6A-4

シャフル表現による非同期イベント系列の記述*

阿部 真也†

† 地方独立行政法人 東京都立産業技術研究センター

1 はじめに

システムの動作テストにおいて，実行時に発生したイベント系列が開発者の意図する系列であるかどうかを判定するのは重要である．イベント系列の記述には正規表現が用いられることが多い．その理由として，開発者が簡潔に記述できること，テストプログラムによる解析が容易なことが挙げられる．

ところが，各々のイベントが非同期並行的に実行されるようなシステムでは，正規表現の記述能力では不十分であることが知られている [1, 2, 3, 4]．たとえば，セマフォによる同期機構の動作を記述するには現在の状態を保持する必要があるが，正規表現では数を記憶できないため記述が困難である．また，系列の並行性は正規表現でも記述可能であるが，多くの和を用いた表現になり簡潔ではない．

本稿では，シャフル表現 [5] による非同期イベント系列の記述法を提案し，代表的な非同期問題であるキュー問題，read/write 問題，セマフォの記述例を与える．以降，2 節でシャフル表現の定義，3 節で非同期問題の記述例，4 節で結論と今後の課題，5 節で関連研究について述べる．

2 シャフル表現

この節では，文献 [5] で定義されたシャフル演算，シャフル閉包演算，シャフル表現，シャフル言語について述べる．

定義 2.1 シャフル演算 Σ を有限記号集合，λ を空語とする．このときシャフル演算は次のように定義される．

- 全ての $u \in \Sigma^*$ について，$u \odot \lambda = \lambda \odot u = \{u\}$
- 全ての $u, v \in \Sigma^*$, $a, b \in \Sigma$ について，$au \odot bv = a(u \odot bv) \cup b(au \odot v)$

また，言語 $L_1, L_2 \subset \Sigma^*$ についてシャフル演算は次のように定義される．

$$L_1 \odot L_2 = \bigcup_{u \in L_1, v \in L_2} u \odot v$$

定義 2.2 シャフル閉包演算 Σ を有限記号集合，λ を空語とする．言語 $L \subset \Sigma^*$ についてシャフル閉包演算は次のように定義される．

$$L^{\otimes} = \bigcup_{i=0}^{\infty} L^{\otimes i}$$

ただし $L^{\otimes 0} = \{\lambda\}$，$L^{\otimes i+1} = L^{\otimes i} \odot L$.

定義 2.3 シャフル表現 空集合 ϕ, 空語 λ, 記号 a はシャフル表現である．S_1, S_2 をシャフル表現とすると，連接 S_1S_2，和 $S_1|S_2$，クリーネ閉包 S_1^*，シャフル $S_1 \odot S_2$，シャフル閉包 $S_1^{\otimes}$ もシャフル表現である．これら以外はシャフル表現ではない．

定義 2.4 シャフル言語 シャフル表現 S_1, S_2 から生成されるシャフル言語 $L(S)$ は次のように定義される．$L(\phi) = \phi$，$L(\lambda) = \{\lambda\}$，$L(a) = \{a\}$，$L(S_1S_2) = L(S_1)L(S_2)$，$L(S_1|S_2) = L(S_1) \cup L(S_2)$，$L(S_1^*) = (L(S_1))^*$，$L(S_1 \odot S_2) = L(S_1) \odot L(S_2)$，$L(S_1^{\otimes}) = (L(S_1))^{\otimes}$.

シャフル演算は正規表現の記述能力を拡張しない．すなわち L_1 と L_2 が正規言語であるとき，$L_1 \odot L_2$ もまた正規言語である [5]．シャフル閉包演算は正規表現の記述能力を拡張する．シャフル表現は全ての正規言語と，文脈自由言語と文脈依存言語の一部を記述可能である．

3 イベント系列の記述

この節では，シャフル表現を用いた代表的な非同期問題の記述例を示す．

キュー問題

無限個のデータを格納できるキューに対して，in イベント（末尾にデータを追加する）と out イベント（先頭のデータを取り出し，それをキューから削除する）が非同期並行的に実行されることを考える．このとき，キューにデータが一つ以上存在するときのみ，out が可能でなければならないというのがキュー問題である．これは，任意の実行時点で in の個数 $\geq$ out の個数を満たすことと同等である．

in イベントを i，out イベントを o とすると，キュー問題の形式的記述は次のようになる．

キュー問題の形式的記述

$$(io)^{\otimes}$$

これにマッチするイベント系列の例として $\{\lambda, io, iioo, iioioo, ioiiioooo\}$ などがある．また，満たさない系列の例として $\{oi, iooio, ioiiooooo\}$ などがある．

より実用的な問題として，マルチユーザシステムにおけるログイン問題や，ネットワークパケットの受信・

*Description of Asynchronous Event Sequence by Shuffle Expression.

†Shinya Abe, Tokyo Metropolitan Industrial Technology Research Institute

転送を扱うパケットリレー問題なども，キュー問題と同様にシャフル表現によって記述可能である．

read/write 問題

高々一つのデータを格納できる共有バッファに対して，read イベント（データの読み込み）や write イベント（データの書き込み）が非同期並行的に実行されることを考える．このとき，read と write，write と write が同時に起こってはいけないというのが read/write 問題である．

read イベントの開始を r，read イベントの終了を r'，write イベントの開始を w，write イベントの終了を w' とすると，read/write 問題の形式的記述は次のようになる．

read/write 問題の形式的記述

$$((rr')^{\otimes} \mid (ww'))^*$$

この記述は，rr' はシャフル閉包演算により同時に実行されることが許されるが，ww' は独立して実行されなければならないを示している．

これにマッチするイベント系列の例として $\{\lambda, rr', ww', ww'rrr'r', rrr'rr'r'ww'ww'rr'\}$ などがある．また，満たさない系列の例として $\{wrr'w', rww'r', rwr'w'\}$ などがある．

セマフォ

セマフォは (s, p, v) で定義される同期機構であり，ある資源に対する同時アクセス数の上限値を規定したい場合に用いられる．s はセマフォ変数と呼ばれ，状態を保持しアトミックな操作が可能なカウンタであり，その初期値は同時アクセス数の上限値である．p はコマンド `when s>0 do s:=s-1` を示し，アクセス要求時に実行される．v はコマンド `s:=s+1` を示し，アクセス終了時に実行される．特に，上限値が 1 のときをバイナリセマフォ，上限値が 2 以上のときをカウンタセマフォあるいは汎用セマフォと呼ぶ．バイナリセマフォとカウンタセマフォの形式的記述は次のようになる．

バイナリセマフォの形式的記述

$$(vp \mid v)^*$$

カウンタセマフォの形式的記述

$$(vp \mid v)^{\otimes}$$

このように，バイナリセマフォのイベント系列は正規表現で記述可能だが，カウンタセマフォの場合はシャフル表現が必要である．

4 おわりに

非同期イベント系列を記述する場合，正規表現の記述能力では不十分であった．本稿ではシャフル表現による記述を提案し，代表的な非同期問題であるキュー問題，read/write 問題，セマフォの記述例を与えた．

今後は，イベント系列のマッチング方法を考える必要がある．文献 [6] でシャフルオートマトンの定義とその実現方法が示されているが，シャフル閉包演算の展開は膨大な時間計算量を要するので，場合によってはシャフル閉包演算の展開を枝切りするなどの対処が必要である．

5 関連研究

文献 [6] では，シャフル言語を受理するシャフルオートマトンの定義とその実現方法が述べられている．有限オートマトン上に以前の状態を記憶するためのマーカを置くことで，シャフルオートマトンが実現可能であることが示されている．しかしながら，システムの動作テストに応用することを考えた場合，シャフルオートマトンが入力記号列を受け取り，それが受理可能か否かを判定するまでに要する時間計算量を考慮する必要がある．

文献 [7] では，シャフル表現の実用的側面として，XML（eXtensive Markup Language）文書のパターンマッチングに対して，シャフル表現を応用した例が述べられている．パターンマッチ構文をシャフル表現によって拡張することで，RELAX NG のようなインターリーブ型のあるスキーマを持つ XML 文書に対しても，パターンマッチが可能になることが示されている．

参考文献

[1] A. C. Shaw, Software description with flow expressions, *IEEE Trans. Software Eng.*, Vol.SE-4, No.3, pp.242-254, 1978.

[2] 西谷 泰昭，伊藤 貴康，同期機構を持つ並行プロセスの記述能力について-フロー表現の拡張とその記述能力-，電子情報通信学会論文誌，Vol.J64-D, No.9，pp.831-838，1981 年．

[3] 西谷 泰昭，伊藤 貴康，同期制御された正規表現の記述能力，電子情報通信学会論文誌，Vol.J64-D, No.12，pp.1089-1096，1981 年．

[4] Vijay K. Garg, M.T. Raghunath, Concurrent Regular Expressions and their Relationship to Petri Nets, *Theoretical Computer Science*, 96:285-334, 1992.

[5] Joanna Jedrzejowicz, Andrzej Szepietowski, Shuffle languages are in P, *Theoretical Computer Science*, 250:31-53, 2001.

[6] Jedrzejowicz J., Structural Properties of Shuffle Automata, *Grammars*, Volume 2, Number 1, pp. 35-51(17), 1999.

[7] 紙名 哲生，玉井 哲雄，Lisp への XML 文書構造変換言語の埋め込みとそれのシャッフル表現への拡張，情報処理学会研究報告，2002-SE-136，pp.127-134，2002 年 3 月．