余帰納法，双模倣

muratak

2021 年 6 月 12 日

本稿は書きかけであり，主に知り合いの勉強会のために使用している資料であり，勉強会の進行に伴って追記・更新されます．（なお，この勉強会に興味のある方は **muratak** まで直接ご連絡ください）

第 1 章
ラベル付き遷移系と双模倣

本章では，ラベル付き遷移系 [Plo81, Plo04] および双模倣 [Par81] を導入する．

§1.1 ラベル付き遷移系と状態システム

ラベル付き遷移系は，端的には，「状態をもち，イベントに伴って状態が遷移していくシステム」を記述するために用いられる数学的構造である．

1.1.1 ラベル付き遷移系の定義

まずは，ラベル付き遷移系（LTS）の定義を述べる．LTS は数学的には単なるラベル付き有向グラフであるが，情報工学の観点では，システムの動作を記述するのに便利である[*1]．

以下，**アルファベット**（alphabet）とは，空でない集合 Σ のことであるとする．直観的には，アルファベットはシステムが使うメッセージ全体の集合であるが，有限である必要はない．

▶**定義 1.1.1**（**ラベル付き遷移系**） Σ をアルファベットとする．$\mathcal{M}$ が Σ 上の**ラベル付き遷移系**（LTS; Labelled Transition System）とは，

- 状態の非空集合 Q および
- 遷移関係 $\Delta \subseteq Q \times \Sigma \times Q$

の組 $\mathcal{M} = (Q, \Delta)$ である．$(q, a, q') \in \Delta$ のとき，$q \xrightarrow{a} q'$ と書くこともある．

$q \xrightarrow{a} q'$ であることは，直観的には「状態 q で，アクション a を伴って，状態 q' に変化した」ということである．ここで「アクション」とは，数学的には単なるラベルに過ぎない．具体的に何を意味するのかは，記述対象のシステムに依って大きく異なるものなので，ここでは敢えてこのような抽象的な言い方

[*1] UML でいえば状態機械図（state-machine diagram）に近いと考えて良い．

にとどめている．

▷ **注意 1.1.2**　例えば，$q \xrightarrow{a} q'$ であるということを，「ある機械が状態 q であるときに，a を入力することで，その機械が状態 q' へ遷移した」と思うべき場合もあるし，「ある機械が状態 q であったが，その機械が a を出力して，さらに状態 q' へ遷移した」と思うべき場合もある．要するに，LTS という構造では，アルファベットの文字 a が，機械に対する入力なのか，機械からの出力なのか，はたまたそれ以外の何かなのかといった区別はしないということである．これはシステムを記述する上で不便な制約のように思えるかもしれないが，回避策はある．例えば，アルファベットに a と $\overline{a}$ のような対を用意しておいて，$q \xrightarrow{a} q'$ と $q \xrightarrow{\overline{a}} q'$ で入力と出力を区別するなどの策が考えられる．⊣

LTS $\mathcal{M} = (Q, \Delta)$ および $a \in \Sigma$，$q \in Q$ に対して，二つの写像

$$\mathsf{Post}_{\mathcal{M},a}, \mathsf{Pre}_{\mathcal{M},a} \colon Q \to \mathcal{P}\,Q$$

を，

$$\mathsf{Pre}_{\mathcal{M},a}\,q = \{q' \mid q' \xrightarrow{a} q\}, \quad \mathsf{Post}_{\mathcal{M},a}\,q = \{q' \mid q \xrightarrow{a} q'\}$$

で定義する（$\mathcal{M}$ が明らかな場合はこれを省略して，Pre_a や Post_a と書くことにする）．すなわち，

- $\mathsf{Pre}_{\mathcal{M},a}\,q$ は，a によって q へと遷移可能な状態全体の集合，
- $\mathsf{Post}_{\mathcal{M},a}\,q$ は，q から a によって遷移可能な状態全体の集合

である．

◇**非決定性**　LTS では，$\mathsf{Post}_a\,q$ が空集合 $\emptyset$ となったり，二元以上の集合となることがある．すなわち，$q \xrightarrow{a} q'$ となる q' が存在しなかったり，$q \xrightarrow{a} q'$ かつ $q \xrightarrow{a} q''$ となるような二つの相異なる状態 q', q'' が存在することがあり得る．このように，$\mathsf{Post}_a\,q$ が空集合あるいは二元以上の集合となるような $a \in \Sigma$ および $q \in Q$ が存在するような LTS は，**非決定的**（nondeterministic）であるという．

▷ **注意 1.1.3**　非決定的な LTS をどう解釈すべきかは，記述対象のシステムによって異なる．例えば，$\mathsf{Post}_a\,q = \emptyset$ の場合は，「システムが状態 q であるときにイベント a は起こりえない」と解釈すべき場合もあるし，「システムが状態 q であるときにイベント a が起こったら，システムが破綻する」と解釈すべき場合もあるだろう．また，$\mathsf{Post}_a\,q = \{q', q''\}$（$q' \neq q''$）の場合も，絶対にこれという解釈があるわけでもないが，たいていの場合は「q' に遷移することも，q'' に遷移することも，どちらも起こりえる」と解釈する．⊣

非決定的でない LTS は，決定的であるという．決定的な LTS では，どの状態のときにどの $a \in \Sigma$ が生じても，次の状態が必ず存在して，しかも唯一に定まる．

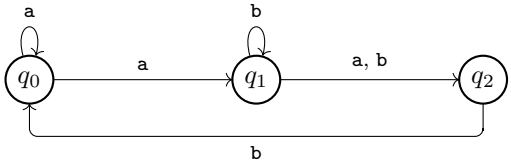
▷ **注意 1.1.4**　非決定的な LTS は，いわゆるべき集合構成 (powerset construction)[*2] によって，決定的な LTS に変換することができる．したがって，理論的には決定的な LTS のみを考えれば良いということもできる．しかし，本稿ではべき集合構成は扱わず，非決定的な LTS は非決定的なまま扱う手法を考える．というのも，今は具体的に述べることは難しいのだが，本稿の主題となる双模倣の理論は，非決定的なシステムが非決定的であるからこそわかる機微について論じるためのモノであると言っても良いからである．双模倣の理論は，非決定的なシステムを非決定的なまま扱うときにこそ，真価を発揮するのである．⊣

◇**状態遷移図**　LTS $\mathcal{M} = (Q, \Delta)$ は，形式的にいえば，Q を頂点の集合，Δ をラベル付き辺の集合とするラベル付き有向グラフである．本稿では，LTS をラベル付き有向グラフとして図で表し，この図を**状態遷移図**ということにする．

例えば，$\Sigma = \{\mathtt{a}, \mathtt{b}\}$ 上の LTS $\mathcal{M} = (Q, \Delta)$ を，

$$Q = \{q_0, q_1, q_2\}, \quad \Delta = \left\{ \begin{array}{l} (q_0, \mathtt{a}, q_0), (q_0, \mathtt{a}, q_1), (q_1, \mathtt{a}, q_2), \\ (q_1, \mathtt{b}, q_1), (q_1, \mathtt{b}, q_2), (q_2, \mathtt{b}, q_0) \end{array} \right\}.$$

で定義したとする．本稿では，この $\mathcal{M}$ の状態遷移図を，以下のようなラベル付き有向きグラフとして表す．



1.1.2　LTS で記述されるシステムの例

以下，簡単な「システム」を実際に LTS を用いて記述した例を，いくつか述べる．

▶**例 1.1.5**（**スタック**）　深さ 2 の**スタック**（stack）を考える．スタックには，0, 1 のいずれかの値を積んでいく．スタックには，一番上の値を取り出す**ポッ**

[*2] 形式言語理論の教科書において，非決定性オートマトン（NFA）を決定性オートマトン（DFA）へ変換する手法として紹介されていることが多い．

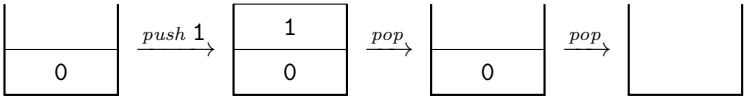


図 1.1 深さ 2 のスタックの状態遷移例

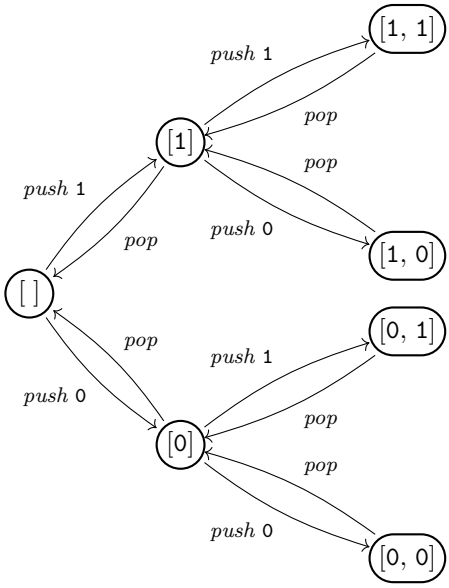


図 1.2 深さ 2 のスタックを表す LTS の例

プ（pop）および，一番上の値に値を押し入れる**プッシュ**（push）の二つの操作がある．このスタックを LTS で表現することを考えよう．

まず，アルファベット Σ として，

$$\Sigma = \{pop, push\ 0, push\ 1\}$$

を考える．ここで，$push\ i$ は，値 $i \in \{0, 1\}$ をスタックの一番上に押し入れる操作を表すとする．スタックとは，このアルファベットによって，状態遷移をしていくシステムだと思うことにする．スタックの状態遷移の例を，図 1.1 に示す．

このシステムを表す LTS の状態遷移図の例を，図 1.2 に示す．状態 $[b_0, \ldots, b_i]$ は，スタックが上から順に値 $b_i, \ldots, b_0$ が積まれている状態であ

ることを表している[*3]. ⊣

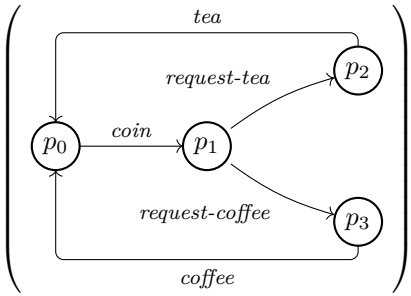
▷ **注意 1.1.6**　例 1.1.5 のスタックは,

- 要素が空の状態からポップしようとした場合,
- 要素が満杯の状態でさらに要素をプッシュした場合（すなわち，スタックオーバーフローを起こした場合）

の振る舞いについて，どちらも気にしていない．例えば後者については，*stack_overflow* などの新たな状態を設けて，要素数が 2 の状態からプッシュすれば状態 *stack_overflow* へと遷移することにすれば，こうしたシステムの挙動を表現することもできるだろう． ⊣

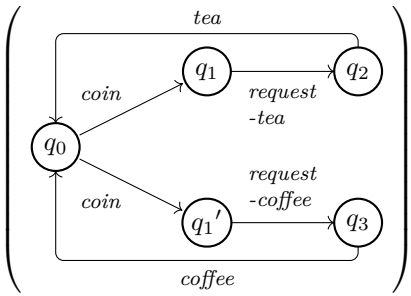
▶**例 1.1.7**　ワンコインでコーヒーとお茶を購入できる自動販売機があったとする．この自動販売機の振る舞いを，LTS を用いて表すことを考えよう．ここでは，次のような二つの LTS $\mathcal{M}_0, \mathcal{M}_1$ を考えることにする．

$\mathcal{M}_0 =$



,

$\mathcal{M}_1 =$



LTS $\mathcal{M}_0$ は，状態 p_0 から *coin* で決定的に p_1 へと遷移し，*request-tea* か *request-coffee* に応じて，*tea* か *coffee* のいずれかを出力する．一方，LTS $\mathcal{M}_1$

[*3] もちろん状態の名前は何でも良いので，これまでと同じように $q_0, q_1, \ldots$ のような名前を用いてもかまわない．が，ここでは状態を直観的に把握できるような状態集合を用意しただけである．

は，状態 q_0 から *coin* で非決定的に q_1 か q_1' へと遷移し，

- q_1 に遷移した場合は *request-tea* のみを受け付けて *tea* を，
- $q_1{}'$ に遷移した場合は *request-coffee* のみを受け付けて *coffee* を

出力する．二つの LTS の違いは，端的に言えば *tea* と *coffee* のどちらを出力するかが決まるタイミングが，コインを入れる前か後かということである． ⊣

> **演習 1**(B)　例 1.1.5 にならって，2 値（0 または 1）を保存する深さ 2 のキューを表す LTS を作れ．ただし，アルファベットは
>
> $$\Sigma = \{enqueue\ i \mid i \in \{0, 1\}\} \cup \{dequeue\}$$
>
> とし，*enqueue* i はリストの先頭に i を追加し，*dequeue* はリストの末尾の要素を取り出す操作を表す．

1.1.3　跡同値とグラフ同型

LTS の二つの状態があったとき，同じ振る舞いをするかどうかについて議論したい．これによって，二つのシステムの等しさについて議論することができるようになる．

LTS の状態の「等しさ」を表す概念はいくつかあり，

- 跡同値（trace equivalence）
- 双模倣性（bisimulality）
- LTS 同型（LTS isomorphism）

などがある．このうち，本稿での主題は双模倣性であり，「システムの振る舞いの等しさを記述する」という観点からは最もよく用いられるものである．双模倣性については次節から詳しく述べることにし，本節では，残る跡同値とグラフ同型について簡単に説明しておく．

◇**跡同値**　$q \in Q$ を始とする**跡**とは，Σ の列 $a_0, \ldots, a_{n-1}$ であって，或る $q_1, \ldots, q_n \in Q$ が存在して

$$q \xrightarrow{a_0} q_1 \xrightarrow{a_1} q_2 \xrightarrow{a_2} \cdots \xrightarrow{a_{n-1}} q_n$$

となるものである．本稿では，$a_0, \ldots, a_{n-1}$ が q を始とする跡であることを，

$$q \vdash a_0 \cdots a_{n-1}$$

と書くことにする．

▶**定義 1.1.8**（**跡同値**）アルファベット Σ 上の二つの LTS $\mathcal{M}_0 = (P, \Delta_P)$ および $\mathcal{M}_1 = (Q, \Delta_Q)$ を考える．$p \in P$ と $q \in Q$ が**跡同値**（trace equivalence）であるとは，任意の $a_0, \ldots, a_{n-1} \in \Sigma$ に対して，

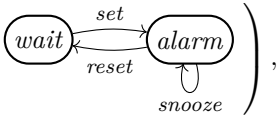
$$\forall (p_0, p_1 \in P).\ p \vdash a_0 \cdots a_{n-1} \iff q \vdash a_0 \cdots a_{n-1}$$

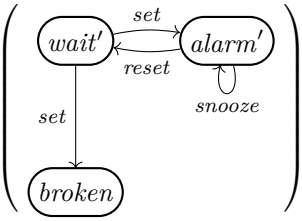
となることである．

▶**例 1.1.9**　例 1.1.7 の二つの LTS において，p_0 と q_0 は跡同値である．このことは，跡同値という概念は，`tea` と `coffee` **のどちらにするかが決まるタイミングを区別しない**ということを意味する．

より一般的に言えば，非決定的な動作のあるシステムにおいて，状態の等しさとして跡同値の概念を用いると，非決定的な選択のうち一つでも跡が等しくなるようなものがあれば，その二つのシステムの状態は等しいと看做してしまうということである．⊣

演習 2(**A**)　以下の二つの LTS $\mathcal{M}_0, \mathcal{M}_1$ を考える．

$\mathcal{M}_0 =$ 

，　$\mathcal{M}_1 =$ 

$wait$ と $wait'$ は跡同値か否か？

▷ **注意 1.1.10**　システムの状態の等しさを跡同値で定義することは，形式言語理論における非決定性オートマトンの等しさを，それらが受理する言語の等しさによって定義するのと極めて似ている．例えば，例 1.1.7 の二つの LTS を，それぞれ

- $\mathcal{M}_0$ を始状態が p_0，終状態の集合を $\{p_0\}$ としたオートマトン，
- $\mathcal{M}_1$ を始状態が q_0，終状態の集合を $\{q_0\}$ としたオートマトン

と看做すと，二つが受理する言語は同じ言語である[*4]．

このような事情から，跡同値のことを，**言語等価性**（language equivalence）ということもある．⊣

[*4] もちろん，始状態および終状態をどのようにとるかによって，受理する言語は異なってくるため，この例はあくまでも「終状態をうまく選べば，受理する言語が同じ DFA とみなせる」という話に過ぎない．しかし，跡同値な状態をもつ LTS であれば，その状態を始状態として，うまく終状態を上手く選べば同じようにできるはずである

◇**LTS 同型**　次に，LTS 同型について述べる．そのための準備として，LTS 準同形について述べる

▶**定義 1.1.11**　アルファベット Σ 上の二つの LTS $\mathcal{M}_0 = (P, \Delta_P)$ および $\mathcal{M}_1 = (Q, \Delta_Q)$ を考える．

(1)　$f\colon P \to Q$ が **LTS 準同形写像**（LTS homomorphism）あるいは単に LTS 準同形，準同形であるとは，

$$p_0 \xrightarrow{a} p_1 \implies f\,p_0 \xrightarrow{a} f\,p_1$$

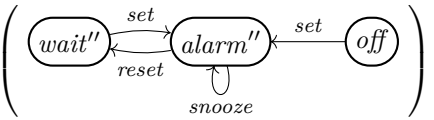
が成り立つことである．$f\colon P \to Q$ が準同形であるとき，$f\colon P \to Q$ の意味で $f\colon \mathcal{M}_0 \to \mathcal{M}_1$ と書くこともある．

(2)　準同形 $f\colon \mathcal{M}_0 \to \mathcal{M}_1$ に対して，f が**同型**（isomorphism）であるとは，f が全単射となっている（すなわち，逆写像 f^{-1} をもつ）ことである．また，同型 $f\colon \mathcal{M}_0 \to \mathcal{M}_1$ が存在するとき，$\mathcal{M}_0$ と $\mathcal{M}_1$ は**同型**（isomorphic）であるという．

▶**補題 1.1.12**　LTS 同型 $f\colon \mathcal{M}_0 \to \mathcal{M}_1$ に対して，その逆写像 $f^{-1}\colon \mathcal{M}_1 \to \mathcal{M}_0$ は準同形である．

証明　$\mathcal{M}_1 = (Q, \Delta_Q)$ に対して，$q_0, q_1 \in Q$ かつ $q_0 \xrightarrow{a} q_1$ とする．このとき，$f \circ f^{-1} = \mathrm{id}_Q$ に注意すれば，$f\,(f^{-1}\,q_0) \xrightarrow{a} f\,(f^{-1}\,q_1)$．さらに f は準同形だから，$f^{-1}\,q_0 \xrightarrow{a} f^{-1}\,q_1$ がしたがう．　□

演習 3(A)　演習 2 の二つの LTS $\mathcal{M}_0, \mathcal{M}_1$ に加え，さらに次の LTS $\mathcal{M}_2$ を考える：

$\mathcal{M}_2 =$ 

.

以下の問いに答えよ．

(1)　$\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2$ の状態のうち，*off* と跡同値な状態をすべて挙げよ．

(2)　$\mathcal{M}_0$ と $\mathcal{M}_2$ は同型でないことを確かめよ．

§1.2　ラベル付き遷移系の双模倣

ラベル付き遷移系の双模倣について述べる．

▶**定義 1.2.1（模倣，双模倣，双模倣性）**　二つの LTS $\mathcal{M}_0 = (P, \Delta_P)$, $\mathcal{M}_1 = (Q, \Delta_Q)$ を考える．

（1）　二項関係 $R \subseteq P \times Q$ が $\mathcal{M}_0$ から $\mathcal{M}_1$ への**模倣**（simulation）であるとは，任意の $(p, q) \in R$ および $a \in \Sigma$ に対して，

$$\forall (p' \in P).\ p \xrightarrow{a} p' \implies \exists (q' \in Q).\ q \xrightarrow{a} q' \wedge (p', q') \in R$$

が成り立つことである．$\mathcal{M}_0$ から $\mathcal{M}_1$ への模倣全体の集合を，$\mathsf{Sim}_{\mathcal{M}_0, \mathcal{M}_1}$ と書く．

（2）　二項関係 $R \subseteq P \times Q$ が $\mathcal{M}_0$ と $\mathcal{M}_1$ の**双模倣**（bisimulation）であるとは，任意の $(p, q) \in R$ および $a \in \Sigma$ に対して，以下の二つの条件が同時に成り立つことである．

- $\forall (p' \in P).\ p \xrightarrow{a} p' \implies \exists (q' \in Q).\ q \xrightarrow{a} q' \wedge (p', q') \in R$
- $\forall (q' \in Q).\ q \xrightarrow{a} q' \implies \exists (p' \in P).\ p \xrightarrow{a} p' \wedge (p', q') \in R$

$\mathcal{M}_0$ と $\mathcal{M}_1$ の双模倣全体の集合を，$\mathsf{BiSim}_{\mathcal{M}_1, \mathcal{M}_0}$ と書く．

あきらかに双模倣は模倣であるから，$\mathsf{BiSim}_{\mathcal{M}_0, \mathcal{M}_1} \subseteq \mathsf{Sim}_{\mathcal{M}_0, \mathcal{M}_1}$ である．

◇**最大模倣，最大双模倣**　以下しばらく，二つの LTS $\mathcal{M}_0 = (P, \Delta_P)$, $\mathcal{M}_1 = (Q, \Delta_Q)$ を固定する．

二つの二項関係 $\lesssim_{\mathcal{M}_0, \mathcal{M}_1}, \sim_{\mathcal{M}_0, \mathcal{M}_1} \subseteq P \times Q$ を，それぞれ

$$\lesssim_{\mathcal{M}_0, \mathcal{M}_1} = \bigcup \mathsf{Sim}_{\mathcal{M}_0, \mathcal{M}_1}, \quad \sim_{\mathcal{M}_0, \mathcal{M}_1} = \bigcup \mathsf{BiSim}_{\mathcal{M}_0, \mathcal{M}_1}$$

で定める．また，二項関係 $\gtrsim_{\mathcal{M}_0, \mathcal{M}_1}$ を，$\gtrsim_{\mathcal{M}_0, \mathcal{M}_1} = \lesssim_{\mathcal{M}_1, \mathcal{M}_0}{}^{-1}$ とする．すなわち，

$$q \gtrsim_{\mathcal{M}_0, \mathcal{M}_1} p \iff p \lesssim_{\mathcal{M}_1, \mathcal{M}_0} q.$$

さらに，$\mathcal{M}_0, \mathcal{M}_1$ が明らかな場合には，これを省略して，単に $\lesssim, \gtrsim, \sim$ などと略記する．

▶**命題 1.2.2**　以下が成り立つ．

（1）　$\lesssim_{\mathcal{M}_0, \mathcal{M}_1} \in \mathsf{Sim}_{\mathcal{M}_0, \mathcal{M}_1}$.

（2）　$\sim_{\mathcal{M}_0, \mathcal{M}_1} \in \mathsf{BiSim}_{\mathcal{M}_0, \mathcal{M}_1}$. ⊣

証明　（2）のみ示す．以下，$\sim_{\mathcal{M}_0, \mathcal{M}_1}$ を $\sim$ と略記する．

$p \sim q$ とする．すなわち，或る双模倣 $R \subseteq \sim$ が存在して，$(p, q) \in R$.

- $p' \in P$ かつ $p \xrightarrow{a} p'$ とする．R は双模倣より，或る $q' \in Q$ が存在して，

$q \xrightarrow{a} q'$ かつ $(p', q') \in R'$．したがって，$q \xrightarrow{a} q'$ かつ $p' \sim q'$．

- 同様に，$q' \in Q$ かつ $q \xrightarrow{a} q'$ とすれば，或る $p' \in P$ が存在して，$p \xrightarrow{a} p'$ かつ $p' \sim q'$．　□

この命題からわかるように，$\lesssim_{\mathcal{M}_0, \mathcal{M}_1}$ は順序集合 $(\mathsf{Sim}_{\mathcal{M}_0, \mathcal{M}_1}, \subseteq)$ の最大元になっているし，$\sim_{\mathcal{M}_0, \mathcal{M}_1}$ は順序集合 $(\mathsf{BiSim}_{\mathcal{M}_0, \mathcal{M}_1}, \subseteq)$ の最大元になっている．$\lesssim_{\mathcal{M}_0, \mathcal{M}_1}$ は**最大模倣**（greatest simulation）といい，$\sim_{\mathcal{M}_0, \mathcal{M}_1}$ は，$\mathcal{M}_0$ と $\mathcal{M}_1$ の間の**双模倣性**（bisimilality）および**最大双模倣**（greatest bisimilation）という．

▶**命題 1.2.3**　以下が成り立つ．

(1)　$\sim_{\mathcal{M}_0, \mathcal{M}_1} = \sim_{\mathcal{M}_1, \mathcal{M}_0}{}^{-1}$．

(2)　$\sim_{\mathcal{M}_0, \mathcal{M}_1} \subseteq \lesssim_{\mathcal{M}_0, \mathcal{M}_1} \cap \gtrsim_{\mathcal{M}_0, \mathcal{M}_1}$．　⊣

証明　(1)　容易に確かめられる．

(2)　以下，$\mathcal{M}_0$ と $\mathcal{M}_1$ を略記する．$p \sim q$ とする．このとき，或る双模倣 $B \subseteq P \times Q$ が存在して，$(p, q) \in B$ である．双模倣は模倣でもあるから，$(p, q) \in B \in \mathsf{Sim}$．したがって，$p \lesssim q$．同様に $p \gtrsim q$（すなわち $q \lesssim p$）であることも容易に示せる．　□

▶**例 1.2.4**　以下の二つの LTS $\mathcal{M}_0, \mathcal{M}_1$ を考える：

$$\mathcal{M}_0 = \left(\begin{array}{ccccc} & & & & p_1 \\ & & \overset{\mathtt{a}}{\nearrow} & & \\ p_0 & & & & \\ & & \underset{\mathtt{a}}{\searrow} & & \\ & & & & p_2 \xrightarrow{\mathtt{b}} p_3 \end{array} \right),$$

$$\mathcal{M}_1 = \left(q_0 \xrightarrow{\mathtt{a}} q_2 \xrightarrow{\mathtt{b}} q_3 \right).$$

このとき，$p_0 \lesssim q_0$ かつ $q_0 \gtrsim p_0$ である．実際，

$$S_0 = \{(p_0, q_0), (p_1, q_2), (p_2, q_2), (p_3, q_3)\},$$
$$S_1 = \{(q_0, p_0), (q_1, p_1), (q_3, p_3)\}$$

とすれば，S_0 は $\mathcal{M}_0$ から $\mathcal{M}_1$ への模倣であり，S_1 は $\mathcal{M}_1$ から $\mathcal{M}_0$ への模倣である．したがって $(p_0, q_0) \in S_0$ だから $p_0 \lesssim q_0$ であり，$(q_0, p_0) \in S_1$ だから $p_0 \gtrsim q_0$．

しかし，$p_0 \not\sim q_0$ である．実際，背理法の仮定として $p_0 \sim q_0$ だったとしてみよう．まず，$p_0 \xrightarrow{\mathtt{a}} p_1$ だから，或る $q_0 \xrightarrow{\mathtt{a}} q'$ かつ $p_1 \sim q'$．ところで，$q_0 \xrightarrow{\mathtt{a}} q'$

となる q' は，q_2 しかあり得ないから，$q' = q_2$．したがって，$p_1 \sim q_2$ が成り立つ．しかし，$q_2 \xrightarrow{\mathtt{b}} q_3$ であるが，$p_1 \xrightarrow{\mathtt{b}} p'$ となる $p' \in P$ は存在しない．これは矛盾である．

なお，この例は，命題 1.2.3 (2)の逆方向の反例になっている．　⊣

▶**例 1.2.5**　次の二つの LTS $\mathcal{M}_0, \mathcal{M}_1$ を考える：

$$\mathcal{M}_0 = \left(\begin{array}{ccccc} p_0 & \xrightarrow{\mathtt{a}} & p_1 & \underset{\mathtt{a}}{\overset{\mathtt{b}}{\rightleftarrows}} & p_2 \end{array} \right),$$

$$\mathcal{M}_1 = \left(\begin{array}{ccc} q_1 & \underset{\mathtt{a}}{\overset{\mathtt{b}}{\rightleftarrows}} & q_2 \end{array} \right).$$

このとき，三つの関係

$$R_0 = \emptyset, \quad R_1 = \{(p_1, q_1), (p_2, q_2)\},$$
$$R_2 = \{(p_0, q_2), (p_1, q_1), (p_2, q_2)\}$$

はいずれも $\mathcal{M}_0$ と $\mathcal{M}_1$ の双模倣であり，特に R_2 は最大双模倣である．　⊣

演習 4$^{(\mathsf{A})}$　演習 3 の三つの LTS $\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2$ を考える．以下のそれぞれは成り立つか否か？（証明はしなくてよい）

(1)	$wait \lesssim wait'$	(2)	$wait \lesssim wait''$	(3)	$wait \lesssim off$
(4)	$wait' \lesssim wait$	(5)	$wait' \lesssim wait''$	(6)	$wait' \lesssim off$
(7)	$wait'' \lesssim wait$	(8)	$wait'' \lesssim wait'$	(9)	$wait'' \lesssim off$
(10)	$off \lesssim wait$	(11)	$off \lesssim wait'$	(12)	$off \lesssim wait''$

演習 5$^{(\mathsf{A})}$　例 1.2.5 の LTS $\mathcal{M}_0$ と $\mathcal{M}_0$ の間の双模倣をすべて求めよ．また，最大双模倣はどれか（証明はしなくてよい）．

演習 6$^{(\mathsf{A})}$　図 1.1 の LTS $\mathcal{M}$ に対して，$\mathcal{M}$ と $\mathcal{M}$ の間の最大双模倣を求めよ（証明はしなくてよい）．

演習 7$^{(\mathsf{B})}$　LTS $\mathcal{M} = (Q, \Delta)$ に対して，$\mathcal{M}$ と $\mathcal{M}$ の間の最大双模倣 $\sim$ は，Q 上の同値関係であることを示せ．

§1.3　文献ノートおよび追加の演習問題

1.3.1　文献ノート

本章の執筆に際しては，教科書 [San11] の第 1 章を参考にした．この教科書は，双模倣と余帰納法についての教科書である．

LTS が最初に導入されたのは，Plotkin によるノート [Plo81]（このリプリント版が [Plo04] である）であると言われている．このノートは，簡単なプログラミング言語の意味論を，状態システムによって与えることを目標にしており，有限状態オートマトンやペトリネットとともに LTS による意味の記述を検討している．また，本章のもう一つのテーマである双模倣の概念は，ω 語を受け付けるオートマトンについての研究の文脈で現れた [Par81] のが初出であると言われる．

状態システムや余帰納法全般についての歴史については，[San$^+$11] の第 1 章も参考になる．

◇追加の演習問題

参考文献

[Par81] David Park. Concurrency and automata on infinite sequences. in *Proceedings of 5th GI Conference on Theoretical Computer Science*. Vol. 104 of *Lecture Notes in Computer Science*. pp. 167–183. Springer Berlin Heidelberg. 1981.

[Plo81] Gordon Plotkin. A structural approach to operational semantics. Report DAIMI FN-19, Aarhus University. 1981.

[Plo04] Gordon Plotkin. A structural approach to operational semantics. *Journal of Logic and Algebraic Programming*. Vol. 60, pp. 17–139. 2004.

[San11] Davide Sangiorgi. *Introduction to Bisimulation and Coinduction*. Cambridge University Press. 2011.

[San+11] Davide Sangiorgi and Jan Rutten eds.. *Advanced Topics in Bisimulation and Coinduction*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press. 2011.