システムの振る舞いと双模倣性

郡 茉友子

1　システムの正しさの保証とシステムの等価性

プログラムなどのシステムが、期待する仕様・性質を満たすかどうかを検証することを考えましょう。例えば、作ったプログラムが意図した動作をするか、バグがないかなどを確認したいことがあります。コードレビューやテストを行うことである程度の信頼を得ることができますが、全ての挙動を網羅的に確認することは一般に困難です。特に、プログラムが大規模で複雑になるほど、手作業やテストで全体の正しさを保証することは難しくなります。そこで、システムの正しさを数学的に保証することを目的として研究されているのが、**形式検証**という分野です。形式検証では、システムの動作や計算を数理モデルとして形式的に表し、その数理モデルが与えられた仕様を満たすことを数学的に示すことを目指します。

1.1　システムを数理モデルで表す

計算を扱う数理モデルは様々ありますが、操作的意味論と呼ばれる枠組みでは、計算の実行を「内部状態 (例えば変数の値やプログラムカウンタ) が変化しながら進行する過程」と捉え、状態遷移 (遷移システム) として記述します。そのようなモデルのなかでも、今回はラベル付き遷移システム (LTS) と呼ばれる、状態の遷移とそれに付随する動作やイベントをラベルとして持つ遷移システムを考えます。定義自体はシンプルですが、計算機科学において重要なモデルの一つで、特に複数のプロセスが同時に動く**並行システム**の振る舞いを表すための基本的なモデルとしてよく用いられています。

定義 1.1. ラベル付き遷移システム (labelled transition system; LTS) とは、三つ組

$$L = (S, A, \to)$$

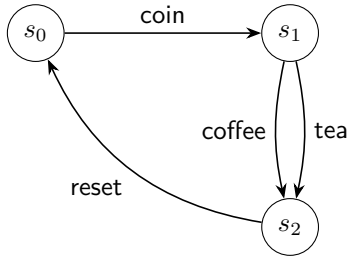
であって、

- 状態の集合 S,
- ラベルの集合 A,
- $\to \subseteq S \times A \times S$,

からなるものをいう。$(s, a, t) \in \to$ のとき、$s \xrightarrow{a} t$ と書き、「状態 s からラベル a で 状態 t に遷移で

きる」という。

例 1.2 (自動販売機の簡単なモデル)**.** 状態 s_0 を「待機中」、s_1 を「硬貨投入済み」、s_2 を「商品提供済み」を表す状態とし、ラベルを $\mathsf{coin}, \mathsf{tea}, \mathsf{coffee}, \mathsf{reset}$ とします。非常に簡易的ではありますが、次のような LTS でコーヒーとお茶を買うことができる自動販売機の挙動を表すことができます。



状態は内部状態を表し、ラベルは外部から観測できる動作やイベントを表していると考えられます。例えば、$s_0 \xrightarrow{\mathsf{coin}} s_1$ は、待機中の状態で硬貨が投入されると、商品を選べる状態に移ることを表しています。

このようにシステムを抽象化して数理モデルで表し、そのモデルが性質を満たすとはどういうことかを定義することで、数学的に検証することが可能になります。特に、**モデル検査** [1, 4, 2] と呼ばれる手法では、システムを LTS のような遷移システムで表し、論理式で表された仕様を満たすかどうかを検査します。

1.2 システムの等価性

今回特に考えたいのは、2 つのシステムが同じように振る舞うことを表す**システムの等価性**です。システムの等価性は非常に基本的な概念で、様々な場面で使うことができます。

例えば、異なる実装をした 2 つのプログラムが同じ動作をするかどうかを等価性を調べることで検査することもできますし、最適化前後のプログラムが同じ振る舞いをするかどうかを調べて最適化の正当性を保証することもできます。

また、等価性はモデル検査の前処理として利用できることもあります。モデル検査では、遷移システムの状態数が大きくなると、探索すべき空間が急激に増大することで検査が長時間終わらなくなるという**状態爆発**とよばれる現象が起こることがあります。これに対し、検証したい性質の観点から区別する必要のない状態を等価な状態としてまとめて、より小さなシステムに変換をしてから検査をすることで効率的なモデル検査を行える場合があります。

1.3 この講義の構成と目的

本講義では、以上のような背景のもとで、LTS の等価性を題材として、数学が計算機科学、特に形式検証においてどのように役に立つのかを見ていきたいと思います。無限的な対象も計算機に扱える有限的な記述で表したい、検証をするために計算機が自動で探しやすい証拠を得たい、などの計算機科学特有の動機に対し、数学がどのように現れ、どのような役割をするのかを、LTS を対象とすることでプログラムの具体的な構文の詳細などを脇において説明できればと思います。

LTS の等価性には、どのような振る舞いに着目したいか、何を等価として扱いたいかによって様々な概念があるほか、グラフ的な視点から得られる**振る舞いの等価性**と、同じ性質を満たすかという視点から得られる**論理的等価性**などがあります。特に振る舞いの等価性としてトレース等価性と双模倣性を、論理的等価性として Hennessy-Milner 論理による等価性を紹介します。そのなかで、束論的不動点を使った有限と無限の橋渡しや、異なる等価性の概念や特徴づけが検証の際にどのように役立つかなどを説明していく予定です。

2　トレース等価性

まずは LTS の振る舞いを比較するための基本的な等価性として、トレース等価性を見ていきましょう。実装の中身までは見ずに、外部から観測できる振る舞いが一致するかどうかを考えたいときなどに便利な概念で、実行によって現れうるラベル列が等しいかどうかを比較します。

2.1　有限トレース等価性

以降、LTS $L = (S, A, \to)$ を一つ固定します。その中で、2 つの状態が等価とはどういうことかを考えていきます。異なる 2 つの LTS の状態を比較したい場合も、それらの直和を取って 1 つの LTS にまとめてしまえば、その直和の中での 2 つの状態の比較に言い換えられます。

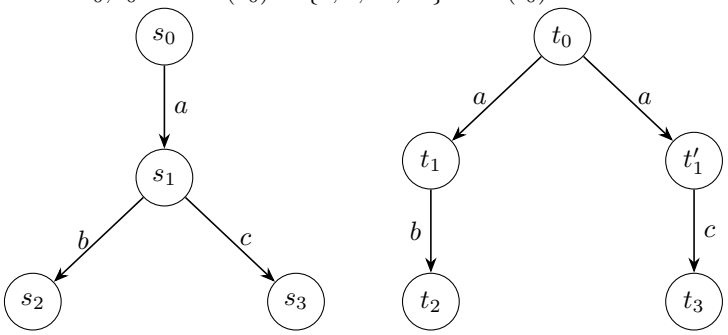
ラベルの有限列全体の集合を A^* で表し、空列を ε と書きます。遷移の列 $s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots \xrightarrow{a_n} s_n$ を**実行**とよびます。状態 $s, t \in S$, $w = a_1 \cdots a_n \in A^*$ に対して、s から t まで w で遷移する実行 $s = s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} s_n = t$ が存在するとき、$s \xrightarrow{w} t$ と書きます。特に、$s \xrightarrow{\varepsilon} s$ は常に成り立ちます。

定義 2.1 (有限トレース等価性)**.** 状態 $s \in S$ から出発して実行できる有限ラベル列全体を

$$\mathsf{Tr}(s) = \{w \in A^* \mid \text{ある } t \in S \text{ が存在して } s \xrightarrow{w} t\}$$

と書き、s の**有限トレース集合**という。2 つの状態 s, t が**有限トレース等価**であるとは、$\mathsf{Tr}(s) = \mathsf{Tr}(t)$ が成り立つことをいう。

例 2.2. 次の LTS の状態 s_0, t_0 は、$\mathsf{Tr}(s_0) = \{\varepsilon, a, ab, ac\} = \mathsf{Tr}(t_0)$ なので有限トレース等価です。



有限トレース集合は、オートマトン理論でいう**言語**として見ることもできます。LTS $(S, A, \to)$

に S の部分集合 $I, \mathrm{Acc} \subseteq S$ を指定すると、**非決定性オートマトン**が得られます。よく S も A も有限であることを課して、非決定性有限オートマトン (NFA) といったりします。I が初期状態の集合、Acc が受理状態の集合を表していて、トレースの始まりと終わりをこれらの集合で制限することを考え、$s \in I$ から始まり $t \in \mathrm{Acc}$ で終わる実行から得られるトレースを集めた $\bigcup_{s\in I, t\in \mathrm{Acc}} \{w \mid s \xrightarrow{w} t\} \subseteq A^*$ を、**受理言語**といいます。有限トレース集合 $\mathsf{Tr}(s)$ は、この受理言語の特別な場合 ($I := \{s\}, \mathrm{Acc} := S$) として表すことができるため、有限トレース等価性は言語等価性とみることができます。

後ほど第 4 節で詳しくみますが、オートマトンの重要な点の一つは、有限的なデータによって、無限集合になりうる言語を表すことができることです。つまり、オートマトンは言語を表すための有限的な記述を与えることができ、オートマトンが与えらると、遷移を有限回繰り返して到達できる受理状態を調べることで、受理言語を得ることができます。

2.2 トレース等価性

多くの検証問題では、有限回で終了する実行だけでなく、無限に続く実行も重要です。例えば、「無限に何度もリクエストを処理する」といった性質は、無限実行に関する性質です。

ラベルの無限列の集合を A^ω で表します。$w = a_0a_1a_2\cdots \in A^\omega$ に対して、$s \xrightarrow{w}$ とは、$s_1, s_2, \ldots \in S$ が存在して $s \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \cdots$ となることをいいます。

定義 2.3 (トレース等価性)**.** 状態 $s \in S$ から出発して実行できる無限ラベル列全体を

$$\mathsf{Tr}^\omega(s) = \{w \in A^\omega \mid s \xrightarrow{w}\}$$

と書き、s の**無限トレース集合**という。有限または無限トレース集合を

$$\mathsf{Tr}^\infty(s) = \mathsf{Tr}(s) \cup \mathsf{Tr}^\omega(s)$$

と書き、単に s の**トレース集合**と呼ぶ。2 つの状態 s, t が $\mathsf{Tr}^\infty(s) = \mathsf{Tr}^\infty(t)$ を満たすとき、s と t は**トレース等価**であるという。

有限トレース等価性とトレース等価性は同じでしょうか？トレース等価ならば有限トレース等価、つまり

$$\mathsf{Tr}^\infty(s) = \mathsf{Tr}^\infty(t) \quad \Longrightarrow \quad \mathsf{Tr}(s) = \mathsf{Tr}(t)$$

は明らかですが、逆は一般には成り立ちません。例えば、一方の状態は ラベル a のループを持ち、もう一方の状態では任意の長さの有限 a 列を実行できるが無限に a が続く列 $aaa\cdots$ は実行できない、という LTS を考えます。このとき、有限トレース集合はどちらも $\{a^n \mid n \geq 0\}$ になりますが、前者には無限トレース $aaa\cdots$ が存在し、後者には存在しないため無限トレースは一致しません。

この例では、同じ状態とラベルに対して無限分岐があることが本質的に使われています。そこで、次の性質を考えます。

定義 2.4. LTS が *image-finite* であるとは、任意の状態 $s \in S$ とラベル $a \in A$ に対して、$\{t \in S \mid s \xrightarrow{a} t\}$ が有限集合であることをいう。

定理 2.5. LTS が image-finite のとき、任意の状態 $s, t \in S$ について、

$$\mathsf{Tr}(s) = \mathsf{Tr}(t) \quad \Longrightarrow \quad \mathsf{Tr}^{\infty}(s) = \mathsf{Tr}^{\infty}(t)$$

が成り立つ。つまり、有限トレース等価性とトレース等価性が等価になる。

証明 $\mathsf{Tr}(s) = \mathsf{Tr}(t)$ とする。任意の無限トレース $a_0 a_1 a_2 \cdots \in \mathsf{Tr}^{\omega}(s)$ をとる。S の無限列 $(t_i)_{i \in \mathbb{N}}$ で、各 i について $\{a_i \cdots a_n \mid n \geq i\} \cap \mathsf{Tr}(t_i)$ が無限であるようなものを以下のように構成する。

- $t_0 \coloneqq t$ とする。各 n について、$a_0 \cdots a_n$ は s の有限トレースである。仮定より $\mathsf{Tr}(s) = \mathsf{Tr}(t)$ なので、$a_0 \cdots a_n \in \mathsf{Tr}(t)$ が成り立つ。
- t_i が定義されているとき、以下のように t_{i+1} を定義する。image-finite 性より、$S' \coloneqq \{t' \in S \mid t_i \xrightarrow{a_i} t'\}$ は有限である。$\{a_i \cdots a_n \mid n \geq i\} \cap \mathsf{Tr}(t_i)$ が無限であることから、$\{a_{i+1} \cdots a_n \mid n \geq i{+}1,\ a_i a_{i+1} \cdots a_n \in \mathsf{Tr}(t_i)\} = \{a_{i+1} \cdots a_n \mid n \geq i{+}1\} \cap (\bigcup_{t' \in S'} \mathsf{Tr}(t'))$ は無限集合である。S' は有限集合であることから、ある $t' \in S'$ があって、$\{a_{i+1} \cdots a_n \mid n \geq i{+}1\} \cap \mathsf{Tr}(t')$ が無限集合である。そのような t' を t_{i+1} とする。

これを繰り返して定義した $(t_i)_{i \in \mathbb{N}}$ によって実行 $t = t_0 \xrightarrow{a_0} t_1 \xrightarrow{a_1} \cdots$ で、$a_0 a_1 a_2 \cdots \in \mathsf{Tr}^{\omega}(t)$ である。以上より $\mathsf{Tr}^{\omega}(s) \subseteq \mathsf{Tr}^{\omega}(t)$ が分かる。逆向きも同様であり、有限トレース部分は仮定より一致しているので、$\mathsf{Tr}^{\infty}(s) = \mathsf{Tr}^{\infty}(t)$ である。 □

上の証明では、image-finite 性を使って，無限に多くの有限 prefix を実行できる遷移先を一つずつ選びました。これは König の補題と呼ばれる、「有限分岐の無限木には無限につづくパスが存在する」という主張の証明をこの状況に合わせて展開したものになっています。

3 双模倣性

前節では、実行できるラベル列に注目してトレース等価性を定義しました。振る舞いを時間にそった一本の列で見ているので、線形時間的な見方をしているといえます。一方で、システムの振る舞いを考えるときには、単にどの実行列が可能かだけではなく、各状態でどのような分岐があるのかという分岐構造を見たいことがあります。例えば、例 2.2 では、状態 s_0 と t_0 はトレース等価でしたが、s_0 側ではラベル a の遷移の後に b の遷移も c の遷移も選べる状態に到達できるのに対し、t_0 側では a の遷移の時点で b の遷移のみ可能な状態か、c の遷移のみ可能な状態かに分かれていました。このような分岐構造の違いまで捉えられる等価性の一つとして、双模倣性を導入します。

双模倣性を定義するために、状態同士を対応付けるため関係を使います。ここで集合 S 上の ***(二項)* 関係**とは、部分集合 $R \subseteq S \times S$ のことです。$(x, y) \in R$ のとき、xRy とも書きます。

二項関係に対し、いくつかの基本的な操作を導入しておきます。まず、恒等関係を

$$\Delta_S = \{(s, s) \mid s \in S\}$$

で表します。また、関係 $R \subseteq S \times S$ の逆関係を

$$R^{-1} = \{(t, s) \mid sRt\}$$

で定め、さらに2つの関係 $R, Q \subseteq S \times S$ の合成を

$$R \circ Q = \{(s, t) \mid \text{ある } u \in S \text{ が存在して } sQu \text{ かつ } uRt\}$$

で定めます。

命題 3.1. 任意の二項関係 $R, Q \subseteq S \times S$ について、次が成り立つ。

1. $(R^{-1})^{-1} = R$,
2. $(R \circ Q)^{-1} = Q^{-1} \circ R^{-1}$.

3.1 双模倣関係・双模倣性

双模倣性は、2 つの状態が互いに遷移を真似できるかを表す性質です。これを定義するために、まずは片方向の模倣を表す関係を定義します。

定義 3.2. 二項関係 $R \subseteq S \times S$ が**模倣関係** (simulation) であるとは、任意の $s, s', t \in S$ と $a \in A$ について、$s \xrightarrow{a} s'$ かつ sRt ならば、ある $t' \in S$ が存在して $t \xrightarrow{a} t'$ かつ $s'Rt'$ が成り立つことをいう。

模倣関係は片方向の条件です。sRt であるとき、s の動きを t が模倣できることは要求しますが、逆に t の動きを s が真似できることは要求していません。双模倣関係では、この条件を両方向に要求します。

定義 3.3. 二項関係 $R \subseteq S \times S$ が**双模倣関係** (bisimulation) であるとは、R と R^{-1} がともに模倣関係であることをいう。言い換えると、任意の $s, t \in S$, $a \in A$ が sRt ならば次の二条件を満たすとき、R を双模倣関係という。

1. 任意の $s' \in S$ について、$s \xrightarrow{a} s'$ ならば、ある t' が存在して $t \xrightarrow{a} t'$ かつ $s'Rt'$ が成り立つ。
2. 任意の $t' \in S$ について、$t \xrightarrow{a} t'$ ならば、ある s' が存在して $s \xrightarrow{a} s'$ かつ $s'Rt'$ が成り立つ。

命題 3.4. 1. Δ_S は双模倣関係である。
2. R が双模倣関係ならば、R^{-1} も双模倣関係である。
3. R, Q が双模倣関係ならば、$R \circ Q$ も双模倣関係である。

双模倣関係は、2 つの状態が互いに模倣でき、同じ振る舞いをもつことを示すための「証拠」と思うことができます。これを使って、この証拠があるときに等価であるとして双模倣性を得ることができます。

定義 3.5 (双模倣性)**.** 状態 $s, t \in S$ が**双模倣** (bisimilar) であるとは、ある双模倣関係 $R \subseteq S \times S$ が存在して sRt が成り立つことをいい、$s \sim t$ と書く。関係 $\sim$ を**双模倣性** (bisimilarity) と呼ぶ。

定義から、双模倣性は以下のように書くことができます。

$$\sim = \bigcup\{R \subseteq S \times S \mid R \text{ は双模倣関係}\}$$

この双模倣性自体が双模倣関係であることは、以下の補題から示すことができます。

補題 3.6. $\{R_i\}_{i\in I}$ を双模倣関係の族とすると、$\bigcup_{i\in I} R_i$ も双模倣関係である。

証明 $s(\bigcup_i R_i)t$ かつ $s \xrightarrow{a} s'$ とする。このとき、ある $i \in I$ が存在して sR_it である。R_i は双模倣関係なので、ある t' が存在して $t \xrightarrow{a} t'$ かつ $s'R_it'$ である。したがって $s'(\bigcup_i R_i)t'$ である。逆向きも同様である。 □

したがって、$\sim$ が双模倣関係であり、任意の双模倣関係 R について $R \subseteq (\sim)$ が成り立つことから、以下の定理が成り立ちます。

定理 3.7. 双模倣性 $\sim$ は最大の双模倣関係である。

3.2 双模倣性とトレース等価性

双模倣性は分岐構造まで見ていて、途中の状態からの遷移の可能性まで等しいことを要求します。一方、トレース等価性は実行して外から観測できる列が等しいかどうかを見ていて、双模倣性よりも粗い等価性になっています。実際、以下のように双模倣であればトレース等価であることが成り立ちます。

命題 3.8. $s \sim t$ ならば $\mathsf{Tr}^\infty(s) = \mathsf{Tr}^\infty(t)$ である。

証明 $s \sim t$ とする。まず、無限トレースについて示す。任意に $(a_0a_1a_2\cdots) \in \mathsf{Tr}^\omega(s)$ をとる。このとき、無限実行 $s = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \cdots$ が存在する。$t = t_0$ とおく。仮定より $s_0 \sim t_0$ が成り立つ。自然数 $n \in \mathbb{N}$ に t_n が定義されていて $s_n \sim t_n$ が成り立つとき、$t_n \xrightarrow{a_n} t_{n+1}$ かつ $s_{n+1} \sim t_{n+1}$ を満たす $t_{n+1} \in S$ が存在する。これを各 n について繰り返すことで、無限実行 $t = t_0 \xrightarrow{a_0} t_1 \xrightarrow{a_1} t_2 \cdots$ が得られる。よって $(a_0a_1a_2\cdots) \in \mathsf{Tr}^\omega(t)$ である。有限トレースに対しても同様に示すことができる。 □

逆は一般には成り立ちません。例えば、例 2.2 の s_0 と t_0 はトレース等価ですが双模倣ではありません。一方で、同じ状態とラベルで進める先が高々一つであるような LTS では、トレース等価性から双模倣性が従います。

命題 3.9. $L = (S, A, \to)$ が決定的であるとする。すなわち、任意の $s \in S$ と $a \in A$ について、$s \xrightarrow{a} u$ と $s \xrightarrow{a} v$ が成り立つなら $u = v$ であるとする。このとき以下が成り立つ。

$$\mathsf{Tr}(s) = \mathsf{Tr}(t) \quad \Longrightarrow \quad s \sim t.$$

証明 関係 $R \subseteq S \times S$ を $R = \{(x, y) \mid \mathsf{Tr}(x) = \mathsf{Tr}(y)\}$ で定める。R が双模倣関係であることを示す。xRy かつ $x \xrightarrow{a} x'$ とする。このとき $a \in \mathsf{Tr}(x)$ なので、$\mathsf{Tr}(x) = \mathsf{Tr}(y)$ より $a \in \mathsf{Tr}(y)$ である。

したがって、ある y' が存在して $y \xrightarrow{a} y'$ である。LTS が決定的なので、任意の $w \in A^*$ について、

$$w \in \mathsf{Tr}(x') \iff aw \in \mathsf{Tr}(x) \iff aw \in \mathsf{Tr}(y) \iff w \in \mathsf{Tr}(y')$$

が成り立つ。よって $\mathsf{Tr}(x') = \mathsf{Tr}(y')$、すなわち $x'Ry'$ である。$R = R^{-1}$ より、R は双模倣関係である。 □

3.3 商 LTS

双模倣性の使い方の一つとして、双模倣な状態を一つにまとめることで振る舞いを保ったまま LTS を小さくする操作を紹介します。この操作は LTS の (あるクラスの) 性質を保つため、LTS が性質を満たすかどうか？を検査する問題を解くための前処理としてこの操作を行えば、元の大きな LTS を直接調べる代わりに、より小さな LTS 上で検査を行うことができます。

命題 3.10. 関係 $\sim$ は S 上の同値関係である。

証明　命題 3.4 と定理 3.7 から示すことができる。

反射性は、Δ_S が双模倣関係であり、$\Delta_S \subseteq (\sim)$ が成り立つことから従う。対称性は、R が双模倣関係ならば R^{-1} も双模倣関係であることから成り立つ。推移性は、$\sim$ が双模倣関係であることから、$\sim \circ \sim$ も双模倣関係であり、$(\sim \circ \sim) \subseteq (\sim)$ が成り立つことからいえる。 □

この命題により、関係 $\sim$ の同値類をとって双模倣な状態をまとめることで、状態集合の分割を与えることができます。また、関係 $\sim$ の定義から、元の LTS の遷移に合わせてこの分割の上で LTS を定義できます。たとえば、元の LTS に $s \xrightarrow{a} t$ という遷移があるときは、分割の上では同値類 $[s]_\sim$ から $[t]_\sim$ へラベル a の遷移がある、というように定義できます。

定義 3.11 (商 LTS)**.** LTS $L = (S, A, \to)$ の**商** *LTS* とは、LTS $(S/\sim, A, \to_\sim)$ のことである。ここで $[s]_\sim \xrightarrow{a}_\sim [t]_\sim$ であるとは、ある $s' \in [s]_\sim$ と $t' \in [t]_\sim$ が存在して $s' \xrightarrow{a} t'$ が成り立つことをいう。

商 LTS は、双模倣的に区別できない状態を一つの状態にまとめた LTS です。元の LTS が大きくても、双模倣同値な状態が多く存在すれば、商 LTS はより小さくなります。

さらに、元の状態 s と商 LTS の状態 $[s]_\sim$ は双模倣です。実際、元の LTS と商 LTS の直和を考え、$B = \{(s, [s]_\sim) \mid s \in S\}$ という関係を取ると、B は双模倣関係になります。したがって、元の LTS と商 LTS は、双模倣の観点から同じ振る舞いをもつと考えられます。

今回は双模倣性 $\sim$ の同値類の上に商 LTS を作りましたが、一般の同値関係で商 LTS を作ることもあります。このように状態を粗く見る操作は**抽象化** (abstraction) と呼ばれ、形式検証やモデル検査で重要な概念になっています。

4 不動点と (余) 帰納法

ここまで、振る舞いの等価性としてトレース等価性と双模倣性を見てきました。特に前節では、双模倣性 $\sim$ が最大の双模倣関係であり (定理 3.7)、このことから任意の双模倣関係 R について

$R \subseteq (\sim)$ が成り立つことを見ました。この事実は、双模倣性を示すための基本的な証明方法を与えています。2 つの状態が双模倣であることを示したいとき、$\sim$ 自体を計算する必要はなく、考えたい 2 つの状態を含むような双模倣関係を 1 つ構成すれば十分です。

本節では、この証明方法を束論的不動点の言葉で整理します。双模倣性が「最大の双模倣関係」として特徴づけられる理由、そして双模倣関係を一つ構成するだけでよい理由を、一般的な不動点の原理から説明できればと思います。

不動点というのは、集合 X 上の関数 $f\colon X \to X$ に対し、$f(x) = x$ を満たす点 $x \in X$ のことです。定義は非常に単純ですが、不動点は計算機科学において重要な役割を果たします。特に、無限になりうる対象を有限の記述で表したいときに自然に現れます。典型的には、有限的に記述された操作を繰り返して得られる対象を、その操作から定義される関数の不動点として表します。

例えば、自然数の集合はそれ自体は無限集合ですが、0 から始めて「1 を足す」という操作を繰り返すことで再帰的に生成される集合 (最小不動点) として見ることができます。また、第 2.1 小節の最後で触れた、有限的な記述であるオートマトンと無限になりうる言語との関係も、不動点の言葉で説明することができます。

4.1 完備束上の不動点

まずは、束論的不動点を導入するために、順序を持つ集合を考えます。集合 X 上の関係 $(\leq) \subseteq X \times X$ が以下の 3 つの条件を満たすとき、$(X, \leq)$ を**半順序集合**と呼びます。

1. 反射性: 任意の $x \in X$ について $x \leq x$ が成り立つ。
2. 反対称性: 任意の $x, y \in X$ について、$x \leq y$ かつ $y \leq x$ ならば $x = y$ が成り立つ。
3. 推移性: 任意の $x, y, z \in X$ について、$x \leq y$ かつ $y \leq z$ ならば $x \leq z$ が成り立つ。

今回は半順序集合の中でも特に、完備束と呼ばれるものを扱います。

定義 4.1. 半順序集合 $(X, \leq)$ が**完備束**であるとは、任意の部分集合 $A \subseteq X$ に対して、上限 $\bigvee A$ と下限 $\bigwedge A$ が存在することをいう。

特に、空集合の上限と下限も存在し、それぞれ $\bot, \top$ と書きます。これらは定義から X の最小元、最大元になっています。

例えば、集合 S の冪集合 $\mathcal{P}(S)$ は包含順序 $\subseteq$ によって完備束になります。このとき、上限と下限はそれぞれ和集合と共通部分で与えられ、最小元 $\bot$ は空集合 $\emptyset$、最大元 $\top$ は S です。

定義 4.2. $(X, \leq)$ を半順序集合とし、関数 $f\colon X \to X$ を考える。

- $f(x) = x$ を満たす要素 $x \in X$ を、f の**不動点**という。
- $f(x) \leq x$ を満たす要素 $x \in X$ を、f の**前不動点**という。
- $x \leq f(x)$ を満たす要素 $x \in X$ を、f の**後不動点**という。

また、任意の $x, y \in X$ に対して $x \leq y$ ならば $f(x) \leq f(y)$ が成り立つとき、f が**単調**であるという。

完備束の上の単調関数の最小不動点と最大不動点の存在は、次の定理によって保証されます。

定理 4.3 (Knaster–Tarski の不動点定理 [5])**.** $(X, \leq)$ を完備束とし、$f\colon X \to X$ を単調関数とする。このとき f は最小不動点 μf と最大不動点 νf が存在して、

$$\mu f = \bigwedge\{x \in X \mid f(x) \leq x\}, \quad \nu f = \bigvee\{x \in X \mid x \leq F(x)\}$$

が成り立つ。また、f の不動点の集合は完備束をなす。

証明　ここでは最小不動点、最大不動点の主張のみ示す。$p \coloneqq \bigwedge\{x \in X \mid f(x) \leq x\}$ とおき、p が最小不動点であることを示す。

まず、$f(p) \leq p$ を示す。任意の前不動点 x を考えると、$p \leq x$ から単調性により $f(p) \leq f(x) \leq x$ が成り立つ。よって p の定義より $f(p) \leq p$ が従う。

次に、$p \leq f(p)$ を示す。$f(p) \leq p$ を満たすことから、$f(f(p)) \leq f(p)$ が成り立つ。したがって、p の定義から $p \leq f(p)$ が従う。

以上より、$f(p) = p$ が成り立つ。さらに、$f(x) = x$ を満たす任意の x に対して、$f(x) \leq x$ も満たすので、$p \leq x$ が従う。したがって、p は f の最小不動点である。

最大不動点は完備束 $(X, \geq)$ の最小不動点であるため、主張が成り立つ。　□

つまり、最小不動点は最小の前不動点、最大不動点は最大の後不動点になります。

例 4.4. 非決定性有限オートマトン $\mathcal{A} = (S, A, \to, I, \mathrm{Acc})$ の言語が最小不動点として書けることを見てみましょう。ここで、$(S, A, \to)$ は LTS、 I, Acc は S の部分集合です。オートマトン $\mathcal{A}$ の各状態 s から受理される言語 $\{w \in A^* \mid \exists t \in \mathrm{Acc}.\ s \xrightarrow{w} t\}$ を不動点として表すために、S から $\mathcal{P}(A^*)$ への関数の集合 $\mathcal{P}(A^*)^S$ を考えます。この集合は各成分ごとの包含順序 $(g \leq h \Leftrightarrow \forall s \in S.\ g(s) \subseteq h(s))$ をとることで、完備束になります。関数 $f_{\mathcal{A}}\colon \mathcal{P}(A^*)^S \to \mathcal{P}(A^*)^S$ を任意の $g \in \mathcal{P}(A^*)^S$ と $s \in S$ に

$$f_{\mathcal{A}}(g)(s) = \{\varepsilon \mid s \in \mathrm{Acc}\} \cup \{aw \mid \text{ある } t \in S \text{ が存在して } s \xrightarrow{a} t \text{ かつ } w \in g(t)\}$$

で定義すると、この関数は単調であり、$\mu f_{\mathcal{A}}$ は各状態に対しその状態からの受理言語を返す関数になります。つまり、オートマトン $\mathcal{A}$ の受理言語は $\bigcup_{s \in I}(\mu f_{\mathcal{A}})(s)$ です。このことから、言語を表すための有限的な表示が $\mathcal{A}$ で、それらをつなぐのが不動点だという見方ができます。

4.2　最小不動点と帰納法

定理 4.3 より、最小不動点はすべての前不動点の下限として与えられます。したがって、前不動点は最小不動点以上であることが成り立ち、これが最小不動点に対する証明原理を与えます。

系 4.5 (帰納法)**.** $f\colon X \to X$ を完備束 $(X, \leq)$ 上の単調関数とする。$x \in X$ が $f(x) \leq x$ を満たすならば、$\mu f \leq x$ である。

例 4.6. この帰納法が、通常の数学的帰納法を特別な場合として含んでいることを見てみましょ

う。関数 $f_{\mathbb{N}}\colon \mathcal{P}(\mathbb{N}) \to \mathcal{P}(\mathbb{N})$ を

$$f_{\mathbb{N}}(A) = \{0\} \cup \{n+1 \mid n \in A\}$$

で定義すると、その最小不動点 $\mu f_{\mathbb{N}}$ は $\mathbb{N}$ になります。ここで、自然数全体についてある性質を示したいとして、$P \subseteq \mathbb{N}$ をその性質を満たす自然数全体の集合とします。すると $f_{\mathbb{N}}$ の定義から、

$$f_{\mathbb{N}}(P) \subseteq P \quad \Longleftrightarrow \quad 0 \in P \text{ かつ } (n \in P \text{ ならば } n+1 \in P)$$

が成り立ちます。したがって、$0 \in P$ かつ $(n \in P$ ならば $n+1 \in P)$ が成り立つとき、系 4.5 より $\mu f_{\mathbb{N}} \subseteq P$、つまり全ての自然数が性質を満たすことが分かります。これは通常の数学的帰納法そのものであり、最小不動点に対する帰納法の特別な場合として理解することができます。

帰納法を最小不動点に対する一般原理として定式化したことにより、自然数だけではなく、様々な帰納的に定義された集合に対して帰納法を適用することができます。

例 4.7. 状態集合 S、辺の集合 $\to \subseteq S \times S$、初期状態集合 $I \subseteq S$ をもつグラフ $G = (S, \to, I)$ を考えます。これはラベルが 1 点集合の LTS に初期状態集合をつけたものだと見ることもできます。関数 $f_G\colon \mathcal{P}S \to \mathcal{P}S$ を、任意の $X \in \mathcal{P}S$ に対し

$$f_G(X) = I \cup \{\, t \in S \mid \text{ある } s \in X \text{ が存在して } s \to t \,\}$$

で定義します。この関数は単調であり、最小不動点 μf_G は到達可能な状態の集合、つまりある初期状態から 0 回以上の遷移で到達できる状態全体を表します。

いま、安全な状態の集合 $\mathrm{Safe} \subseteq S$ が与えられたときに、到達可能な状態が安全かどうか？という安全性検証をしたいとします。つまり、

$$\mu f_G \subseteq \mathrm{Safe}$$

を示したいわけです。この包含関係を直接示そうと思うと、μf_G を計算する必要がありますが、状態空間が非常に大きいときは計算が難しいです。しかし、帰納法を使って Safe の部分集合 P で $f_G(P) \subseteq P$ を満たすものを構成できれば、$\mu f_G \subseteq P \subseteq \mathrm{Safe}$ が成り立つため安全性を検証することができます。このような P は**不変条件**と呼ばれ、初期状態で成り立ち、1 ステップ遷移によって保存される性質になっています。実際、f_G の定義より

$$f_G(P) \subseteq P \quad \Longleftrightarrow \quad \begin{cases} I \subseteq P, \\ s \in P \text{ かつ } s \to t \text{ ならば } t \in P, \end{cases}$$

が成り立ちます。このように、帰納法によって「すべての到達可能状態が安全である」という大域的な主張を、「初期状態で成り立ち、1 ステップの遷移で保存される」という局所的な条件を満たす P を証拠として与えることで示せるようになります。

4.3 最大不動点と余帰納法

前節では、最小不動点に対する証明原理として帰納法を見ました。その双対として得られる、最大不動点に対する証明原理が**余帰納法**です。

系 4.8 (余帰納法)**.** $f\colon X \to X$ を完備束 $(X, \leq)$ 上の単調関数とする。$x \in X$ が $x \leq f(x)$ を満たすならば、$x \leq \nu f$ である。

では、双模倣性と余帰納法との関係を見ていきましょう。LTS $(S, A, \to)$ を考えます。関係 $R \subseteq S \times S$ に対して、$\mathcal{B}(R) \subseteq S \times S$ を次の二条件を満たすペア (s, t) 全体として定めます。

1. 任意の $a \in A$ と $s' \in S$ について、$s \xrightarrow{a} s'$ ならば、ある t' が存在して $t \xrightarrow{a} t'$ かつ $(s', t') \in R$ が成り立つ。
2. 任意の $a \in A$ と $t' \in S$ について、$t \xrightarrow{a} t'$ ならば、ある s' が存在して $s \xrightarrow{a} s'$ かつ $(s', t') \in R$ が成り立つ。

この定義により、双模倣関係は $\mathcal{B}$ の後不動点として特徴づけられることが分かります。

命題 4.9. $\mathcal{B}\colon \mathcal{P}(S \times S) \to \mathcal{P}(S \times S)$ は単調関数であり、任意の $R \subseteq S \times S$ に対し、

$$R \text{ は双模倣関係} \quad \Longleftrightarrow \quad R \subseteq \mathcal{B}(R)$$

が成り立つ。

定理 4.3 より、最大不動点は最大の後不動点でした。双模倣性は最大の双模倣関係である (定理 3.7) ことを思い出すと、以下が成り立ちます。

系 4.10. 双模倣性 $\sim$ は関数 $\mathcal{B}$ の最大不動点である。

関数 $\mathcal{B}$ は、入力の関係 R のもとで一歩先の遷移を互いに模倣できるような状態のペアを集める操作でした。そのため、直感的には、最大不動点 $\nu\mathcal{B}$ は一歩ごとに互いに模倣でき、模倣後も同じ条件で対応を続けられる状態のペアを集めた関係として解釈できます。上の系は、双模倣性 $\sim$ がこの最大不動点であることを述べています。

この系により、

$$R \text{ が双模倣関係} \quad \Longrightarrow \quad R \subseteq (\sim)$$

が余帰納法の特別な場合であることが分かります。この主張は、双模倣性 $\sim$ の定義 (定義 3.5) から自明に導かれるものではありますが、$\nu\mathcal{B}$ に対する余帰納法だと思うと、例 4.7 と同様に、振る舞いの一致 ($(s, t) \in \nu\mathcal{B}$) という大域的な性質を 1 ステップ分についての条件を満たす証拠 ($(s, t) \in R$ を満たす双模倣関係 R) によって示すという証明原理だと見ることができます。

5 Hennessy–Milner 論理による論理的等価性

第 2 節と第 3 節では、LTS の振る舞いの等価性として、トレース等価性と双模倣性を見てきました。トレース等価性では実行できるラベル列が同じかどうか、双模倣性では互いに模倣できるかどうかに注目し、等価性を定義していました。本節では、2 つの状態が同じ性質を満たすかどうかに基づいている、論理的等価性を考えます。

どのような性質を考えるかを形式的に定めるために、LTS の状態の性質を記述するための基本的な論理である *Hennessy–Milner* **論理** *(HML)* を導入します。この論理を用いて論理的等価性を定義し、それが双模倣性とどのように関係するかを見ていきます。

5.1 Hennessy–Milner 論理

まずは HML を導入していきます。そのために、性質をどのような記号列 (論理式) で表すかを定め、その記号列が LTS の状態に対してどのような意味を持つかを定めます。例えば、記号列 $\langle a\rangle\top$ によって「ラベル a の遷移ができる」という状態の性質を表すというように、LTS の状態がもつ性質を形式的に記述します。

定義 5.1. 集合 A 上の *HML* **の論理式**の集合 $\mathcal{L}_{\mathrm{HML}}$ は、次の文法で帰納的に定義される。

$$\varphi, \psi ::= \top \mid (\neg\varphi) \mid (\varphi \wedge \psi) \mid (\langle a\rangle\varphi) \qquad (a \in A).$$

すなわち、$\mathcal{L}_{\mathrm{HML}}$ は次を満たす最小の集合である。

1. $\top \in \mathcal{L}_{\mathrm{HML}}$,
2. $\varphi \in \mathcal{L}_{\mathrm{HML}}$ ならば $(\neg\varphi) \in \mathcal{L}_{\mathrm{HML}}$,
3. $\varphi, \psi \in \mathcal{L}_{\mathrm{HML}}$ ならば $(\varphi \wedge \psi) \in \mathcal{L}_{\mathrm{HML}}$,
4. $\varphi \in \mathcal{L}_{\mathrm{HML}}$ かつ $a \in A$ ならば $(\langle a\rangle\varphi) \in \mathcal{L}_{\mathrm{HML}}$.

ここで、A は固定している LTS$L = (S, A, \to)$ のラベルの集合です。論理式に含まれる括弧は、適宜省略して書きます。上で定義されているように、$\mathcal{L}_{\mathrm{HML}}$ は $\top$ から始めて、論理結合子と呼ばれる $\neg$, $\wedge$, $\langle a\rangle$ を有限回使って構成できる論理式の集合です。例えば、$\langle a\rangle\top$ や $\langle a\rangle(\langle b\rangle\top \wedge \neg\top)$ は論理式になります。

これらの基本的な論理結合子から、いくつかの略記を導入します。

$$\bot := \neg\top, \qquad \varphi \vee \psi := \neg(\neg\varphi \wedge \neg\psi), \qquad [a]\varphi := \neg\langle a\rangle\neg\varphi.$$

これらは新しい構文を追加しているわけではなく、すでに定義した論理式を使った略記です。

定義 5.2. 論理式 $\varphi \in \mathcal{L}_{\mathrm{HML}}$ に対し、$[\![\varphi]\!] \subseteq S$ を以下のように帰納的に定義する。

$$\begin{aligned}
[\![\top]\!] &:= S, \\
[\![\neg\varphi]\!] &:= S \setminus [\![\varphi]\!], \\
[\![\varphi \wedge \psi]\!] &:= [\![\varphi]\!] \cap [\![\psi]\!], \\
[\![\langle a \rangle \varphi]\!] &:= \{s \in S \mid \text{ある } s' \in S \text{ が存在して } s \xrightarrow{a} s' \text{ かつ } s' \in [\![\varphi]\!]\}.
\end{aligned}$$

状態 $s \in S$ が論理式 φ を**充足する**とは、$s \in [\![\varphi]\!]$ であることをいう。このとき、$s \models \varphi$ と書く。

この定義は、論理式の構造に沿って再帰的に意味を定めています。例えば、$[\![\langle a \rangle \top]\!]$ の意味は以下のように計算でき、ラベル a の遷移ができる状態の集合を表します。

$$\begin{aligned}
[\![\langle a \rangle \top]\!] &= \{s \in S \mid \text{ある } s' \in S \text{ が存在して } s \xrightarrow{a} s' \text{ かつ } s' \in [\![\top]\!]\} \\
&= \{s \in S \mid \text{ある } s' \in S \text{ が存在して } s \xrightarrow{a} s'\},
\end{aligned}$$

また、略記として導入した論理式の意味も計算できます。例えば、$[a]\varphi$ の意味は以下のように計算できます。

$$\begin{aligned}
[\![[a]\varphi]\!] &= [\![\neg\langle a \rangle \neg\varphi]\!] \\
&= S \setminus [\![\langle a \rangle \neg\varphi]\!] \\
&= S \setminus \{s \in S \mid \text{ある } s' \in S \text{ が存在して } s \xrightarrow{a} s' \text{ かつ } s' \in [\![\neg\varphi]\!]\} \\
&= \{s \in S \mid \text{すべての } s' \in S \text{ について、} s \xrightarrow{a} s' \text{ ならば } s' \notin [\![\neg\varphi]\!]\} \\
&= \{s \in S \mid \text{すべての } s' \in S \text{ について、} s \xrightarrow{a} s' \text{ ならば } s' \in [\![\varphi]\!]\}.
\end{aligned}$$

このことから、$[a]\varphi$ の意味はラベル a の遷移先で必ず φ が成り立つ状態の集合になります。

5.2 HML 論理的等価性

論理式が LTS の状態の性質、つまり振る舞いを表すと考えると、ある論理式が状態 s では成り立ち、状態 t では成り立たないときは、s と t は異なる振る舞いをしていると考えることができます。逆に、HML のどの論理式を使っても 2 つの状態を区別できないとき、少なくとも HML で観測できる範囲では同じ振る舞いをしていると考えることができます。

定義 5.3. LTS $(S, A, \to)$ の 2 つの状態 $s, t \in S$ について、任意の論理式 $\varphi \in \mathcal{L}_{\mathrm{HML}}$ に対して

$$s \models \varphi \quad \Longleftrightarrow \quad t \models \varphi$$

が成り立つとき、s と t は *HML* **論理的に等価**であるといい、$s \equiv_{\mathcal{L}_{\mathrm{HML}}} t$ と書く。

つまり、$s \equiv_{\mathcal{L}_{\mathrm{HML}}} t$ であるとは、どの論理式を使っても s と t を区別できないという意味です。一方、ある HML 論理式 φ が存在して

$$s \models \varphi \quad \text{かつ} \quad t \not\models \varphi$$

となるならば、s と t は φ によって区別できます。このとき、論理式 φ は s と t が論理的等価でないことの証拠になっています。

例 5.4. 例 2.2 の LTS の状態 s_0, t_0 を考えます。これらは有限トレース等価でしたが、次の論理式で区別できます。

$$\varphi := \langle a \rangle (\langle b \rangle \top \wedge \langle c \rangle \top).$$

この論理式は、「b 遷移も c 遷移も可能な状態へ、a 遷移で行ける」という性質を表します。左の LTS では $s_0 \models \varphi$ が成り立ちます。実際、$s_0 \xrightarrow{a} s_1$ であり、s_1 では b 遷移も c 遷移もできます。一方、右の LTS では $t_0 \not\models \varphi$ が成り立ちます。t_0 から a で行ける状態は、b だけできる状態 t_1 と、c だけできる状態 t_1' のみで、b と c の両方を選べる状態には行けないからです。したがって、$s_0 \equiv_{\mathcal{L}_{\mathrm{HML}}} t_0$ は成り立ちません。

5.3 双模倣性と HML 論理的等価性

では最後に、HML による論理的等価性と双模倣性の関係を見ていきましょう。以下の定理は、2 つの状態が双模倣ならば論理的等価である、つまり HML のどの論理式を使っても区別できないことを示しています。

定理 5.5. LTS $(S, A, \to)$ の状態 $s, t \in S$ について、$s \sim t$ ならば $s \equiv_{\mathcal{L}_{\mathrm{HML}}} t$ が成り立つ。

証明 各 HML 論理式 φ が次の命題を満たすことを、φ の構造に関する帰納法で示す:

$$\forall x, y \in S.\ \big(x \sim y \Longrightarrow (x \models \varphi \Longleftrightarrow y \models \varphi)\big). \tag{1}$$

HML 論理式 φ, ψ が (1) を満たすとする。$\top$, $\neg\varphi$, $\varphi \wedge \psi$ は定義から (1) を満たす。$s \sim t$ かつ $s \models \langle a \rangle \varphi$ とする。このとき、ある s' が存在して $s \xrightarrow{a} s'$ かつ $s' \models \varphi$ である。$s \sim t$ なので、ある t' が存在して $t \xrightarrow{a} t'$ かつ $s' \sim t'$ である。帰納法の仮定から $t' \models \varphi$ である。したがって、$t \models \langle a \rangle \varphi$ が成り立つ。逆向きは $t \sim s$ に対する同様の議論で成り立つ。

以上により、(1) が成り立つ。したがって、$s \sim t$ ならば $s \equiv_{\mathcal{L}_{\mathrm{HML}}} t$ が成り立つ。 □

この定理により、双模倣である状態のペアは、HML のどの論理式を使っても区別できないことが分かります。特に、定義 3.11 のあとで見たように、商 LTS の状態と元の LTS の状態が双模倣であることから、元の LTS での HML に関する検査を商 LTS での検査に置き換えることができます。また、この定理は証拠という観点からも重要です。論理的等価であることを示したいとき、定義どおりに示すなら全ての論理式について検査する必要がありますが、この定理により考える状態のペアを含む双模倣関係を 1 つ構成するだけでよいことが分かります。同様に、双模倣でないことを示したいときも、定義どおりに示すと考える状態のペアを含む双模倣関係が存在しないことを示す必要がありますが、この定理により区別できる論理式を 1 つ見つけるだけでよいことが分かります。

逆向き、すなわち $s \equiv_{\mathcal{L}_{\mathrm{HML}}} t$ ならば $s \sim t$ は、一般の LTS では必ずしも成り立ちません。しかし、定義 2.4 で定義した image-finite 性のもとでは成り立ちます。

定理 5.6 (Hennessy–Milner の定理 [3])**.** LTS $(S, A, \to)$ が image-finite であるとする。このとき、任意の状態 $s, t \in S$ について

$$s \sim t \quad \Longleftrightarrow \quad s \equiv_{\mathcal{L}_{\mathrm{HML}}} t$$

が成り立つ。

証明　左から右はすでに示した。右から左を示す。関係 $\equiv_{\mathcal{L}_{\mathrm{HML}}} \subseteq S \times S$ が双模倣関係であることを示せばよい。

$x \equiv_{\mathcal{L}_{\mathrm{HML}}} y$ かつ $x \xrightarrow{a} x'$ とする。y の a 遷移先を $\mathsf{Succ}_a(y) \coloneqq \{z \mid y \xrightarrow{a} z\}$ と書く。$x \xrightarrow{a} x'$ なので、$x \models \langle a \rangle \top$ である。また、$x \equiv_{\mathcal{L}_{\mathrm{HML}}} y$ も成り立つため、$\mathsf{Succ}_a(y)$ は空ではない。さらに、image-finite 性により、これは有限集合である。したがって、自然数 $n\,(> 0)$ を用いて $\mathsf{Succ}_a(y) = \{y_1, \ldots, y_n\}$ と書くことができる。

どの $i \in \{1, \cdots, n\}$ についても $x' \not\equiv_{\mathcal{L}_{\mathrm{HML}}} y_i$ であると仮定する。すると、各 i について、x' と y_i を区別する HML 論理式が存在する。必要なら否定を取ることで、$x' \models \varphi_i$ かつ $y_i \not\models \varphi_i$ となる論理式 φ_i が存在することがわかる。このとき $x' \models \bigwedge_{i=1}^{n} \varphi_i$ であり、任意の $i \in \{1, \cdots, n\}$ に $y_i \not\models \bigwedge_{i=1}^{n} \varphi_i$ が成り立つ。したがって $x \models \langle a \rangle \Big(\bigwedge_{i=1}^{n} \varphi_i \Big)$ である一方、$y \not\models \langle a \rangle \Big(\bigwedge_{i=1}^{n} \varphi_i \Big)$ である。これは $x \equiv_{\mathcal{L}_{\mathrm{HML}}} y$ に反する。

よって、ある i が存在して $x' \equiv_{\mathcal{L}_{\mathrm{HML}}} y_i$ である。逆向きの模倣条件も、$\equiv_{\mathcal{L}_{\mathrm{HML}}}$ は対称であることから分かる。よって、$\equiv_{\mathcal{L}_{\mathrm{HML}}}$ は双模倣関係であるため、$s \equiv_{\mathcal{L}_{\mathrm{HML}}} t$ ならば $s \sim t$ が成り立つ。　□

参考文献

[1] Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs*, Lecture Notes in Computer Science, pages 52–71. Springer, 1981.

[2] Edmund M. Clarke, E. Allen Emerson, and Joseph Sifakis. Model checking: algorithmic verification and debugging. *Commun. ACM*, 52(11):74–84, 2009.

[3] Matthew Hennessy and Robin Milner. Algebraic laws for nondeterminism and concurrency. *J. ACM*, 32(1):137–161, 1985.

[4] Jean-Pierre Queille and Joseph Sifakis. Specification and verification of concurrent systems in CESAR. In *Symposium on Programming*, Lecture Notes in Computer Science, pages 337–351. Springer, 1982.

[5] Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.

[6] Rob J Van Glabbeek. The linear time–branching time spectrum i. the semantics of concrete, sequential processes. In *Handbook of process algebra*, pages 3–99. Elsevier, 2001.