マルチメディア通信と分散処理 79－13
（１９９６．１１．１５）

時間表現を含む分散システムの仕様記述法とそのテストの実現

岡田 康治
kokada@etl.go.jp
電子技術総合研究所 情報アーキテクチャ部
〒305 茨城県つくば市梅園 1-1-4

概要: システムの高信頼化のため、分散システムの形式仕様記述の重要性が認識されて、記述法が研究されているが、時間概念を的確に表現できる方法がないことが問題となっている。プロセス代数に基づく言語の時間拡張が試みられているが、記述性、理解性に対する危惧がある。
本稿では、より古典的な方法である、有限状態機械モデルに基づく方法の時間拡張を検討している。時間付き有限状態機械モデル TFSM を定義し、それの具体形を代数仕様言語 OBJ で記述し、更に分散システムのテストシステムを作成して、実行例等を示し、本モデルに基づく仕様記述が、有用なものであることを示した。

Formal Specifications with Time for Distributed Systems and Its Application to Interoperability Testing

Koji OKADA
kokada@etl.go.jp
Computer Science Division, Electrotechnical Laboratory
1-1-4 Umezono, Tsukuba, Ibaraki 305, JAPAN

Abstract: Formal description techniques for distributed systems have been developed and many specifications in their languages have been accumulated, lacking in the concept of absolute time.
This paper tries to develope a description technique with absolute time by augmenting the finite state machines. A new model TFSM (Timed Finite Stete Machine) is introduced and realized by using the OBJ algebraic specification language as a user language. A test systems generator in OBJ is also shown with results of testing for a time sensitive protocol.

1 はじめに

1.1 分散システム仕様記述のモデル

分散システムを高信頼化するためには、その仕様を形式的に記述しそれに基づいてソフトウェアを開発することが必要だと考えられ、様々な仕様記述法が研究されてきている。分散システムの仕様を記述する有力な方法として、有限状態機械モデルに基づくものとプロセス代数モデルに基づくものがある。

分散システムの振る舞いが、有限状態（遷移）機械で自然に記述できるということは、古くから多

くの技術者が気付いていた。この考えに基づく多くの研究を集大成する形で1989年7月にISO国際規格となったのが形式記述技法(formal description technique、以下FDT)Estelle[2]である。Estelleは、チャネルやSAPなどのキーワードを持つ、いわば通信プロトコル専用の言語であるが、しかし、有限状態遷移機械モデルに基づいて分散システムを仕様記述するのにこれらに依らねばならないことはなく、より一般的な仕様記述言語を用いても可能で、代数仕様言語のASL[5]やOBJ[1]、モデル指向型仕様言語のZ記法を用いた分散システムの仕様記述も行われている[6]。

プロセス代数モデルにおいては、2つのプロセス間に起こるイベントの時間順序について述べることによってシステムの振る舞いを高い抽象度をもって定義する。種々のものが提案されているが、代表的なCCSを元にして、これも1989年の2月にLOTOS[3]がFDTとして国際規格になっている。

1.2 時間の表現の必要性

これらFDTsや種々の記述法を得て、仕様記述技術は発展期に入るべきであるが、問題点がひとつある。それは「時間」の記述である。「何単位時間内にデータの授受を終える」、「何ミリ秒後に再送する」など、時間の記述は分散システムの定義にとって必須ともいえる問題であるが、上記両モデルとも、その形式的な意味モデルの性質から、イベントの時間的な順序を述べるのみで、時刻や絶対的な時間量を扱えない。Estelleには、delay節というものがあって遷移の発火の条件の一つとして、「delay時間」を述べるのに使えるが、時間の取り扱いはここだけのごく限定的なものである。既発表の仕様の多くは、FDTの構文要素や、記述のある概念単位を時間として「解釈」して間接的に表現しているか、コメント的な扱いとして消極的に表現するかしている。

ISO/IEC JTC 1/SC 21/WG 7に「LOTOS拡張(Enhancements to LOTOS)」というプロジェクト[4]があり、設けられて既に満3年になるが、この活動で検討されている各拡張領域のうち、最重要視されているのが時間の導入である。現在、時間付きLOTOSの候補として絞り込まれつつあるのは、ベルギーとスペインから提案されたTE-LOTOS([4]のAnnex C)というものであるが、これは、現LOTOSに対する上位互換性を目指し、時間関連のアクション（演算子）を付加し、それらに対する公理と推論規則を与えるものである。

時間概念を付加したプロセス代数には、大きな期待が寄せられているが、TE-LOTOSを観察する限りにおいて、アクションの選び方や意味の与え方に恣意的なところがある、意味定義がより難解である、処理系の作成が難しそうで実行も重そうである、などの批判ないし危惧がある。E-LOTOSプロジェクトも遅れ気味であり、言語の完成、さらにはその普及ということになるといつになるか、現在は目途が立たない状況である。

1.3 時間付き有限状態機械モデル

そこで、本稿では、表現しやすく理解しやすい仕様と、それを実行系で実行させて低コストでテストシステムを実現することを目指して、有限状態機械の方の時間付きモデルを検討する。

2節で、有限状態機械を拡張して、データ授受などのイベント生起時に時刻が記録され、その記録に基づいて挙動が規定される、新しいモデルTFSMを導入する。ここでは、イベント生起時以外に時間の経過によっても機械の状態は遷移する。

このモデルに基づく機械の仕様を合成すれば、それは分散システム全体の仕様になる。それを実行することによる一種のテストシステムを作成した。具体的には、代数仕様言語OBJ[1]で記述し、その実行系を用いた。これを3節に述べている。

4節では、時間を陽に含むプロトコルの例を引いて、テストの実施の具体例を示し、このテストの方式が分散システムの設計段階に役立つ、事前評価の良い手段となる可能性を示した。

2 時間付き有限状態機械

2.1 時間付き有限状態機械モデル

まずグローバルな時間(時刻)を表すデータ型$T=\{t\}$を定義する。最小値を0とする自然数に最大値∞を加えたものとする。

時間付き有限状態機械(Timed Finite State Machine、以下TFSM)の仕様は以下で定義する6字組 $M=(S,V,P,D,R,init)$ で記述される。ここで、S は内部状態の有限集合 $\{s_1,...,s_n\}$、V は内部変数の集合 $\{v_i\}$、P は入出力ポートの有限集合 $\{p_1,...,p_k\}$、D は入出力ポートを介して他機械と交換されるデータの集合 $\{d_i\}$、R は機械の状態遷移を定義する遷移規則の集合 $\{r_i\}$、$init$ は機械の初期状態で、S と V の直積の一つの値である。

機械へのポート p を介した入力を $p?d$ で表す。機械からのポート p を介した出力を $p!d$ で表す。S と V の直積 $< S,V >$ を、拡張した内部状態とよぶ。遷移規則 r は

$$r:\ (< s,v >, p?d, t) \Longrightarrow (< s',v' >, p'!d', t')$$

の形をしている(ただし、$t \leq t'$、かつ、$p?d$ と $p'!d'$ は空である特別な値 ϕ を許容するものとする)。

2.2 時間付き有限状態機械モデルの解釈

データ D としては、$ConReq$ だとか ACK などのように信号的に考えられているものと、数値など狭義のデータを一緒にしたものを想定している。

遷移規則で、遷移は一般的には出力を伴うことを示している。もし複数の出力をさせたいときは、A への出力 $p'!d'$ を行う遷移を起こし、続いて時間をおかず B への出力 $p''!d''$ を行う遷移を起こすという記述でよいであろう。

遷移規則で、$t=t'$ のときは、時間を要しない遷移の規則であると考える。遷移は一般に機械への入力によって起こされここで、時間部分が変数 t のままで、かつ、s',v',p',d' が t に依存しないならば、それはこの遷移が時刻と関係なく起こり、かつ所要時間も0であるということをあらわしていることになり、すなわち、時間概念を付加しない元の状態遷移機械の遷移規則である。その意味で、TFSMは時間の概念のない原形の有限状態機械を含んでいる。

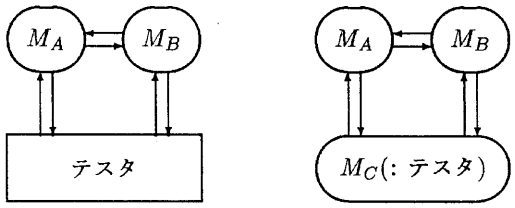
入力 $p?d$ が空 ϕ の場合は、遷移は入力がなくても起こり得ることを示している(sponteneousな発火)。ある時刻 t になったらある信号を発するというような動きがこれで記述しうる。

3 テストの実現

3.1 形式仕様の駆動法

形式仕様を書く目的のひとつに、シミュレーションをしたい、仕様を模擬的に実行して振る舞いについて知りたい、ということがある。そうして得られた挙動を調べて仕様の正当性、妥当性を確認することは、ソフトウェア開発過程の中の有効な一段階である。

実際のプロトコルエンティティの適合性試験などにおいては、テストシステムが、テスト列などを用意して、回線等を介して実時間にデータのやりとりをするが、前小節で定義したものはTFSMの仕様(文書)であって、これと異なるので、「記述」されている時間などを記述として利用して、そのまま自動的に実行してしまう方法を考えるべきである。



(a) テストシステム　(b) TFSM分散システム

図1 TFSMモデルによるテストの実現

TFSMを動かすにはこれにデータを与えて受け取ってやるテスタが必要である(図1(a))。しかし、TFSMは、データを予定されたスケジュールに従って自律的に送出する等の能力を備えているので、このテスタは一つのTFSMとみなして差し支えないので、一般の分散システムの形態で考えて良い(図1(b))。本節の以下においてもテスタという特別な概念を用いることなくあるシステムの振る舞いをテストしている。

いくつかのTFSMが互いに結合された分散システムの挙動を知るためには、各TFSMの仕様を調べて、最も早い遷移を検出し、その遷移を「起こさせれ」ばよい。すなわち、1) 全体系の時計をその時刻まで進め、2) 拡張された状態の変化を起こさせ、3) 出力を正しい送り先に配ればよい。この一回の遷移に相当する基本的な操作を繰り返

すことによって、そのシステムにおいて最終と規定されている状態まで進めばよい。これをたとえば関数型言語で再帰的な関数として定義することはたやすい。

いま、TFSM の A, B, C がしかるべく結合された分散システムがあるとして、その仕様は、MA, MB, MC、その初期値はそれぞれ、MA_0, MB_0, MC_0、その時刻 t_0 からの振る舞いの最終状態がどうなるかを知りたければ、次のようにすればよい。

1. 各 MX と時間 T の要素を引数にもつ、sys という名の構成子を定義して，ある時刻の全体系のコンフィギュレーションを表す、複雑なデータ型 $System$ を構成する。

$$sys : MA, MB, MC, T \rightarrow System$$

2. $System$ を引数とし、$System$ を値とする、全体系の最終状態を計算する関数 $proceed$ を定義する。

$$proceed : System \rightarrow System$$

3. $proceed(MA_0, MB_0, MC_0, t_0)$ の値を計算する。

普通は、最終状態でなく途中の様子が知りたいものであるが、そのときは、$System$ をもう 1 引数増やし、系の遷移のログを蓄積記録していくように改めればよい（後述）。

3.2 OBJ 言語による実現

筆者らは前小節までに示したことを代数仕様言語 OBJ を用いて実現し、様々に実験してきた [7, 8]。個々の TFSM の仕様は、たとえば各プロトコルなどごとに記述するが、それらを結合させたテストを実行する部分、テスト駆動系は、汎用性をもった一つの記述があるだけである。これに具体的なプロトコル (TFSM) の仕様を当てはめればそのプロトコルのテストが出来るので、この OBJ による駆動系は、プロトコルテストの生成系になっているといえる。駆動系の記述のうち、最も重要なオブジェクト（OBJ の記述単位）は SYSTEM といい、次の様な構文をしている。

```
obj
  SYSTEM[ A B :: USER, C :: SERVICE] is
  sort SysLog . sort System . sort  Log .
  ...
  op  slg : System Log -> SysLog .
  op  sys : User.A User.B Service Time
            -> System .
  op  proceed : SysLog -> SysLog .
  op  s-proceed : Time System Log
                  -> SysLog .
  ...
  eq  proceed(slg(sys(A1, B1, Serv1, T1),
                  LOG1))
      = if (fin?(A1, T1) and fin?(B1, ...
        then slg(sys(....), LOG1)
        else s-proceed(...)
endo
```

ここで、System はパラメタ型オブジェクトといい、A, B, C の 3 つの形式パラメタを実オブジェクトで実例化すればいいようになっている。A と B は USER という型に属し、先述のテスタ（テストケースの列を保持するもの）に相当する。C は SERVICE という型に属し、テスト対象の分散システムに相当する。これらを統合した全体系である System と遷移のログ Log を併合した SysLog が proceed という関数によって計算されるようになっている。

この形式パラメタ C に対して実在のオブジェクトを選ぶことによって、たとえば、プロトコルエンティティ、ネットワークアーキテクチャのある層のサービス、ネットワークなど、いかなる分散システムでもテストすることが出来る。

4 Sliding Window Protocol についての実験

OBJ の記述による分散システムのテストの一例として、Sliding Window Protocol（以下 SWP）[9] についての実験を示す。

4.1 SWP の問題

SWP の基本構造を図 2 に示す。

SWP では、2種のプロトコルエンティティ Transmitter と Receiver が協力しあって、データの漏れ等の転送異常が生じる可能性がある不確実な双方向通信路 Medium の上に、確実な、順序を保存したデータ列の片方向転送を実現する。

両者は、Sliding Window と呼ばれる長さに制限のあるテーブルを用いて、各データが転送に成功した（Ack を得た）か否かを管理しながら、処理を進める。返事がないデータに対して一定時間後に再送する点において、時間概念を陽に含むプロトコルである。

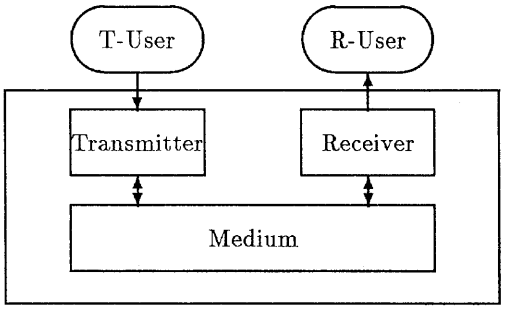


図2 SWP の構造

4.2 SWP の OBJ による記述

まず、Transmitter、Receiver、Medium を表現した各オブジェクト、TRANSMITTER、RECEIVER、SWP-MEDIUM を作成した。ついでこれらを合成して、SWP のサービスに相当するオブジェクト SWP-SERV を作成した。これがテストの対象である。テスタに変わるものとして、送り手ユーザ T-User、受け手ユーザ R-User を表現したオブジェクト T-USER、R-USER を記述した。これらの内容の中心部分は、適合性試験等におけるテストケースに相当するもので、テストの度ごとに書き換える（本例題は特殊で、送り手側のみ）。以上の準備を経て最終的な記述は先述のパラメタ型オブジェクト SYSTEM の実例化で得られる。

```
make  SWP-SYSTEM  is
  SYSTEM
    [ view to  T-USER  is
      sort User  to T-User . … endv,
      view to  R-USER  is
      sort User  to R-User . … endv,
      view to  SWP-SERV is   … endv]
endm
```

その遷移を遷移規則の集合で与えられているので、TFSM の振る舞いは、一般的には、非決定的である。SWP の例題における通信路 Medium は、非決定的にデータを落とす性質を持っているが、1)（現在の OBJ の実行系を用いる実現法では、）本当に非決定的に実行することは難しい、2) データを落とす確率を「決定的に」管理したかった、の理由から実際には乱数生成によって通信路の振る舞いを制御した。

4.3 SWP のテストの実験

もし、送り手ユーザ T-USER にある時刻が来るとデータを次々に出力するような遷移を定めてあれば、上記の SYSTEM を実例化したオブジェクトを OBJ の実行系で実行することによって、SWP の動作の様子をログの中に得ることが出来る。すなわち、(OBJ で記述された)SWP がどのように振舞い、最終的にどのような状態にいたるかを知ることができ、記述した内容が意図通りのサービスを提供するものであるかを確認することができる。また、各データがユーザから層へ送られた時刻、層からユーザへ送られた時刻、セッションが終了した時刻も見ることができるので、記述したシステムのいわばパフォーマンスを知ることができる。これによって、両プロトコルエンティティに与えられるパラメータ（window サイズや timeout までのリミットの値など）の適正値を求める目安を得ることも可能である。

実行の例のごく一部を示す。1, 3, ... という時刻に発せられた 10 個のデータがそれぞれ、順に、10, ..., 122, 188, 188, 250 という時刻に転送されたことを読みとることが出来る。

```
reduce in SWP-SYSTEM :
      proceed(slg(sys(t-user(Tinit ;
        is(in-u((ut('data1)),1)) ;
        is(in-u((ut('data1)),3)) ;
   ..........................
result SysLog:
    slg(sys(t-user(Tinit),
```

```
        r-user(Rinit ;
    os(out-u((ur('data1)),10)) ;
    ......
    os(out-u((ur('data7)),122)) ;
    os(out-u((ur('data8)),188)) ;
    os(out-u((ur('data9)),188)) ;
    os(out-u((ur('data10)),250))),
......
```

4.4 SWP のテストの利用

通信路 Medium のデータ漏洩率、または sliding window の長さを様々に変えて、それが転送効率にどう影響するかの実験をした。次に、Transmitter 側での時間切れによる再送をバッファ内の最初のデータに関してのみ2回行なう改定版を作成して、効率を比較する実験を行なってみた。以上のようなプロトコルの振る舞いに関する仕様の実行の結果については、稿を改めて、報告したい。

このような形式仕様の実行は、ソフトウェア作成過程において一般に（ラピッド）プロトタイピングと呼ばれているものと同質のものであり、機能が複雑である、分散システムの設計段階での事前評価に期待されているものである。特に、時間の要素を直接に表現できる本記述技法は、時間が重要な要素となりがちな、分散システムの仕様記述に有用なのではないかと考えられる。

5 あとがき

本稿では、表現しやすく理解しやすい仕様と、それを実行系で実行させて低コストでテストシステムを実現することを目指して、有限状態機械モデルを拡張した時間付きモデル TFSM 導入した。さらに、このモデルに基づく仕様を対象に模擬的に実行することによる一種のテストシステムを作成した。具体的には、代数仕様言語 OBJ で作成し、そして実際のプロトコル SWP の例の実験を示したが、本稿で述べようとしたことは、有限状態機械モデルの時間拡張の有益性、特に容易性であり、一般的なものである。

TFSM モデルにおいては、時間 T は、一種類のグローバルなものとした。これは、分かれて存在している各システムが、同じ時刻を指す時計をみることが可能であることを意味する。これが当てはまらない応用も多くあるはずで、実用化のためこの課題をどう進化させるかは重要であろう。次の課題として、例えば、階層的な分散システムで、子プロセスは親プロセスの時計を見ることが出来るが兄弟の時計は見られない、などのモデルを検討したい。

本稿では、TFSM モデルに基づく機械を適当にポート間を固定的に結線して合成したシステムを調べた。すなわち、機械の結合は個々の機械の振る舞いの外にある。機械の起こす動きの一部としてシステムの構成を能動的に変更するようなモデルを考察することは今後の課題としたい。

参考文献

[1] Goguen, J.A.,Winkler, T.,: Introducing OBJ3, Technical Report SRI-CSL-88-9, SRI International, Computer Science Lab, 1988.

[2] ISO: Estelle – A formal description technique based on an extended state transition model, ISO 9074, 1989.

[3] ISO: LOTOS – Formal description technique based on the temporal ordering of observational behaviour, ISO 8807, 1989.

[4] ISO/IEC JTC 1/SC 21/WG 7 N1053: Revised Working Draft on Enhancements to LOTOS (V3), 1996.

[5] 東野輝夫, 関浩之, 谷口健一: 代数的仕様から関数型プログラムの導出とその実行, 情報処理, Vol.29, No.8, 881-896, 1989.

[6] 水野忠則（監）： プロトコル言語, カットシステム, 1994.

[7] Okada, K., Ishigamori, M., Futatsugi, K.: Systemtic Construction of Services from Protocol Specifications by a Parameterized Description according to the OSI Layered Structure, JWCC-6: Joint Workshop on Computer Communications, 347-354, 1991.

[8] Okada, K.: Realizing Interoperability Testing on Formal Specifications, 情処学会マルチメディア通信と分散処理研究会, 72-3, 1995.

[9] Turner, K. J.(ed.): Using Formal Description Techniques, John Wiley & Sons, 1993.