2T−1

イベントとアニメーション表示の対応付けによるLOTOSプログラムの実行の可視化

安本慶一　塩屋佳美　東野輝夫　松浦敏雄　谷口健一

大阪大学 基礎工学部 情報工学科

1 まえがき

通信プロトコルの動的振る舞いの把握には，仕様の可視化が有効である．本稿では，通信システムの仕様記述言語であるLOTOS[1]を用いた一つの可視化の方法を提案し，「5人の哲学者」のLOTOS仕様などを用いて評価実験を行なう．

2 可視化の要求とその実現法

2.1 可視化の要求

通信プロトコルなどの動的振る舞いを記述した仕様(被可視化仕様)が与えられ，その可視化を行なう際には，次のような要求が生じる．

- 被可視化仕様における特定のイベントの実行前後および実行時に，あらかじめ用意しておいたアニメーションを表示したい．
- 同じイベントが実行された場合でも，入出力されるデータ値に応じてアニメーション表示を変えたり，被可視化仕様から受け取ったデータをアニメーション表示したい．
- 被可視化仕様の構造を変更することなく，独立に作成した可視化プログラムを並行に実行することで可視化を実現したい．

また，可視化システムを実現する上で，

- 並行に動作する複数のプロセスをリアルタイムで可視化したい．
- システム自体を簡潔に作成したい．

などの要求も生じる．

2.2 可視化の方法

そこで提案する方法では，まずLOTOS[1]をアニメーション(文字や図形の表示，消去，移動，コピーなど)が記述できるよう機能拡張した．

LOTOSではイベントの時間的実行順序を指定することにより，システムの仕様を記述する．実行順序の指定にはアクションプレフィクス(a;β)，選択(α[]β)，並列(α|||β)，同期(α|[G]|β)，逐次(α >> β)，割り込み(α[> β)が使用できる[1]．

例えば，動作式がa!E1;b!E2;c!E3;exitであるLOTOS仕様Pに対して，a?x:t;(アニメーション記述)のような別プロセスPAを定義し，$P|[a]|PA$のように同期実行させることによって，イベントa!E1を可視化する．この時，LOTOSの同期機構[1]により，変数xに，a!E1の出力値E1が代入されるため，イベントの入出力値に応じたアニメーションの表示が可能である．

アニメーション動作は，プリミティブなアニメーション操作の系列として記述する．各アニメーションプリミティブ，例えば，「アニメーションオブジェクトAを座標P_1からP_2へt秒間で移動させる」は以下のようなLOTOSイベントとして記述する．

$AE!MoveCast(A, P_1, P_2, t)$

ゲートAEはアニメーションサーバ[3]への通信路であり，MoveCastという命令をアニメーションサーバに送信することを表す．

被可視化仕様の各イベントの実行の前後にアニメーション用のイベントが実行されるようにアニメーション記述部を作成し，それらの記述を我々が作成したLOTOSコンパイラ[2]およびアニメーションサーバ[3]を用いてマルチスレッド化されたオブジェクトコードに変換する．各アニメーション用イベントはアニメーションを表示するための適当なオブジェクトコードに置き換えられ，高速に実行される．

アニメーション記述言語としてLOTOSを用いた理由は以下の2つである．

- 既存のLOTOSコンパイラなどを用いて可視化システムを容易に実現可能．
- 被可視化仕様とのデータ交換に基づくアニメーション動作の変更などが容易に記述可能．

3 LOTOSプログラムの可視化例

3.1 5人の哲学者のLOTOS記述

哲学者の食事問題は排他制御の典型的な例題としてしばしば使用される．各哲学者は以下の制限のもとで，食事あるいは瞑想を繰り返す．

(1) 食事を行なうのは左右のフォークを持っている時のみ．

Visualization of LOTOS Specifications Executing Protocol Behaviors and Animation Behaviors in Parallel
Keiichi YASUMOTO, Yoshimi ENYA, Teruo HIGASHINO, Toshio MATSUURA and Kenichi TANIGUCHI
Department of Information and Computer Sciences, Faculty of Engineering Science, Osaka University
Toyonaka-shi, Osaka 560, Japan

(2) 瞑想を行なうのは左右どちらのフォークも持っていない時のみ.

哲学者の動作 (イベント) を「左/右のフォークを取る (lf!take/rf!take)」, 「左/右のフォークを戻す (lf!release/rf!release)」, 「食べる (eat)」, 「瞑想する (meditate)」とすると, その挙動は LOTOS では例えば以下のように記述できる.

```
process Ph[lf, rf]: noexit :=
  hide eat, meditate in
 ( lf!take; rf!take; eat;
     rf!release; lf!release; exit
[] meditate; exit ) >> Ph[lf, rf, eat, meditate]
endproc
```

一方, 各フォークは「取る」「戻す」の繰り返しであるから, 以下のように記述できる.

```
process Fork[f]:noexit :=
  f!take; f!release; Fork[f]
endproc
```

よって, 5 人の哲学者の動作と 5 つのフォークの動作を 5 つのフォーク (f1, ..., f5) で同期させることによって, この問題の LOTOS 記述が完成する.

```
specification Philosophers[f1,f2,f3,f4,f5]:noexit
behavior
  (Ph[f1,f2] ||| ... ||| Ph[f5,f1])
|[f1,f2,f3,f4,f5]|
  (Fork[f1] ||| ... ||| Fork[f5])
endspec
```

3.2 可視化用アニメーション記述

Ph[f1,f2] を実行する哲学者の左側のフォーク (f1 とする) に関するイベントを本手法によって可視化してみよう. 今, f1!take (f1!release) が実行された時, フォークの形をした物体がウィンドウ上をある座標 Pos1 から Pos2 まで t 秒間で移動させたいものとする. フォークの移動アニメーションは例えば以下のように記述できる.

```
process MvFork[AE](Pos1,Pos2:pos_t, t:time)
  :exit :=
  AE?fid:obj[fid=CreateCast("fork.xbm")];
  AE!MoveCast(fid, Pos1, Pos2, t);
  AE!DestroyCast(fid); exit
endproc
("fork.xbm"はフォークを描いたビットマップファイル)
```

始点, 終点の座標をプロセス MvFork のパラメータとして設定することによって, 5 本のフォークのアニメーションを 1 つのプロセスとして簡潔に記述できた.

次に上記のアニメーションをイベント f1!take (f1!release) の実行時に表示させるため, 同期点 (f1?x:fk_t) を設定する. ここでは, 被可視化仕様で実行されたイベントの出力値 (take, release) をアニメーション記述側で受け取り, 「取る」と「戻す」に対応するアニメーションをプロセス内で選択実行できるよう記述している.

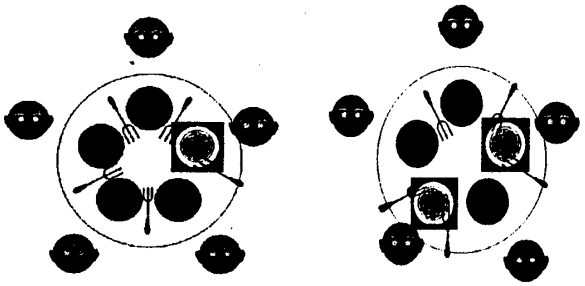

図 1: 「5 人の哲学者」の可視化例

```
process ForkA[f,AE]: noexit :=
  f?x:fk_t;
  ([x=take]-> MvFork[AE](Pos1,Pos2,t)
[] [x=release]-> MvFork[AE](Pos2,Pos1,t)
  ) >> ForkA[f,AE]
endproc
(fk_t は値 take, release をとる変数の型)
```

("[条件式]− >" はガード [1] と呼ばれ, 条件が成立する時のみ, その後の動作式が実行可能)

最後に元のプロセスとゲート f1 に関して同期させることにより, 可視化プログラムが完成する.

```
process Fork'[f1,AE]: noexit :=
  Fork[f1] |[f1]| ForkA[f1,AE]
endproc
```

同様に他のフォークや, 他の「食べる」「瞑想する」などのイベントも容易に可視化可能である. 上記手順で作成した可視化プログラムから本システムを用いて導出したオブジェクトコードを実行した様子を図 1に示す.

4 評価と今後の課題

図 1に示すように, 可視化により各フォーク, 哲学者の動作が容易に把握できるようになった. 可視化プログラムは, 我々の作成した LOTOS コンパイラ [2], アニメーションサーバ [3] を用いることによって, マルチスレッド化プログラムとして高速に動作する. アニメーション記述に LOTOS を用いたため, 被可視化プログラムの動作に従った可視化が容易に実現できた. また可視化システムは前述の LOTOS コンパイラに若干の変更を加えることによって実現した. 以上より, 本手法は既存 LOTOS 仕様の可視化に有効であると思われる.

参考文献

[1] ISO: "LOTOS - A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour", IS 8807 (1989).

[2] 安本慶一, 由雄宏明, 東野輝夫, 谷口健一: "マルチスレッド化オブジェクトコードを生成する LOTOS コンパイラの試作", 情処研報 (94-PRG-18)(1994-07).

[3] 城島貴弘, 松浦敏雄, 谷口健一: "対話型アニメーション用サーバの実現", 情処全大 (3R-10)(1994-09).