実時間システム仕様記述法について†

陳 昭煬†† 有澤 誠††

実時間システムの仕様記述をするための手法として，JSD 法を基にして拡張した拡張 JSD 法 (Extended JSD) を提案する．実時間システムはシステムの運行や外部実体とのかかわり合いが時間，時刻と深く関連しているシステムであり，システム仕様の記述には動的な視点からの記述法が必要である．JSD 法は並列プロセスモデルに基づく記述法であり，望ましい性格を多く有しているが，時間制約や同期通信プリミティヴを含まないため，実時間システムの仕様記述には不十分である．そこで拡張 JSD 法は元の JSD 法のタイミングステップを強化する方向で，多様な通信プリミティヴを提供し，また時間制約をその特性に合わせて記述できるように，JSD 法を拡張した．

1. はじめに

実時間システムとはシステムの運行や外部実体とのかかわりあいが時間，時刻と深く関連しているシステムである．たとえば自動車生産ラインの工作機械などに埋め込むシステムや，航空管制制御などの時間臨界システム (Time-Critical System) がある．

実時間システムを開発するための方法に操作的アプローチ[10),11)]がある．システム開発で仕様の段階に重点をおくこと，実現に自動化あるいは半自動化の手法を利用すること，開発の早い段階でユーザへフィードバックをかけることなどがその特徴であり，適切な仕様記述法の提供と支援環境の整備が重要な課題である．

実時間システムの仕様記述法の具体例として，以下のものをあげることができる．Zave, Schell らの PAISLey[10)~12)] は，並行通信モデル，関数的な仕様記述，通信の交換関数 (exchange function)，動的な仕様検証を特徴とする．Gomma の DARTS[5)] は，並行通信モデル，図式的な仕様記述，プロトタイピング手法を特徴とする．Alford の SREM[1),2)] は，構造的有限状態機械モデル，RSL による要求記述，精度要求と性能要求 path の R-net 図式による表現を特徴とする．Jahanian の RTL[8)] は，event-action モデルの時間要求記述，一階述語論理へ変換と解析を特徴とする．

これらの研究成果を踏まえた上で，本論文では Jackson, Cameron らの JSD 法[7),3)] を基にして拡張した実時間システムの仕様記述方法である拡張 JSD 法 (Extended JSD) について述べる．JSD 法は図式記述，実世界を重視した開発などの特徴をもつが，実時間システムの仕様記述には不十分である．特に通信プリミティヴ，時間制約の形式記述が満足すべきものでない．拡張 JSD 法は，同期通信を含む多様な通信プリミティヴ記法によって，プロセス間の関係の記述を可能にし，また制約プロセスの記法やプロセスの制御構造に埋め込む記法による時間要求の記述も可能であることが特徴である．

以下第 2 章で拡張 JSD 法の基本的な考え方を整理する．第 3 章でプロセスの結合関係を表す通信プリミティヴを述べ，通信待ちを検討し，特殊な同期手法を述べる．第 4 章で時間制約のアプローチ，時間制約の記述法を述べる．最後に第 5 章で拡張 JSD 法の使用方法を例題を用いて簡単に説明する．

2. 拡張 JSD 法の基本的な考え方

2.1 並行プロセスモデルの 4 種類のプロセス

情報システムの機能仕様は問題領域向き (problem area oriented) であるから，システムがおかれた環境をモデル化することが重要である．一般の情報システムに比べると，実時間システムはシステムのおかれている環境とシステムのかかわりあいがより複雑で，時間の要求も厳しい．このため環境をシステムの仕様の中にとりいれるようにモデル化することが一層重要になる．環境をうまくモデル化するためには，問題の領域の本質的な特性である並行性をそのまま記述できることが望ましい．そこで JSD 法で採用しているような複数の並行通信プロセスを用いたモデル構築が便利である．拡張 JSD 法では，これらのプロセスを，実世界の実体を記述する外部プロセス，外部プロセスを

† A Specification's Description Method of Real-Time Systems by JOAU-YANG CHEN and MAKOTO ARISAWA (Department of Computer Science, Faculty of Engineering, Yamanashi University).

†† 山梨大学工学部計算機科学科

システムの内部で模倣するモデルプロセス，システムの主な機能を記述する機能プロセス，制約を記述する制約プロセスの4種類のプロセスに分ける．これに応じて機能仕様を外部層，モデル層，機能層の3層に分ける．制約プロセスはモデル層と機能層の中間におく．これら各種のプロセスの間で状態を参照し，同期し，通信する関係を，後述する多様な通信プリミティヴで記述する．

2.2 事象によるプロセス構成

各プロセスは状態 (state) と事象 (event) の二つの面をもち，事象が状態遷移の起因であり，また結果でもある．プロセスの活動を細分したとき，基本要素となる介入不能，分割不能な活動を事象と呼ぶ．以下では事象をシステムの基本的な要素として仕様を構築する．

一般にはシステムの外部の動きを事象といい，内部の動きをアクション (action) と区別することが多いが，拡張 JSD 法ではそれをすべて事象としてとらえる．

仕様は外部実体と事象の検討と記述に始まり，内部のモデルプロセスの構築，機能プロセスの導入，制約プロセスのとりいれまで，外部から内部へという過程で記述する．外部エンティティをモデルプロセスで模倣するには外部事象の段階的詳細化が必要であり，粗い外部事象を十分に詳細化されるまで内部事象へ細分する．機能プロセスや制約プロセスは他の複数のプロセスの事象や状態をまとめて合成して参照することもある．

2.3 3段階の仕様記述と時計モデル

拡張 JSD 法は，基本的に元の JSD 法と同じ手順ですすめ，5番目のタイミングステップでより形式的な仕様記述を行うように考えている．ここでの仕様記述には順序段階，時間段階，時刻段階の3つがある．各段階では視点の水準の違いから，それに用いる時計は抽象的時計のこともあり，実時計のこともある．

順序段階では事象を時間線上での瞬時的な区切り点であると考えて，各プロセスの中に事象間の順序関係，因果関係を記述する．このとき用いる時計は速度を考えずただ順序よく自然数を作り出す抽象的時計であり，各事象インスタンスにひとつの数 (tick) を割り当てる．

システムに対する処理速度の性能要求，あるいは実時間システムの応答時間などの時間制約の要求に対して，時間を具体化しなければならない．時間段階では，事象の経過時間の長さや，事象と事象の間の時間の長さを明記する．このときに用いる時計は実時計である．

時刻段階では，締切時刻など事象の発生点や終了点がある実時間の時点に合わなければならない制約を記述する．

仕様記述は，順序，時間，時刻という順に行い，抽象的時計と実時計を共に用いる．時計モデルは，集中型制御システムに対して global timer モデルがあり，分散システムに対して local timer モデルがある．

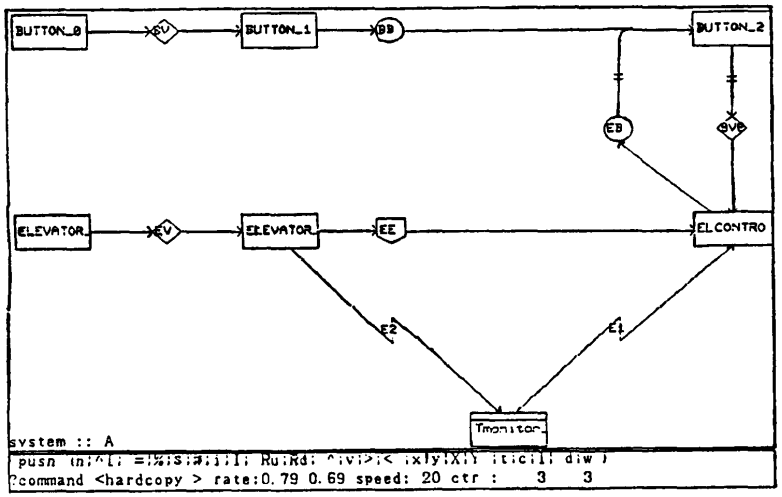
Global timer モデルはシステムの中に一つの時計だけを仮定して事象の順序関係，時間制約，時刻制約を記述する．このモデルで順序関係を記述する場合には，半順序の関係であるために余分な順序関係が生じて過剰制約になることがある．事象の間に不定の順序関係が多い分散システムの記述には不適当なモデルである．また時間あるいは時刻の要求を記述する場合には，プロセス内の local 時間の要求でも中央の時計から時刻を持ち出すように書かなければならないため仕様が読みにくくなる欠点がある．

Local timer モデルはシステムの中に多数の時計があると考えて事象の順序関係，時間制約，時刻制約を記述する．このモデルは各時計間に時刻のずれが生じる欠点がある．しかし複数の並行プロセスから成り立つ分散システムでは，各プロセスがそれぞれの時計をもつことは自然であり，時間制約と時刻制約を記述しやすい．時計の間に時刻ずれの問題は，プロセスの実行スケジュールの問題と合わせて設計段階で考慮し，仕様記述段階では問題にしない．

2.4 プロセス構造図と拡張システム仕様図

多くの仕様記述がネットワーク図とプロセス内部ロジック図の組合せであることに従い，拡張 JSD 法での仕様をプロセス構造図 (Process Structure Diagram) と拡張システム仕様図 (Extended System Specification Diagram) とに分ける．プロセス内部での処理の流れの記述にはプロセス構造図を使用し，プロセス間の関係の記述には拡張システム仕様図を使用する．

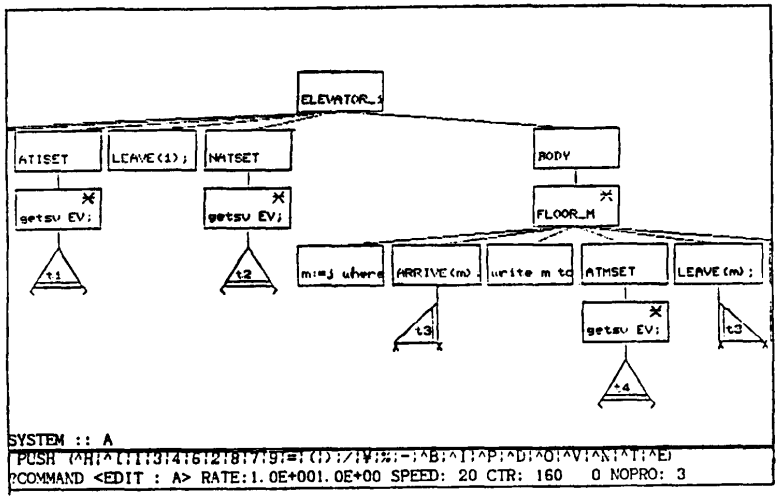
プロセス構造図は Jackson 構造図を基にしているが，末端節点として事象のほかに，状態設定操作，通信操作，同期通信法，時間制約も書けるようにした図である．ここで状態設定操作は，他のプロセスから参照されるプロセス内の状態変化を示し，通信操作はプロセスの状態を他のプロセスへ知らせる時点や他のプ



Extended system specification diagram

```
 3>   ELEVATOR_1 seq
 4>     m:=1;
 5>     getsv EV;
 6>     ATISET itr while(AT.1)
 7>       e1 : getsv EV;
 8>     ATISET end
 9>     LEAVE(1);
10>     NATSET itr while(NAT)
11>       e2 : getsv EV;
12>     NATSET end
13>     BODY itr
14>       FLOOR_M seq
15>         m:=j where EV=AT.j;
16>         e3 : ARRIVE(m);
17>         write m to EE;
18>         ATMSET itr while(AT.m)
19>           e4 : getsv EV;
20>         ATMSET end
21>         e5 : LEAVE(m);
22>         NATSET itr while(NAT)
23>           e6 : getsv EV;
24>         NATSET end
25>       FLOOR_M end
26>     BODY end
27>     t1 : cycle e1 max 5 usec ;
28>     t2 : cycle e2 max 5 usec ;
29>     t3 : set timeinterval from ( e5) to ( e3 ) mean 30 SEC ;
30>     t4 : cycle e4 max 5 usec ;
31>     t5 : cycle e6 max 5 usec ;
32>   ELEVATOR_1 end
33>
```

Extended structure text



Extended structure diagram

図 1 拡張 JSD 法での仕様記述例（支援システムからの出力）

Fig. 1 An example specification in Extended JSD method (outputs from the supporting system).

ロセスから状態を読み込む時点を示す．いっぽう拡張システム仕様図もやはり JSD 法の SSD 図を基にしており，local 時計モデルでの同期通信法と時間制約の記述もできるようにした図である．**図 1** に拡張システム仕様図，プロセス構造図，および拡張構造テクストの例を示す．

この図は Jackson[7] のエレベータ制御問題を記述したものであり，PC 9801/VM 2 の MS-DOS 上に TurboPascal で作成した拡張 JSD 用編集プログラムからの出力である．

プロセスには単一インスタンス型，複数のインスタンスを動的に生成できる型，複数のインスタンスをあらかじめ定まった個数だけもつ型の 3 つがある．

3. プロセス間の通信方法と同期の手法

3.1 通信プリミティヴ

並行通信プロセスモデルを基にした言語は，プログラミング言語か仕様記述言語かにかかわらず，通信の方法が言語の主な特徴になる．並行プロセスモデルの中で最も難しい部分がプロセス同士の同期とプロセス間での情報交換であり，しかもそれらが通信の方法で定まるからである．特に仕様記述言語は問題領域での多数のエンティティの間の関係を忠実に記述しなければならないため，通信方法も種々の方法を用意する必要がある．

Ada のランデヴのシミュレーション等いろいろな例題を記述してみた結果，実世界のタイミング情報を含めたモデル化を記述するための通信プリミティヴとして，拡張 JSD 法では**図 2** に示す 5 つを用意することにした．元の JSD 法を含み，その自然な拡張となるように留意して，この 5 つを定めてある．送信側および受信側は単一のプロセスか，複数のプロセスからなるクラスである．以下の通信プリミティヴはいずれも送信側が複数である場合には受信側にキュー機能を仮定し，また受信側が複数の場合には送信側の指定に従って決められた受信相手と通信をするという配分機能を仮定している．

情報を生成するプロセスが情報を消費するプロセスの動きを関知しないまま，自分自身の状態やその環境に依存して行動する非同期通信プリミティヴとして，次の 2 つを用意している．

（1） データストリーム結合 (data stream connection)：送信側がメッセージの内容，順序を定め，受信側は送信されたメッセージを順序どおりに受け取る．送信側は受信側の動きと関係なく送信し続ける．受信側はメッセージが送信されていないときに受信しようとすると，送信があるまで待つ．

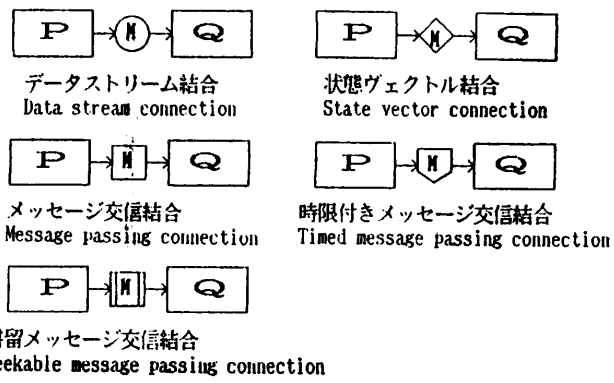


図 2 拡張 JSD 法の通信方法
Fig. 2 Communication methods in Extended JSD.

この通信は，通信プロセスの間に自動的な無限の大きさのバッファを必要とする欠陥をもつ．バッチシステムのプロセスの間の非同期の結合や，実時間システムのプロセス間の緩い結合に使用する．

（2） 状態ヴェクトル結合 (state vector connection)：受信側（調べる側）が通信の主体であり，送信側（調べられる側）は通信を引き起こすことはできない．送信側は受信の有無によらずにメッセージを一方的に書き換える．受信側は送信側がメッセージを書き換えている間に受信することはできない．受信側が通信を行う周期を送信側のメッセージの書き換えサイクルと合わせないと，メッセージを読み落とすことがある．

これは静的なエンティティの状態を読みとる場合や，送信活動を行えない非知能的な外界エンティティの状態を読み取るために使用する．以上 2 つの通信方法は，元の JSD 法にあるものと同じである．

実時間システムには同期あるいは同期通信の要求が多くあり，それを明確に記述したい．拡張 JSD 法では，同期通信としては通信待ちのしかたに応じて以下の 3 つの通信プリミティヴを用意している．

（3） メッセージ交信結合 (message passing connection)：送信側と受信側のどちらか先に通信の用意ができた側が，相手側の通信の用意ができるまで待ち続ける．通信の両側とも通信用意ができると，送信側がメッセージを受信側へ渡して通信を完了する．この通信ではデッドロックが生じやすい．実時間システム

での独立なプロセスの間の同期通信に使用する．

（4） 時限付きメッセージ交信結合(timed message passing connection)：この通信プリミティヴは基本的にメッセージ交信結合と同じ方式の通信法であるが，通信待ちの時間が無限ではなく上限値をもつ．この上限を越えると通信を断念して通信の時間フォールトを引き起こし，フォールト処理に切り替える．この通信はメッセージ交信結合の改良であり，時間フォールトが起こった場合の対処の記述を必要とする．通信の相手側の異常に対して早急に対応する必要のある場合や，時限付き情報を伝達するために使用する．

（5） 書留メッセージ交信結合 (Peekable message passing connection)：受信側は，送信側からメッセージが送られているか否かを常に調べながら仕事をする．送信があると現在の仕事の小さな区切りまで進んでメッセージを読み取り，それを処理する．いっぽう送信側は送信した後は受信側がメッセージを読み取るまで待つ．しかし受信側が現在行っている仕事の断片を終了するまでであるから，この待ち時間は短い時間ですむ．

送信の周期が予知できず，送信側が通信待ちしてでも同期をとりたい場合に使用する．たとえば生産制御システムで生産計画プロセスと生産ライン制御プロセスとの間の通信がある．システム全体の製品を管理するために，生産計画プロセスは管理する生産ライン制御プロセスの状態を調べながら生産計画プロセス同士と通信する．全体の製品の量，質などの情報によって管理する生産ライン制御プロセスにいろいろな指令を送る．生産ライン制御プロセスは，生産ラインを制御しながら生産計画プロセスからの新たな指令を調べる．新たな指令を受信すると現在生産している半製品を完成したあと，その指令に従って別の動きをとる．

この例題を拡張 JSD 法で記述すると**図3**のようになる．

3.2 通信待ちと非同期通信によるプロセスの同期

同期通信，非同期通信のいずれの方法でも，通信待ちの現象が生じる．(1)のデータストリーム結合は受信側だけが通信待ちになることがある．(2)の状態ヴェクトル結合については，情報の書き込みと読み出しの間の相互排除のために，わずかな時間ではあるが送信側に通信待ちが生じることがある．たとえば readers/writers の問題では，一般に writer が優先であるから，reader が通信待ちになることがある．(3)のメッセージ交信結合では受信側，送信側の両方

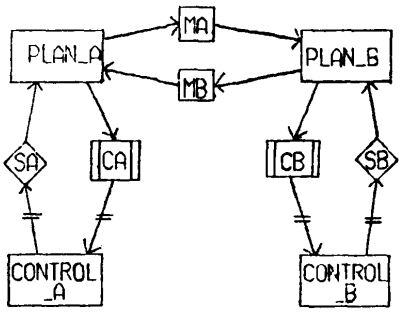


図 3 生産制御システムの仕様図
Fig. 3 System specification diagram of production control system.

とも通信待ちになることがある．(4)の時限付きメッセージ交信結合では受信側，送信側の両方とも通信待ち，あるいは通信待ちの時限フォールトになることがある．(5)の書留メッセージ交信結合では送信側は送ったメッセージが読み取られるまで少し待つ必要がある．しかし受信側はメッセージがない場合には別の仕事を進めるか，メッセージが来るまで待つかについて自由に決めることができ，通信待ちになることはない．

プロセス間の同期関係は，通常は同期通信で記述するが，非同期通信を用いて記述することも可能であり，次の2つのやりかたがある．

（i） 状態ヴェクトル結合による同期の記述：

状態ヴェクトル結合では受信側と送信側と同期するため，受信活動に時間制約をかけて受信のリズムを送信のサイクルと合わせる．しかしこの時間制約をかけるためには送信側のメッセージの書換え周期や最短の書換え間隔の予知が可能でなければならない．

（ii） データストリーム結合による同期の記述：

データストリーム結合でプロセス間の同期を記述する[3]には，データストリーム結合を2度書かなければならない．こうすると，同期のための結合か非同期の通信結合かを見分けにくい．特にプロセス間の結合が複雑だったり同期の要求が多いときは，仕様が読みに

```
Process_P1 seq                Process_P1 seq
  ... ..                        ... .
  write invoke to D1;           read D1;
  read R;                        (仕事 alpha ... .
  ... .                             ... .          )
                                write response to R;
  ... ..                        ... .
Process_P1 end                Process_P2 end
```

図 4 Ada の rendezvous に相当する通信
Fig. 4 Ada's rendezvous using data stream connection.

くくなる欠点をもつ.

データストリーム結合では図4に示すように Ada の rendezvous に相当する通信も記述できる.

4. 時間制約の形式化

4.1 Stimuli-Response アプローチ

プロセスは事象と状態という2面をもっているため，状態に対する時間制約と事象に対する時間制約とがある．しかし状態に対する時間制約は，その状態をはさんだ2つの事象に対する時間制約で表現できることから，拡張 JSD 法では事象に対する時間制約の記述だけを採用している.

時間制約を形式化する手法として，Dasarathy[6] の Stimuli-Response アプローチがある．システムへの入力となる事象を Stimuli (誘因) と呼び，システムから環境への出力の事象を Response (応答) と呼ぶ．時間制約の要求をこの2種類の事象の組合せで記述する．ただしこの2種だけでは入力や出力以外の reference (参照) を含まないため，システムと環境が互いに参照しあう事象の時間制約を記述できない．拡張 JSD 法では，環境とシステムの間で Stimuli, Response の事象の組合せ4通りと，参照事象同士の計5通り，Stimuli-Stimuli (SS), Stimuli-Response (SR), Response-Stimuli (RS), Response-Response (RR), reference-reference (rr) を用いる.

また時間制約は制約の働きから次の3種類に分類できる.

(a) 行為時間制約：これは外界エンティティがシステムへ誘因を供給する周期と，外界エンティティがシステムの状態を調べる周期に対する時間制約である．可能な時間制約の事象の組は，SS, RS, rr の3通りがある．これは環境に対する制約であるため，形式的には記述できない場合もある．そのときに注釈の形に書く．行為時間制約はシステムの運行の時間的な条件を示す意味ももつ.

(b) 性能時間制約：これはシステムが外界エンティティの状態を調べる周期と，システムが環境へ応答する周期に対する時間制約である．可能な時間制約の事象の組は，RR, SR, rr の3通りがある.

(c) 機能時間制約：これはシステムが外界エンティティの誘因供給周期を監視する機能と，システムが自身の動きを監視する機能である．可能な時間制約の事象の組は SS, RS, RR, SR, rr の5通りがある．システムが外界エンティティの動きや自身の動きを監視したり，システムの内部に埋め込んだ故障に対する処理に関する時間制約などがこの例である.

実時間システムの仕様記述は，時間制約を形式的に記述するだけでなく，時間制約を満たさなかった場合の異常処理をシステムの機能として記述できなければならない．特に時間的な制約は実際の運行環境により変動することが多い．この場合にはフォールト処理を含む形で記述する.

これら時間制約の分類とそれらを仕様中でどこに記述するかを図5にまとめておく.

4.2 Local 時間制約の記述

時間軸上に並んだ事象，事象と事象の間隔，さらに事象グループと事象グループの間にある時間変量 (timing variation) に課した要求を時間定量と呼ぶ．時間定量は特定の時刻を示す場合と，ある時間幅を示す場合とがある．時刻を示す場合はキイワード at を用い，時間幅を示す場合はキイワードの just, max, min, mean を用いる．ただしこの4つは，時間幅その

行為時間制約	組合せ		SS	RS	rr
	記述場所		システム外部	システム外部	システム外部
機能時間制約	外界への制約	組合せ	SS	RS	
		記述場所	モデルプロセス	モデルプロセスより内部	
	システム自身への制約	組合せ	RR	SR	rr
		記述場所	モデルプロセスより内部	モデルプロセスより内部	モデルプロセス
性能時間制約	組合せ		RR	SR	rr
	記述場所		モデルプロセスより内部	モデルプロセスより内部	モデルプロセス

図5 時間制約の分類
Fig. 5 Timing constrain's classification.

もの，時間幅の最大値，最小値，平均値をそれぞれ表す．

拡張 JSD 法の仕様は多数の並行通信プロセスで構成する．そこで時間制約の記述は，各プロセス独立な時間軸を用意する local 時間制約の記述と，システム全体に共通の時間軸を前提とする global 時間制約の記述の2つがある．local 時間制約の記述は，ひとつのプロセス内での事象の生起時刻，終了時刻，事象と事象の間の時間幅に対する制約の記述に用いる．また global 時間制約の記述は，複数のプロセス間での事象に関する時間制約の記述に用いる．local 時間制約は拡張プロセス構造図に書き，global 時間制約は拡張システム仕様図に書く．この2つの時間制約を書き分けるために，拡張 JSD 法では構文や図式の中に事象を表現するための事象識別子と，異なる時間制約を見分けるための時間制約識別子を使う．

Local 時間制約の記述には，ある事象の生起時刻や終了時刻の記述と，2つの事象の時間幅の記述がある．前者はキイワード occur, cycle, complete を事象識別子に添え，時間定量と組み合わせる．

occur はある事象をある時刻で発生するという制約であり，時間定量キイワード at と組み合わせる．cycle は繰り返しの事象の周期を表し，キイワード just, max, min, mean と組み合わせる．complete は特定事象の経過時間を表し，キイワード just, max, min, mean と組み合わせる．

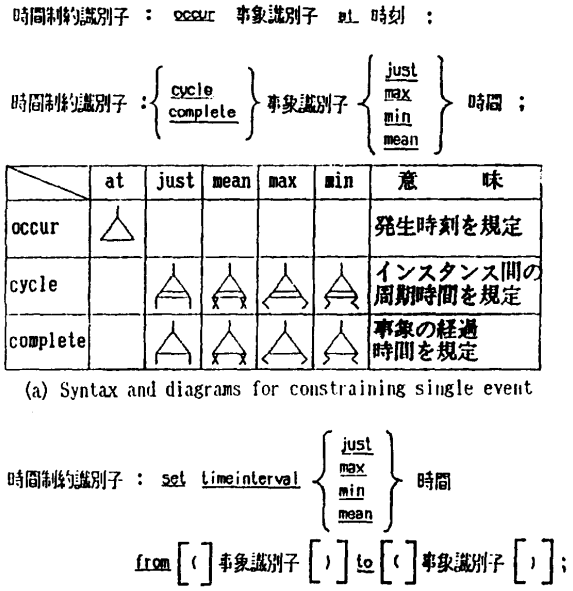
2つの事象に対する制約は，キイワード set timeinterval とキイワード from, to を組み合わせて記述する．from と to は時計の開始および終了を表す．時間の計りかたは，原則として事象そのものが費やす経過時間も含む．すなわち from は事象の始めの時刻から，to は事象の終わりの時刻までそれぞれ計算する．事象の経過時間を含まないことを示すにはかっこの対で事象を囲む．この場合，from は事象の終わりの時刻から，to は事象の始めの時刻までそれぞれ計算する．

以上の時間制約の記述の構文と図式を**図 6** にまとめておく．

4.3 Global 時間制約

Global 時間制約には，2つのプロセスの間の時間制約と，3個以上のプロセスの間の時間制約に分けて考える．Global 時間制約を記述するためには，新たに時間制約プロセス t_monitor が必要になる．

t_monitor プロセスは他のプロセスの事象の発生を



(a) Syntax and diagrams for constraining single event

(b) Syntax and diagrams for contraining two events

図 6 Local 時間制約の構文と図式
Fig. 6 Syntax and diagram of local timing constrain.

見守り，時刻の経過を t_monitor 自身が持つ時計で計る．t_monitor の記述の中にキイワード monitor timeinterval とキイワード from と to または between と and を用いて時間制約を記述する．並行して動いているプロセスの互いに順序を定められないような事象同士の時間制約にはキイワード between, and を用いる．これは事象と事象の間を表す．時間の計算は，先に発生した事象の始めの時刻から，あとに発生した事象の終わり時刻まで計算する．from, to の意味は前と同じである．

3個以上のプロセスの間の時間制約も2つの事象グループの間の時間制約に帰着させることができる．事象オペレータ ∩(and) と ∪(or) で事象をまとめて事象グループにする．事象を∩で結んだものは，その事象のすべてが発生したときにグループ全体の事象が発生したとみなす．すなわち from の記述に∩があれば最初に発生した事象から時刻を計り，to の記述に ∩があれば最後に発生した事象までの時刻を計る．事象

をUで結んだものはそのグループの中のひとつの事象が発生したとき全体の事象が発生したとみなす，すなわち from の記述にUがあれば最初に発生した事象から時刻を計り，to の記述に U があれば最初に発生した事象まで時刻を計る．

3個以上のプロセス間の時間制約に，いくつかのプロセスの状態が特定の値である場合のみ制約が働く場合もある．そのときは t_monitor の中に∩とUを用いて他のプロセスの状態の合成を記述する．

5. 例題によるシステム使用方法の概要

拡張 JSD 法の使用方法を，例題を用いて簡単に説明する．とりあげる例題は，Zave による文献 10) や Alford による文献 1) 等で引用している，患者看護システムである．このシステムは入院患者の看護を自動化することで，センサによって患者の状態を監視し，医師があらかじめ設定した安全基準を越えると自動的に警報を出すしかけになっている．また患者に関する情報を医師や看護婦たちの問い合わせにいつでも応答できるように記憶しておく．

この看護システムの機能的要求は次のとおりである．

- センサから異常な情報があれば，センサの故障警報を出す．
- センサから正常な情報があれば，設定基準値と比較し，それを越えていれば警報を出す．
- 医師あるいは看護婦は設定基準値の変更ができる．
- センサからの情報は記憶部に格納する．
- 医師および看護婦からの問い合わせには，記憶部の情報を利用して回答する．

Jackson の JSD 法と同様に，まずエンティティとアクションによる実世界のモデル化からはじめ，システム機能を拡張システム仕様図で記述し，プロセスの逐次的ロジックは拡張プロセス構造図で記述する．これらを合わせて，拡張構造テクストの形で疑似言語コードが作成できる．

実世界のモデル化プロセスとして Patient, Nurse, Doctor を用意し，機能プロセスとしてセンサ情報の監視をする Monitor, 問い合わせ応答のための Database を導入する．ここまでが Jackson の第4番目の機能ステップに相当する．第5番目のタイミングステップで，システムの時間制約を詳しく記述する．

患者の状態の情報を一定時間内にシステムに取り込むのは local 時間制約であり，Patient_1 の拡張プロセス構造図部分に書く．また Monitor が Patient_1 からの情報を読み取る部分は，時限付きメッセージ結合によって，一定時間内に行うことを，拡張システム仕様図に記述する．問い合わせに対してある時間幅以内に応答することは global 時間制約であり，Tmonitor という新しいプロセスを2個導入して，やはり拡張システム仕様図に記述する．

紙幅の都合で，看護システムの拡張システム仕様図だけを図7にあげ，拡張プロセス構造図は省く．この仕様図は，看護システムのプロセス同士の関係を，豊富なプリミティヴを利用して記述できている．

なお拡張 JSD 法は，Zave[11] の言うオペレーショナ

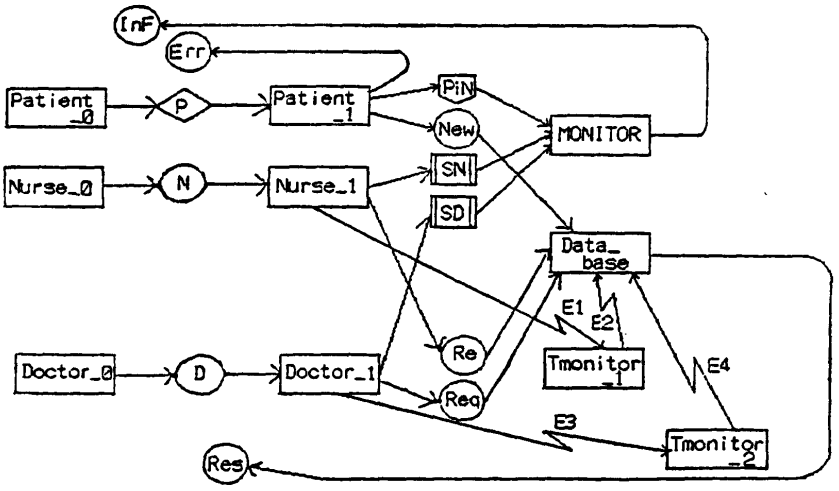


図 7 拡張 JSD 法による看護システム記述例
Fig. 7 Nursing system example in the Extended JSD.

ルな方法のひとつであり，拡張システム仕様図と拡張プロセス構造図で記述した機能仕様を仕様実行することも十分考えられる．プロトタイピングを仕様実行の形で実現すれば，それだけ早い時点で仕様とユーザからの要求とのギャップを検出できる．

6. おわりに

本論文は実時間システムを対象とする並行プロセス仕様記述法として拡張 JSD 法を提案してその詳細を述べた．この手法は，実時間システムを並行プロセスでモデル化し，多様な通信プリミティブな使い分けすることと，時間制約記述を形式化することによって元の JSD 法を拡張するアプローチをとっている．現実の実時間システム仕様記述での経験を積み重ねて，より良い仕様記述方法に改善していきたい．

謝辞 本研究を支援していただいた，山梨大学工学部計算機科学科有澤研究室の皆様と建設的なコメントをくださった査読委員とに感謝いたします．本研究の一部は情報処理教育研修助成財団および米山奨学財団の援助を受けており，両財団および甲府ロータリークラブに感謝します．

参考文献

1) Alford, M. W.: A Requirements Engineering Methodology for Real-Time Processing Requirements, *IEEE Trans. Softw. Eng.*, Vol. SE-3, No. 1, pp. 60-69 (1977).
2) Alford, M. W.: SREM at the Age of Eight; The Distributed Computing Design System, *IEEE Trans. Comput.*, Vol. 18, No. 4, pp. 36-46 (1985).
3) Cameron, John R.: An Overview of JSD, *IEEE Trans. Softw. Eng.*, Vol. SE-12, No. 2, pp. 222-240 (1986).
4) Chen, Bo-Shoe and Yeh, Raymond T.: Formal Specification and Verification of Distributed Systems, *IEEE Trans. Softw. Eng.*, Vol. SE-9, No. 6, pp. 710-722 (1983).
5) Dasarathy, B.: Timing Constrains of Real-Time Systems: Constructs for Expressing Them, Methods of Validating Them, *IEEE Trans. Softw. Eng.*, Vol. SE-11, No. 1, pp. 80-86 (1985).
6) Gomma, Hassan: Software Development of Real-Time Systems, *Comm. ACM.*, Vol. 29, No. 7, pp. 657-668 (1986).
7) Jackson, M. A.: *System Development*, 418 pp., Prentice-Hall, Englewood Cliffs, NJ (1983).
8) Jahanian, F. and Mok, A. KA-Lau: Safety Analysis of Timing Properties in Real-Time Systems, *IEEE Trans. Softw. Eng.*, Vol. SE-12, No. 9, pp. 890-904 (1986).
9) Lamport, L.: Time, Clocks, and the Ordering of Events in a Distributed System, *Comm. ACM.*, Vol. 21, No. 7, pp. 558-565 (1978).
10) Zave, P.: An Operational Approach to Requirements Specification for Embedded System, *IEEE Trans. Softw. Eng.*, Vol. SE-8, No. 3, pp. 250-269 (1982).
11) Zave, P.: The Operational Versus The Conventional Approach to Software Development, *Comm. ACM.*, Vol. 27, No. 2, pp. 104-118 (1984).
12) Zave, P. and Schell, W.: Salient Features of Executable Specification Language and Its Environment, *IEEE Trans. Softw. Eng.*, Vol. SE-12, No. 2, pp. 312-325 (1982).
13) 陳 昭場，有澤 誠：実時間並行プロセス仕様記述法の一考察，第 33 回情報処理学会全国大会論文集，3 W-6, p. 2067 (1986).

(昭和 62 年 1 月 19 日受付)
(昭和 62 年 9 月 9 日採録)

陳 昭 場（正会員）

1956 年生．中華民国（台湾）出身．1979 年台湾東吳大学商学部計算機科学科卒業．来日後 1987 年山梨大学大学院工学研究科計算機科学専攻修士課程修了．現在（株）CSK 技術開発事業本部研究開発部に勤務．ソフトウェア工学および人工知能の諸分野に興味をもっている．

有澤 誠（正会員）

1944 年生．1967 年東京大学工学部計数工学科卒業．電子技術総合研究所を経て，現在山梨大学工学部計算機科学科に勤務．工学博士．ソフトウェア工学，特にソフトウェアの評価，アルゴリズムの解析，オブジェクト指向システムなどに興味をもっている．