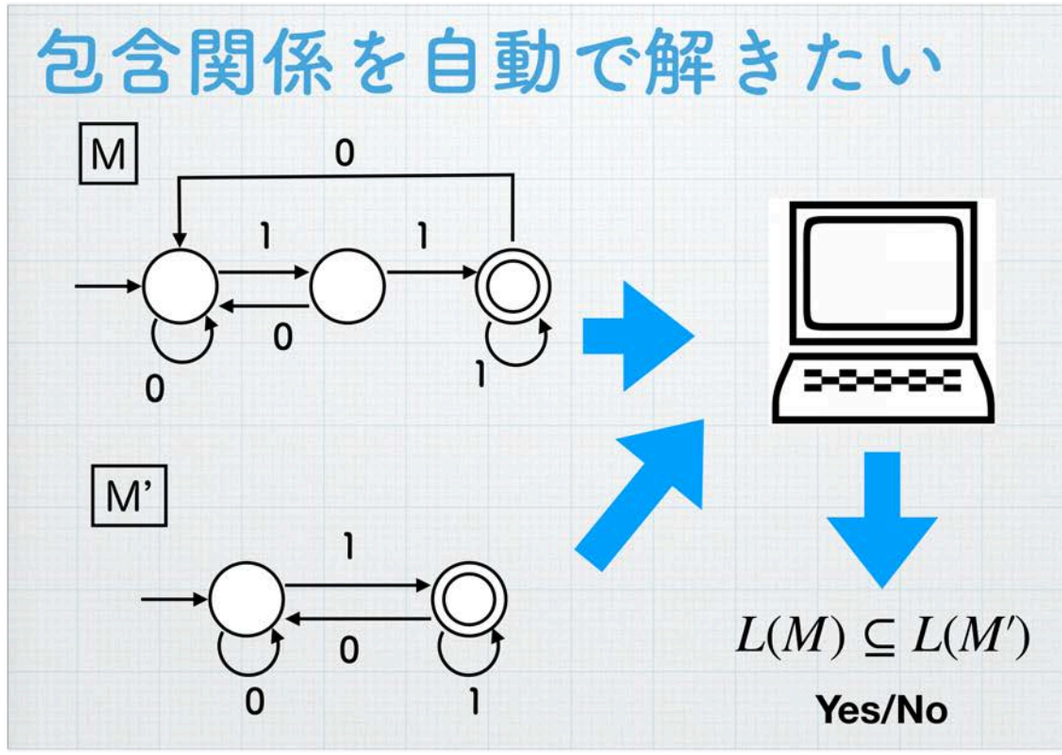


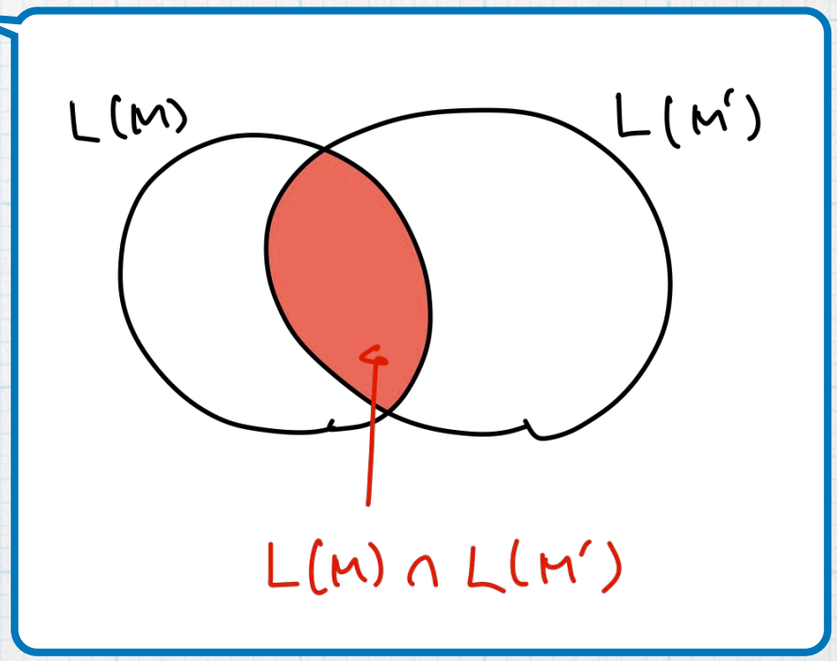
材料2：
オートマトンの同期積

* 入力： 有限オートマトン M, M′
* 出力：

$$L(M \otimes M') = L(M) \cap L(M')$$

となるオートマトン $M \otimes M'$





$M \otimes M'$ は，Mと M′ の両方に受理される文字列（そしてそれらのみ）を受理

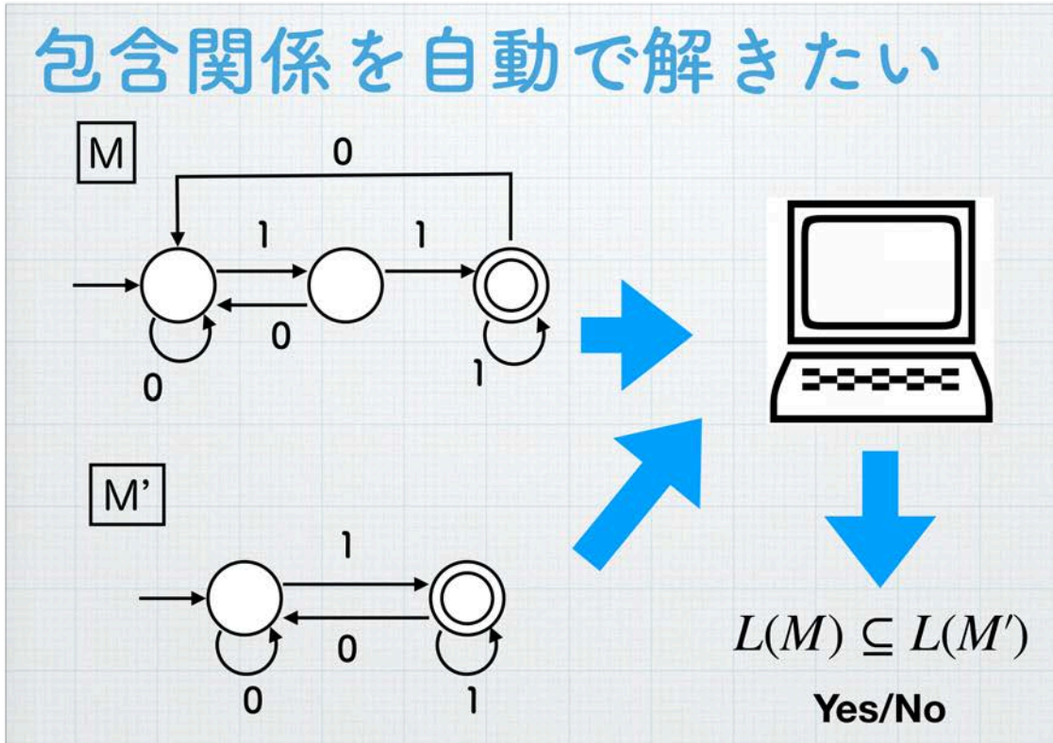
文字列集合の交わり intersection
$L(M) \cap L(M')$ を表現

材料2：
オートマトンの同期積

* 入力： 有限オートマトン M, M′
* 出力：

$$L(M \otimes M') = L(M) \cap L(M')$$

となるオートマトン $M \otimes M'$



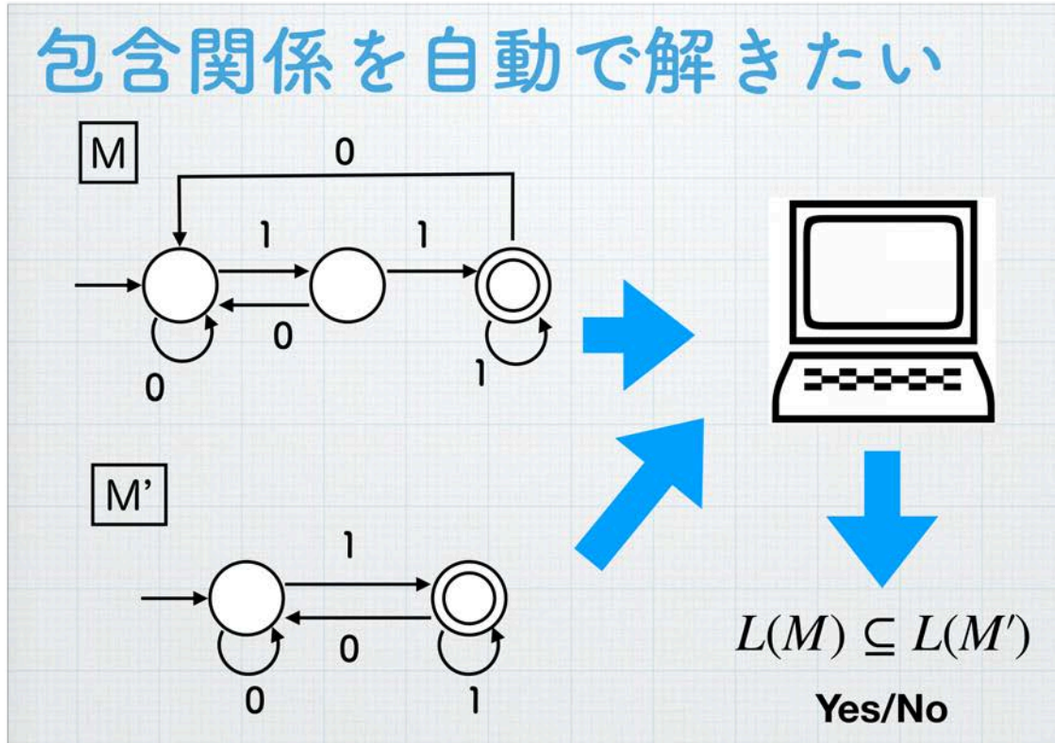
文字列集合の交わり intersection
$L(M) \cap L(M')$ を表現

材料2：
オートマトンの同期積

* 入力： 有限オートマトン M, M′
* 出力：

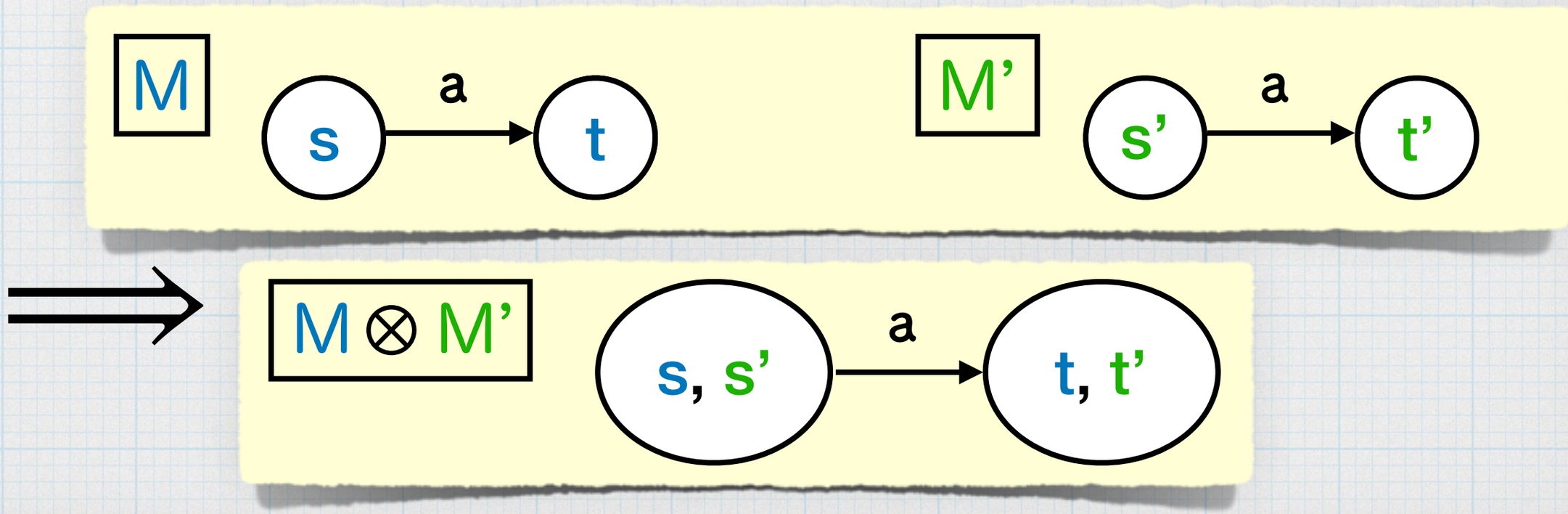
$$L(M \otimes M') = L(M) \cap L(M')$$

となるオートマトン $M \otimes M'$



文字列集合の交わり intersection
$L(M) \cap L(M')$ を表現

* 作り方： 「2つのオートマトンを両方同時に動かす」

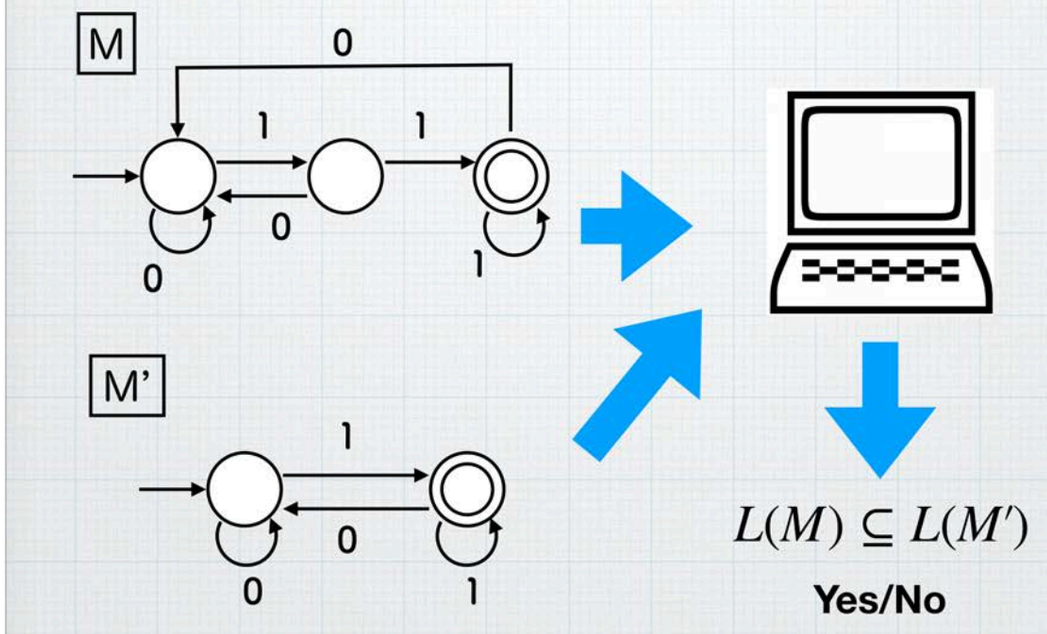


材料2：
オートマトンの同期積

* 入力： 有限オートマトン M, M′
* 出力：

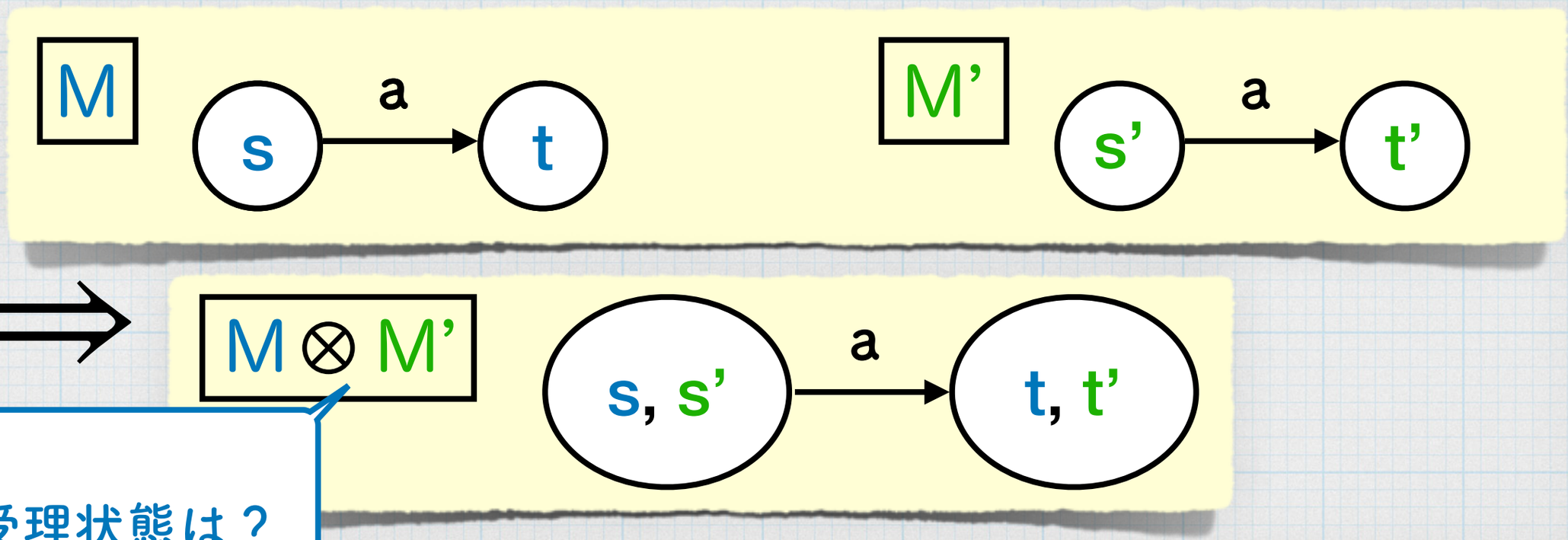
$$L(M \otimes M') = L(M) \cap L(M')$$

となるオートマトン $M \otimes M'$



文字列集合の交わり intersection
$L(M) \cap L(M')$ を表現

* 作り方： 「2つのオートマトンを両方同時に動かす」



クイズ：
初期状態は？ 受理状態は？

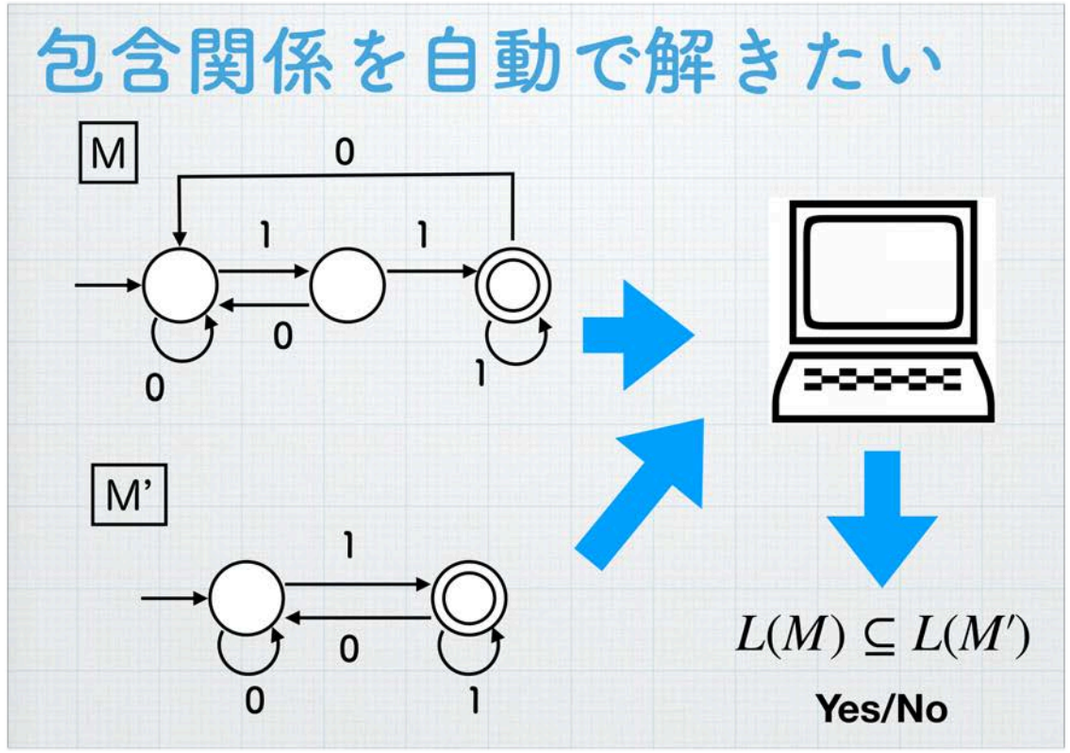
材料3：
オートマトンの言語反転

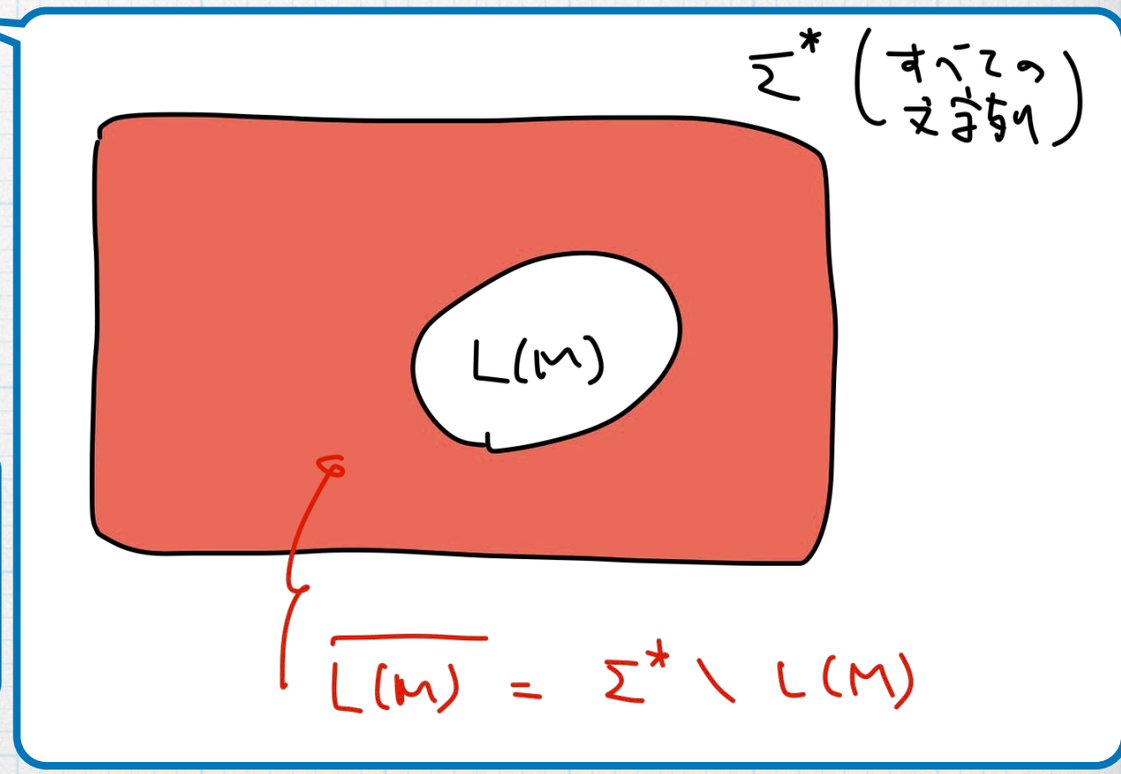
* 入力：有限オートマトン M
* 出力：

$$L(M^c) = \overline{L(M)} = \Sigma^* \backslash L(M)$$

となるオートマトン M^c

Mc と M とでは，受理と非受理がひっくり返っている





材料3：
オートマトンの言語反転

* 入力：有限オートマトン M
* 出力：

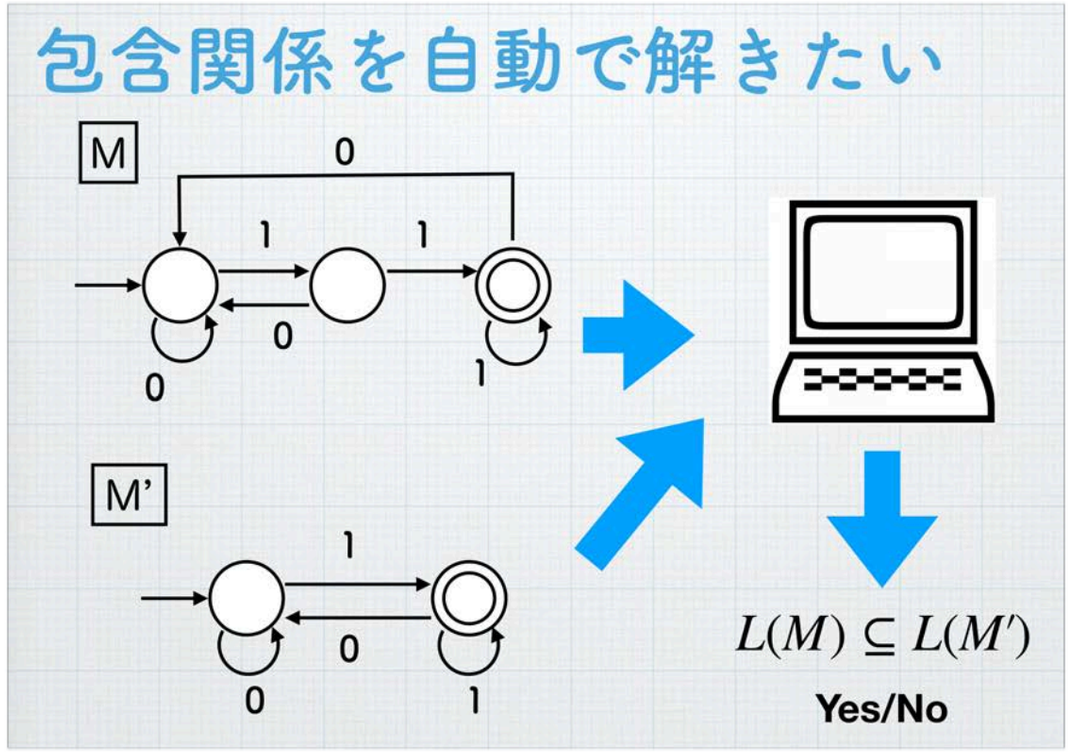
$$L(M^c) = \overline{L(M)} = \Sigma^* \backslash L(M)$$

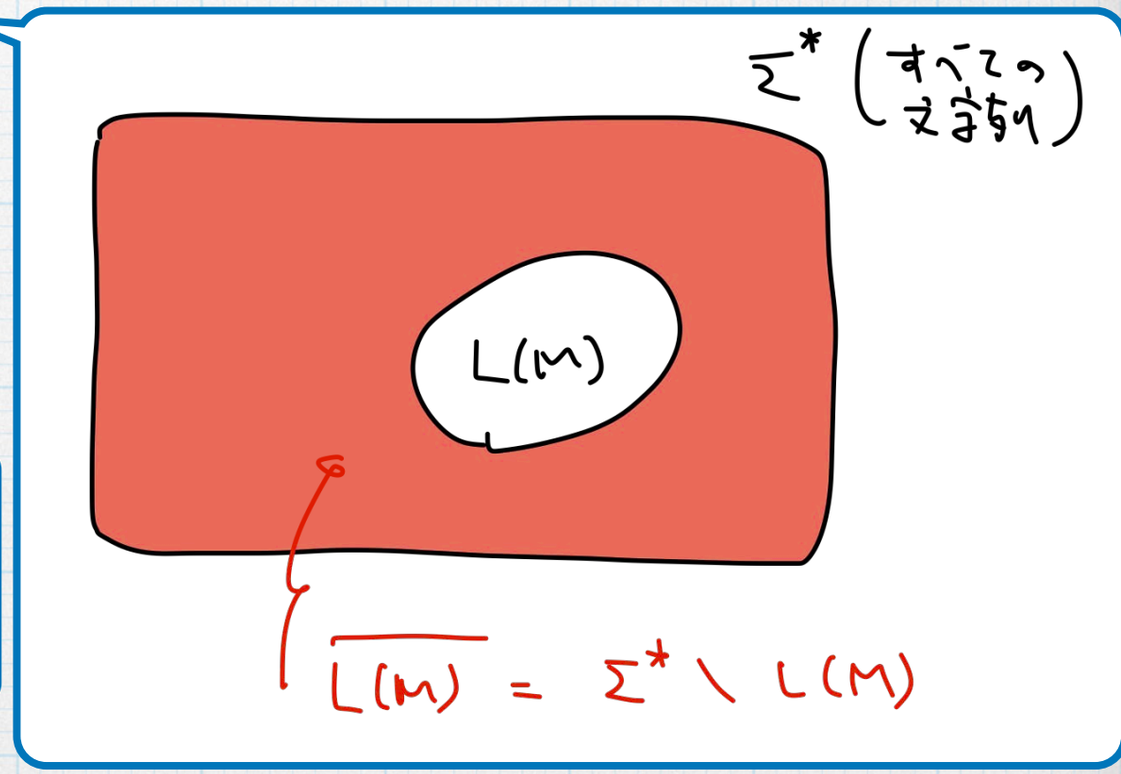
となるオートマトン M^c

* 作り方：

○と◎を反転

M^c と M とでは，受理と非受理がひっくり返っている





材料3：
オートマトンの言語反転

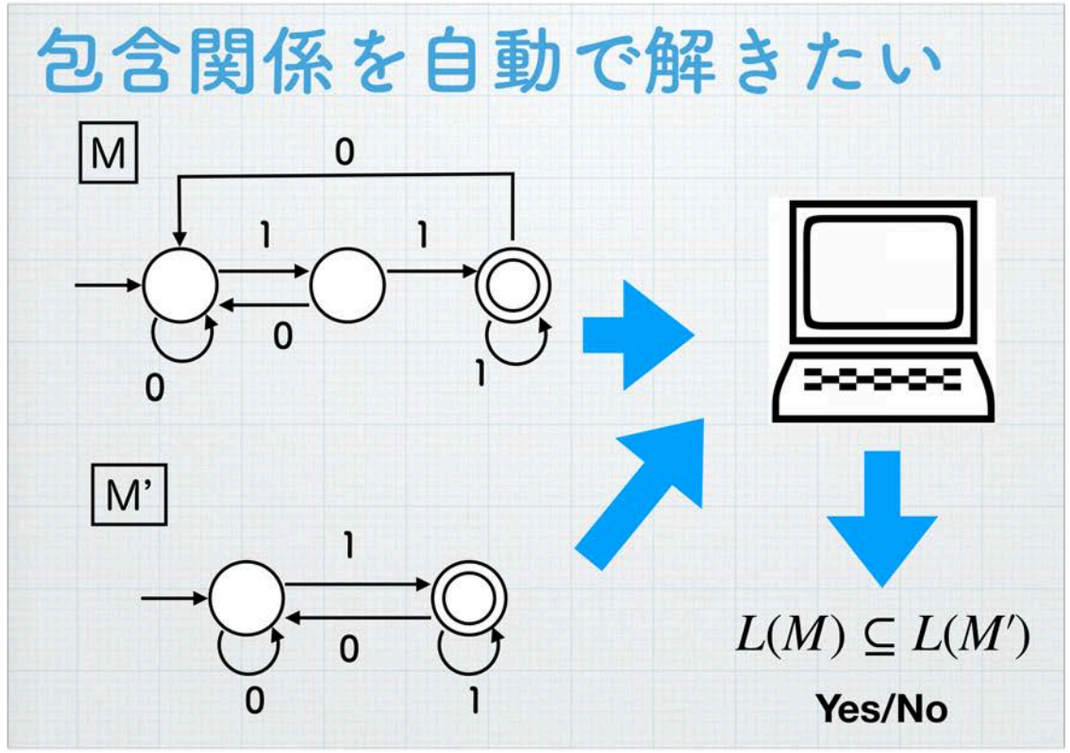
* 入力：有限オートマトン M
* 出力：

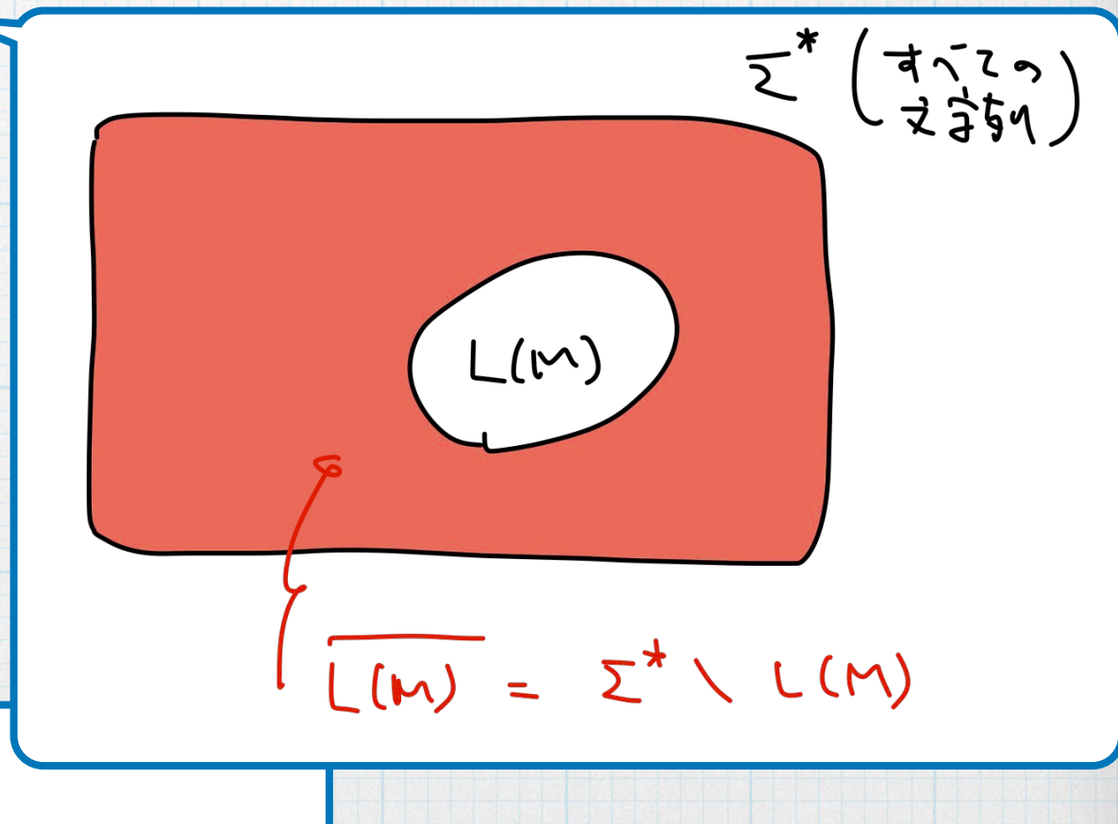
$$L(M^c) = \overline{L(M)} = \Sigma^* \backslash L(M)$$

となるオートマトン M^c

* 作り方：

○と◎を反転





M^c と M とでは，受理と非受理がひっくり返っている

Warning!（詳細，念の為）
決定性オートマトンだからこれでOK
非決定性オートマトンではダメ（一度決定化が必要）

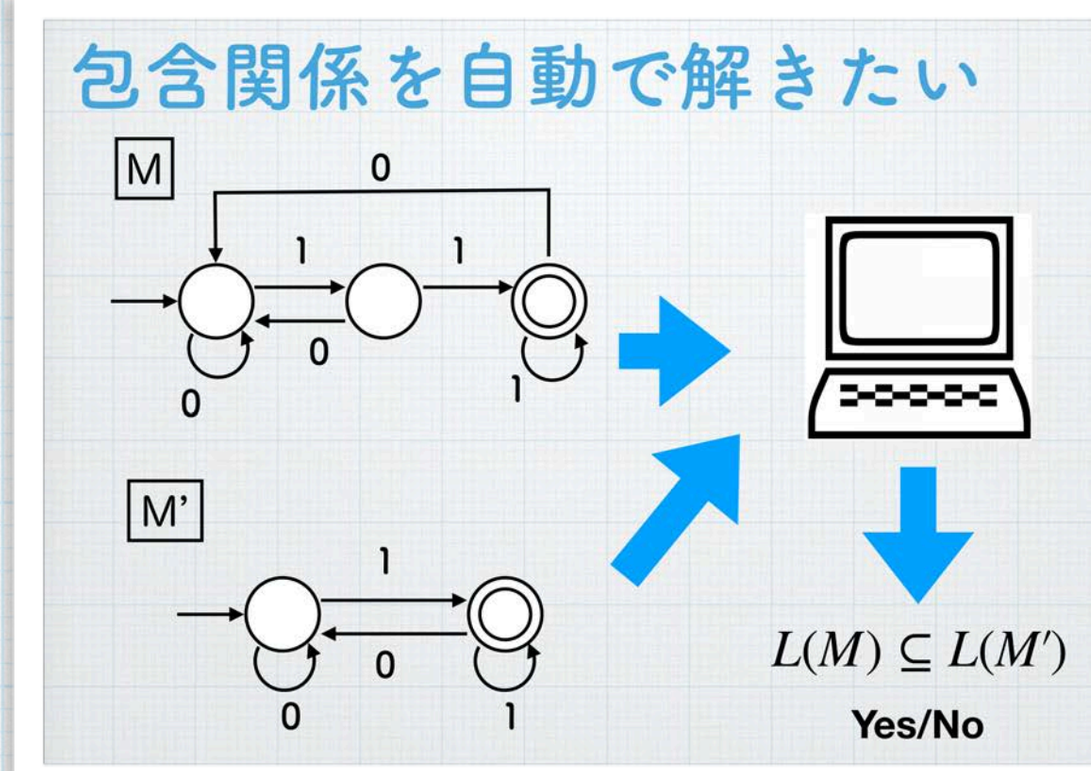
解答：

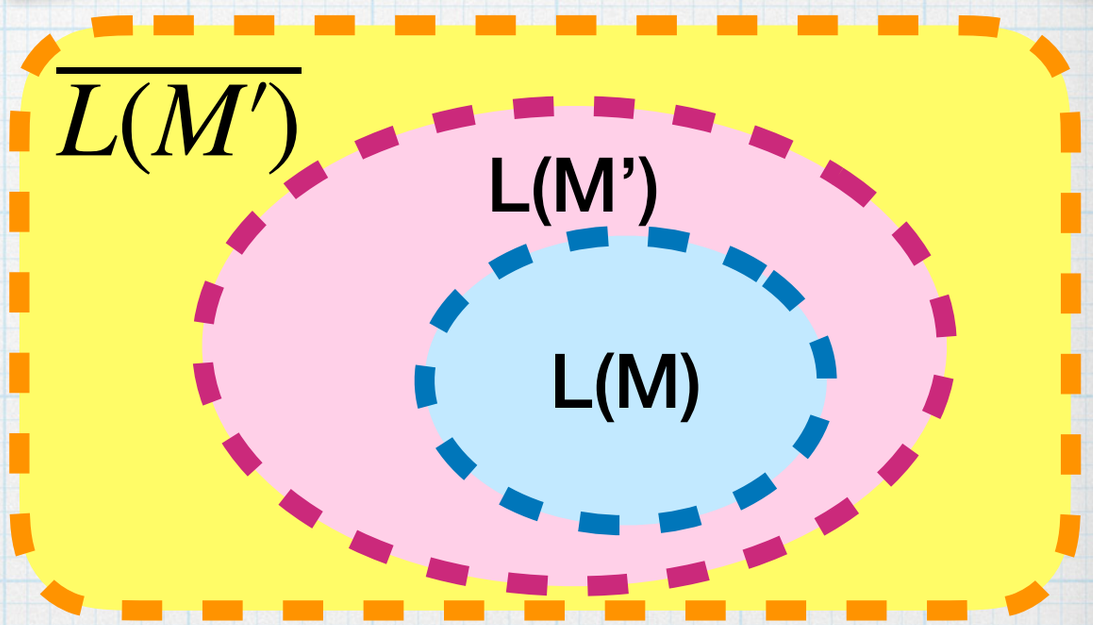
包含関係を解く

$$L(M) \subseteq L(M')$$

$$\iff L(M) \cap \overline{L(M')} = \emptyset$$

$$\iff L(M \otimes (M')^c) = \emptyset$$





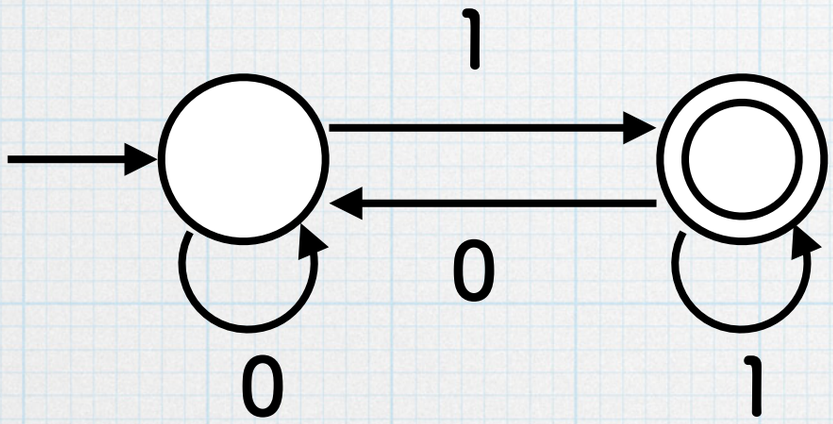
* アルゴリズム：
  * $(M')^c$ を作る （○↔◎，材料３）
  * $M \otimes (M')^c$ を作る （オートマトンの同期積，材料２）
  * $M \otimes (M')^c$ の空判定
    （◎が到達可能かどうか探索，材料１）

理論計算機科学入門 — 有限と無限のあいだ

アウトライン

* 理論計算機科学とは — 有限と無限のせめぎあい
* いろいろな無限 — 対角線論法
* オートマトン理論入門
  * 包含問題を解くアルゴリズム — 有限の方法で無限の対象を操作
  * オートマトンの表現能力 — 有限性の限界
* 形式検証 — オートマトンを用いたシステム品質保証
* AI 時代の形式検証 — 研究紹介

有限状態オートマトン



* 機能：
  * 有限の文字列を読んで，
  * 「受理 OK!」あるいは「非受理 NG!」と言う

* すなわち，オートマトン M は，文字列の集合 L(M) を表現する（ひきおこす）．
  ここでは，
  L(M) =
  {w1： w は任意の文字列}

有限状態オートマトン

* オートマトン M は，文字列の集合 L(M) を表現する．

L(M) = { Mに受理される文字列全体 }

有限状態オートマトン

* オートマトン M は，文字列の集合 L(M) を表現する．

  L(M) = { Mに受理される文字列全体 }

* 逆に，次は成り立つか？
  * かってな文字列集合 L について，
  * これを表現するオートマトン M がある
    （L = L(M)）

有限状態オートマトン

* オートマトン M は，文字列の集合 L(M) を表現する．

  L(M) = { Mに受理される文字列全体 }

* 逆に，次は成り立つか？

  答え： 成立しない！

  * かってな文字列集合 L について，
  * これを表現するオートマトン M がある
    （L = L(M)）

無限の濃度がちがう

* $\mathbb{N}$：自然数全体の集合

  $\equiv$ ：「一対一対応がある」

  文字集合（アルファベット）は有限，とする

無限の濃度がちがう

* ℕ：自然数全体の集合

  ≡ ：「一対一対応がある」

  文字集合（アルファベット）は有限，とする

* { 文字列全体 } ≡ ℕ

* 長さ n の文字列の個数は有限，∀ n

無限の濃度がちがう

* ℕ：自然数全体の集合

  ≡ ：「一対一対応がある」

  文字集合（アルファベット）は有限，とする

* { 文字列全体 } ≡ ℕ

  * 長さ n の文字列の個数は有限，∀ n

* { 文字列集合 L 全体 } ≡ P(ℕ)

無限の濃度がちがう

* ℕ：自然数全体の集合

  ≡ ：「一対一対応がある」

  文字集合（アルファベット）は有限，とする

* { 文字列全体 } ≡ ℕ

  * 長さ n の文字列の個数は有限，∀ n

* { 文字列集合 L 全体 } ≡ P(ℕ)

* { 有限オートマトン全体 } ≡ ℕ

  * 状態数 n の有限オートマトンは（同型をのぞいて）

    有限個しかない，∀ n

無限の濃度がちがう

* ℕ：自然数全体の集合

  ≡ ：「一対一対応がある」

  文字集合（アルファベット）は有限，とする

* { 文字列全体 } ≡ ℕ

  * 長さ n の文字列の個数は有限，∀ n

* { 文字列集合 L 全体 } ≡ P(ℕ)

* { 有限オートマトン全体 } ≡ ℕ

ずれ

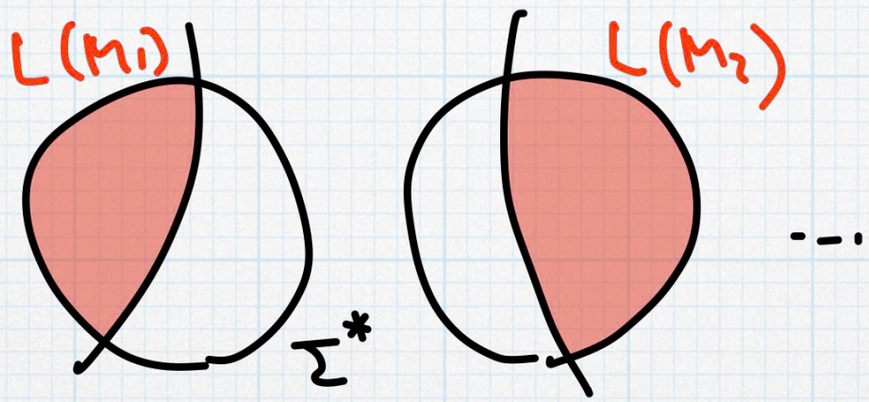
ℕ ≡ P(ℕ) は成り立たない！

  * 状態数 n の有限オートマトンは（同型をのぞいて）

    有限個しかない，∀ n

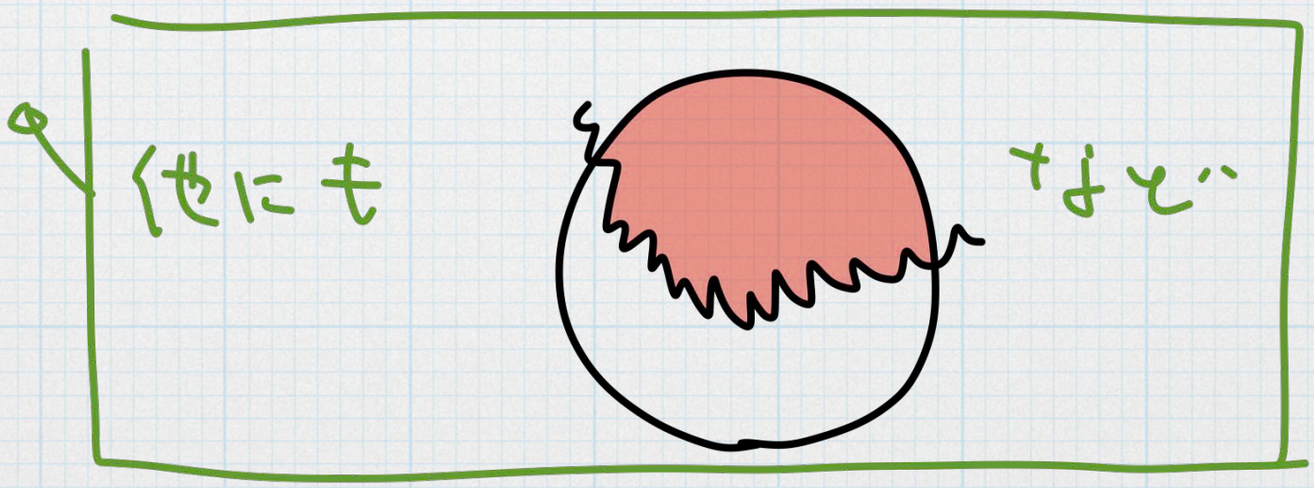
可算個 (≡ N)

オートマトン　M_1　M_2　…



言語

非可算個
(≡ 𝒫(N))



対角線論法（再掲）

* 定理　任意の集合 X に対して，X と P(X) の間に1対1対応はない

* 証明（背理法）
仮に，1対1対応 f: X → P(X) があるとしよう．（矛盾を導く）

（アイデア： ヤバそうな P(X) の元を持ってくる）

U = {x | x は f(x) に含まれない } と定義する．U は P(X) の元．

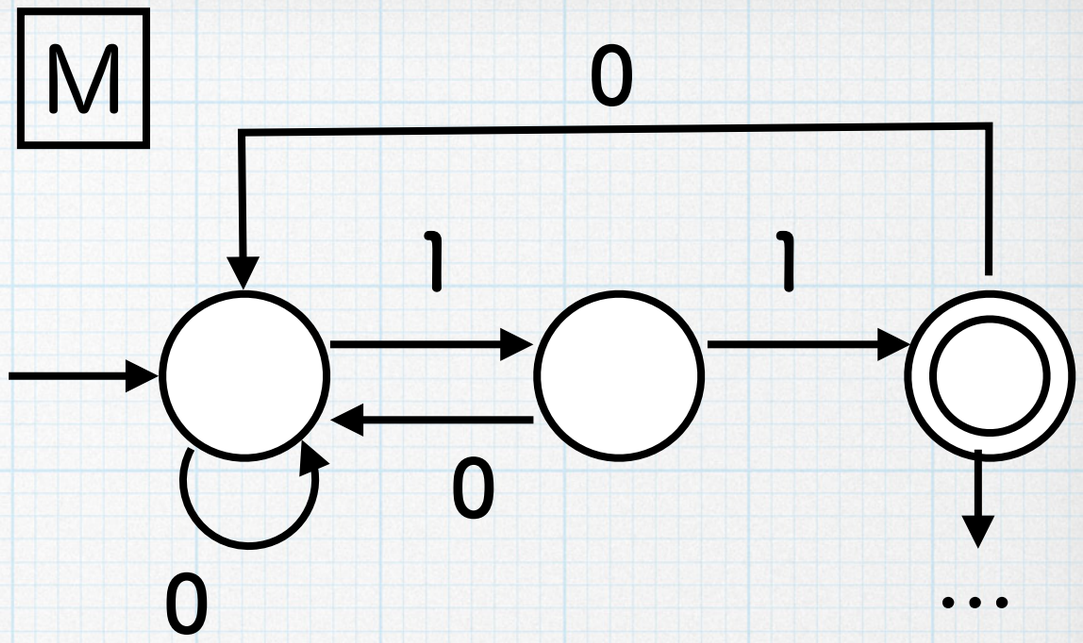
f: X → P(X) は1対1対応なので，X の元 u であって，f(u) = U
となるものが存在する．ここで，u が U に含まれるかを考える．

* u が U に含まれる場合： f(u) = U より，u は f(u) に含まれる．
U の定義より，u は U に含まれない ➜ 矛盾！

* u が U に含まれない場合： f(u) = U より，u は f(u) に含まれない．
U の定義より，u は U に含まれる ➜ 矛盾！

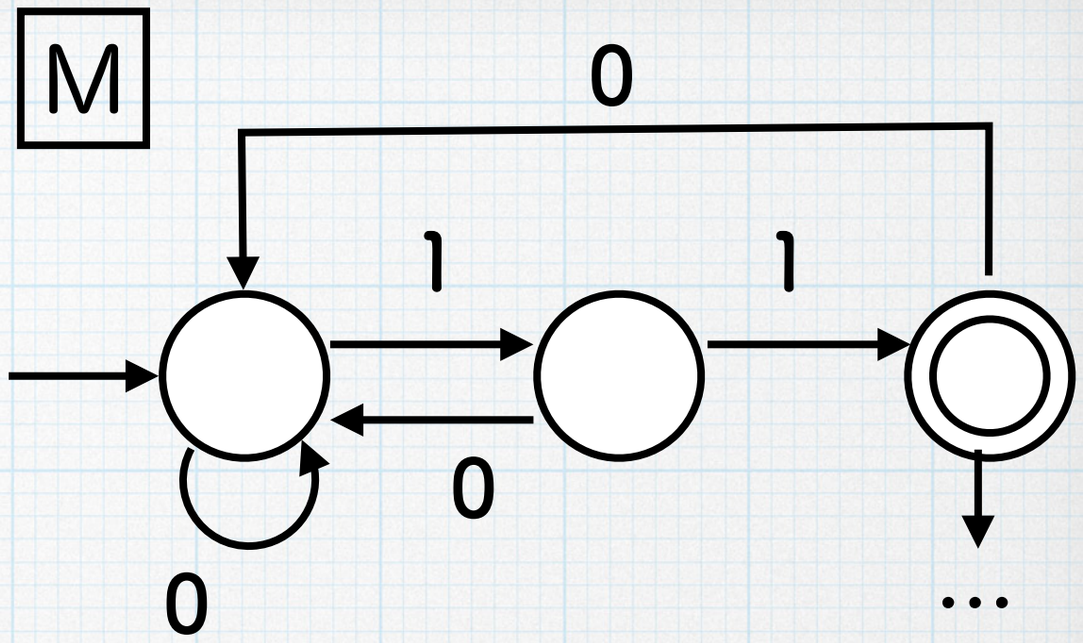
オートマトンの表現能力の限界

* クイズ 具体例は？

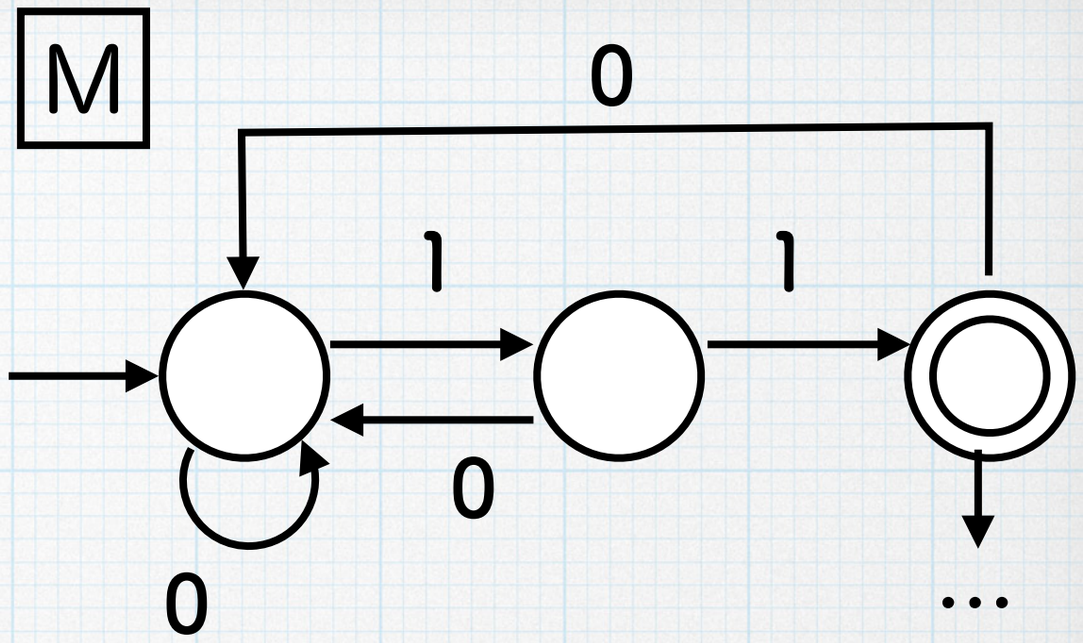


オートマトンの表現能力の限界

* クイズ 具体例は？
* $L = \{ 0^n 1^n : \quad n \text{ は 自然数} \}$
  $= \{ \varepsilon, 01, 0011, 000111, \cdots \}$
  は有限オートマトンで表現不可



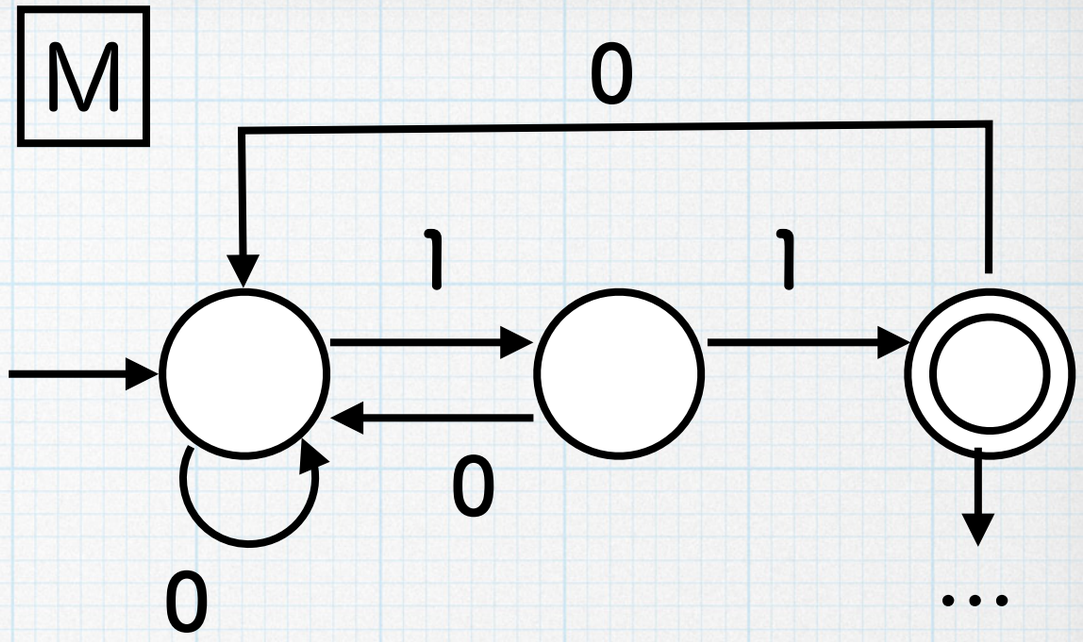
オートマトンの表現能力の限界



* <u>クイズ</u> 具体例は？
* L = { 0^n1^n :　n は 自然数 }
  = { ε , 01, 0011, 000111, … }
  は有限オートマトンで表現不可

* <u>証明</u> （アイデア）
  * オートマトンは左から右に文字を読む
  * 読んだ文字は次々に忘れていく
  * それでもなんとか記憶を残そうとする
  * 唯一の記憶装置は「今どの状態にいるか？」
    ➔ 決められた個数（有限個）しかない！
    ➔ たくさん 0 を読んでいるうちに「あふれる」

オートマトンの表現能力の限界



* 定理

  $L = \{ 0^n 1^n : \ n$ は自然数 $\}$

  $= \{ \varepsilon, 01, 0011, 000111, \cdots \}$

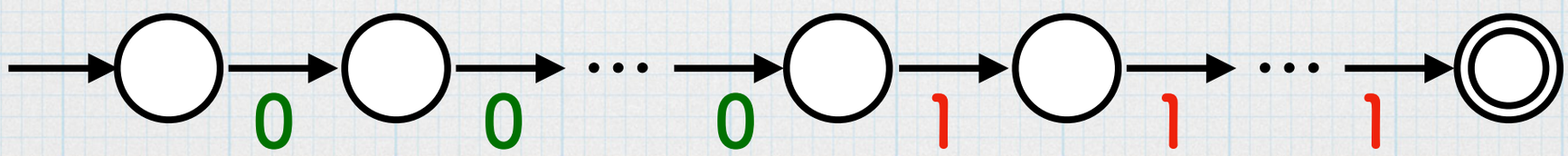
  は有限オートマトンで表現不可

* 証明　（背理法）

  $L = L(M)$ となる有限オートマトン M が存在すると仮定

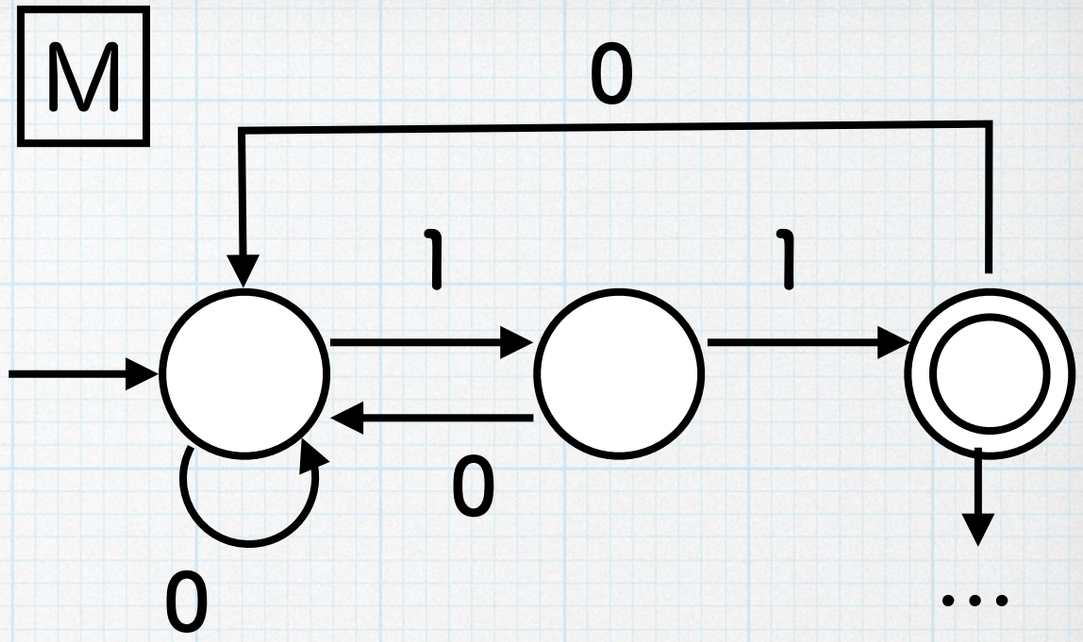
  * M の状態数を n とする
  * 文字列 $0^n 1^n$ を考える．これは M に受理されるはず．

    オートマトンの実行トレース



    がとれる．

オートマトンの表現能力の限界



* 定理

  $L = \{ 0^n 1^n : \ n \text{ は 自然数} \}$

  $= \{ \varepsilon, 01, 0011, 000111, \cdots \}$

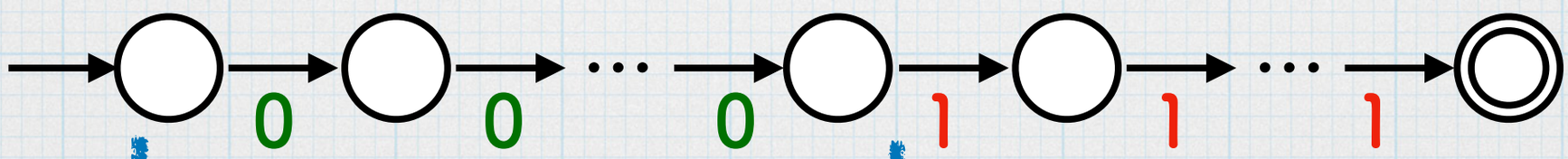
  は有限オートマトンで表現不可

* 証明　（背理法）

  $L = L(M)$ となる有限オートマトン M が存在すると仮定

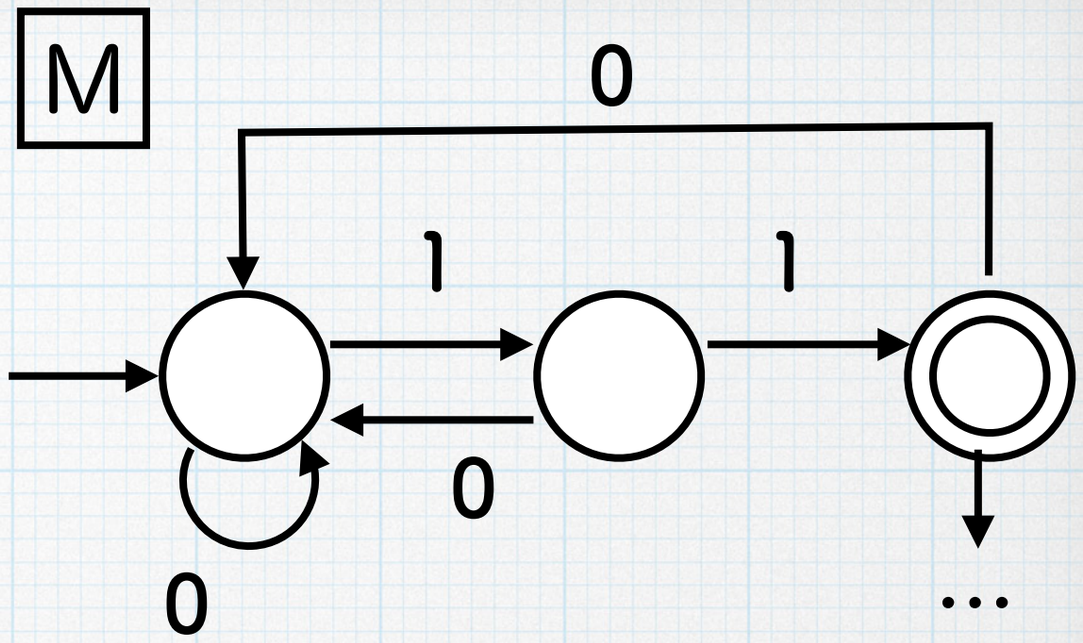
  * M の状態数を n とする
  * 文字列 $0^n 1^n$ を考える．これは M に受理されるはず．

    オートマトンの実行トレース



    がとれる．

オートマトンの表現能力の限界



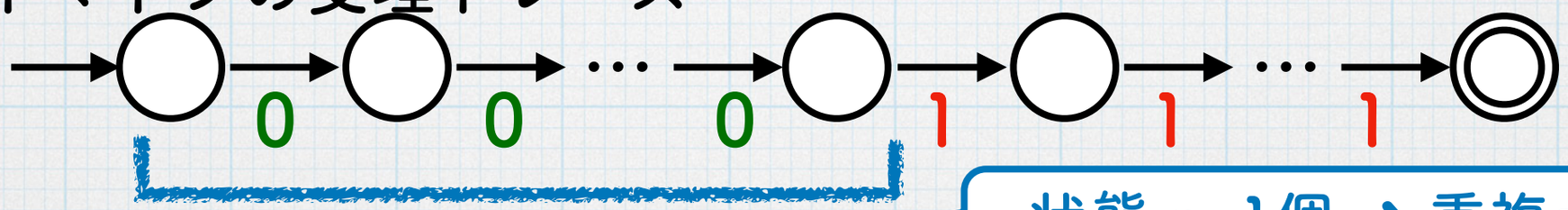
* 証明　（背理法）

  L = L(M) となる有限オートマトン M が存在すると仮定

  * M の状態数を n とする

  * 文字列 0^n1^n を考える．これは M に受理されるはず．

    オートマトンの受理トレース



    がとれる．

  * 重複をショートカットしたら，$0^{n-k}1^n$ の受理トレースができる（k はショートカットの長さ）．

    ➜ L = L(M) に矛盾！ 証明終．

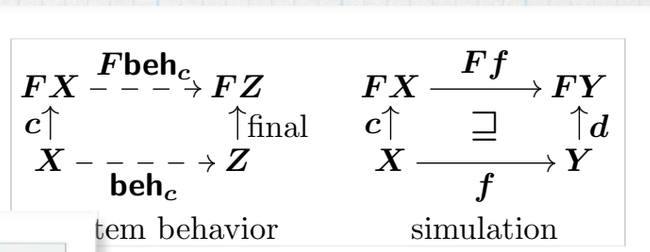
理論計算機科学入門 — 有限と無限のあいだ

アウトライン

* 理論計算機科学とは — 有限と無限のせめぎあい
* いろいろな無限 — 対角線論法
* オートマトン理論入門
  * 包含問題を解くアルゴリズム — 有限の方法で無限の対象を操作
  * オートマトンの表現能力 — 有限性の限界
* 形式検証 — オートマトンを用いたシステム品質保証
* AI 時代の形式検証 — 研究紹介

自己紹介

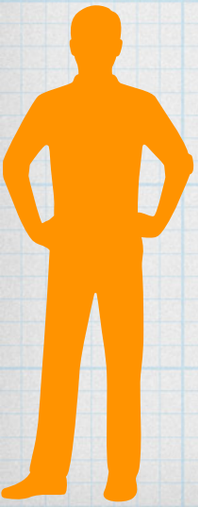
- 理学士（東大 数学）
  理学修士（東工大 数理・計算科学）
  PhD （Computer Science, Radboud University Nijmegen, NL）
- <u>専門分野</u>
  - 理論計算機科学における圏論的構造（category theory）
  - 特に，オートマトンの圏論的余代数によるモデリングと一般論
  - 数学的抽象性が生み出す新奇な応用
    ➜ 自動運転システムの品質保証手法
  - 特に，定理証明とモデル検査.
    「システムが安全なことを証明」

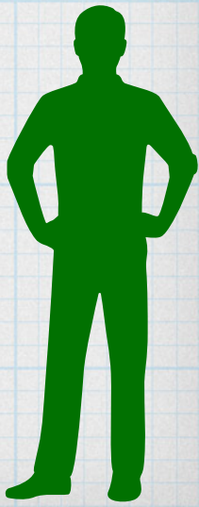


圏論の歩き方
（日本評論社）

ビジネスでの証明

https://www.pexels.com/ja-jp/photo/63294/

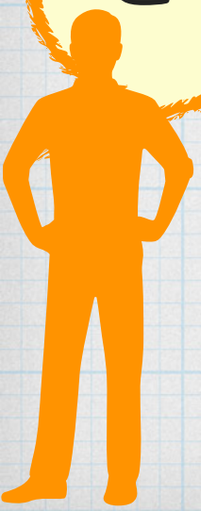


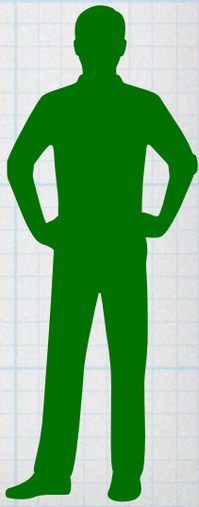


ビジネスでの証明

https://www.pexels.com/ja-jp/photo/63294/

この駐車場,
どうっすか！？





ビジネスでの証明

https://www.pexels.com/ja-jp/photo/63294/

この駐車場,
どうっすか！？

ゴンドラが衝突したりしない？ 大丈夫？

ビジネスでの証明

絶対大丈夫っす！

この駐車場,
どうっすか！？

https://www.pexels.com/ja-jp/photo/63294/

ゴンドラが衝突したりしない？ 大丈夫？

ビジネスでの証明

絶対大丈夫っす！

この駐車場,
どうっすか！？

ホントに？ なんで？

ゴンドラが衝突したりしない？ 大丈夫？

https://www.pexels.com/ja-jp/photo/63294/

ビジネスでの証明

オレが責任持つっす！

絶対大丈夫っす！

この駐車場,
どうっすか！？

ホントに？ なんで？

ゴンドラが衝突したりしない？ 大丈夫？

https://www.pexels.com/ja-jp/photo/63294/

ビジネスでの証明

オレが責任持つっす！

絶対大丈夫っす！

この駐車場,
どうっすか！？

https://www.pexels.com/ja-jp/photo/63294/

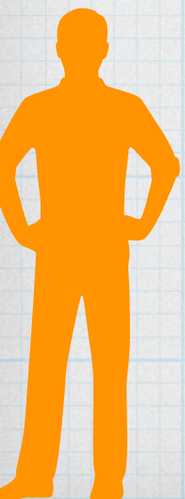
・・・
（ダメだこいつ）

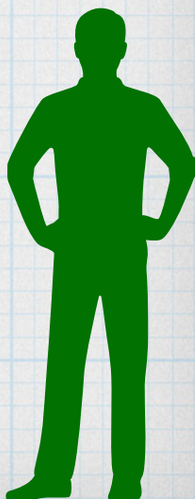
ホントに？ なんで？

ゴンドラが衝突したりしない？ 大丈夫？

ビジネスでの証明

https://www.pexels.com/ja-jp/photo/63294/





ビジネスでの証明

https://www.pexels.com/ja-jp/photo/63294/

この駐車場,
どうでしょう.

ビジネスでの証明

https://www.pexels.com/ja-jp/photo/63294/

この駐車場,
どうでしょう.

ゴンドラが衝突したりしない？ 大丈夫？

ビジネスでの証明

はい，大丈夫です.

なぜなら，任意の状態 s に対して，ゴンドラ g_1 の位置を x_1 とすると，．．．

この駐車場，
どうでしょう.

https://www.pexels.com/ja-jp/photo/63294/

ゴンドラが衝突したりしない？ 大丈夫？

ビジネスでの証明

はい，大丈夫です．
なぜなら，任意の状態 s に対して，ゴンドラ g_1 の位置を x_1 とすると，．．．

この駐車場，
どうでしょう．

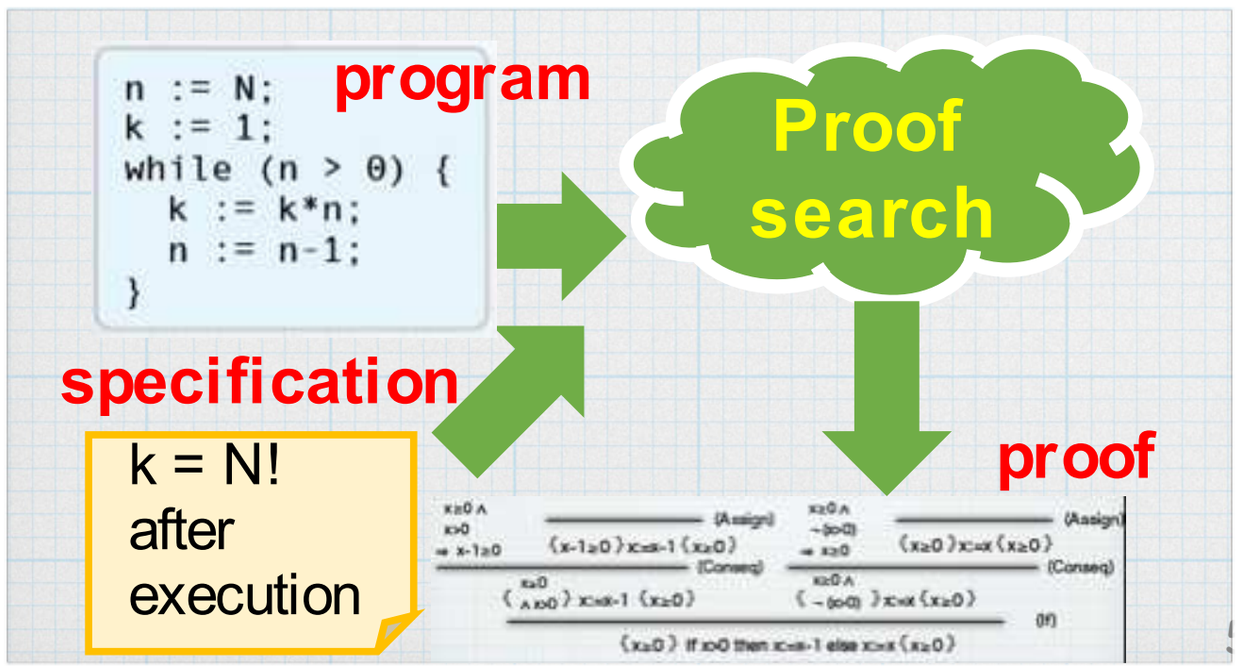
．．．（読んでる）
なるほど．それでは1基いただこう．

ゴンドラが衝突したりしない？ 大丈夫？

https://www.pexels.com/ja-jp/photo/63294/

定理証明によるシステム品質保証

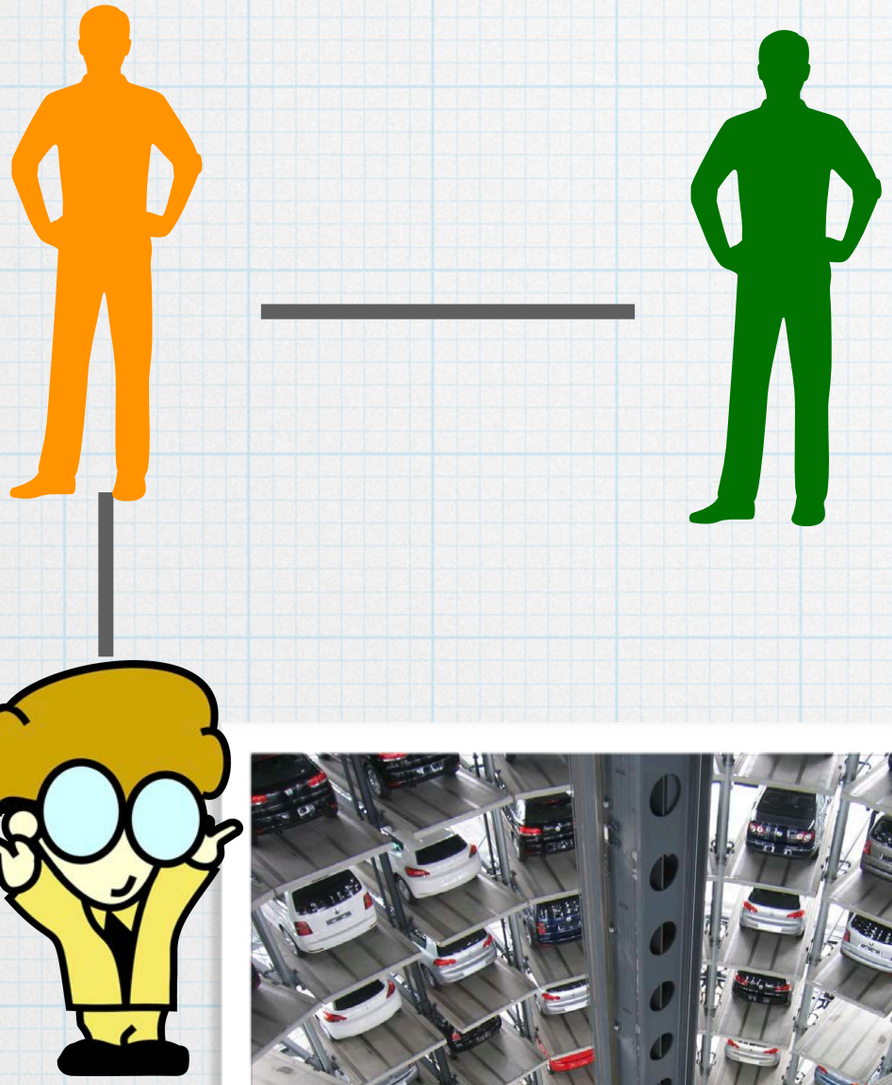
* 例: Hoare 論理による定理証明，プログラムの品質保証
  * イメージ： 紙とペンで証明を書くのと同じ
    * しかし，人力だと間違えるので，計算機上で.
      記号列で証明を表現 ➜ 各ステップの正当性をプログラム（証明支援系）がチェック
    * できれば証明の自動探索（定理証明器）
  * （原理上は）無限の入力空間をすべてカバーできる
    * 証明では「i を入力の自然数とする」とかけますね
  * 数学的証明という，強い品質保証




```
Theorem forall_exists : (forall P : Set->Prop,
          (forall x, ~(P x)) -> ~(exists x, P x)).
Proof.
  intros P.
  intros forall_x_not_Px.
  unfold not.
  intros exists_x_Px.
  destruct exists_x_Px as [ witness proof_of_Pwitness].
  pose (not_Pwitness := forall_x_not_Px witness).
  unfold not in not_Pwitness.
  pose (proof_of_False := not_Pwitness proof_of_Pwitness).
  case proof_of_False.
Qed.
```

A Coq proof of ∀x. ¬P(x) → ¬ ∃x. P(x)

証明で金もうけ



https://www.needpix.com/photo/178834/nerd-cartoon-geek-character-glasses-isolated-funny-face-smart

* 定理証明のコスト：
  * たとえば，ヨーロッパの計算機科学の博士課程（有給，3–4年）
* 1000万円をゆうに超える

モデル検査によるシステム品質保証

* 基本的ワークフロー
  * システムモデルを，オートマトン M として表現
  * 求める安全性をオートマトン M′ として表現．

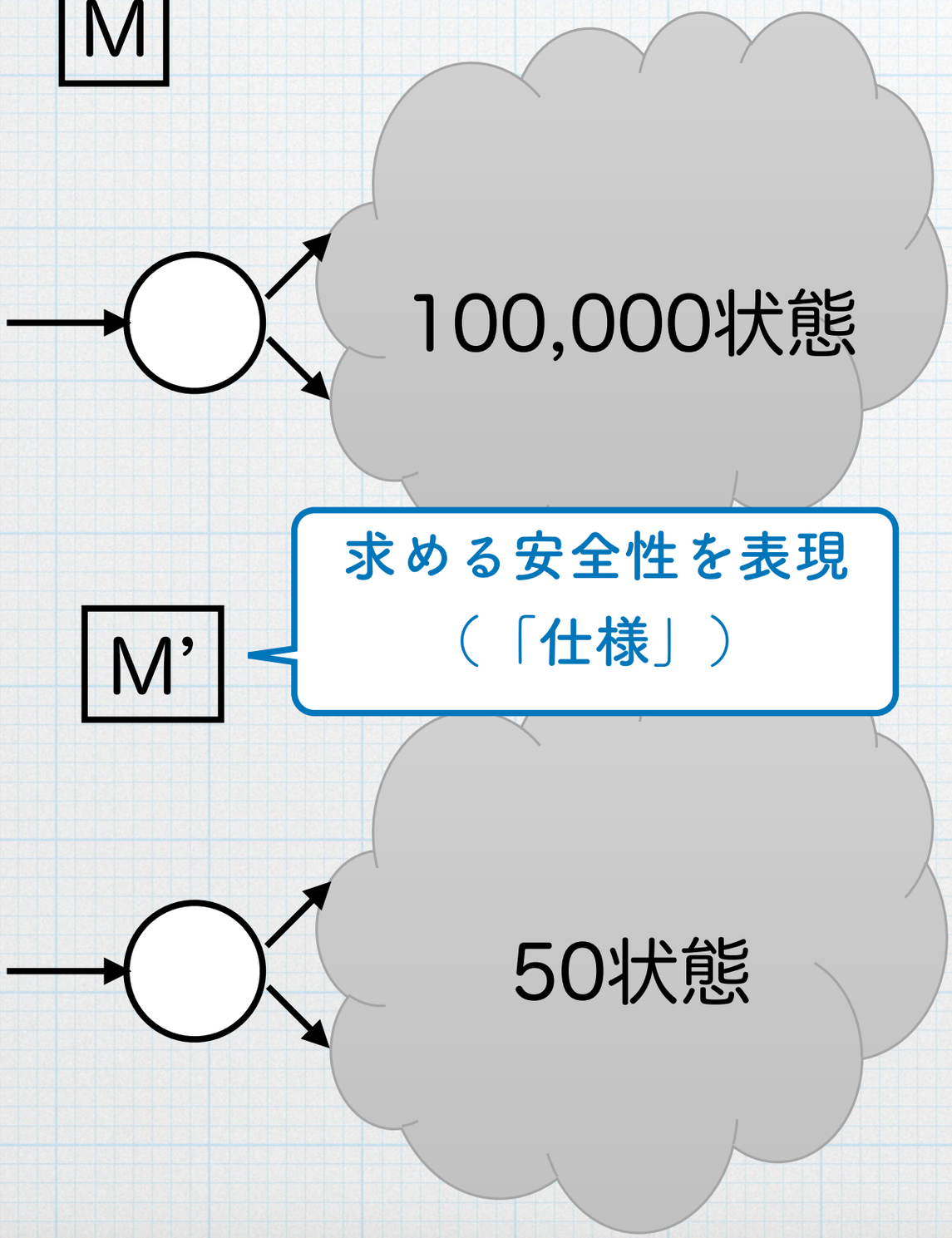
「M′ に受理される文字列はすべて安全」

* $L(M) \subseteq L(M')$ を示す（今日のアルゴリズムで）
* すると，M の振る舞い（＝受理される文字列）がすべて安全であると結論できる

オートマトンによる自動証明

システムモデル（駐車場の設計図など）

M



$L(M) \subseteq L(M')$

Yes/No

ソフトウェア品質保証におけるスペクトル

形式手法

テスト test | モデル検査 | 自動定理証明 | 対話定理証明

低 コスト 高

ブラックボックスモデルを許容

ホワイトボックスモデルを要求

高 自動化の度合い 低

自動

専門家を月・年単位で拘束

低 保証の網羅性 高

低 保証の数学的厳密性 高

経験論的保証

数学的保証 ➜ 「形式検証」

理論計算機科学入門 — 有限と無限のあいだ

アウトライン

* 理論計算機科学とは — 有限と無限のせめぎあい
* いろいろな無限 — 対角線論法
* オートマトン理論入門
  * 包含問題を解くアルゴリズム — 有限の方法で無限の対象を操作
  * オートマトンの表現能力 — 有限性の限界
* 形式検証 — オートマトンを用いたシステム品質保証
* AI 時代の形式検証 — 研究紹介

形式手法の歴史（超ダイジェスト）

ソフトウェアの大規模化

ネットワーク，分散並列

物理システムとの融合

統計的機械学習の能力向上

* 1970頃 形式手法によるソフトウェア品質保証（ソフトウェア検証）の研究開始．定理証明，モデル検査

* 1990頃 ソフトウェア検証のツール化・実応用加速
  定理証明： Isabelle, Coq, PVS, …
  モデル検査： SPIN, SMV/NuSMV, mCRL2, PRISM, Uppaal, …

* 2006 物理情報システム (Cyber-Physical Systems, CPS) への応用開始．ソフトウェア検証 ＋ 制御理論
  （この経緯は［奥村，研究 技術 計画 2017］に詳しい）

* 2016 機械学習システムへの応用．
  形式的推論と統計的推論

ソフトウェア品質保証におけるスペクトル

形式手法

テスト test | モデル検査 | 自動定理証明 | 対話定理証明

低 コスト 高

ブラックボックスモデルを許容 | ホワイトボックスモデルを要求

高 自動化の度合い 低

自動 | 専門家を月・年単位で拘束

低 保証の網羅性 高

低 保証の数学的厳密性 高

経験論的保証 | 数学的保証 ➜ 「形式検証」

ソフトウェア品質保証におけるスペクトル

形式手法

テスト test | モデル検査 | 自動定理証明 | 対話定理証明

低

ブラックボックス
モデルを許容

高

低

を月・年
で拘束

低

経験論的保証

数学的保証 ➜ 「形式検証」

システムの正しさを数学的に証明
➜ システムの「定義」が必要.

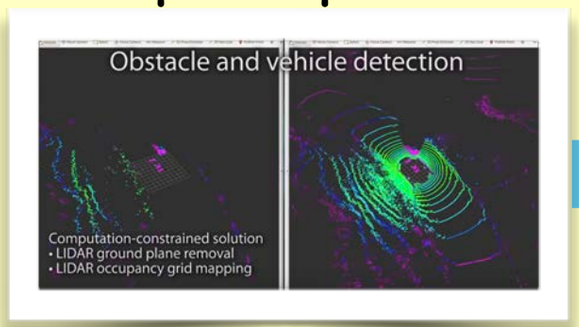
「定義」 = システムモデル.
設計図,
システムの完全な記述

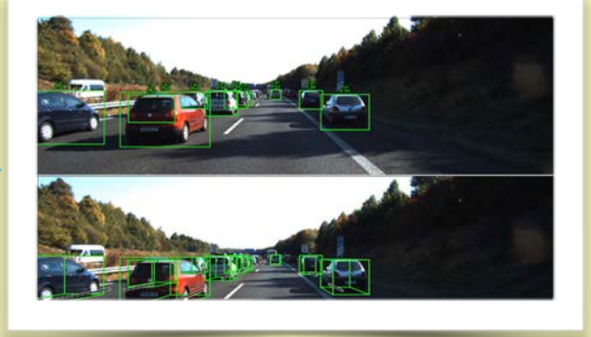
Uncertainties in Cyber-Physical Systems 2020

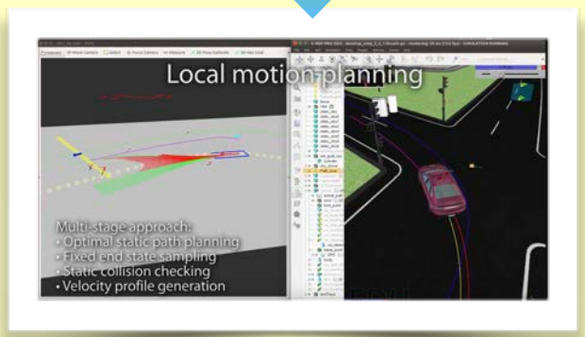
* Take automated driving…

physical reality

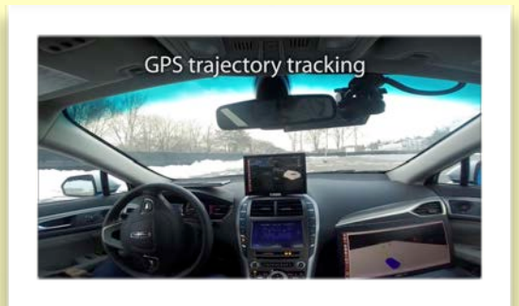
sensing







behavior planning
path planning



local planning
path tracking

actuation

Uncertainties in Cyber-Physical Systems 2020

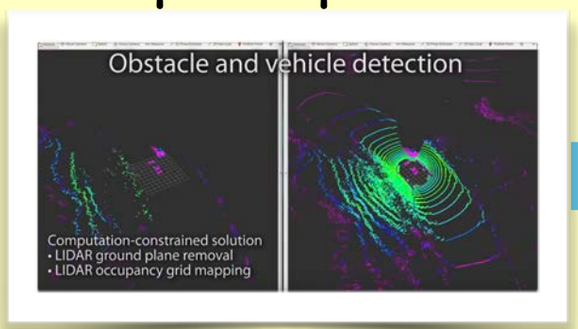
* Take automated driving…

physical reality

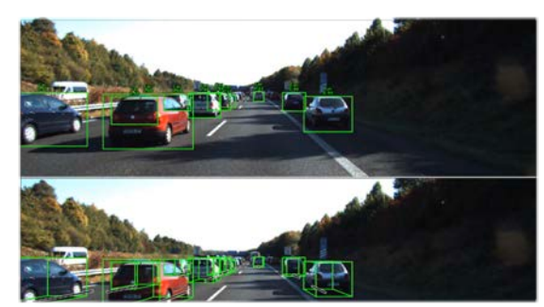
sensing

actuation

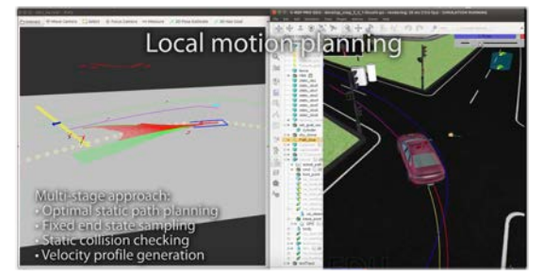
perception



object recognition



local planning
path tracking



behavior planning
path planning

External uncertainties:
other traffic participants, weather, sunlight, comets, earthquakes, …

Uncertainties in Cyber-Physical Systems 2020

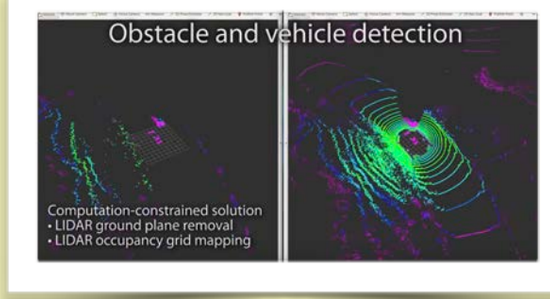
* Take automated driving…

Internal uncertainties:
never trust neural nets!

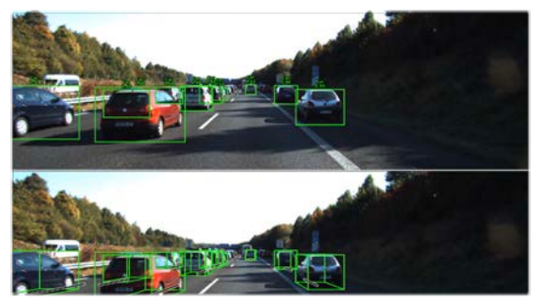
physical reality

sensing

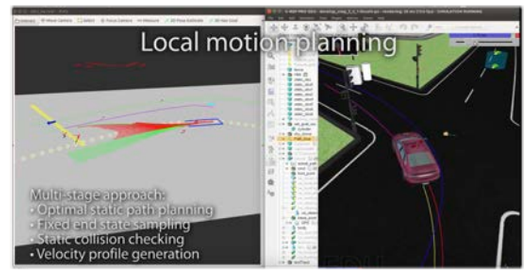
perception



object recognition



local planning
path tracking

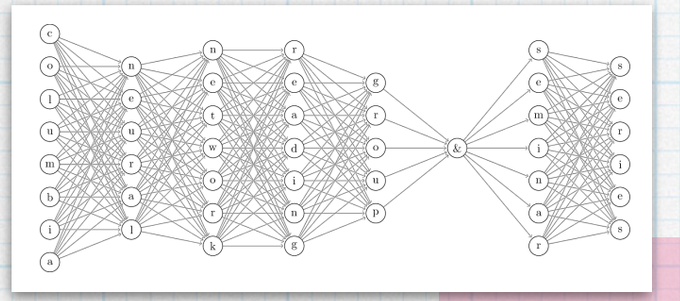


behavior planning
path planning

actuation

External uncertainties:
other traffic participants, weather,
sunlight, comets, earthquakes, …

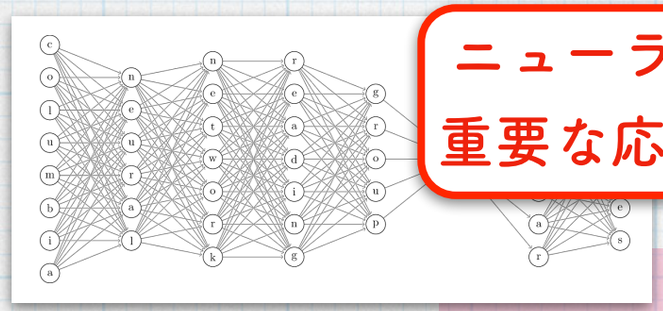
統計的機械学習 vs 演繹的形式推論



$$\frac{A \quad A \supset B}{B}$$

統計的機械学習		演繹的形式推論
データのノイズを許容	入力の誤り	公理は絶対誤りは想定せず
保証されない	結論の正しさ	論理的に保証（cf. 数学的証明）
高い データから自動で特徴量発見	スケーラビリティ	低い 公理の準備は人力（cf. エキスパートシステム）
低い 判断の理由はパラメータ（重み）	説明可能性	高い 推論過程が証明として明示的

統計的機械学習 vs 演繹的形式推論



ニューラルネットなど.
重要な応用対象

今日の話

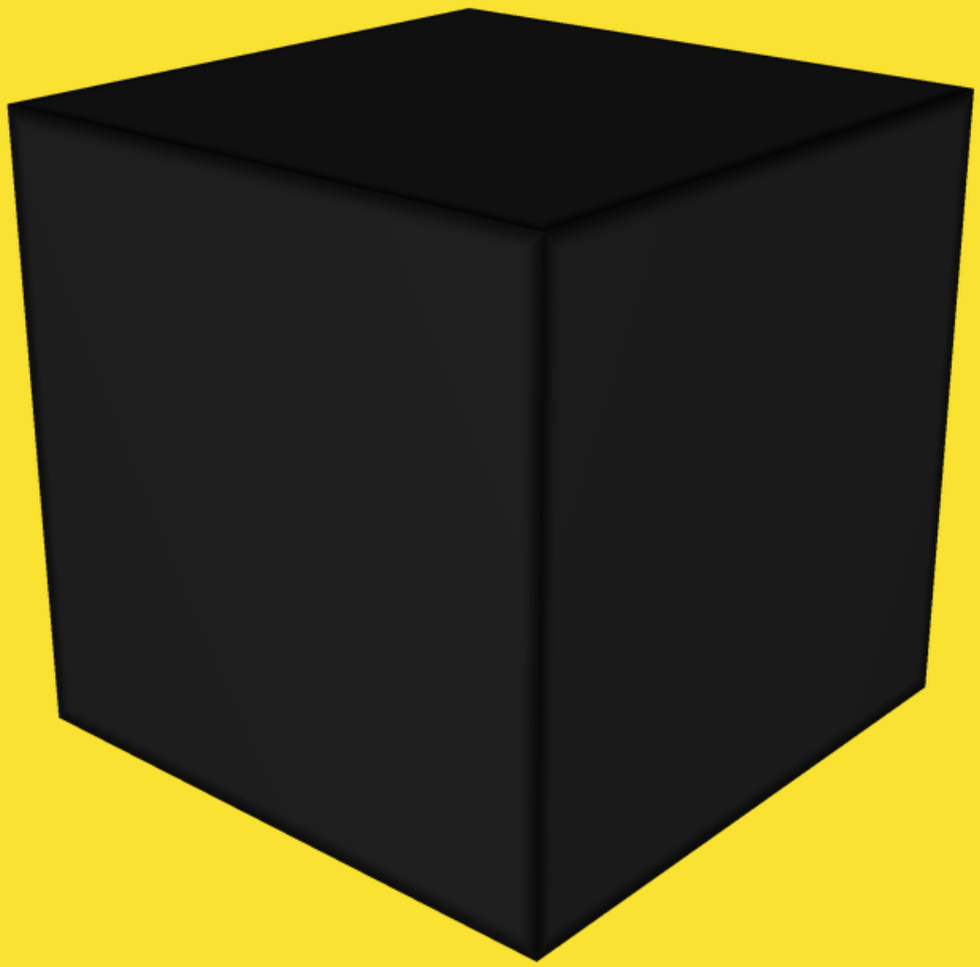
$$\frac{A \quad A \supset B}{B}$$

統計的機械学習		演繹的形式推論
データのノイズを許容	入力の誤り	公理は絶対誤りは想定せず
保証されない	結論の正しさ	論理的に保証（cf. 数学的証明）
高い データから自動で特徴量発見	スケーラビリティ	低い 公理の準備は人力 （cf. エキスパートシステム）
低い 判断の理由はパラメータ（重み）	説明可能性	高い 推論過程が証明として明示的

You walk in the dark,

You walk in the dark,
guided by an apparatus

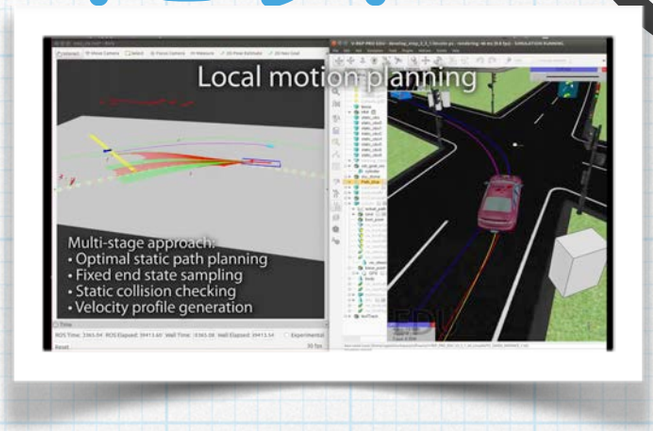
… which itself is a black box!



https://pixabay.com/illustrations/cube-block-black-box-3d-geometric-250082/

ERATO　蓮尾メタ数理システムデザインプロジェクト
ERATO Metamathematics for Systems Design Project
国立情報学研究所 & 科学技術振興機構　National Institute of Informatics & Japan Science and Technology Agency

ERATO MMSD 紹介

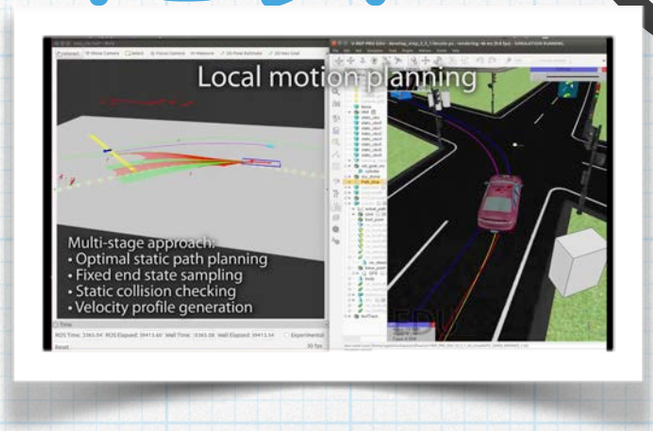


* JST ERATO プロジェクト
  * 2016/10-2022/03.
    総勢50名規模の基礎研究プロジェクト

* プロジェクト目標：工業製品の設計サポート
  * 形式手法の拡張．ソフトウェアから物理情報システムへ
  * 安全性・信頼性，Verification & Validation.
    「システムが期待通り動作するか」
  * 特に自動運転を戦略的ターゲットに．U Waterloo と協働
    www.autonomoose.net

ERATO　蓮尾メタ数理システムデザインプロジェクト
ERATO Metamathematics for Systems Design Project
国立情報学研究所 & 科学技術振興機構　National Institute of Informatics & Japan Science and Technology Agency

ERATO MMSD 紹介



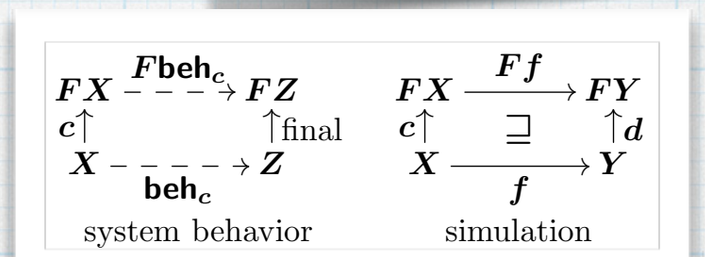
* JST ERATO プロジェクト
  * 2016/10-2022/03.
    総勢50名規模の基礎研究プロジェクト

* プロジェクト目標：工業製品の設計サポート
  * 形式手法の拡張．ソフトウェアから物理情報システムへ
  * 安全性・信頼性，Verification & Validation.
    「システムが期待通り動作するか」
  * 特に自動運転を戦略的ターゲットに．U Waterloo と協働
    www.autonomoose.net

ERATO　蓮尾メタ数理システムデザインプロジェクト
ERATO Metamathematics for Systems Design Project
国立情報学研究所 & 科学技術振興機構　National Institute of Informatics & Japan Science and Technology Agency

ERATO MMSD 紹介

* アプローチ
  * テストと形式検証（証明）のミックス
    * 環境の不確かさ，統計的機械学習，…
      ➜ すべてを証明することは不可能
    * それでもできるだけ証明・説明を
      （cf. 自動運転車の安全性）
    * 証明を用いたテスト効率化
* 現代数学（特に圏論）を軸にした
  異分野協働，科学的ブレークスルー
  * たとえばオートマトンは余代数として
    抽象化できます
  * （… また次の機会に）



圏論の歩き方
（圏論の歩き方委員会 編）
日本評論社，2015

理論計算機科学入門 — 有限と無限のあいだ

アウトライン

* 理論計算機科学とは — 有限と無限のせめぎあい
* いろいろな無限 — 対角線論法
* オートマトン理論入門
  * 包含問題を解くアルゴリズム — 有限の方法で無限の対象を操作
  * オートマトンの表現能力 — 有限性の限界
* 形式検証 — オートマトンを用いたシステム品質保証
* AI 時代の形式検証 — 研究紹介

参考文献

* 思いつくところを少し挙げます．完全なリストには程遠いです．
* いろいろな無限 — 対角線論法
  * 数学科の2年で習う「集合・位相」の冒頭部分です
  * 集合への30講，志賀 浩二，朝倉書店 （読みやすい）
    集合・位相入門，松坂 和夫，岩波書店 （ハードコア）
* オートマトン理論入門
  * 情報科学科の2年で習う「形式言語理論」の最初の半分を，形式検証向けにアレンジしました
  * J. Hopcroft, R. Motowani, J. Ullman 著，野崎 昭弘， 高橋 正子，町田 元，山崎 秀記 訳
    『オートマトン，言語理論，計算論 I，第2版』 サイエンス社
* 形式検証
  * ツールがたくさんあるので，ダウンロードして遊んでみると楽しいのでは
  * SPIN モデル検査：検証モデリング技法，中島 震，近代科学社 （モデル検査）
    コンピュータは数学者になれるのか？ 数学基礎論から証明とプログラムの理論へ，照井 一成，青土社（定理証明と論理学入門，おすすめ）

理論計算機科学入門 — 有限と無限のあいだ

まとめ

* 理論計算機科学とは — 有限と無限のせめぎあい
* いろいろな無限 — 対角線論法
* オートマトン理論入門
  * 包含問題を解くアルゴリズム — 有限の方法で無限の対象を操作
  * オートマトンの表現能力 — 有限性の限界
* 形式検証 — オートマトンを用いたシステム品質保証
* AI 時代の形式検証 — 研究紹介

ご清聴ありがとうございました

http://group-mmm.org/~ichiro/

さっそく脱線：
数学基礎論，メタ数学

Hilbert, Gödel, Church, Gentzen, Cohen, Turing, ...
→ computers!

Definition. A *proof* is a finite tree subject to the derivation rules:

$$\cdots \qquad \frac{\Gamma \Rightarrow \Delta, A \quad B, \Pi \Rightarrow \Sigma}{A \supset B, \Gamma, \Pi \Rightarrow \Delta, \Sigma}\ (\supset\text{-L}) \qquad \frac{A, \Gamma \Rightarrow \Delta, B}{\Gamma \Rightarrow \Delta, A \supset B}\ (\supset\text{-R}) \qquad \cdots$$

$$\frac{\Gamma \Rightarrow \Delta, A \quad A, \Pi \Rightarrow \Sigma}{\Gamma, \Pi \Rightarrow \Delta, \Sigma}\ (\textsc{Cut})$$

$$\vdots$$

$$\Rightarrow A$$

A *theorem* is a formula A with a proof $\Rightarrow A$.

Theorem (cut elimination).

$$\vdots$$

$$\Rightarrow A$$

Any theorem A has a proof $\Rightarrow A$ where the $(\textsc{Cut})$ rule is never used.

Proof. Let Π be a proof of a theorem A. We turn Π into a cut-free proof Π' by induction. . .

* 数学者の営為の数学的研究
* 定義，定理，証明などの概念が数学的に定義される

さっそく脱線：
数学基礎論，メタ数学

Hilbert, Gödel, Church, Gentzen, Cohen, Turing, ...
→ computers!

Definition. A *proof* is a finite tree subject to the derivation rules:

$$\cdots \qquad \frac{\Gamma \Rightarrow \Delta, A \quad B, \Pi \Rightarrow \Sigma}{A \supset B, \Gamma, \Pi \Rightarrow \Delta, \Sigma}\ (\supset\text{-L}) \qquad \frac{A, \Gamma \Rightarrow \Delta, B}{\Gamma \Rightarrow \Delta, A \supset B}\ (\supset\text{-R}) \qquad \cdots$$

$$\frac{\Gamma \Rightarrow \Delta, A \quad A, \Pi \Rightarrow \Sigma}{\Gamma, \Pi \Rightarrow \Delta, \Sigma}\ (\textsc{Cut})$$

$$\vdots$$

$$\Rightarrow A$$

A *theorem* is a formula A with a proof $\dfrac{}{\Rightarrow A}$.

Theorem (cut elimination).

$$\vdots$$

$$\Rightarrow A$$

Any theorem A has a proof $\dfrac{}{\Rightarrow A}$ where the $(\textsc{Cut})$ rule is never used.

Proof. Let Π be a proof of a theorem A. We turn Π into a cut-free proof Π' by induction. . .

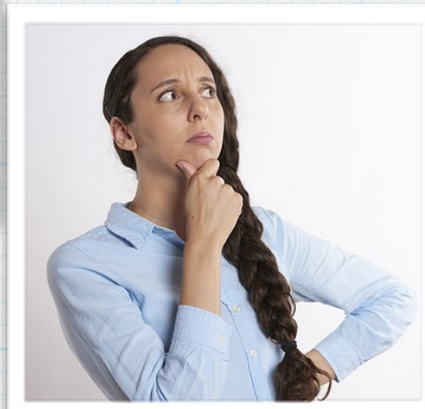


Image by Robin Higgins from Pixabay

数学者

* 数学者の営為の数学的研究
* 定義，定理，証明などの概念が数学的に定義される

さっそく脱線：
数学基礎論，メタ数学

Hilbert, Gödel, Church, Gentzen, Cohen, Turing, ...
→ computers!

Definition. A *proof* is a finite tree subject to the derivation rules:

$$\cdots \qquad \frac{\Gamma \Rightarrow \Delta, A \quad B, \Pi \Rightarrow \Sigma}{A \supset B, \Gamma, \Pi \Rightarrow \Delta, \Sigma}\ (\supset\text{-L}) \qquad \frac{A, \Gamma \Rightarrow \Delta, B}{\Gamma \Rightarrow \Delta, A \supset B}\ (\supset\text{-R}) \qquad \cdots$$

$$\frac{\Gamma \Rightarrow \Delta, A \quad A, \Pi \Rightarrow \Sigma}{\Gamma, \Pi \Rightarrow \Delta, \Sigma}\ (\textsc{Cut})$$

A *theorem* is a formula A with a proof $\dfrac{\vdots}{\Rightarrow A}$.

Theorem (cut elimination). Any theorem A has a proof $\dfrac{\vdots}{\Rightarrow A}$ where the $(\textsc{Cut})$ rule is never used.

Proof. Let Π be a proof of a theorem A. We turn Π into a cut-free proof Π' by induction. . .

メタ数学者

Image by Robin Higgins from Pixabay

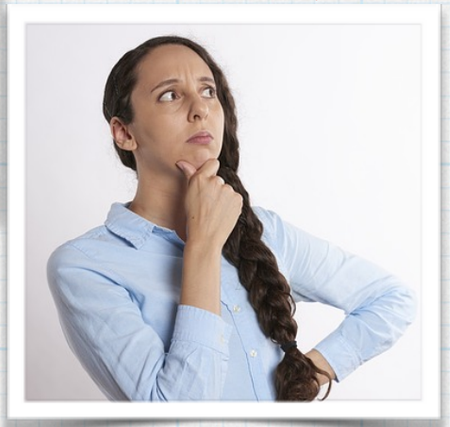


Image by Robin Higgins from Pixabay

数学者

* 数学者の営為の数学的研究
* 定義，定理，証明などの概念が数学的に定義される

さっそく脱線：
数学基礎論，メタ数学

Hilbert, Gödel, Church, Gentzen, Cohen, Turing, ...
→ computers!

どんなにがんばっても
それは証明できないよ…

メタ数学者

Image by Robin Higgins from Pixabay

Definition. A *proof* is a finite tree subject to th

$$\frac{\Gamma \Rightarrow \Delta, A \quad B, \Pi \Rightarrow \Sigma}{A \supset B, \Gamma, \Pi \Rightarrow \Delta, \Sigma}\ (\supset\text{-L}) \qquad \frac{A, \Gamma \Rightarrow \Delta, }{\Gamma \Rightarrow \Delta, A \supset B}$$

$$\frac{\Gamma \Rightarrow \Delta, A \quad A, \Pi \Rightarrow \Sigma}{\Gamma, \Pi \Rightarrow \Delta, \Sigma}\ (\textsc{Cut})$$

A *theorem* is a formula A with a proof $\dfrac{\vdots}{\Rightarrow A}$.

Theorem (cut elimination). Any theorem A has a proof $\dfrac{\vdots}{\Rightarrow A}$ where the $(\textsc{Cut})$ rule is never used.

Proof. Let Π be a proof of a theorem A. We turn Π into a cut-free proof Π' by induction. . .

Image by Robin Higgins from Pixabay

数学者

* 数学者の営為の数学的研究
* 定義，定理，証明などの概念が数学的に定義される

さっそく脱線：
数学基礎論，メタ数学

Hilbert, Gödel, Church, Gentzen, Cohen, Turing, ...
→ computers!

Definition. A *proof* is a finite tree subject to th

$$\frac{\Gamma \Rightarrow \Delta, A \quad B, \Pi \Rightarrow \Sigma}{A \supset B, \Gamma, \Pi \Rightarrow \Delta, \Sigma}\ (\supset\text{-L}) \qquad \frac{A, \Gamma \Rightarrow \Delta, }{\Gamma \Rightarrow \Delta, A \supset B}$$

$$\cdots$$

$$\frac{\Gamma \Rightarrow \Delta, A \quad A, \Pi \Rightarrow \Sigma}{\Gamma, \Pi \Rightarrow \Delta, \Sigma}\ (\textsc{Cut})$$

$$\vdots$$

$$\frac{}{\Rightarrow A}$$

A *theorem* is a formula A with a proof $\dfrac{}{\Rightarrow A}$.

どんなにがんばっても
それは証明できないよ…

…ということを
私は証明できます！

メタ数学者

Image by Robin Higgins from Pixabay

Theorem (cut elimination).

Any theorem A has a proof $\dfrac{}{\Rightarrow A}$ where the (CUT) rule is never used.

Proof. Let Π be a proof of a theorem A. We turn Π into a cut-free proof Π' by induction. . .

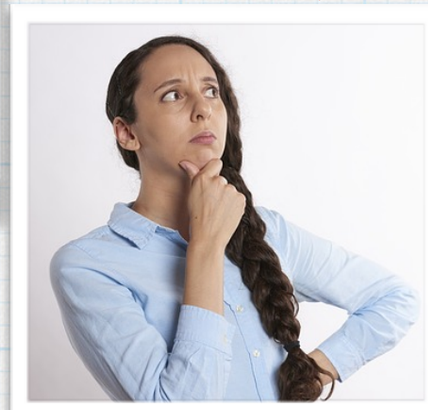


Image by Robin Higgins from Pixabay

数学者

* 数学者の営為の数学的研究
* 定義，定理，証明などの概念が数学的に定義される

理論計算機科学とは？

* 応用数学の一分野です
  * ↔ 純粋数学
    * 「応用する気のない数学」
    * 「応用できない数学」「役に立たない数学」とは違う
* 本人にその気がなくても，
  時々応用で大ホームランを飛ばす
* あなどれない！！