ソフトウェア工学 86－T３
（１９９２．７．２１）

状態遷移モデルをベースとした制御ソフトウェアのオブジェクト指向設計

Object-Oriented Design Method of Control Software
Based on State-Transition Model

村田智洋、赤津雅晴、栗原兼三（日立製作所システム開発研究所）

１．はじめに

高機能なマイクロプロセッサが潤沢に供給されるようになるにつれ、マルチマイクロプロセッサシステムにより高性能、高信頼なシステムを実現するアプローチが多くなり、システム制御ソフトウェア開発の規模や複雑性が増大してきている。[1],[2]

このような制御ソフトウェアの設計を正しく行なうには、各プロセスが並行作し、その結果としてシステムのグローバルな状態遷移仕様が正しく満たされることを設計段階で十分検証しておく必要がある。

制御ソフトウェアの設計を対象とするリアルタイムシステムSA(Structured Analysis)手法が提案されている。[3],[4],[5]しかし、並行動作プロセスのインタラクションの表現方法としては不十分な面がある。

本論文では並行動作するプロセスを互いに通信するオブジェクトととらえ、その仕様を状態遷移機械として記述することにより、制御ソフトウェアの設計と検証を支援する方法について述べる。

２．オブジェクト指向モデルによる制御ソフトウェアの設計･検証支援

2.1 制御ソフトウェアとオブジェクト

制御ソフトウェアを概念的に、次の３種類の要素オブジェクトで構成することを考える。

（１）リソースオブジェクト

共通資源として管理すべき状態を内部状態として持ち、メッセージまたはイベント受信を契機にして独立に内部状態を変化させるプリミティブなオブジェクト。

（２）写像オブジェクト

リソースオブジェクトの上位オブジェクトであり、独立に遷移する複数のリソースオブジェクトの状態変化パタンに応じ、必要なシステム内部動作を起動し、グローバルなシステム状態の遷移を制御する。内部状態としてシステムのグローバルな状態を表すシステム制御状態を持ち、リソースオブジェクトの状態変化パタンの組合せをシステム制御状態の遷移にマッピングする写像ルール群を持つ。写像オブジェクトの上位に写像オブジェクトを設けて良い。

（３）タスクオブジェクト

テンポラリな内部状態を有し、リソースオブジェクトの状態を参照更新し、互いに通信しながら自律的に動作する。

（４）システム動作モデル

(a)オブジェクト動作の関連

タスクオブジェクトはリソースオブジェクトに対してメッセージを出力することでリソースを確保し、その状態を参照更新しながら並行動作する。リソースオブジェクトは外部イベントやメッセージによりその内部状態を自律的に変更する。リソースオブジェクトはその内部状態を遷移した契機で、必要に応じて上位の写像オブジェクトに対しメッセージを上げる。写像オブジェクトは関連するオブジェクトの内部状態を参照し、自らのシステム制御状態を遷移し、遷移にともなって必要となる処理をタスクオブジェクトにメッセージで依頼する。タスクオブジェクトはシステム制御状態に関しては参照するのみで、自らシステム制御状態を変化させることはなく、状態遷移に付随して必要となる処理を行なった後、状態遷移の契機となるメッセージを写像オブジェクトに出力する。これらのオブジェクトの関連を図１に示す。

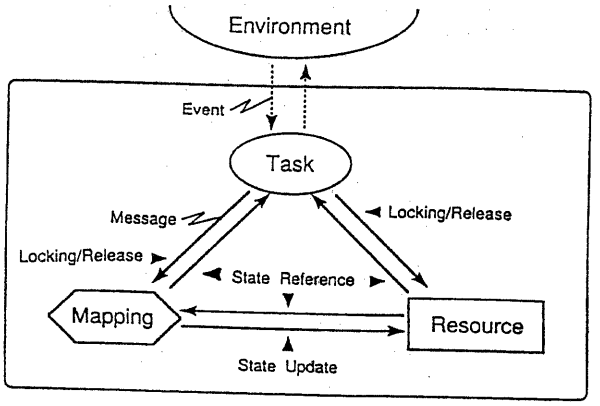


図１ オブジェクトの相互関係

〒215 川崎市麻生区王禅寺1099
TEL (044)966-9111 FAX (044)966-6832

(b)オブジェクトのロック

オブジェクトが他オブジェクトの内部状態を参照/更新する場合、対象となるオブジェクトのロックを取得する。他オブジェクトがロック中のオブジェクトの内部状態を参照/更新することはしない（ルール）。すなわち、オブジェクトのロックは、過渡的な内部状態を他オブジェクトから隠蔽する。ロックの取得は各オブジェクト内に後述のメッセージ式で記述する。また、動作中のオブジェクトは他オブジェクトに対し、自らをロック状態におく。

2.2 制御ソフトウェア設計手順

一般的には次のステップからなる。

[ステップ1]システム要求仕様の作成
制御システムが全体として満たすべき仕様を定義。

[ステップ2]オブジェクト仕様の記述
システム要求仕様を上述した3種のオブジェクト群の機能に分割し、各オブジェクトの仕様を記述。

[ステップ3]システム内部動作仕様の記述
オブジェクトの組合せでシステムとして実現されるべきグローバルな状態遷移仕様を記述。(本ステップは［ステップ2]と並行して進められる。)

［ステップ4] オブジェクト仕様の検証
オブジェクト仕様を連動させ、システム内部動作仕様が満たされることを検証。

[ステップ5]オブジェクトの実現
(S5-1)各オブジェクトのコーディングと単体テスト。
(S5-2)サブシステム機能を構成するオブジェクト群の組合せテスト。
(S5-3)システムテスト。

本論文における主題は[ステップ2]、[ステップ3]、[ステップ4]の支援である。以下ではまず、[ステップ2]におけるオブジェクト仕様記述方式SCOPE (Specification of Concur-rent Objects for Process Extraction)、[ステップ3]におけるシステム制御仕様記述方式 STM(State-Transition Matrix-diagram)について述べ、続いてディスク制御装置におけるデータ保証制御仕様をSCOPEとSTMにより記述した例を示し、最後に[ステップ4]において、オブジェクト仕様検証を支援するシステムADRES (Aid for Design of Real-time Embedded Software)の概要について述べる。

3．オブジェクト仕様記述：SCOPE

SCOPEにおける、上述した3種のオブジェクトの仕様記述について述べる。これらは全て状態遷移機械として記述される。
(記述例は5.1参照)

3.1 タスクオブジェクト記述

タスクオブジェクトの内部状態は局所的であり、許された内部状態のみが他のオブジェクトから参照できる。図2にタスクオブジェクトの構造を示し、以下これらの記述内容について述べる。

（1）遷移構造記述
（a）図的記述

内部状態遷移仕様は、ステート（円で示す）、トランジション（四角で示す）、アーク（矢印でしめす）の三つの要素を用いて図3に示すような状態遷移図として記述する。各トランジションの入出力アークは必ず一本である。

（b）リスト形式記述

階層構造化された遷移図はif then else型の制御構造となり正規表現によりトップダウンに記述できる。SCOPEでは遷移構造のリスト形式記述として正規表現を用いる。図3に対応する記述を図4に示す。

（2）遷移制御記述

遷移構造記述における各トランジション対応に、次の内容を記述する。

（a）遷移条件

タスク内部状態遷移のトリガとなる条件を後述するメッセージ式で指定。遷移条件が成立すると後述する行動記述部を実行した後、トランジションの入力側ステートから出力側ステートに状態が遷移する。遷移条件はその条件が指定されているトランジ

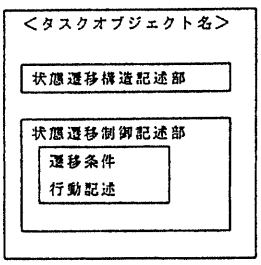


図2 タスクオブジェクトの構造

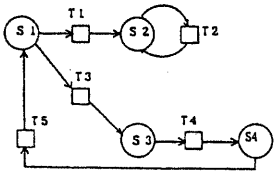


図3 状態遷移記述(図的記述)

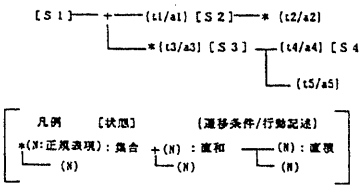


図4 状態遷移記述(リスト記述)

ションの入力側ステートが成立しているときのみ評価されるため、１つのオブジェクト内で同時に複数の行動記述部が動作することはない。トランジションに遷移条件が定義されていない場合、その遷移元状態が成立した時点で無条件に行動記述が実行され、タスク内部状態が遷移する。

（ｂ）行動記述

内部状態の遷移にともなって行なう処理を記述する。処理は、自然言語で任意の内容を記述した処理文をシーケンシャルに並べた擬似プログラムで記述する。また、他オブジェクトとのメッセージ通信を行なう場合は、後述するメッセージ式を独立した処理文として記述する。処理文は先述した遷移条件の成立時点で逐次実行されるが、実行がエラー終了した場合、残りの処理文の実行は中断される。この場合、状態遷移は起こらずエラー処理文に制御が渡る。エラー処理文には自タスクの内部状態を適当な状態に再設定するためのメッセージ式を記述することができる。

3.2 リソースオブジェクト記述

(1)状態定義

各資源の状態を構成する要素状態名とその値域を、資源ごとに定義する。定義したリソース状態は、メッセージ式により任意のタスクオブジェクト、または写像オブジェクトから参照更新できる。

（２）状態遷移定義

リソースオブジェクトはメッセージ(またはイベント)で指示された状態遷移をおこなうのみで、自ら状態遷移することはない。リソースオブジェクトの場合、イベントやメッセージ対応の状態遷移が中心であり、その遷移は一般に階層構造化された状態遷移図にはならない。SCOPEでは、可能な状態遷移を(メッセージパタン、遷移前状態、遷移後状態)の３つ組で定義する。なお、受信したメッセージとカレントなリソース状態が状態遷移定義にマッチしない場合は状態遷移は起こらない。

3.3 写像オブジェクト

(1) 記述モデル

相互作用をもつ２つのオブジェクトについて、その内部状態の集合をそれぞれα、βと表す。

$$\alpha=(\alpha_1、\ \alpha_2、\ \cdots\ \alpha n) \qquad [1]$$
$$\beta=(\beta_1、\ \beta_2、\ \cdots\ \beta m) \qquad [2]$$

一方、これらのオブジェクト状態の遷移に関連して変化するシステム制御状態の集合

$$\gamma=(\gamma_1、\ \gamma_2、\ \cdots\ \gamma i) \qquad [3]$$

を写像オブジェクトの内部状態とする。$\alpha\times\beta$における状態遷移とγの要素間の状態遷移との関係を、

$$f:\alpha\times\beta\rightarrow\gamma \qquad [4]$$

で記述する。ｆを写像ルールとよぶ。一方それに続くγの内部状態遷移は

$$g:\gamma\rightarrow\gamma \qquad [5]$$

により記述する。このとき写像オブジェクトの状態遷移は関数ｈ＝ｇ(ｆ)で定まる。

３つ以上のオブジェクトの相互作用を記述したい場合、オブジェクト群を２組に分け、それぞれの組について内部状態の直積をとり、それぞれを新たなオブジェクトの状態とする。オブジェクト状態の直積のサイズが大きい場合、各オブジェクトの状態集合をいくつかの部分集合に分け、各部分集合をマクロなオブジェクト状態とみなすことで、階層的にオブジェクト群の相互作用を記述する。

(2)写像ルールの記述

写像ルールは、下位オブジェクトの状態の組合せ条件と、それが満たされた場合の自オブジェクトの状態遷移を記述したルールの集合により与える。各ルールのIF部には下位オブジェクト状態の成立を参照するメッセージ式の論理積を、またTHEN部にはIF部成立時に自オブジェクトの状態更新を行なうためのメッセージ式をそれぞれ記述する。ルール群はいくつかのルールセットに分けることができ、リソースオブジェクトからの状態変化通知のメッセージ受信時に実行される。

3.4 オブジェクト関連図

オブジェクト間のメッセージ交信関係を陽に記述するものである。各オブジェクト間に必要なメッセージ更新パスを定義し、双方向通信に用いるパスにはチャネル名を、片方向通信に用いるパスにはバッファ名を定義する。定義したチャネル、バッファについては、後述のメッセージ式により、メッセージ交換の動作仕様を記述する。

3.5 メッセージ式

メッセージ式は上記オブジェクト間の通信仕様を記述するためのマクロである。メッセージ式にはZave、Fitzwaterらが提案した３種の双方向通信（Ｘ、ＸＲ、ＸＭ）[1]と、本論文で導入する４つの片方向通信（Ｓ、Ｃ、Ｒ、Ｅ）を用いる。

3.5.1 双方向通信

(1)１対１通信：Ｘα(ｍ)

仮想チャネルαに関してあるオブジェクトＡがＸを実行した時、他オブジェクトＢがすでにＸαを実行していれば２つのオブジェクトＡ，Ｂはメッセージｍ（ｍが空なら同期信号）を交換し、それぞれ次の状態に制御を移す。もしＢがＸαを実行してい

なければ、ＢがＸαを実行するまでＡの実行はブロックされる。なおＸはＸ自身および後述するＸＲ、ＸＭとメッセージ交換可能である。

(2)１対ｎ通信：ＸＭα(ｍ)

チャネルαに関してあるあるオブジェクトが実行したＸＭは、他オブジェクトが実行したＸα、ＸＲαとメッセージを交換する。他オブジェクトが実行したＸＭαとはメッセージを交換しない。それ以外はＸと同じである。

(1)１対１即時通信：Ｒα(ｍ)

チャネルαに関し、あるオブジェクトがＸＲαを実行した時、同じチャネルに関して他オブジェクトがＸα、ＸＭαを実行していれば２つのオブジェクトはメッセージｍを交換し、それぞれ次の状態に制御を移す。Ｘα、ＸＭαを実行しているオブジェクトがなければＸＲを実行したオブジェクトにメッセージｍが返される。なお、ＸＲ同志はメッセージ交換できない。

3.5.2 片方向通信

(1)メッセージ送信：Ｓβ(ｍ)

バッファβに関しＳを実行すると、βの内容をメッセージｍで更新する。βにリソースオブジェクトの状態名を指定すると当該状態の値をｍに設定する。

(2)照合付き送信：Ｒβ(ｍ1、ｍ2)

バッファβに関しＲを実行すると、βにｍ1と等しいメッセージが存在していれば真、存在していなければ偽の値が返る。結果が真の場合、Ｒβの実行によりバッファβ内のメッセージはｍ2に置き換えられる。ｍ2を略せば、内のメッセージは無効化される。(オブジェクトロック用メッセージとしても用いる。)

(3)メッセージ照合：Ｃβ(ｍ)

バッファβに関しＣを実行した場合、βにｍと等しいメッセージが存在していれば真、存在していなければ偽の値が返る。なお、Ｃの実行はβ内のメッセージには何の影響もおよぼさない。なおβにオブジェクト名を指定した場合、オブジェクトの内部状態をβ内のメッセージとして同様の動作をおこなう。

(4)イベント照合：Ｅβ(ｍ)

バッファβ（βは最大一つのメッセージを記憶する）に関しＥβを実行した場合、βにｍと等しいメッセージが存在していれば真、存在していなければ偽の値が返る。結果が偽の場合、βにメッセージｍが到着するまでＥβを実行したオブジェクトの実行はブロックされる。結果が真の場合、Ｅβの実行によりβ内のメッセージは無効化される。

４．システム内部動作仕様記述:STM(マトリクス型状態遷移図)

オブジェクトの集合としてシステム要求仕様が実現されることを検証するには、各写像オブジェクトの振舞を理解しやすい形で記述しておく必要がある。今、２つのオブジェクトの状態α、βとその相互作用を制御する写像オブジェクトの状態γに関し、

$$S=\alpha\times\beta\times\gamma \qquad [6]$$

をシステム状態と定義すると、写像オブジェクトの振舞はシステム状態の遷移として記述される。システム状態の遷移仕様をシステム内部動作仕様とよぶ。以下、システム内部動作仕様をシステマティックに記述する一方法を述べる。(記述例は5.2参照)

(1)マトリクス型状態遷移図(STM)

システム状態の遷移を図５に示すようにフラットな状態遷移図として記述すると、状態遷移図が平面的に任意の構造で拡大してしまう。状態数が小さい場合は特に問題がないが、状態数が大きくなると遷移図の意味する内容が非常に理解しづらくなる。そこで、システム状態を構成する３つの状態α、β、γをそれぞれ３つの座標軸上に割り当て、図６のように状態遷移を３次元空間上に構造化して表現する。図６で点線で囲まれた状態群をドメインと呼ぶ。

図６において３次元空間を構成する座標軸α、β、γのうち、２つのオブジェクト状態を表す座標軸α、βからなる平面をキー平面と定める。キー平面に直行する直線上に並ぶ３次元空間上のシステム状態群をクラスタリングし、それらの状態におけるシステム制御状態γの要素の遷移をキー平面上の対応するドメインに写像すると、図７に示すような状態遷移表現が得られる。これをマトリックス型状態遷移図(STM)とよぶ。α、βのサイズが大きい場合、α、βの要素をいくつかの部分集合に分け、それをα、βのマクロな要素としてSTMのキー平面を構成し、更に各ドメインを部分集合要素で展開して詳細STMのキー平面を構成するという方法を繰り返すことでSTMを階層的に記述できる。

STMを用いることにより、システム状態の遷移記述がドメイン内遷移とドメイン間遷移に分けて構造化されるため、フラットな状態遷移図(図５)に比べ、複雑なシステム内部動作仕様を平面（設計用紙）上に見通しよく記述できる。

(3)STMによるシステム内部動作仕様記述

システム内部動作仕様の記述は、タスクオブジェクトの内部状態をブラックボックスとみなし、システム内外で発生するイベントメッセージに対するリソースオブジェクト群と写像オブジェクト群の状態遷移の関連をSTMで記述することにより行なう。

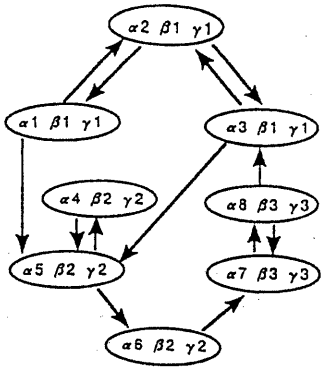


図５ 構造化されていない状態遷移図

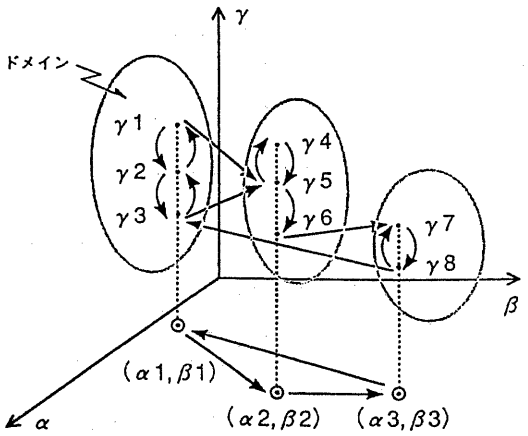


図６ 状態遷移の３次元構造化

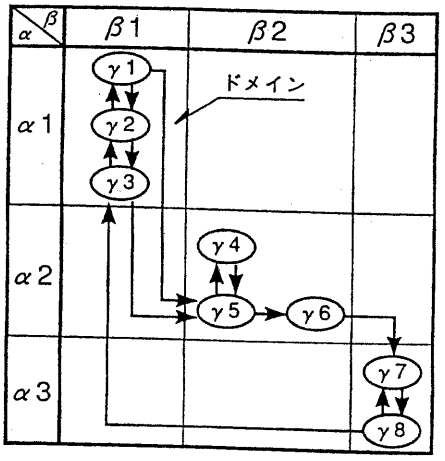


図７ マトリクス型状態遷移図(STM)

この際、ある時点で遷移可能な状態遷移は全て同時に遷移するものとして仕様を記述する（同時遷移の仮定）。この仮定は物理的には非現実的なものであるが論理的には妥当なものである。システム内部動作仕様に状態遷移に関する非決定性を持ち込むと、状態爆発の問題が起るからである。

状態遷移の非決定性に対する制御はタスクオブジェクトや写像オブジェクト仕様において、ロックされているオブジェクトの状態を他オブジェクトが参照更新しないというルールのもとに、同時遷移でのシステム内部動作仕様と実質的に等価な状態遷移が実現できるようにロック制御を行なう。

５．記述例

マルチプロセッサアーキテクチャ(図８)を有するディスク制御装置（ＤＫＣ）におけるデータ保証のための制御ソフトウェアにおけるオブジェクト仕様とシステム内部動作仕様の一部をSCOPE、およびSTMで記述した例を示す。ＤＫＣはキャッシュ（Ｃ）とそのバックアップメモリ（Ｎ）を内蔵しており、ライト処理時にライトデータをＣとＮに格納した後、ライト処理とは非同期にライトデータをディスク媒体上に書き込む。データ保証制御は、ＣまたはＮの一方が障害となった場合にそれらのうちの正常な方に格納してあるライトデータを緊急にディスク媒体上に書き込む制御である。

5.1 オブジェクト仕様記述

（１）オブジェクト関連図

データ保証制御を構成するオブジェクトの関連図を図９に示す。楕円で囲まれているのがオブジェクトであり、Rがリソースオブジェクト、Pがタスクオブジェクト、Fが写像オブジェクトを示す。オブジェクト間を結ぶ線は、オブジェクト間の通信が存在することを示す。

（２）タスクオブジェクト

緊急デステージサーバの仕様記述を図10に示す。緊急デステージサーバは、キャッシュに格納されているライトデータをドライブ媒体に書き込む処理の進捗制御を行なうものである。。

（３）リソースオブジェクト

キャッシュオブジェクトの仕様記述を図11に示す。キャッシュオブジェクトはキャッシュの状態を記憶する。括弧で囲まれた内容は状態の値域を示す。

（４）写像オブジェクト

アクセスレベル制御の仕様記述を図12に示す。アクセスレベル制御は、キャッシュやバックアップメモリ障害状態でのリード要求に対し、リードデータが矛盾しないように、キャッシュやバックアップメモリの状態とドライブ状態間の整合性を管理するものである。

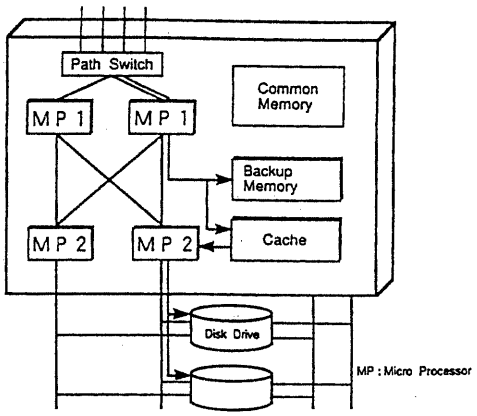


図8 ディスク制御装置

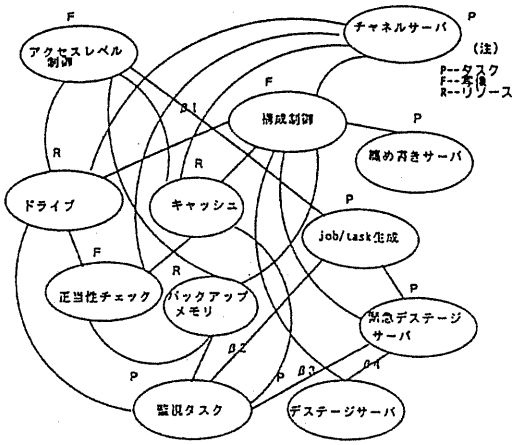
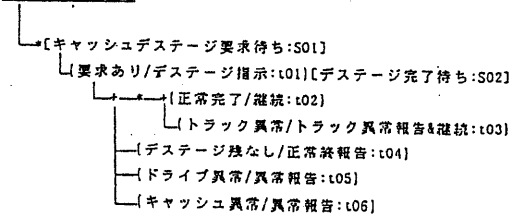


図9 オブジェクト関連図

状態遷移構造記述



状態遷移制御記述

遷移条件

t01: E<β1>(デステージ対象デバイス(dev_no, キャッシュ))
t02: E<β2>(デステージ完了報告(dev_no, 正常終了))
t03: E<β3>(デステージ完了報告(dev_no, 異常終了))

行動記述

t01: S<β0>(デステージ指示(dev_no, キャッシュ))
t02: S<β0>(デステージ指示(dev_no, キャッシュ))
t03: S<β0>(デステージ指示(dev_no, キャッシュ))

図10 タスクオブジェクト記述例

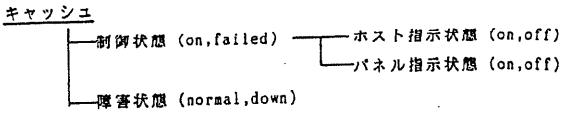


図11 リソースオブジェクト記述例

アクセスレベル制御

RULE SET 1(アクセスレベル決定:)

起動条件: E<β9>(キャッシュ障害) /...

(ルール1)IF キャッシュ. 制御状態 (on)
NVS. 制御状態 (on)
デバイス(i).キャッシュ.制御状態(on)
デバイス(i).DFW.制御状態(on)
THEN デバイス(i).アクセスレベル:=3

(ルール2)IF キャッシュ. 障害状態 (down)
NVS. 制御状態 (on)
デバイス(i).キャッシュ.制御状態(on)
デバイス(i).DFW.制御状態(on)
デバイス(i). データ状態(なし)
THEN デバイス(i).アクセスレベル:=2.5

図12 写像オブジェクト記述例

5.2 システム内部動作仕様記述

データ保証制御のシステム内部動作仕様のSTMによる記述例を図13に示す。キャッシュ、バックアップメモリの各障害状態とそれらに格納されているライトデータ状態の3つのリソースオブジェクトとそれらのオブジェクト状態間の関連を管理するための写像オブジェクトの状態遷移が記述されている。写像オブジェクトの状態として、キャッシュとバックアップメモリに関するシステム制御状態を定義している。

相互作用を有するオブジェクトが3つあるので、キャッシュおよびバックアップメモリの2つのオブジェクト状態の直積要素をSTMのキー平面の縦軸に記述し、横軸には3つめのオブジェクトであるライトデータの状態をとっている。これらのオブジェクト状態の組合せで定義されるキー平面上に、データ保障制御のために識別すべきシステム制御状態の遷移が記述されている。キー平面上のそれぞれの丸がシステム制御状態であり状態値が中に記述されている。

6. 制御ソフトウェア検証支援システム

6.1 ADRESの概要

ADRESはSCOPEで記述された制御ソフトウェア仕様をもとにその仕様検証とプログラミングの支援を行なうシステムである。ADRESの構成を図14に示す。ADRESはまず、

SCOPE記述された仕様をペトリネット[6]をベースとする実行可能モデルに変換する。そして、それを用いて次の支援を行なう。
(STAGE1) 写像オブジェクトの仕様検証、
(STAGE2)タスクオブジェクトの仕様検証
(STAGE3)骨格プログラムの生成
以下、SCOPE記述のペトリネット変換、および(STAGEP1)と(STAGE2)における支援機能について述べる。((STAGE3)については別の機会に譲る。)

6.2 SCOPE記述の実行可能モデルへの変換

(1)タスクオブジェクトの変換
(手順１)遷移構造記述のペトリネット変換
遷移構造記述に定義された内部状態をプレースに、遷移をトランジションに対応づけ、そのままペトリネットに変換する。
(手順２)遷移制御記述のペトリネット変換
遷移制御記述に含まれる遷移条件、行動記述をそれぞれペトリネットに変換する。遷移条件はそのまま対応するトランジションの発火条件とする。行動記述のうちメッセージ式を含まない処理文（自然言語で記述された処理文）の列は１つの処理名に縮約し、1つのトランジションに変換する。

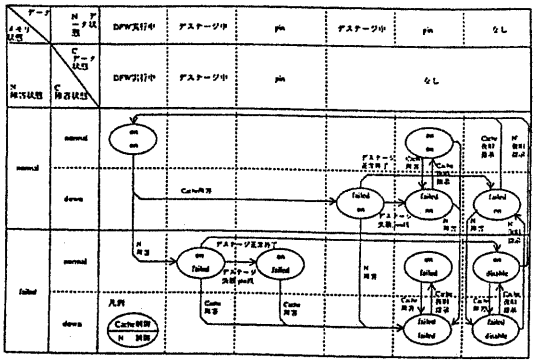


図13 システム内部動作のSTM記述例

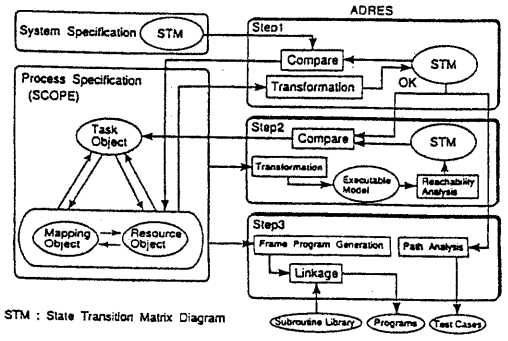


図14 ADRESの構成

メッセージ式からなる処理文は、1つのトランジションに変換し、通信に使用するチャネル、またはバッファ名を記憶する。
手順１～２の変換を指定されたタスクオブジェクトに関して行う。
(手順３)メッセージ式のリンク
同一のチャネル、またはバッファを介して互いに通信するトランジションペアを単位として以下の変換を行なう。メッセージ式X<α>, XM<α>, XR<α>及びE<β>に対応するトランジションペア(発火に同期制約あり)を図15に示すサブペトリネットに展開する。メッセージ式内のメッセージ内容は、カラートークンとして実行時に反映させる。メッセージ式S<β>, R<β>, C<β>に対応するトランジション（自由発火可能）については当該トランジションが発火した時点で、それらのメッセージ式を実行し、バッファβに対応するプレースに対し必要なトークン操作を行なう。
(2)その他のオブジェクトの変換
リソースオブジェクト、写像オブジェクトはメッセージを受信して内部状態を遷移させる実行可能なルールに変換する。

6.3 オブジェクト仕様の検証支援

STM記述されたシステム内部動作仕様をもとに仕様検証の支援を行なう。
(STAGE1) 写像オブジェクトの仕様検証
同時遷移の仮定のもとに写像オブジェクトが生成するシステム制御状態遷移を出力し検証を支援する。
(1)写像オブジェクトの実行
リソースオブジェクトと写像オブジェクトを組み合わせて動作シミュレーションを行ない、システム制御状態の遷移を生成する。これらのオブジェクトが動作するには、本来タスクオブジェクトからのメッセージが必要であるが、初期状態を起点に、次の状態に遷移するために必要なメッセージを自動的に供給してやることでシミュレートすることができる。もし、状態遷移に必要なメッセージが複数同時に要求された場合は、必要なメッセージをすべて同時に供給し、遷移可能な状態遷移を同時に発生させる。(同時遷移の仮定に対応)
(2)検証
シミュレーションの結果得られたシステム制御状態遷移の集合を、与えられたシステム内部動作仕様と同一形式のSTMに編集し出力する。もし、両者に違いがあれば、写像オブジェクトまたはリソースオブジェクトの状態遷移仕様の不備と考えられる。
(STAGE2)タスクオブジェクトの仕様検証
タスクオブジェクトの出力するメッセージシーケンスのもとで写像オブジェクトが生成するシステム制御状態遷移を出力し検証を支援する。

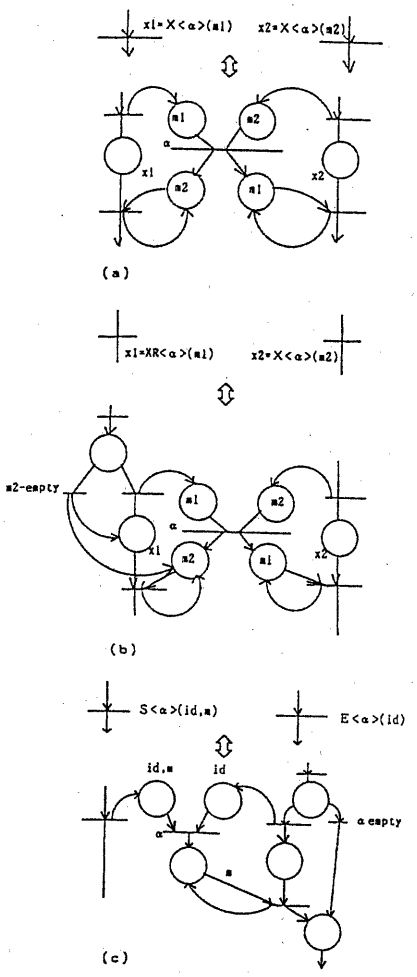


図15 メッセージ式のペトリネット変換

(1)オブジェクトの実行

ペトリネット変換されたタスクオブジェクトをリソースオブジェクトおよび写像オブジェクトと結合して連動シミュレーションを行なう。

タクオブジェクト仕様から得られたペトリネットはセーフであり、到達可能な状態は有限である。初期状態に対応するプレースに初期トークンを投入し、可能なトランジションの発火列をすべて生成することによりメッセージの出力シーケンスを求める。メッセージ式の実行と無関係なトランジションが複数同時に発火可能になった場合、それらの発火順序は考慮する必要がなく、それらを全て発火した後のマーキングのみ記憶すれば良い。これは、メッセージ式と無関係なトランジションの発火がメッセージの出力シーケンスに影響しないためである。これらのメッセージ出力シーケンスのもとで写像オブジェクトが生成するシステム制御状態遷移を求める。

(2)検証

シミュレーションの結果得られたシステム制御状態遷移を与えられたシステム内部動作仕様と同一形式のSTMに編集出力する。この際、ロックされているオブジェクトについてはロック解放時の内部状態のみ表示し、ロック中の過渡状態は表示しない。両者を比較し違いがあれば、

(i)タスクオブジェクトまたは写像オブジェクトによるオブジェクトのロック不備

(ii)タスクオブジェクトのメッセージ出力シーケスの不備

が考えられる。

7．おわりに

本論文では、並行動作システムにおける制御ソフトウェアの設計と検証を支援する一方法として、状態遷移の概念に基づくオブジェクト仕様記述方法、システム内部動作仕様記述方法、オブジェクト仕様を変換した実行可能モデル(ペトリネット)を用いたオブジェクト仕様の検証支援方法等について述べた。現在、ADRESの(STAGE1)の機能をワークステーション上で開発中である。

参考文献

1)R.E.Filman,D.P.Friedman :Coordinate computing Tools and Technique for Distributed Software, McGraw-Hill(1984)(雨宮、他(訳):協調型計算システム、マグロウヒル(1986))

2)本位田、他: リアルタイムシステムにおけるプロトタイピングの一手法、情報処理学会論文誌、Vol.24、No.5

3)Ward,P.T.and Mellor,S.J.:Structured Development for Real-Time Systems,Vol.1,2,3,Yourdon Press(1985) 4)Hatley,D.J. and Pirbhai,I.A.:Strategies for Real-Time Systems Specification(1988)

4)片岡:ソフトウェアモデリング、日科技連出版社(1988)

5)S.Shlayer, S.Mellor:An Object-Oriented Approachto Domain Analysis, Software Engineering Notes,ACM Press, Vol.14, No.5, pp.66-67(1989)

6) J.L.Peterson: Petri Net Theory and the Modeling of Systems, Prentice Hall Inc.,Englewood Cliffs(1982)(市川、小林(訳): ネット理論入門、共立出版(1984))

7)M.Akatsu, T.Murata, K.Kurihara : Verification of Error Recovery Specification for Distributed Data by Using Colored Petri Nets, IEICE Transactions, Vol.E-74,No.10,pp.3159-3167(1991)