ソフトウェア工学 64－11
（１９８９．２．２）

通信ソフトウェア仕様における制約指向記述の適用法

林　　潔，西園 敏弘

ＡＴＲ通信システム研究所

通信システムを構成するプロセスの相互作用の仕様記述法を提案する。本手法は、各プロセス動作の重ね合わせにより、相互作用を仕様化するものであり、プロセス毎の機能仕様と相互作用仕様との分離記述が可能である。動作の重ね合わせを実現する同期機構として、制約指向における遅延評価を適用する。本記述法による仕様では、従来のプロセス間の信号授受による同期機構が不要となる。各プロセスは、他プロセスとのインタラクション無しにその動作を決定でき、プロセス機能仕様の再利用性の向上が期待できる。以上の利点を示すために、CCITT 勧告X.25レイヤ２手順を本記述法により仕様化するとともに、その機能追加時の影響を議論している。

Constraint Oriented Description for Communication Software Specifications

Kiyoshi HAYASHI , Toshihiro NISHIZONO

ATR Communication Systems Research Laboratories

Twin 21 Bldg. MID Tower, 2-1-61 Shiromi,
Higashi-ku Osaka 540 JAPAN

A method for specifying interaction between processes for communication software is proposed. This method allows to separately specify process functions and interactions, by the new concept which regards interactions as process action superposition. The delay evaluation mechanism in constraint oriented logic programing is employed to implement the synchronization for action superposition. In specifications by this method, message passing for synchronizing each process execution can be eliminated. Since each process can independently perform functions, reusability of process function specifications is imploved. To show this capability, the HDLC protocol is specified by this method and the influence of its modification is discussed.

1．まえがき

通信システムのソフトウェアは、通常、順序機械モデルに基づく状態遷移図で仕様記述される。

この仕様記述法は、システムへのそれまでの入力イベントと出力動作の履歴を状態に対応させ、新たな入力イベントにより、その時の状態から定まる動作および状態遷移をグラフを用いて図的に表現するものである。すなわち、各状態(ノード)をその入力イベントと出力動作を付した矢印(アーク)で接続することにより、外界や内部からの刺激(イベント)に継続的に対応する通信システムの振る舞いを視覚的に表現できるという利点を持つ。しかし、この記述法単独では以下の様な問題がある。

(1)システムの機能が複雑化するにつれて、状態数が指数関数的に増大し、状態遷移図の規模が膨大となる。このため、システム全体の理解が難しい。

(2)多くの状態から同一の動作を行うイベントについては、その全てに対して遷移アークを記述する必要があり、冗長な記述となる。

上述(1)の問題については、機能的に分割された単純な状態機械（プロセス）の集まりとして記述する構造化設計手法がとられる[1)]。この手法の利点は、各プロセス毎に定義される状態の直積でシステム全体の状態を表すことにより、全体の状態数を削減できる点にある（図1）。しかし、この手法では、プロセス間に相互作用が必要な場合、各プロセスの状態遷移の中に相互作用を行う論理を埋め込んで記述する必要がある（図2）。

また、(2)の問題については、状態に集合的な包含関係を持たせ、状態の集合に対して、一括したアークにより、状態遷移と出力動作を記述する方法[2)]が提案されている。しかし、ある状態集合に対する動作が同一でも、その遷移先が異なる場合（例えば、あるイベントに対して、状態は遷移せずに同一動作を行う場合）には、各状態に対する遷移アークが必要であるという問題が残る。

本稿では、まず、通信システムの特徴を考察して、上述の問題も含め、その仕様記述法への要求条件を整理する。また、その要求条件を満たすために、動作の起動条件に着目し[3)]、相互作用を個別動作の重畳で定義する手法について提案する。次に、この手法を用いた仕様記述を実現するためのメカニズムとなる制約指向の考え方について概説し、制約指向プログラミングの通信ソフトウェアの仕様記述への適用法について述べる。最後に、本手法による通信ソフトウェアの仕様記述例を示し、評価する。

2．仕様記述法の要求条件と動作重畳手法

2.1　通信システム仕様記述法の要求条件

i）イベントドリブンな記述

大量の入力データを特定のアルゴリズムに従って加工する変換形システムとは異なり、通信システムは、外界や内部の刺激に対して、継続的に対応する必要がある。

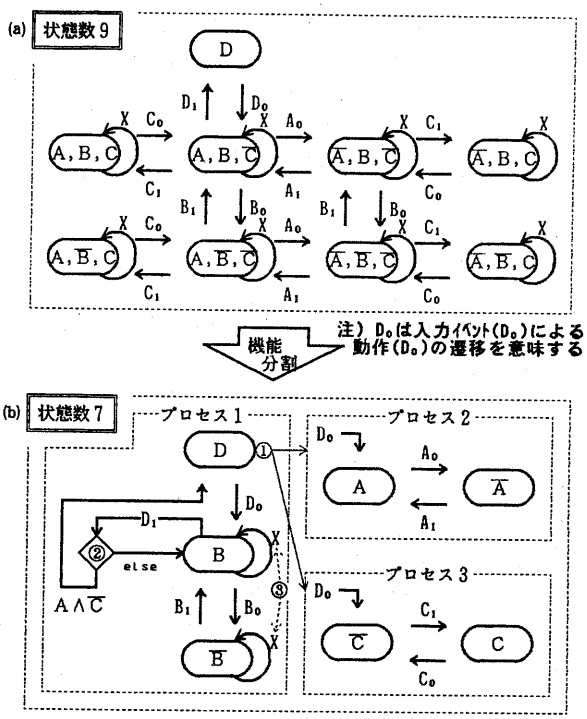


図1．状態の統合による状態数の削減

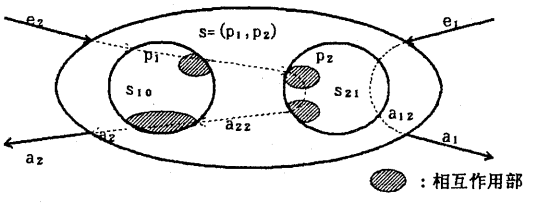


図2．機能分割された状態機械モデル

すなわち、イベントの発生を認識し、その内容と、その時のシステム状態に応じて、動作を決定する機構を表現できることが、通信システムの仕様記述への基本的要求条件となる。

ii）機能分割によるシステムの記述

複雑な機能を持つシステムの仕様化には、システムを機能的にまとまりのあるいくつかの単純なプロセスに分割した記述により、システム状態を表現するために必要な状態数の削減と、プロセス機能の単純化が必要である。例えば、図1(a)では、状態変数 A,B,Cの値の組合せすべてに対応して9個の状態が必要であるが、図1(b)では、3プロセスに分割したことにより、7個の状態で記述できている。

従来の状態遷移記法と構造化設計は、以上の2点を満足する重要な手法である。

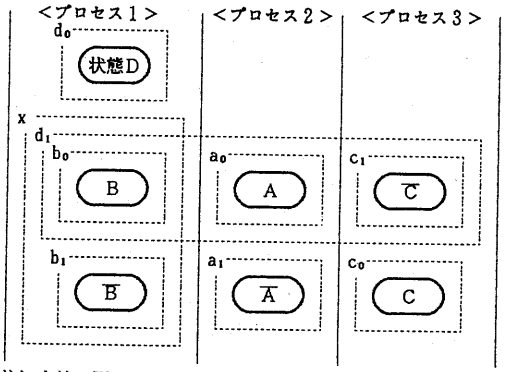
iii）機能部と相互作用部との分離記述

通信システムは、多数の呼を制御する必要があり、また、各呼の制御は、端末や通信リソース等、制御対象が本質的に持つ動作の並行性を考慮したものとなる。この特質から、通信ソフトウェアは、並行プロセスの概念に基づくものとなる[4]。並行プロセスは、機能部と相互作用部に分けることができる[5]。機能部は、状態変数を用いて、各プロセスの要求機能を実現するものであり、相互作用部は、プロセス間のインタラクションにより、同期や排他制御を行うものである。

構造化設計における機能分割では、機能部は、各プロセスに整然とマッピングされるが、相互作用部が各プロセスに分散配置される。すなわち、同一イベントが複数プロセスの動作に関与する場合(図1(b)①)や、他プロセスの状態が自プロセスの動作に影響する場合(図1(b)②)には、相互作用部を各プロセスに埋め込む必要がある(図2)。このため、各プロセスの機能仕様の独立性が阻害され、あるプロセスの機能変更が他プロセスの相互作用部に波及することになり、仕様の再利用が困難になるという問題がある。この解決のためには、相互作用部を抽象化し、機能部から分離して記述できる手法が必要である[6]。本問題の解決法が、本稿の主な目的である。

iv）動作に着目した記述

状態遷移図の記法では、多くの状態から同一の動作を行うイベントについては、そのすべてに対して遷移アークを記述する必要があり、冗長な記述となる（図1(a)遷移X参照）。構造化設計による機能分割により、この問題は一部緩和できる（図1(b)③では8アークから2アークになっている）が、その完全な解決とはいえない。一方、各動作に着目すると[3]、その起動条件は図3のように整理できる。したがって、記述の簡潔性のためには、状態のみに着目した記述だけでなく、動作も重要視する必要がある。



注）点線で囲まれた部分（d_0～x）が、図1の遷移（D_0～X）の起動条件に対応する
（例：遷移Xの起動条件は図1ではB∨$\bar{B}$であったが、この図で見るとプロセス1の状態D以外となる）

図3．状態の枠組みと動作の起動条件

2.2 動作重畳手法

前節の要求条件、特に、機能部と相互作用部との分離記述を実現するために、動作重畳の概念を用いた仕様記述法を提案する。この記述法は、構造化設計に従い、システムをいくつかの機能部（プロセス）に分割して記述する。また、機能部とは独立に、各プロセスに入力イベントを分配するとともに、各プロセスからの出力動作をまとめる相互作用部を記述するものである。

従来の仕様記述では、各イベントは、何れかのプロセスに入力され、必要な動作（状態遷移と出力動作）が行われる。そのイベントが他のプロセスに関与する場合や、他プロセスの状態を参照して動作を決定する必要がある場合には、関連するプロセスに信号が送られ、それに応じた動作が行われる。さらに、必要に応じて、別のプロセスに信号を送り、所要の動作を行わせる手続き（機能の実行と相互作用）を直列的に繰り返した結果として、システムの出力動作が行われる（図2参照）。

各プロセスは、これら、一連の手続きの過程で、信号を入力し、システムの出力動作に関与する部分的な動作を行っていると見なせる。また、入力イベントは、各プロセスへの入力信号を加え合わせたものであり、システムの出力動作は、各プロセスが行った部分動作の集まりで規定されると考えることができる。本記述法の概念は、図4に示すように、入力イベントを各プロセスの入力信号とする成分（部分イベント）に分解し、それをもとに各プロセスが並行的に行った部分動作を重畳した結果により、システムの出力動作を規定するものである。イベントの分解と動作の重畳を相互作用部としてまとめることにより、各プロセスが行う機能部の仕様の独立性が高まる。

機能部は、状態遷移記法の枠組みに従い、

〔部分イベント、状態、状態遷移、部分動作〕

の4つ組により、各プロセス毎の機能仕様を記述する。また、その記述形式としては、状態遷移および動作を行うための条件として、部分イベントの入力とプロセス状態を用いる。相互作用部は、部分イベントの分解と部分動作の重畳で決まるシステム動作定義であり、

入力イベント＝Σ部分イベント

システム動作＝F（Σ部分動作）

として記述する。

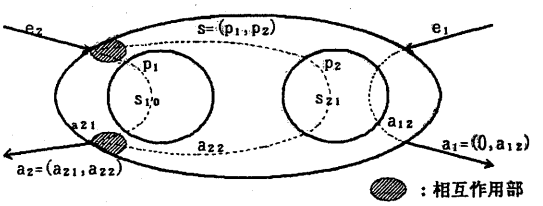


図4．動作重畳型モデル

3．制約指向と動作重畳手法

前節の記述の内、相互作用部では、各プロセスの部分動作を待ち合わせ、一つのシステム動作として統合する同期機構が必要である。これを状態遷移で記述すると、一連の動作決定待ち状態を定義する必要があり、複雑となる。この解決と実現機構の明確化のために、以下では、制約指向記述における遅延評価を用いて、上述の仕様記述に適合させる枠組みを示す。

3.1 制約論理プログラミング

制約論理プログラミング[7),8)]では、連立方程式に帰着できる問題を扱う場合、変数間の関係を論理式による制約の形で記述する（図5）。これに特定のゴール(<定数，未知数>の組)が与えられた時、意味単化と遅延評価（制約伝播）を基本とした制約評価の副作用として連立方程式が得られる。最終的にはシステムで用意した戦略（図5の場合、線型方程式の解法）に従って連立方程式を解いた結果として、その未知数の値が求まる。なお、意味単化は、構文上では等しく無くても、意味的には等しいもの（例えば、"2+3" と"5"）を等しいと見なす機構である。また、遅延評価は、あるゴールが与えられた時に直ちに評価出来ない制約を、制約の形のままで伝播させ、評価できる時点で解消する機構である。

```
制 約
  鶴亀（H，L，C，T）：－
    H＝C＋T，
    L＝2C＋4T．

ゴール
  鶴亀（4，10，鶴，亀）

副作用
    4  ＝  鶴  ＋  亀
   10  ＝  2＊鶴  ＋  4＊亀

解
  鶴  ＝  3
  亀  ＝  1
```

図5．制約論理プログラミングの例

3.2 制約指向による動作重畳手法

ここでは、制約論理プログラミングの枠組みにおける［(定数、未知数)、制約評価］を［(イベントと状態、状態遷移と出力動作)、状態分析］に対応させる（表1）。また、各プロセスの機能制約と、部分動作の重畳によるシステム動作を規定した相互作用制約の2階層に分離する。

機能制約は、部分イベントの入力とその時のプロセス状態を起動条件として、プロセスが行う状態遷移と部分動作を定義する論理式の集合であり、

［部分イベント∧状態　＝＞　状態遷移::部分動作］

の形の並列記述とする。なお、この論理式は、それぞれのプロセスごとにグループ化する。また、相互作用制約は、各プロセスの部分動作値の論理積を条件として、シ

表1．制約論理と通信システム仕様記述との対応

手法 / 項目	制約論理プログラミング	通信システムの仕様記述
制約	未知数間の論理的関係	条件（イベントと状態）と動作（出力と遷移）の関係
ゴール	既知の未知数の値	入力イベント（状態はシステムで既知）
制約評価の副作用	連立方程式	状態遷移タスク
制約評価の結果	未知数の値	動作（出力と遷移）

ステムの出力動作を定義する論理式の集合であり、

［∧部分動作　=>　システム動作］

の並列記述とする。

機能制約の条件に現れる部分イベント変数の論理値は、イベントの入力時に決定される。イベントの入力をゴールが与えられることと解釈し、意味単化により、その値が対応する部分イベント変数に代入され(値が決定する)、その論理式の評価が行われる。すなわち、イベント待ちのときは、部分イベント値が不定であり、論理式の評価が遅延されることになる。条件が真の場合には、評価の副作用として、右辺の論理変数(状態遷移、部分動作)に値(真)が代入される。相互作用制約の条件に現れる部分動作変数の論理値は、それを右辺に含む制約式の評価が行われた時点で決定する。何れかの変数値が不定の間は、論理式の評価が遅延される。したがって、システムの出力動作は、個別動作が揃った段階での遅延評価により定まる。

このように、遅延評価の機構により、イベントドリブンな記述と部分動作の統合を行うための同期が簡単に記述可能となる。なお、各式の条件に現れた論理変数は、その評価が完了した時点で不定となり、再びイベントが入力されるまで、論理式の評価が遅延される。

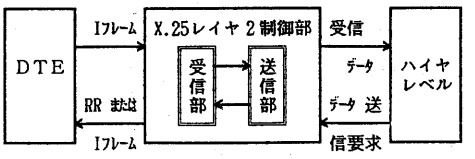


図6．プロセス構成図

4.. 仕様記述例

4.1 Iフレーム受信確認

例としてCCITT勧告によるX.25レイヤ2[9]仕様を送信部と受信部の2プロセスに機能分割(図6)して仕様記述する場合のIフレーム受信確認応答のシーケンス(図7)を採り上げる。勧告では、Iフレームを受信するとその確認応答をRRフレームまたは送信Iフレームの受信シーケンス番号N(R)で返送する必要がある。受信部がその何れを用いるかは、送信部の状態で定まる(図8)。

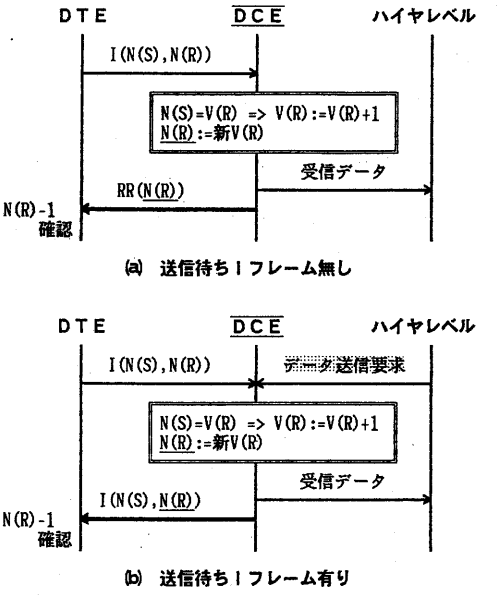


図7．Iフレーム受信確認応答シーケンス

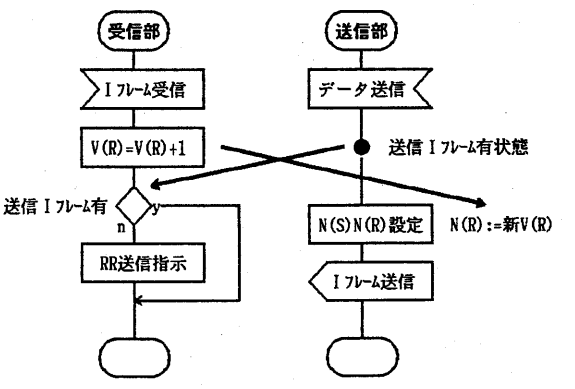


図8．状態遷移図による記述

すなわち、送信可能なＩフレームが無い場合には、RRフレームで返送し、有る場合には、そのＩフレームにN(R)の値を相乗りさせて返送する。また、送信部がＩフレームに設定するN(R)値は、受信部の受信状態変数vrを用いる必要があるという相互関係がある。

図６の受信部は、端末（ＤＴＥ）から受信したＩフレームの送信シーケンス番号N(S)をチェックし、それが正常なＩフレームの情報部をハイヤレベルに渡す機能を行う（このため、次に受信すべきN(S)値を状態変数vrに記憶しておき、N(S)=vr の場合に、そのＩフレームをハイヤレベルに渡し、状態変数vrを１増加させる）。

送信部は、ハイヤレベルからの送信データをＩフレームに編集して、ＤＴＥに送信する機能を行う。この時、送信ＩフレームのN(S)には、状態変数vsの値を設定する(その後、vsを１増加させる)。また、ＤＴＥからの受信フレームに設定された受信シーケンス番号N(R)の最大値を記憶するための状態変数L を持ち、vs-L>kの場合には、Ｉフレームの送信を待ち合わせる(ＤＴＥからの確認応答なしに連続して送信できるIフレーム数はk個である)。なお、その後、ＤＴＥから受信したフレームのN(R)値が状態変数L より大きい場合には、待ち合わせたＩフレームがvs-L≦kの範囲内で、送信可能となる。

本手法による記述例を図９に示す。受信部の機能制約は、Ｉフレーム受信により、RRフレームのN(R)で確認応答する記述となる（図９(a)第１式）。また、ハイヤレベルからのデータ送信要求に対しても、送信ＩフレームにVr値を設定する（第２式）。

送信部の機能制約は、ハイヤレベルからのデータ送信要求に対して、送信状態変数vsをN(S)に設定したＩフレームを送信するという個別動作の記述となる（図９(b)第１式及び第２式）。また、受信フレームの新N(R)値により、新たなＩフレームが送信可能となる（第３式）。

相互作用制約として、受信部と送信部の個別動作を重畳して、送信待ちのＩフレームがあればN(R)を相乗りさせ、無い場合はRRフレームにより確認応答を送信するシステム動作が記述される（図９(c)）。

この仕様記述における制約評価は、以下の順序となる。たとえば、Ｉフレーム受信時には、意味単化により、受信部の第１式の左辺の未知数<(I"S,*")受信>および、送

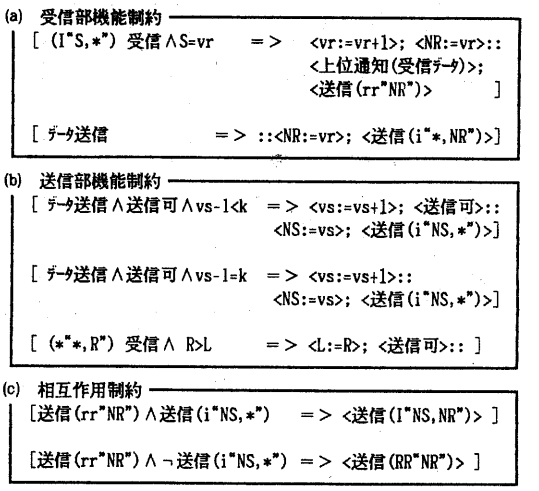


図９．Ｉフレーム受信確認応答の制約指向による仕様記述

信部の第３式の未知数<(*"*,R")受信>の値が真となる。この結果、受信部では右辺の未知数<送信(rr"NR")>の値が真となる。ただし、この様に制約が評価され右辺の値が決まると、左辺の未知数の値は不定に戻る。また、送信部において、仮に、送信待ちIフレームが有り(注参照)、条件R>Lが満たされると、未知数<送信可>の値が決定し、第１式または第２式が評価され、未知数<送信(i"NS,*")>が決定する。最後に、相互作用制約の出力部第１式が遅延評価され、未知数（出力動作)<送信(I"NS,NR")> が真となり、Ｉフレームの送信により、受信シーケンス番号N(R)を返送する動作が行われる。

（注）送信部第２式は、ハイヤレベルからのデータ送信要求時に、vs-1=kであれば、未知数<送信可>の値の代入が行われず、不定となる。このため、次のデータ送信要求に対しての第１式または第２式の評価は、前述のＩフレーム受信による未知数<送信可>の値の決定（第３式の評価）まで遅延される。この状態が、送信部において、送信待ちＩフレームがある状態を意味する。

4.2 P／Fビットの処理

X.25レイヤ2では、通信相手の状態を問い合わせるためにP／F（ポール／ファイナル）ビットが用意されている。勧告では、これを用いて、Pビット「1」のコマンドフレームを受信した場合はFビット「1」のレスポンスフレームを送出する必要があると規定している。

この機能を制約指向記述により記述すると図10の様になる。すなわち、受信部の機能制約は第3式に示す様にPビットが「1」のフレーム受信時に、Fビット「1」のフレームを送信するという制約記述となる。また、相互作用制約としては、送信フレームがRRフレームの場合、そのRRフレームにFビット「1」を設定して送信する（図10(c)第3式）。送信フレームがIフレームの場合は、Fビット「1」のRRフレームを送信後、Iフレームを送信するという制約になる（図10(c)第4式）。

この機能を、従来の状態遷移図により仕様化する場合には、送信部はIフレーム送信時に、受信部の状態を判定し、もし、Pビット受信後であれば、Fビット「1」のRRフレームの送信を待ち合わせるという相互作用を埋め込む必要がある。

P／Fビット処理の仕様追加（図10）では、前節と同様の制約評価により、未知数の値が求まる。この時、新たに追加された受信部第3式の未知数<(*.P)受信> の値も真となる。この結果、相互作用制約により、たとえば、未知数 <送信(RR"NR")> が真となった場合には、未知数 <(*.P)受信> との重畳により相互作用制約の第3式が評価され、Fビットを付加したRRフレームが送信される。

(a) 受信部機能制約

```
[ (I"S,*") 受信∧S=vr     ＝＞   <vr:=vr+1>; <NR:=vr>::
                                 <上位通知(受信ﾃﾞｰﾀ)>;
                                 <送信(rr"NR")>          ]

[ ﾃﾞｰﾀ送信     ＝＞ ::<NR:=vr>; <送信(i"*,NR")>]

[[(*.P)受信   ＝＞ ::<送信(*.F)>  ]
```

(b) 送信部機能制約

```
[ ﾃﾞｰﾀ送信∧送信可∧vs-1<k   ＝＞ <vs:=vs+1>; <送信可>::
                                 <NS:=vs>; <送信(i"NS,*")>]

[ ﾃﾞｰﾀ送信∧送信可∧vs-1=k   ＝＞ <vs:=vs+1>::
                                 <NS:=vs>; <送信(i"NS,*")>]

[ (*"*,R") 受信∧ R>L       ＝＞ <L:=R>; <送信可>:: ]
```

(c) 相互作用制約

```
[送信(rr"NR")∧送信(i"NS,*")     ＝＞ <送信(I"NS,NR")> ]

[送信(rr"NR")∧¬送信(i"NS,*")  ＝＞ <送信(RR"NR")> ]

[送信(*.F)    ∧送信(RR"NR")      ＝＞ <送信(RR"NR",F)> ]

[送信(*.F)    ∧送信(I"NS,NR")   ＝＞ <送信(RR"NR",F)>;
                                     <送信(I"NS,NR")> ]
```

注）網かけ部：追加仕様

図10. P／Fビット処理の機能追加

5．考　察

5.1 機能追加時の仕様の独立性

本手法を用いると、P／Fビット処理の機能追加は、受信部機能制約と相互作用制約の追加(図10の網かけ部)によって記述できる。この時、機能追加前の仕様(図9)の送信部には変更が不要となっている。これは、各プロセス(機能部)が独立に動作可能であることによる。

また、仕様追加は各機能部及び相互作用部において既に記述されている仕様とは独立に、制約を追加するだけで実現できている。したがって、本手法による仕様記述では、機能分割されたプロセス単位の仕様または、各個々の制約単位の独立性が保てるため、それらの単位で仕様の再利用の対象とできる。

5.2 簡潔な仕様記述

状態遷移図による仕様記述では、同様の遷移が複数状態で起こりうる場合（図1の遷移X参照）、各状態に該当する遷移を記述する必要がある。

本手法では、動作をその起動条件に着目して仕様記述するため、結果として、遷移の冗長記述が不要になる。たとえば、図1の例では、遷移Xの起動条件はプロセス1において状態D以外である（図3参照）。したがって、本手法による仕様記述では、プロセス1の個別制約として、【¬状態D＝＞遷移X】を記述するだけで済む。

なお、CCITT 勧告の記述は、前章の例の様に動作の起動条件と動作をサービス単位に分けて、冗長を省いた形で書かれているため、本手法との親和性が高い。

5.3 検討課題

本手法による仕様記述では、システムの動作が複数プ

ロセスの状態によって決まる場合も相互作用制約により記述可能である。しかし、たとえば、自プロセスの状態遷移が他プロセスの状態によって決まる場合等、相互作用制約に個別プロセスの遷移動作を記述する必要が出てくる。この様な場合、本来、各プロセスの動作はそのプロセスに関する記述だけで決まるという独立性が保証できなくなる。この対策としては、該当するプロセスの個別制約にそのプロセス内の遷移を行うためのイベント（オブジェクト指向のメソッド相当）を追加することが考えられるが、記述量が増える等の問題が残る。

また、サービスの実現のためには、制約の評価順序や評価の戦略をシステムで持つ必要がある。これについては、適用領域に大きく依存するため、適用領域を明確にした上で戦略等を実現する必要がある。

なお、本手法では、仕様を制約として小間切れ、かつ、独立に定義するため、システム全体としての仕様の理解性に欠ける。したがって、論理構造の可視化も今後の課題である。

6．むすび

状態遷移記述法と構造化設計は、複雑な機能をもつ通信システムの仕様記述として、重要な手法である。しかし、各プロセスの機能仕様に、プロセス間の相互作用機構が埋め込まれ、各機能仕様の再利用が困難になるという問題がある。本稿は、機能仕様と相互作用仕様を分離して記述するために、動作重畳手法を提案した。この手法は、入力イベントを各プロセスの入力信号とする成分（部分イベント）に分解し、それをもとに各プロセスが並行的に行った部分動作を重畳した結果により、システムの出力動作を規定するものである。また、部分動作重畳の同期機構を制約指向記述における遅延評価を適用することにより実現する枠組みを示した。さらに、この記述法をCCITT 勧告X.25レイヤ２の仕様化に適用するとともに、あるプロセスの機能追加による変更が、他のプロセスの機能仕様に波及しない例を示した。

本手法は、従来、プロセス間の信号授受により、動作の同期を行っていた機構を、動作の重畳による相互作用制約の遅延評価で代替するものである。これにより、各プロセスは、他プロセスとのインタラクション無しに、その動作を決定でき、プロセスの機能仕様の独立性が向上しており、仕様の再利用が期待できる。今後は、種々の仕様を記述してみることにより、その適用条件を明確化し、整理拡張していく必要がある。また、論理構造の視覚化と制約解消系も重要な課題である。

謝　辞

通信ソフトウェアの問題点を整理する上で貴重な御意見をいただいた、葉原耕平国際電気通信基礎技術研究所副社長に感謝いたします。また、熱心な討論をしていただいた当研究所山下部長、門田室長および、通信ソフトウェア研究室各位に感謝します。

【参考文献】

1) 川島 "交換ソフトウェアの基礎" 電子情報通信学会, 交換ｼｽﾃﾑ研究専門委員会, 専門講習会資料, (1988.6)

2) HAREL,D "On Visual Formalisms" CACM, Vol.31, No.5, (1988)

3) 林,西園,門田 "通信ソフトウェア仕様の制約論理型記述" 情報処理学会, 第37回全国大会, 5L-5, (1988)

4) 伊藤,市川 "通信分野における自動プログラミング" 情報処理学会誌, Vol.28, No.10, (1987)

5) Manna,Z & Wolper,P "Synthesis of Communicating Processes from Temporal Logic Specification" ACM TOPLAS, Vol.6, No.1, (1984)

6) Nishizono,T et al. "A Process Interaction Specification Method for Communication Software" Joint Conference on CNSS'88, Seoul Korea, (1988)

7) 相場 "制約論理プログラミング" bit, Vol.20, No.1, (1988)

8) 淵監修 "知識プログラミング" 知識情報処理ｼﾘｰｽﾞ8, 共立出版, (1988)

9) CCITT "Data Communication Networks; Interfaces. Recommendations X.20 - X.32." Red Book, VIII, Fascicle VIII.3, Malaga-Torremolinos, (1984)