並列分散システム論配布資料 (3)
Milner [1]: 第3章，第4章

京都大学 大学院情報学研究科 通信情報システム専攻
五十嵐 淳
e-mail: igarashi@kuis.kyoto-u.ac.jp

平成 25 年 10 月 22 日

3 Sequential Processes and Bisimulation (逐次プロセスと双模倣)

アクションの定義の変更 (詳細化):

名前の集合	$a, b, \ldots \in \mathcal{N} = \{\mathtt{tea}, \mathtt{2p}, \ldots\}$
余名 (co-names) の集合	$\bar{\mathcal{N}} = \{\bar{a} \mid a \in \mathcal{N}\}$
ラベルの集合	$\mathcal{L} = \mathcal{N} \cup \bar{\mathcal{N}}$
アクションの集合	$\alpha, \beta, \ldots \in Act = \mathcal{L}$

a と $\bar{a}$ は相補的な (例えば「2 ペンスを投入する」に対する「2 ペンスを受け取る」) 動作を表す．相補的な動作は並行性を導入する際に同期を伴う動作として扱われる．ラベルはブラックボックスシステムにおけるボタンの名前を表すと考えられる．アクションは今のところラベルのみだが後で，(ボタンの押下以外の) その他のアクションが加えられる．

3.1 定義 [ラベル付遷移システム (labelled transition system, LTS)]: Act 上の LTS は

- 状態の集合 $\mathcal{Q}$
- 遷移関係 $\mathcal{T} \subseteq \mathcal{Q} \times Act \times \mathcal{Q}$

で与えられる．

LTS は初期状態，受理状態を考えないオートマトンだといえる．

3.2 定義 [強模倣 (strong simulation)]: $(\mathcal{Q}, \mathcal{T})$ を LTS とする．$\mathcal{S}$ を $\mathcal{Q}$ 上の二項関係とする．$\mathcal{S} \subseteq \mathcal{Q} \times \mathcal{Q}$ が以下の条件をみたす時，$\mathcal{S}$ は LTS $(\mathcal{Q}, \mathcal{T})$ 上の**強模倣** *(関係)* であるという:

もし $p\,\mathcal{S}\,q$ かつ $p \xrightarrow{\alpha} p'$ ならば，$q' \in \mathcal{Q}$ が存在して $q \xrightarrow{\alpha} q'$ かつ $p'\,\mathcal{S}\,q'$ である

$p\,\mathcal{S}\,q$ なる強模倣関係 $\mathcal{S}$ が存在する時，q は p を**強く模倣する** *(strongly similar)* という．

(注) ふたつの LTS の関係ではなく，ひとつの LTS 上での状態の二項関係として定義される点に注意．自動販売機の例は (遷移関係をグラフとして見た時) 連結でない状態があるようなひとつのシステムと捉える．状態定義中の $\mathcal{S}$ に関する条件は,

$$\begin{array}{ccc} p & \mathcal{S} & q \\ \downarrow{\scriptstyle \alpha} & & \vdots{\scriptstyle \alpha} \\ p' & \mathcal{S} & \exists q' \end{array}$$

のように図示されることも多い．(本当は下側の $\mathcal{S}$ も薄く (？) 書きたい．)

3.3 定義 [強双模倣 (strong bisimulation)]: $(\mathcal{Q}, \mathcal{T})$ を LTS とする．$\mathcal{S}$ を $\mathcal{Q}$ 上の二項関係とする．$\mathcal{S}, \mathcal{S}^{-1}$ ともに $(\mathcal{Q}, \mathcal{T})$ 上の強模倣関係である時 $\mathcal{S}$ は LTS $(\mathcal{Q}, \mathcal{T})$ 上の**強双模倣** *(**関係**)* であるという．

$p\ \mathcal{S}\ q$ なる強双模倣関係 $\mathcal{S}$ が存在する時，p と q は**強双模倣** *(strongly bisimilar, strongly equivalent)* である，といい，$p \sim q$ と書く．

(クイズ) $p \sim q$ と「p は q を強く模倣し，かつ，q は p を強く模倣する」は同値な条件か？

3.4 Proposition:

1. $\sim$ は同値関係 (反射的，対称的，推移的関係)
2. $\sim$ は強双模倣関係

逐次プロセス式 以下では，LTS に代わる，プロセスの表記法を導入する．正則表現のようにある種のプログラム的な表現 (逐次プロセス式) を使って状態遷移を記述する．$A, B, C, \ldots$ をプロセスの名前を表す識別子とする．プロセス名には (パラメータ化された) 逐次プロセス式が関連付けられる ($\Rightarrow$ プロセスの定義) ことになる．

3.5 定義 [逐次プロセス式, sequential process expression]: 逐次プロセス式の集合 $P \in \mathcal{P}^{\mathrm{seq}}$ を以下の文法で定義する．

$$P ::= A\langle a_1, \ldots, a_n \rangle \mid \sum_{i \in I} \alpha_i.P_i$$

ただし I は有限の添字集合とする．以下，$P, Q, R, \ldots$ を (逐次) プロセス式を表すために使う．

$I = \{1, 2, 3\}$ の時，$\sum_{i \in I} \alpha_i.P_i$ を $\alpha_1.P_1 + \alpha_2.P_2 + \alpha_3.P_3$ と書く．(+ の順番は重要ではない．) また，$I = \emptyset$ の時 $\mathtt{0}$ と書く．

各プロセスの名前には,

$$A(a_1, \ldots, a_n) \stackrel{def}{=} P_A$$

という形で定義がされている (P_A に現れる名前は a_i のみである)．$A\langle a_1, \ldots, a_n \rangle$ はプロセス A のパラメータを $a_1, \ldots, a_n$ で具体化したプロセスを表す (定義と使用で括弧の種類が違うことに注意!)．以下，名前の列は略記して $\vec{a}$ と書くこともある．また，$\{\vec{b}/\vec{a}\}P$ で P 中の a_i を b_i で置き換えたようなプロセス式を表すとする．

3.6 定義 [構造的合同性 (structural congruence)]: ふたつの逐次プロセス式 P, Q が構造的合同 ($\equiv Q$ と書く) であるとは，片方 (例えば P) に現れる $A\langle\vec{b}\rangle$ を $\{\vec{b}/\vec{a}\}P_A$ (ただし $A(\vec{a}) = P_A$ と定義されている) で置き換えてもう片方が得られることをいう.

3.7 例:

$$\begin{aligned} A(a,b) &\stackrel{def}{=} a.A\langle a,b\rangle + b.B\langle a,a\rangle \\ B(c,d) &\stackrel{def}{=} c.d.\mathtt{0} \end{aligned}$$

の下で， $B\langle a,a\rangle \equiv a.a.\mathtt{0}$ であり，$A\langle a,b\rangle \equiv a.(a.A\langle a,b\rangle + b.B\langle a,a\rangle) + b.a.a.\mathtt{0}$ である.

3.8 定義 [逐次プロセス式から生成される LTS]: 逐次プロセス式とプロセス定義 (の集合) から生成される (Act 上の)LTS とは,

- $\mathcal{Q} = \mathcal{P}^{\mathrm{seq}}$
- $\mathcal{T} = \{P \stackrel{\alpha_i}{\to} P_i \mid P \equiv \sum_{i\in I} \alpha_i.P_i\}$

で与える LTS のことである.

3.9 例 [Boolean Buffer, Scheduler, Counter]:

4 Concurrent Processes and Reaction

前章の逐次プロセスを拡張し，並行動作するプロセスについて考える．この時，オートマトンにおける文字の入力のようなプロセスと外界との相互作用だけでなく，プロセス同士の相互作用も発生するため，外部観測可能な動作と内部的動作の区別が重要になってくる.

この章では並行プロセス式を導入し，内部動作がどのように定義されるかをみていく．(外部観測可能な動作については次章で扱う．)

名前 a もしくは余名 $\bar{a}$ に対し，その補名 (complement) をそれぞれ $\bar{a}, a$ とする．$(a, \bar{a})$ の組は，ブラックボックスシステム同士で相互作用・協調動作・ハンドシェイク同期をするための手段となる．例えば,

$$\begin{aligned} A &\stackrel{def}{=} a.\bar{b}.A \\ B &\stackrel{def}{=} b.\bar{c}.B \end{aligned}$$

というふたつの逐次プロセス定義に対し，A, B を並行に動作にさせると，A の b による遷移と，B の $\bar{b}$ の遷移が**同期して**発生する．このような並行システムの同期動作は外部との相互作用ではなく内部的な動作として考えられる．この内部動作 (リアクションという) は τ という記号 (Act の新たな要素) で表す.

以下，$Act \stackrel{def}{=} \mathcal{L} \cup \{\tau\}$ とし，$\alpha, \beta, \ldots$ をアクションを表す記号，$\lambda, \mu, \ldots$ をラベル (外部との相互作用，ブラックボックスのボタン) を表す記号として用いる.

4.1 定義 [(並行) プロセス式]: 並行プロセス式 (または，単にプロセス式) の集合 $\mathcal{P}$ を以下の文法で定義[1]する．

$$P ::= A\langle a_1, \ldots, a_n\rangle \mid \sum_{i\in I} \alpha_i.P_i \mid (P_1 \parallel P_2) \mid \mathsf{new}\, a\, P$$

ただし I は有限の添字集合とする．以下，$P, Q, R, \ldots$ を (並行) プロセス式を表すために使う．

注: $P_1 \parallel P_2$ はプロセス P_1, P_2 が並行して実行されていることを示す．また，$\mathsf{new}\, a\, P$ は名前 a の有効範囲が P に制限されている局所的なものであることを示す．接頭辞的な構文 "α." や "$\mathsf{new}\, a$" は，$+$ や $\parallel$ よりも強く結合する．すなわち，$\mathsf{new}\, a\, P \parallel Q$ は $(\mathsf{new}\, a\, P) \parallel Q$ のことであり，$\mathsf{new}\, a\, (P \parallel Q)$ ではない．

$\sum_{i\in I} \alpha_i.P_i$ は prefix sum とも呼ばれる．sum は直感的には非決定的な動作選択を表しているが，必ず $\alpha_i.$ という動作が伴っていることに注意．$(P \parallel Q) + R$ のようなものはプロセス式ではない．今後 M, N は prefix sum を示す記号として使われる．

束縛されている名前と自由な名前，α 変換 $\mathsf{new}\, a\, P$ において， a は**束縛されている名前** *(bound name)* という．束縛されている名前以外の名前を自由な名前という．以下，$\mathsf{fn}(P)$ で P 中の自由な名前すべての集合を表す．

プロセス式で，束縛されている名前を (自由な名前と衝突しないように) 変更することをを α 変換といい，α 変換前後のプロセス式は (後で定義する) 構造的合同関係にある．

$$\mathsf{new}\, a\, (b.a.\mathsf{0} + \bar{a}.c.\bar{b}.\mathsf{0}) \equiv \mathsf{new}\, d\, (b.d.\mathsf{0} + \bar{d}.c.\bar{b}.\mathsf{0}) \not\equiv \mathsf{new}\, b\, (b.b.\mathsf{0} + \bar{b}.c.\bar{b}.\mathsf{0})$$

逐次プロセスと同様，プロセス識別子には定義

$$A(\vec{a}) \stackrel{def}{=} P_A$$

が割り当てられているが，$\mathsf{fn}(P_a) \subseteq \{a_1, \ldots, a_n\}$ かつ P_A は和の形 $\sum_{i\in I} \alpha_i.P_i$ をしているとする．

4.2 定義 [構造的合同]: $P \equiv Q$ は以下の規則で定義[2]される．

$$\frac{(\text{if } A(\vec{a}) \stackrel{def}{=} P_A)}{A\langle\vec{b}\rangle \equiv \{\vec{b}/\vec{a}\}P_A} \qquad \frac{}{P \parallel \mathsf{0} \equiv P}$$

$$\frac{}{P \parallel Q \equiv Q \parallel P}$$

$$\frac{(P \text{ から } Q \text{ は } \alpha \text{ 変換で得られる})}{P \equiv Q} \qquad \frac{}{P \parallel (Q \parallel R) \equiv (P \parallel Q) \parallel R}$$

[1]原著では $\parallel$ は $\mid$ だが，文法定義と紛らわしいので $\parallel$ を使う．

[2]各規則は，水平線の上にあることが成立するならば，下にあることが成立する，と読む．詳しくは拙著「プログラミング言語の基礎概念」(サイエンス社) などを参考にされたし．

$$\frac{(\text{if } a \notin \mathsf{fn}(P))}{\mathsf{new}\, a\, (P \parallel Q) \equiv P \parallel \mathsf{new}\, a\, Q} \qquad \frac{}{\mathsf{new}\, a\, 0 \equiv 0}$$

$$\frac{}{\mathsf{new}\, ab\, P \equiv \mathsf{new}\, ba\, P}$$

以下の規則は，構造的合同が合同関係，すなわち，同値関係であり，かつ，プロセスの構成に関して閉じていることを示している．(つまりプロセス式の一部分を構造的合同な別のプロセスに置き換えた結果全体は下のプロセスと構造的合同になる)

$$\frac{}{P \equiv P} \qquad \frac{P \equiv Q}{\mathsf{new}\, a\, P \equiv \mathsf{new}\, a\, Q}$$

$$\frac{P \equiv Q}{Q \equiv P} \qquad \frac{P \equiv Q}{P \parallel R \equiv P \parallel R}$$

$$\frac{P \equiv Q \qquad Q \equiv R}{P \equiv R} \qquad \frac{P \equiv Q}{R \parallel P \equiv R \parallel Q}$$

$$\frac{P \equiv Q}{\alpha.P + M \equiv \alpha.Q + M}$$

ちなみに，Milner[1] では $M_1 + M_2$ と $M_2 + M_1$ などのように，和に現れるプロセスの順序を変えたものも構造的合同としているが，どちらも結局は $\sum_{i\in\{1,2\}} M_i$ のことなので特に明示的に規則を与える必要はない．

4.3 定義 [プロセス標準形]: プロセス式 $\mathsf{new}\, \vec{a}\, (M_1 \parallel \cdots \parallel M_n)$ (ただし各 M_i は非空の和) を標準形である，という．($n = 0$ の時 $M_1 \parallel \cdots \parallel M_n$ は 0 を意味する．また，$\vec{a}$ が空の場合は $\mathsf{new}\, \vec{a}$ 自体がない，と考える．)

4.4 定理: 任意のプロセスに対し，それと構造的合同な標準形プロセスが存在する．

4.5 定義 [リアクション関係]: プロセス式の二項関係 $P \longrightarrow Q$ を以下の規則で定義する．

$$\frac{}{\tau.P + M \longrightarrow P}\ (\text{TAU}) \qquad \frac{P \longrightarrow P'}{\mathsf{new}\, a\, P \longrightarrow \mathsf{new}\, a\, P'}\ (\text{RES})$$

$$\frac{}{(a.P + M) \parallel (\bar{a}.Q + N) \longrightarrow P \parallel Q}\ (\text{REACT})$$

$$\frac{P \longrightarrow P'}{P \parallel Q \longrightarrow P' \parallel Q}\ (\text{PAR}) \qquad \frac{P \equiv Q \qquad Q \longrightarrow Q' \qquad Q' \equiv P'}{P \longrightarrow P'}\ (\text{STRUCT})$$

リアクション関係は文献によっては簡約関係 (reduction relation) と呼ばれることも多い．プロセス P について $P \longrightarrow P'$ なる P' が存在しない時，$P \not\longrightarrow$ と書き，P は**安定**であるという．

4.6 例 [Lottery]:

参考文献

[1] Robin Milner. Communicating and Mobile Systems: The π-Calculus. Cambridge University Press. 1999.