分散システムの設計における形式記述技法の適用とその実践

技術解説

東　野　輝　夫*

Formal Description Technique for Designing Distributed Systems

Key Words：FDT, Distributed Systems, Reliability, Verification

1．はじめに

近年の計算機ネットワークの発展に伴い，様々なネットワーク関連製品が開発されている．それらのシステムは年々高度化され，複雑になってきている．また，「情報家電」という言葉に代表されるように，家庭内の様々な電気製品にも通信機能が組み込まれ，インターネット回線などとも容易に接続できるようになってきている．自動車の中にインターネット回線を引いて，携帯電話などを介して様々なネットワークサービスを提供しようとする試みも進められている．このように，通信機能の組み込まれた電気製品が数多く開発されてきている．ごく近い将来，電話やインターネット，携帯電話などを介して自宅やオフィスの家電製品を遠隔制御できるようになると考えられる．しかし，パソコンなどのソフトウェアと違って，頼みもしないのにお風呂が空焚きになったり，小さな子供の寝ている部屋の室温が異常に高くなったり，などといったことが起きれば大問題である．もちろん，こういった装置にはフェイルセーフ機構が働くので，簡単にはそういった事態は生じないと考えられる．しかし，そういった装置に万一異常が発生した場合，「製造物責任」を問われるかもしれない．パソコンソフトなどソフトウェアの分野では，バグフィックス版CDの配布で問題解決が計られることも多い．しかし，高信頼性が要求されるネットワークシステムや通信機能が組み込まれた家電製品など組み込みシステムではそうはいかない．その意味で，上述のような分散システムの設計開発の分野において，信頼性の高いシステムを設計するための技術は今後ますます重要になってくると考えられる．

FDT(Formal Description Technique：形式記述技法)は，このような複雑なシステムの仕様を，あらかじめその意味が定義されている適当な言語で記述し，その記述から仕様の無矛盾性や曖昧性，完全性(記述に漏れのないこと)などを検証や試験手法を用いて解析することにより，開発するシステムの信頼性を向上させようとするものである．従来，これらの解析にはスーパーコンピュータを何日も占有しなければならない場合も多く，人工衛星のソフトウェアのように高度の信頼性を要求されたり，電話の交換機のように多人数の利用者があるような特殊なシステムを除いて，時間とコストの面で採算があわないことも多かった．しかし，最近は安価で高速なワークステーションやパソコンが登場してきたため，それらの計算機を複数台使って，検証や試験を効率よく行おうとする試みも数多くなされている．また，検証・試験手法自身の高速化も進んでおり，市販の計算機を数時間から数日間動かすことにより，実用規模の問題に対する検証や試験が行えるようになってきている．たとえば，(1) AT＆Tで開発した通信ソフトウェアやATM用の通信システム，(2) NASAの人工衛星用プログラム，(3) ジャンボジェットなどの航空機のフライト制御システム，などといった実用システムの仕様をFDTで記述し，機械的な検証・試験手法を用いることにより「数多くのバグが検出できた」とか，「致命的な誤りを発見できた」，「システムの重要な部分の安全性の検

* Teruo HIGASHINO
1956年6月8日生
1984年大阪大学大学院基礎工学研究科(情報工学専攻)博士課程修了
現在，大阪大学大学院・基礎工学研究科・情報数理系専攻・計算機科学分野，教授，工学博士，分散システム，計算機ネットワーク
TEL 06－6850－6590
FAX 06－6850－6594
E-Mail higashino@ics.es.osaka-u.ac.jp

証が行えた」，などといったFDTの適用事例が数多く報告されている．以下では，このような分散システムの信頼性向上のための形式記述技法について概説する．

2．形式記述技法

一般に複雑なシステムの仕様は，英語や日本語などの自然語で書かれてたり，理解を深めるために図や表を交えて記述されている場合が多い．自然語や図表による表現は，多くの人にとって馴染みやすく，理解しやすいものであるが，一方では相矛盾する記述をしてしまう場合や，記述に曖昧さが残る場合など，記述した仕様に問題がある場合も多い．しかも，そのような問題がある場合でも，自然語の記述からその矛盾などを見つけだすのはそれほど容易ではない．その結果，作成したシステムに検出困難なバグが内在したり，同じ仕様に基づいて作成された通信装置同士であっても異なるメーカーの装置間ではうまく通信できない，などといった問題が生じる．FDTは，このような複雑なシステムの仕様を，あらかじめその意味が定義されている適当な言語で記述し，その記述から仕様の無矛盾性や曖昧性，完全性(記述に漏れのないこと)などを解析できるようにすることにより，開発するシステムの信頼性を向上させようとする手法である．

特に，通信システムやグループウェアなどの分散システムや，ソフトウェアプロセスなど複数の人間がお互いに通信しながら並行に作業を行うような分散協調システムを設計する場合は，システム全体をどのようにモデル化するかのみならず，複数ノード(プロセス)間の通信や各ノード(プロセス)の仕様をどのように記述すれば良いかが重要になる．また，そのモデル化の善し悪しによって，無矛盾性や完全性などの解析が容易になったり難しくなったりする．

このように，FDT(形式記述技法)には(1)モデル化と(2)形式仕様記述，の2つの側面がある．通信システムなどの分散システムの仕様を記述する場合，(1)のモデル化は，例えば，各ノード(プロセス)の動作を有限状態機械や拡張有限状態機械などでモデル化し，システム全体をそれら有限状態機械群の並行モデルとしてモデル化したり，ペトリネットやプロセス代数などでモデル化したりする場合が多い．また，プロセス間通信は，非同期通信としてモデル化される場合が多いが，同期通信としてモデル化される場合もある．一方，(2)の形式仕様記述のための言語については，その構文と意味が厳密に定義されていれば，原理的にはどのような言語でもかまわない．しかし，一般に広く利用されている形式仕様記述言語を用いて記述する方が汎用性も高く，多くの人にとって分かりやすい．また，そのような汎用の形式仕様記述言語には，仕様記述のためのエディタや検証，試験，実装のためのツールが数多く開発されている可能性が高いので，システム開発の目的に応じて適当な形式仕様記述言語を選ぶことが望ま

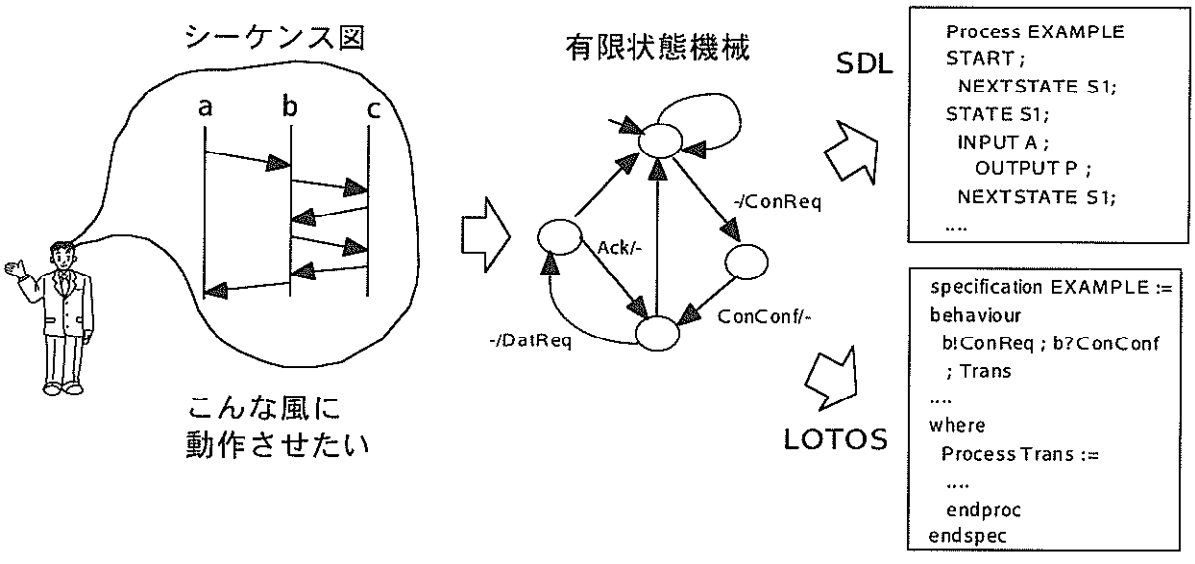

図1 分散システムのモデル化

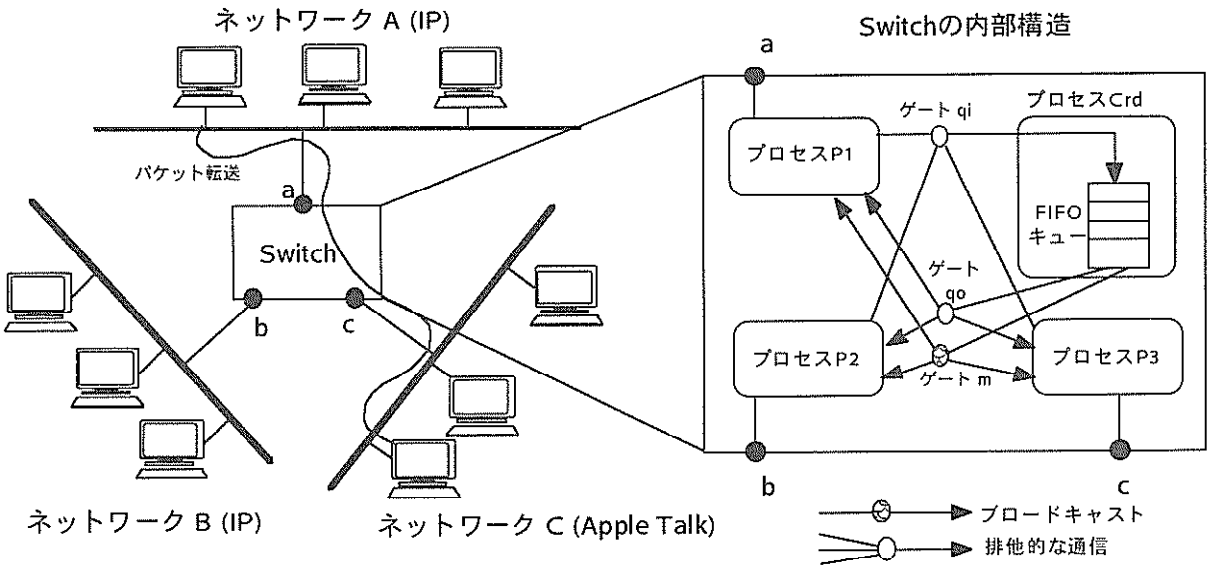

図2　ネットワークスイッチ

しい．例えば，通信システムの形式仕様記述言語としては，SDL[1]やEstelle[2]，LOTOS[3]，MSC[4](メッセージシーケンスチャート)などが考案されている[5,6]．交換システムなどのように，システムの動作が(拡張)有限状態機械としてモデル化され，プロセス間の通信が非同期通信としてモデル化するのが自然な場合，SDLなど有限状態機械をベースにした形式記述言語が有力である．一方，同期通信を用いて並行システムをモデル化する場合，LOTOSなどが有力である．また，ハードウェアの設計の分野では，VHDLやVerilog，SFLなどの言語があるが，これらも，ハードウェア設計用の形式仕様記述言語とみなすことができる．

3．システムの仕様記述

図2に示すような簡単なネットワークスイッチ(ルータ)の仕様をFDTで記述する方法について考えてみよう．図2のスイッチは3つの入出力ポート(ゲート)a, b, cを持ち，それぞれ異なるネットワークA, B, Cに接続されている．各ゲートの動作は他のゲートの動作とは独立して実行できることが望ましいので，a, b, cの3つのゲートをそれぞれP1, P2, P3の3つのプロセスによって制御することを考える．また，このスイッチでは，各ゲートから入力されたパケットは内部ゲートqiを介して共通のFIFOキューにいったん蓄えられるものとし，キューの先頭から取り出されるパケットは内部ゲートqoを介して順次該当するゲートを制御するプロセスに配送されるものとする．また，ゲートqi, qoは同時にはP1, P2, P3のいずれか1つのプロセスからしかアクセスできないものと仮定する．さらに，ブロードキャストの場合の出力ゲートとしてゲートmを用いる．ゲートmではFIFOキューを制御するコーディネータプロセスCrdから送られたパケットが3つのプロセスP1, P2, P3に同時にブロードキャストされるものとする．

このレベルでのネットワークスイッチの仕様を考える場合，ゲートmでのブロードキャストやゲートqi, qoでの排他的な通信のメカニズムはできるだけ抽象的な通信機構を持つ仕様記述言語を用いて簡潔に記述できることが望ましい．このような方針で仕様を書く場合，以下で紹介するLOTOSなど抽象的な通信機構を持つ仕様記述言語でシステム全体の仕様を記述することが望ましい．

LOTOS(Language Of Temporal Ordering Specification)[3]は，プロセス代数と呼ばれる数学的なモデルに基づく言語で，1989年にISOの国際標準として規格制定された．LOTOSでは，各入出力動作をイベントと呼び，各イベントはa?xやb!Eのようにゲート名と入出力データの組で構成される．また，イベントの実行順序を定めるオペレータとして，逐次実行(;, >>)，選択([])，非同期並列実行(|||)，同期並列実行(|[G]|)，割り込み([>)などのオペレータがある．このうち，同期並列実行を表す動作式

B1|[G]|B2では，B1とB2の動作の中で集合Gに属する動作(イベント)の実行は同期して並列に(同時)実行されなければならない．このような抽象的な通信機構を持つ仕様記述言語を用いると，例えば，前述のネットワークスイッチの仕様は

(P1|[m]|P2|[m]|P3)|[qi, qo, m]|Crd

のように記述できる．この例では，ブロードキャストのためのゲートmでの動作はP1, P2, P3, Crdの4つのプロセスが同期して実行しなければならないことを表し，ゲートqi, qoでの入出力動作は{P1とCrd}, {P2とCrd}, あるいは，{P3とCrd}の3つの組のいずれかの組で同期実行しなければならないことを表しており，上述のようなブロードキャストと排他的な入出力動作が簡潔に表現できる．

アルゴリズムやデータ構造の教科書ではCやPascal，あるいは，それらに近い疑似言語を用いてアルゴリズムを説明している場合が多い．その意味では，CやPascalも立派な仕様記述言語である．それらの言語の意味も形式的に定義されているし，何より，言語の形式的な意味定義を具体的に示さなくても多くの人が同じように理解できるという利点がある．しかし，上記のネットワークスイッチの例など少し複雑な分散システムの仕様を記述する場合，複数のプロセス間での通信を記述する必要が出てくる．もちろん，C言語などでもソケットやパイプを用いた通信機能を用いることができるが，それらの機能だけでは，上述のようなゲートmでのブロードキャストやゲートqi, qoでの排他的なプロセス間通信などを簡潔に記述できない．これは，C言語などのプログラム言語では，あまり高度な通信機能が利用できないことに起因する．SDLやEstelle, LOTOS，メッセージシーケンスチャートなど通信システム(分散システム)用の仕様記述言語を用いることにより，複数プロセス間の通信が簡潔に記述できる．

4．試験・検証技術

通信システムの分野では，作成したシステムが要求仕様を満足するかどうかの適合性試験(Conformance Testing)が重要視されている．適合性試験では，システムの仕様をいくつかの状態変数を含む拡張有限状態機械，あるいは，それらの組で記述し，その記述からシステム全体として実行可能なすべての全状態を機械的に構成し，それらの全状態から実行可能な遷移を少なくとも1回は実行するような試験系列を構成し，その試験系列を実際に実装したシステムに与えてシステムの誤りを発見する，などといった手法がよく用いられる．たとえば，Bell研やフランステレコム，オタワ大学などでSSCOP-ATMプロトコルや移動体通信システムの試験系列を自動生成し，開発したシステムのバグを発見するのに役立ったなどといった事例報告がある[7,8]．

ただし，試験手法はトレースした遷移系列の観測から発見できるタイプのバグの検出は得意であるが，「どのように遷移してもデッドロックしないか」や「ある性質が成り立つような遷移系列が存在するか」などといった実行可能なすべての遷移系列に依存するようなタイプの性質の検証は得意ではない．そういったタイプの検証問題に対しては，モデルチェッキング[9]などの形式的検証技術が有効であり，Affirma, CV, KRONOS, PVS, SMV, VISなど幾つかの有用なツールも開発されている．また，取り扱えるデータ型を整数型まで拡張しようとする研究(シンボリックモデルチェッキング[10])もかなり進んできており，omegaなどのツールも開発されている．さらに，ソフトウェア組み込みシステムでは，プログラムを一定のルールで抽象化して拡張有限状態機械に変換し，その上でモデルチェッキングを行うソフトウェアモデルチェッキングの利用も有効であり，Bell研などで幾つかのツールが開発されている(VeriSoft, SPINなど)．一般にモデルチェッキングでは，与えられた仕様から到達可能なすべての全状態を機械的に網羅し，それらの全状態に対してシステムがある性質を満たすかどうかを検証する．全状態を機械的に網羅しようとすると，すぐにその数が数十億通りなどといったオーダーになるが，最近は安価で高速なパソコンが登場し，それらのパソコン上で検証が可能になってきている．特に，システム全体を2の数十乗程度の状態数の有限状態機械としてモデル化できる場合，BDDベースのモデルチェッキング手法を用いてかなり安価に高速に検証できるようになってきた．一方でソフトウェア組み込みシステムなどでは，ソフトウェアの検証と同様の難しさがあり，ソフトウェアモデルチェッキングなどの手法に関してさらなる研究が必要である．

なお，時間制約を含んだシステムの検証，例えば，「どう遷移しても2秒以内に必ず指定された動作が実行されるか」といった性質の検証のため，時間オー

トマトン[11]などの実時間モデルとその上でのモデルチェッキング手法などが活発に研究されているが，実用化の観点からはもう少し研究が進んでいく必要があると思われる．

5．おわりに

安価なパソコンの普及と形式的検証技術の発展に伴い，ようやく実用的な問題に形式的検証技術を適用できる環境が整ってきたように思われる．もちろん，現状では，簡単に利用できるFDTやその処理系，ユーザインタフェースの充実した検証ツール，などが十分そろっているとはいえない部分もある．また，SDLやEstelle，LOTOS，MSCといった複数の言語が並行してISOなどで標準化される背景には，モデル化したい問題に多様性があり，対象とする問題の記述に適したモデルや言語を複数考えざるを得ないという側面もある．ソフトウェア開発者がどのモデルや言語を選べばよいのかや，どのようなモデル化が検証や試験に適しているのか，などに関する解説書やチュートリアル的な書物，実際の応用例にFDTを適用した結果に関する事例報告，などの充実が望まれる．さらに，ISOなどで標準化された言語の機能拡張などさらなる仕様記述能力の充実が望まれる．今後のこの分野の発展を期待したい．

参考文献

[1] ITU-T：“Functional Specification and Description Language”, Recommendation Z.100 (1992).

[2] ISO：“Estelle：A Formal Description Technique Based on an ExtendedState Transition Model”, ISO 9074 (1989).

[3] ISO：“Information Processing System, Open Systems Interconnection, LOTOS-A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour”, IS 8807 (1989).

[4] ITU-T：“Message Sequence Chart (MSC)”, Recommendation Z.120 (1992).

[5] 水野 忠則 監修：“プロトコル言語”，カットシステム社(1994).

[6] 佐藤，長谷川，東野，大蒔：“形式的手法による分散システムへの取り組み”，日本ソフトウエア科学会，講習会資料シリーズ，No.15 (1998).

[7] D. Lee and M. Yannakakis：“Principles and Methods of Testing Finite State Machines-A survey”, Proc. of the IEEE, Vol.84, No.8, pp.1090-1123(1996).

[8] A. Cavalli, B.-H. Lee and T. Macavei：“Test Generation for the SSCOP-ATM Networks Protocol”, SDL Forum'97 (1997).

[9] 平石，浜口：“論理関数処理に基づく形式的検証手法”，情報処理，Vol.35, No.8, pp.710-718 (1994).

[10] K. L. McMillan：“Symbolic Model Checking”, Kluewer Academic Publishers (1993).

[11] R. Alur and D.L. Dill：“A theory of timed automata”, Theoretical Computer Science, Vol.126, pp.183-235 (1994).