〔公　開〕

ＴＲ－Ｃ－０１３２

# 分散ネットワークにおける通信ソフトウェア仕様の生成法

世良　孝文
Takafumi SERA

１９９６　３．１

ＡＴＲ通信システム研究所

# 分散ネットワークにおける通信ソフトウェア仕様の生成法

(株)ATR 通信システム研究所

1996 年 3 月

世良孝文

目次

## 1 はじめに

近年の通信ネットワークの高速化，大容量化の進展に伴い，これらネットワークを使った通信サービスも高度化が急速に進みつつある．また，通信端末の普及と高機能化の影響により，通信サービスも個人別の要求を満たすためのサービス，すなわちパーソナル化指向の通信サービスの実現が望まれている．このような要求のパーソナル化への対応の一つとして，通信サービスのユーザ自身が欲する通信サービスを定義することが可能になることは，ユーザがイメージし要求する新しい通信サービスを提供するための一つの方法である．

一方，通信サービスを提供する側からみると，ネットワークの高速化，大容量化の進展にともない，より高度な通信サービスを提供するための環境が揃ってきた．通信サービスが実際にネットワーク上で稼働する為には，新たに提供される通信サービスの仕様が既存のサービスに悪影響を与えないなどサービス仕様自身の正しさを保証しなければならないだけでなく，実際のネットワーク上に分散された各々のノードの通信ソフトウェアに変換してやらなければならない．通信サービスを実現するネットワークを構成する各々の分散ノードの通信ソフトウェアを誤りなく実現するには，通信サービスの設計者がノード間の同期など高度なネットワーク内部の知識に精通している必要がある．通信サービスは，新たにサービスが追加されることは多いが，既存のサービスが削除されることはほとんどなく，時代の流れとともに通信ソフトウェアは肥大化してきた．従来から高度な知識を持った専門技術者によって設計が進められてきたが，そこでも高度化，多様化の影響を受け人手ですべての設計上の誤りを取り除くために多大な労力を必要とするようになっている．機械的にサービス仕様から通信ソフトウェアの生成を支援したり，自動生成を行うことは，それらの問題に対する解決策の１つである．

我々は，通信サービス仕様から通信ソフトウェアを自動生成する研究の一貫として，機能が分散された複数のノードからなるネットワークにおいて，各々のノード毎の通信ソフトウェアに分割する手法について研究をすすめている．本稿では，アーキテクチャの違いを，論理的な機能の物理配置として定義し，ネットワークの外部から認識可能な端末の状態遷移として定義された通信サービス仕様から，ネットワーク内の機能の異なる複数のノードにおける通信ソフトウェアを自動的に分割して生成する手法について述べる．

## 2 分散ネットワークでの通信サービス実現

従来から通信サービス仕様を記述するために，いくつかのサービス仕様記述言語が提案されている．サービス仕様記述言語としては，SDL[1] やLOTOS[2, 3, 4]，L.0[5]，FRORL[6] などが使用されている．しかし，これらの仕様記述言語によって記述された通信サービス仕様を，通信ソフトウェア開発に適用して通信ソフトウェアを得るためには仕様の段階的な詳細化を行っていく必要がある．詳細化の手法としては，Cameron 他 [5] は，規則ベースの仕様記述言語を用いて実際のプロトコルを導出している．Tsai 他 [6] は，フレームと規則ベースの仕様記述言語 FRORL を用いている．いずれの方法も段階的な詳細化を行うことはできるが，その際にネットワーク内部の動作を理解している必要がある．つまり従来の詳細化手法は，主に通信システムの専門家によって人手で行なわれているという事実があり，以下のような問題を含んでいる．

1. サービスが複雑化すると，従来とのサービスの制御を矛盾なく処理できるようソフトウェアに詳細化するには，通信システム全体に対する広範囲な知識と技術が必要である．
2. 詳細化して設計を進める上で，人手によるソフトウェア誤りの混入する恐れがある．
3. サービス仕様を定義してからサービスをネットワーク上で提供するまでの間に長い開発期間がかかる．
4. 通信サービスのパーソナル化要求に対応することが困難である．

このような問題を解決するために，プロダクションシステムを用いた通信サービス仕様記述 STR が提案されている [7]．このシステムではサービス状態遷移の規則として通信サービスを定義する．このプロダクションシステムは，IF-THEN 形式の規則を用いているので，人間の断片的な知識を表現するのに適しているという特徴を持っている．通信サービスは状態遷移機械としてモデル化することができる [8]．この

ことを利用してネットワークを一つのブラックボックスと見なし，ネットワーク外部から認識可能な端末の状態遷移のみから通信サービスをとらえると，ネットワーク内部に精通しない非専門家でも通信サービス仕様を記述することが可能となる．端末の状態遷移としてSTR言語を用いて記述された通信サービス仕様は，端末単位にプロセスを割り当てるプロセス仕様に変換することができる[9, 10, 11, 12]．この変換により，通信ネットワーク内部を端末単位のプロセスがサービスの動作を行うソフトウェア仕様に詳細化を行うことができることが分かっている．これら端末対応のプロセスは，お互いに通信を行い他の端末の状態を知ることを前提にしており，サービスに必要な機能をすべて該当端末のプロセス内で保有しているという特徴を持っている．しかし，実際の通信ネットワークでサービスに必要なあらゆる機能を1つのプロセスが保有する場合はほとんどなく，何らかの機能の分散化が行われている．しかも，機能の分散化は地理的に分離された物理的な分散を伴うことが多い．

通信ネットワークは，通常複数の物理ノードから構成されているため，通信ソフトウェアの開発では，通信サービス仕様はネットワーク内のノードの間の協調動作を実現するために，明確に詳細化していかなければならない．通信ネットワークの果たす役割は，接続によって情報の送信，受信を行うことであり，一般に地理的に離れた場所にある端末同士が関係を持ってサービスが進行する．このように地理的に分離された通信サービスの実行環境においては，通信サービスの進行にあたり，地理的に分離された他の端末の状態を調べる必要があり，状態を探索する目的で通信が行われる．つまり，通信サービス仕様の詳細化を行うには，ネットワークを構成する各々のノード間で行う通信プロトコルを定義し，ソフトウェア仕様を求めることが必要である．

本稿の分散ネットワークにおける通信ソフトウェア仕様の生成法は，与えられたアーキテクチャに基づき生成された通信サービス制御ソフトウェアを用いてソフトウェアの合成法を確立することを主眼に置く．通信サービスの実現に必要な機能が物理的に分散されたノードからなるネットワークにおいて，各々のノードごとの通信ソフトウェアを自動生成する手法の確立を目的としている．

## 3　ネットワークアーキテクチャへの詳細化モデル

### 3.1　通信サービス機能

通信サービスの進行に必要なネットワークの制御を行うことにより通信サービス仕様がネットワーク内で実現可能となる．このネットワークの制御は，機器の制御を始めとして情報変換，加工などの情報処理，条件判断が含まれる．サービスの進行に必要な各々の制御をサービス機能と呼ぶ．一般にネットワークでは，各々のサービス機能は複数のノード上に点在し，お互いに協調してサービスを進行させる．すなわち機能分散が発生する．通信ソフトウェアの開発においては，各々のサービス機能が矛盾なく適宜に実行されるよう考慮しなければならないが，機能が分散して配備されることがネットワークを構成している各ノードに異なる通信ソフトウェアを設計しなければならない．しかもこれらサービス機能の分散は，各々ネットワークごとにアーキテクチャの配置条件が異なり，ネットワーク各々に対応した通信ソフトウェアを開発しなければならない．

通信サービス仕様から機能分散された通信ネットワークにおけるノードの通信ソフトウェアを生成するために段階的に詳細化を行う手法が有効である，通信サービス仕様は，通信サービス定義に記述された仕様を実現するために，ネットワーク内部の論理的な分散単位である論理エンティティがサービス状態に応じた状態遷移をすることによってネットワークサービスが実現されていると仮定する．サービス機能のうちネットワークのアーキテクチャの差異によって生じない論理的な機能を考える．つまり物理的なノードにサービス機能が配備されることによってアーキテクチャの条件が確定されると考える．論理的な機能を種別毎に分類し，その分類単位毎に仮想的なノードを構成し，論理機能の分散のみを考える．この仮想的なノードを論理エンティティと呼ぶ．論理エンティティは物理的な分散には依存しないため，あらゆるネットワークアーキテクチャに共通に通信サービス仕様を詳細化することができる．

アーキテクチャに依存した物理的な分散は，それら論理エンティティを物理ノードにどのように配置するかのみを与えることによって得られる．このように通信サービス仕様を論理的な機能の分散と物理的な機能の分散の2段階の詳細化を行うモデルを機能分散モデルとして提案している[13, 14]．

### 3.2 論理エンティティ

通信サービスの実現に必要な論理的な機能を種別毎に分類し，その分類単位毎に仮想的なノードを論理エンティティ (Logical Entity) と呼ぶ (図1). 論理エンティティは物理的な分散には依存しない. 通信サービスの持つ論理的な機能は，以下の4種類の論理エンティティに分類され，現在存在している通信サービス仕様を実現する論理機能はこれらのいずれかの論理エンティティに属している.

1. 接続制御論理エンティティ
   通信サービスのもっとも基本的な接続に関する制御を行う機能が属する. 主として物理的な通話パスなどの接続切断を制御したり，論理的なリンク接続制御などの機能が含まれる. 従来の交換機が所有しているハード制御と密着した機能はほとんどはこの論理エンティティに含まれる.

2. リソース管理論理エンティティ
   サービスの実現に必要となるリソースの管理に関わる機能が属する. 接続制御に直接関与しないハードウェアの制御に関する機能は，リソースとして管理され，それらの制御を行う機能としてこの論理エンティティに収容される.

3. データ管理論理エンティティ
   ハードウェアの実体はないが，サービスの実行に際し参照されたり変更されたりするデータを管理する論理エンティティである. 加入者に対する契約やサービス状態の一時的な管理のためのデータなど情報を制御する機能がこの論理エンティティに収容される.

4. サービス制御論理エンティティ
   サービスの実行を制御するための論理エンティティである. 他の論理エンティティのサービス状態の組み合わせによってサービスの進行を制御させるための機能を収容する. 条件判断，他の論理エンティティへの状態変更指示などの機能も収容される.

これらの4種類の論理エンティティは，通信サービスに必要な基本的な機能を，現時点で認められる通信サービス仕様の範囲で十分サービスの機能を実現することができる. しかしこれは将来に渡って4種類の論理エンティティのみであらゆる通信サービスが実現できるという保証を持つものではない. もし，これらの4種類の論理エンティティのいずれにも属さない新しい機能の概念が通信サービス仕様に組み入れられた場合には，論理エンティティの追加を行わなければならない. また，追加された論理エンティティが不要となり削除の場合も有り得る.

サービスの進行に伴い，ネットワーク上であるイベントが発生すると，端末のサービス状態として各論理エンティティがどういうサービス状態にあるかを知るためにお互いに通信を行いサービスの状態を探索する. この探索によってネットワークのサービスの状態が明らかになった時点で通信サービス仕様に定義された次のサービス状態へと遷移を行う.

### 3.3 物理ノード

物理ノードは最終的にネットワーク上の物理的な位置にハードウェア設備に対応して存在するノードである. 物理ノードの数，種類，接続関係などの条件は通信ネットワークのアーキテクチャごとに異なっており，一つの物理ノードは通信サービスを進行させるために，一つ以上の論理エンティティを収容する. すなわち，ネットワーク内のそれぞれの物理ノードに収容される論理エンティティを配置定義することによりその物理ノードが持つ論理機能が決定される.

### 3.4 機能分散モデル

通信サービス仕様定義をネットワーク内部の分散した物理ノードの仕様に変換する詳細化を段階的に行なうために，我々は，機能分散モデル (図2) を提案してきた [13, 14]. 機能分散モデルは，通信サービス仕様が要求するネットワークの機能を，アーキテクチャの種別に依らない論理的な機能配置で詳細化を行う. すべてのネットワークアーキテクチャに共通な論理的なプラットホームを提供することによって，要

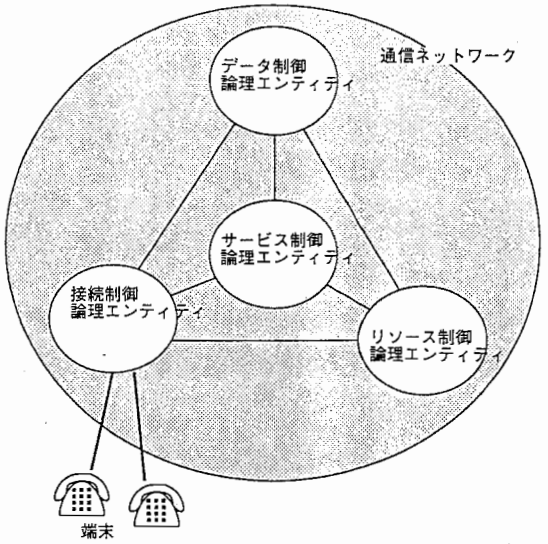


図 1: 論理エンティティ

求された一つの通信サービス仕様を様々なアーキテクチャ特性を持つネットワーク上のソフトウェア仕様を生成することが容易となる.

機能分散モデルは, 3階層から成る. 最初の階層は, サービス定義階層であり, ネットワーク内部を不可視としたブラックボックスと見なし, ネットワーク外部から認識できる端末の状態遷移で通信サービス状態遷移を規定する.

次の階層である論理分散階層は, 各サービスを実現するのには必要であるが, 各々のネットワークアーキテクチャによって異なる物理的な分散には依存しない機能を与える階層である. 通信サービス仕様は, 物理的な分散に依存しない論理的な機能に分類される. そしてサービス仕様はベンダーやネットワークのアーキテクチャに依存しない論理的なサービスアクションの定義へと変換される. 論理分散階層は, 通信サービスの実現のためにネットワークが持つ機能を論理的な機能種別で分類した論理エンティティから構成される. 各々の論理機能は, 物理的な分散には依存しない論理的な機能を示す. 各々の論理機能は, 論理エンティティ (Logical Entity) と呼ばれるエンティティにグループ化される.

詳細化最後の階層は, 物理分散階層とよぶ. 物理分散階層はアーキテクチャに依存した機能を実現する. この階層は論理機能を収容する物理ノードから構成される. この階層では, ネットワーク内のノードの分散や機能の配置状況を決定する. ネットワーク内のノードの分散や, 機能の配置状況を決定する. ネットワークのアーキテクチャの差異は, 物理的な分散形態によって引き起こされる機能であるから, この階層で様々なアーキテクチャ毎の物理的な配置や通信などの条件が反映される. 局所的な機能の物理的配置によって, 分散ネットワーク上の各々の物理ノードで実行すべき通信ソフトウェアを生成することが可能となる.

IN(Intelligent Network)[15, 16] は世界各地のネットワーク市場に浸透しつつあり, いくつかのサービスは既に市場へと導入されている. IN でのサービス仕様は, 論理プログラムであるサービスロジック (SIB) の起動制御として定義され実現される. IN のサービス生成環境 (SCE) では主として SIB の組合せによる起動制御の定義を支援するので, 既存の SIB の組合せだけでは実現できない新しいサービスに対してはサービス仕様を定義することができない. あらかじめ, 専門家の手によって SIB のインタフェースを用意しておく必要があるためである.

これに対して機能分散モデルでは, 通信サービス仕様は通信ネットワーク内部を隠蔽し端末の状態遷移として定義されるため, サービス仕様がネットワーク内部に実装された機能の有無によって影響を受けず, 独立して仕様定義が行われる. 新しいサービスが追加されることによって生じる論理エンティティの追加削除は, 機能分散モデルにおける段階的な詳細化においてアルゴリズムに影響を与えるものではない. 通信サービス仕様を論理エンティティのソフトウェア仕様に変換する詳細化は, 詳細化対象とする論理エン

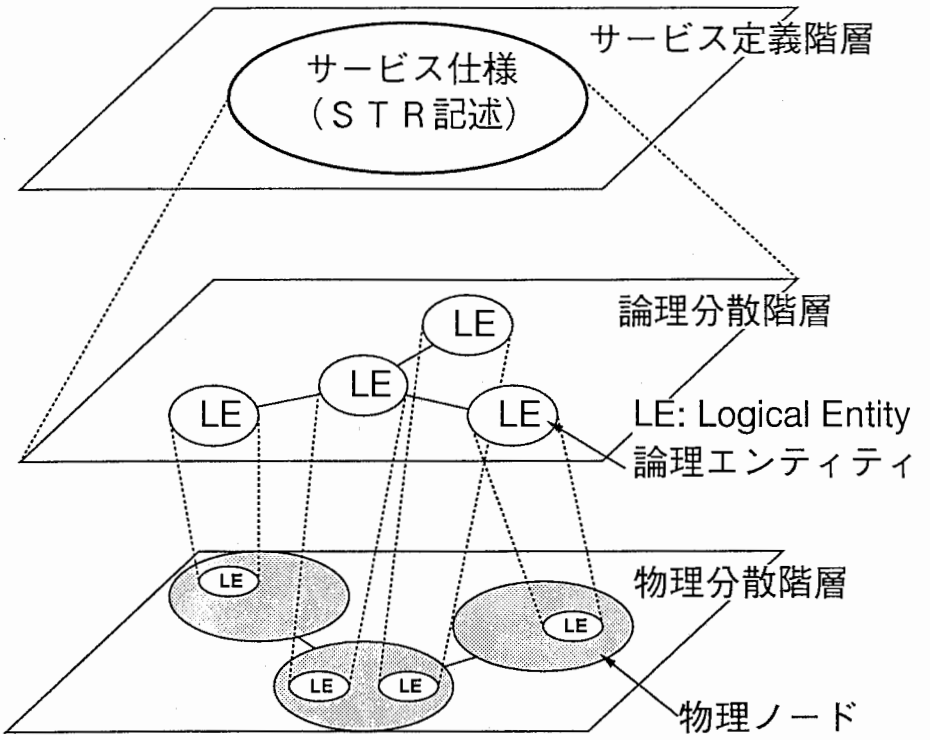


図 2: 機能分散モデル

ティティの種類が増えたり減ったりするだけであり問題なく追加削除を行っても良い.

### 3.5 サービス仕様記述

通信サービスの記述のための言語としてプロダクションシステムを用いたSTR言語について簡単に説明する.

STR言語では，プロダクション規則を記述する際に，サービス状態を表現する基本となるものを状態記述要素と呼び，ネットワークのサービス状態を複数の状態記述要素で表す．ある通信サービス状態でプロダクション規則が適用されるトリガーとなる条件をイベントと呼ぶ．通信サービス仕様はネットワークの外部から認識可能な端末の状態遷移規則の集合として定義する.

STRの規則においては，通信サービスの進行に伴って変化する端末のサービスの状態変化を現状態，イベント，次状態の組であらわす．現状態，次状態は構成要素である状態記述要素の組であらわす．状態記述要素は一つまたは二つの引数を有する．第一引数で指定される端末が，その状態記述要素で指定される状態にあることを表す．第二引数が記述されている場合には，その状態記述要素は第一引数の端末が第二引数の端末とその状態記述要素で表される関係にあることを意味する．イベントは一つあるいは二つの引数を有し，第一引数で指定される端末においてそのイベントが発生したことを表す．イベントの第二引数は，そのイベントにおいて入力された他端末の識別情報を表す．通常の電話サービスではダイヤルされた相手の電話番号である．規則はイベントが発生したときに，そのイベントにより端末の状態変化が起こる端末群の状態が現状態から次状態に遷移することを表す．一つの規則内で同じ端末変数は同一の端末を表す．STRで使用する状態記述要素とイベントは，同じ意味を持つものは唯一の表現をとり，異なった意味を持つものが同一の表現をとることはない.

図3にSTR言語によるサービス仕様記述の例を示す.

```
pots-1) idle(A)    offhook(A): dialtone(A).
pots-2) dialtone(A),idle(A)    dial(A,B):
                    ringback(A.B),ringing(B,A).
pots-3) ringback(B,A),ringing(B,A)
                    offhook(B): path(A,B),path(B,A).
```

図 3: STR 記述の例

### 3.6 タスク定義

通信サービス仕様は，端末の状態遷移を引き起こすネットワーク全体の状態変化として実現され，ネットワーク上で稼働する通信ソフトウェアは，これらサービス進行に伴うネットワークの状態遷移を引き起こすための処理を実行するように作られる．　この状態遷移の際に実行すべき一連の処理を我々は「タスク」と呼んでいる．すなわちタスクとは，ネットワークのグローバル状態のある現状態からある次状態に遷移する場合に，ネットワークが実行すべき処理をいう，と定義する．

ネットワーク内部をブラックボックスとして隠蔽し記述した通信サービス仕様では，ネットワーク内部の制御を行うタスクとサービス状態の対応が仕様中に定義されないため，通信サービス仕様を実際にネットワーク上で動作可能なソフトウェア仕様にするためには詳細化の段階で通信サービス仕様にタスク定義を加えてソフトウェア仕様を生成しなければならない．しかし，通信サービスに必要なタスクの定義は，各々の通信サービスの進行に伴うネットワーク内部の動作やノード間の同期などのさまざまなネットワーク内部の仕組みについての専門知識を必要するという理由から従来，ネットワーク内部の動作に精通した専門家が人手により行なっていた．なお，通信サービス仕様を定義する人を「通信サービス設計者」，ネットワーク内部のタスク記述を行なう人を「通信システム設計者」と呼んで区別する．

### 3.7 仮想リソース

ネットワーク内ではいろいろな仮想リソースが状態を遷移させることによってサービスを進行させる．サービスの進行を行なうためにネットワーク側が用意しなければならない仮想リソースを定義する [17, 18]．仮想リソースとは，一般の物理的なリソース以外に情報や条件成立有無なども含んだより論理的な仮想リソースと定義し，機能分散モデルの論理機能階層における各々の論理機能に対応する．しかし，仮想リソースはネットワーク内のアーキテクチャに依存しない機能の集合を考える．

STR 記述で用いられる通信サービス仕様は，状態記述要素によってサービスを定義する．状態記述要素は，通信サービスの進行に伴うネットワークの広域状態を表現している．すなわち，仕様定義から導き出されるサービス状態は，ネットワークを構成する論理エンティティ各々のサービス状態を表していることから，状態記述要素がいずれかのネットワーク内仮想リソースの状態と対応する．状態記述要素は，通信サービスの進行に伴うネットワークの広域状態を表現している．すなわち，仕様定義から導き出されるサービス状態は，ネットワークを構成する論理エンティティ各々のサービス状態を表していることから，状態記述要素がいずれかのネットワーク内仮想リソースの状態と対応する．この仮想リソースの状態遷移とその遷移に対応に関する知識をあらかじめ通信システム設計者によって作成し用意することによって，通信サービス仕様に対するタスク定義を求めることができる [19]．

## 4 分散ネットワークの詳細化アルゴリズム

### 4.1 状態遷移規則の生成

通信サービス仕様記述で用いている STR は，ネットワーク全体の状態遷移を規則で記述し表現する．分散ネットワークのソフトウェア仕様は，詳細化を段階的に実施してより詳細な仕様を求めることになるため，入力となる状態遷移規則で記述された通信サービス仕様に対して，求めるソフトウェア仕様は詳細

化した状態遷移規則として記述する (図 4). このことにより，詳細化のアルゴリズムを段階的に何度も適用することが可能となる [20, 21, 22].

また，既に述べたようにSTRで記述されたサービス仕様から，端末対応にプロセスを割り当てるソフトウェアを自動的に生成することができる．この方法を用いるとSTR言語で記述された仕様は，さらに詳細化する必要がなくなった時点で適宜にソフトウェアに変換することが可能となる．

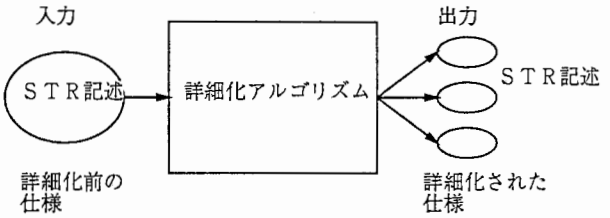


図 4: 詳細化による状態遷移規則の生成

## 4.2 論理エンティティのソフトウェア仕様の導出

我々は，STRで記述されたサービス仕様から通信プロトコルを生成するために，STR規則をグラフ表現として用いている [23]. 一つのSTR規則は，現状態とイベントと次状態記述のグラフの2つのグラフとして表される． STR規則中の現状態とイベントは，合わせて1つのラベル付きの有向グラフとして表現できる．同様に次状態も1つの有向グラフで表現することができることから，STRで記述された通信サービス仕様は，有向グラフの書き換え規則の集合とみなすことができる．グラフ表現されたサービス仕様は，端末対応に一つのプログラムを割り当てるサービス制御プログラムに変換される．これは，有向グラフの書き換え規則の集合として表現された通信サービス仕様から，適用条件を満足する規則探索のための通信を行なう端末対応プロセスのソフトウェア仕様としてを導出することができる．各端末対応プロセスのソフトウェア仕様は，通信システム全体を表す有向グラフに含まれる同形な部分グラフを探索する分散アルゴリズムを実現する．

端末に対応するプロセスは，他のプロセスの状態をプロセス間の通信によって知ることができる．プロセス間の信号の一つは，他のプロトコルエンティティの端末状態を問い合わせるREQUEST信号であり，もう一つの信号は，端末状態と状態遷移を通知するためのRESPOND信号である (図 5).

実際の通信ネットワークでは，機能の異なる複数のノードがお互いの協調作業によってサービスを実現している．一つの端末に関するサービスの状態遷移においても，サービスに必要な機能に応じて複数のノードが関与する．すなわち，一つの端末の状態がネットワーク内の複数のノードの状態により表現されることになる．端末に入力される端末イベント，状態探索信号，状態遷移通知信号をイベントとみなすと，プロセスのソフトウェア仕様は，一つの端末の状態を広域状態とするサービスの状態遷移機械とみなすことができる．したがって，プロセスのソフトウェア仕様もまたSTR規則として表現することができる．また，STRで使用する状態記述要素とイベントは，同じ意味を持つものは唯一の表現をとり，異なった意味を持つものが同一の表現をとることはない． STR記述で用いる状態記述要素とイベントは，サービスの意味を実現するのに必要な機能に対応しており，かつ一意に決まる．従って，状態記述要素とイベントを機能分類対応と，物理的な分散ノードへの対応を与えることによって，分散ネットワーク上の各ノードで動作する通信サービスソフトウェアに変換することができる．

またこのとき，通信サービス仕様からプロセスのソフトウェア仕様に変換する際に生成された，状態探索信号や状態遷移通知信号の送信のタイミングや，適用条件を判定する条件判断などは，STR規則によるプロセスのソフトウェア仕様には含まれない．したがって，これらはSTR規則とは別に保持しておく必要がある．我々は，サービス状態の遷移の際に実行すべき一連の処理をタスクとよんでいる．タスクは，サービスの進行に必要であるが，詳細化における分散の段階に依存しないし，サービス状態遷移による仕様に対してライブラリとして用意されるものである．

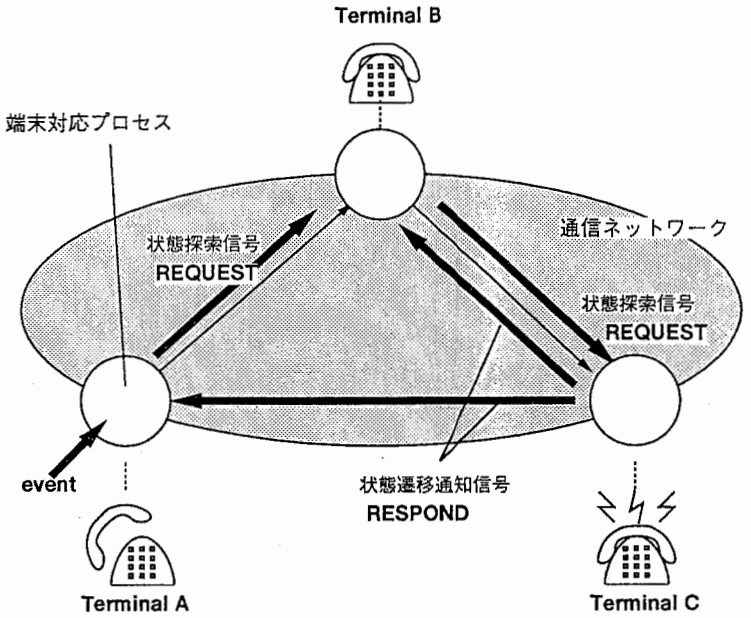


図 5: 端末対応プロトコルエンティティ間通信

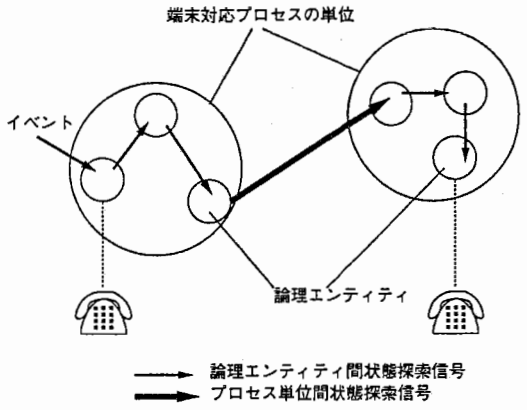


図 6: 論理エンティティ間のプロトコル自動生成

機能分散モデルの論理分散階層では，物理的な分散に依存しない論理的な機能で分類した論理エンティティを定義している．機能分散モデルの論理分散階層において，通信サービス仕様を満たす一つの端末状態の遷移を行なわせるための論理エンティティの機能は一意であり，同じ機能が複数存在することは機能の物理的な分離を意味するため許されないことを意味する．状態記述要素とイベントは，STR 仕様上唯一の表現をとるので，STR 記述された通信サービス仕様では，サービスの進行に伴う状態記述要素の増減変化と生起イベントがサービス実行に必要な機能を表現する．各々の状態記述要素やイベントが担う論理的な機能に分類して，これらを論理分散階層における論理エンティティの分割単位とすることができる．したがって，ネットワークをブラックボックスとしてとらえた通信サービス仕様は，状態記述要素とイベントごとにそれが示すサービス機能の論理的な分類を行なうことができる．このことから論理機能の配置を考えるとき，状態記述要素とイベントで論理機能の配置を行なうことができる．

通信サービス仕様から変換されたプロセスのソフトウェア仕様を，各々の機能分散階層の論理分散階層の論理エンティティのソフトウェア仕様に変換する．このとき論理機能の異なる複数の論理エンティティ毎のソフトウェア仕様に分解する際に，通信サービス仕様を記述した状態記述要素とイベントの各々に論理エンティティを対応させるための論理機能対応知識を与える．ネットワーク上のある端末が通信サービスの進行で状態遷移する時，サービス状態遷移に必要な機能がどの論理エンティティに対応し実行されるかを決定可能である．サービス動作の各論理エンティティで実行すべき仕様に分割するための定義として，各々の状態記述要素とイベントに対する収容論理エンティティを与える．これを論理機能対応定義と呼ぶ．論理機能対応定義は，各論理エンティティ毎に収容する状態記述要素とイベントを記述したもので，一つの論理エンティティに対する論理機能対応定義は以下の形式で記述する．

Primitives:

$$entity = p_1, \cdots, p_n$$

Events:

$$entity = e_1, \cdots, e_m$$

上の形式において，$entity$ は論理エンティティの名称，$p_i(1 \leq i \leq n)$ はサービス状態における状態記述要素である．また，$e_j(1 \leq j \leq m)$ は，イベントである．一つの状態記述要素が，複数の論理エンティティで対応定義されることは許さない．また，一つの論理エンティティでは，複数の論理機能を実行することができる．論理機能対応定義は，通信サービス仕様に含まれる論理的なサービス機能の分類を与えるものであり，ネットワークのアーキテクチャ種別に依らずに定義することができる．

### 4.3 物理ノードのソフトウェア仕様の導出

論理機能対応定義に対し，各々の論理エンティティのソフトウェア仕様をどの物理ノードで実行するかをアーキテクチャに従って定義する．これを物理機能対応定義とよぶ．アーキテクチャの違いによって論理エンティティの物理配置が異なる場合は異なる物理機能対応定義を用意しなければならない．各物理ノードへの分散形態は，それぞれのアーキテクチャごとに異なっており，アーキテクチャが異なれば，各々の物理ノードへの機能の配置や構成が異なるため，各物理ノードへの機能配備に従った詳細化を行なう必要がある (図 7).

論理機能対応定義と物理機能対応定義の 2 種類の配置定義を用いて，2 段階の詳細化を行なうことによって，ネットワーク上の分散したノードごとの機能に応じた通信ソフトウェアを生成することができる．第 1 段階では，通信サービス仕様は論理的な配置構成に基づく論理エンティティのソフトウェア仕様に変換される．第 2 段階では，その論理エンティティのソフトウェア仕様は，物理的な配置定義に従って，ネットワーク内の物理ノードのソフトウェア仕様に変換される．配置構成定義は，各々の端末の状態を定義するための状態記述要素とその端末に対して生起するイベントによって指定される．つまり，配置構成定義は各々のサービス仕様とは独立して定義可能となっている．最初の詳細化で求められる論理エンティティのソフトウェア仕様は，ネットワークの物理的な機能配置に依存しないので，各々のネットワークのアーキテクチャに依存しない，通信サービスを実現するための共通のプラットホーム上でのソフトウェア仕様を導き出す．

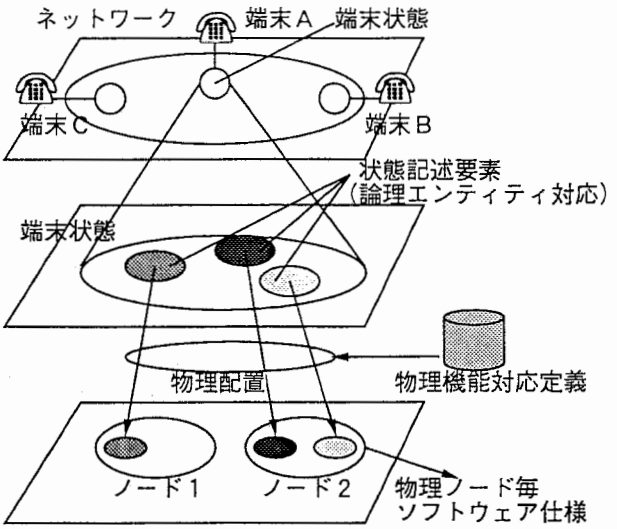


図 7: 物理機能対応定義によるソフトウェア仕様の生成

### 4.4 ネットワーク仮想リソース知識の構築

STR 言語では，通信サービス状態遷移を定義するために端末状態を状態記述要素で表現する．仕様定義から導き出されるサービス状態は，ネットワークを構成する論理エンティティ各々のサービス状態を表していることから，状態記述要素がいずれかのネットワーク内仮想リソースの状態と対応する．この対応を状態記述要素とネットワークの仮想リソース状態の対応知識として定義し用意する．これを，状態記述要素－仮想リソース状態対応知識と呼ぶ．図 8は，状態記述要素と仮想ネットワークリソースの関係を示す．この図では，状態記述要素は 3 種類のリソースの状態と関係を持っている; 各々の状態記述要素は，各々 3 つのリソースの仮想リソース状態と対応している．たとえば，状態記述要素 $p1$ はリソース状態 $s1\_1, s2\_1, s3\_1$ と対応している．

状態記述要素－仮想リソース状態対応知識では，ある状態記述要素がどの仮想リソースのどの状態に対応するかを定義してあり，一つの状態記述要素が複数の異なる種類の仮想リソースにおける仮想リソース状態に対応することがある．ただし，一つの状態記述要素が一つの仮想リソースの異なる複数の仮想リソース状態に対応することは許されない．

サービスの進行に伴い，各々の仮想リソースの状態変化が発生する．さらに，各々の仮想リソースに対応した論理機能サービスの進行は，各々の仮想リソースの状態変化が組み合わされる．この仮想リソースの状態変化を行なわせるのに必要な手続き処理をタスク部品と呼ぶ．タスクとは，サービスの進行に伴って変化した状態の遷移の際に実行しなければならない一連の手続き処理であり，仮想リソースの状態変化を行なわせるタスク部品の集合で構成される．図 9は，仮想ネットワークリソース"resource1" の状態遷移を示している．この例においては，各々のリソース状態遷移 $t1, t2, t3$ はそれぞれ $T1, T2, T3$ のタスク部品に対応している．

次に述べる手順によりこの通信サービス仕様から，ネットワークが実行すべきタスクを導出することが可能となる．

- 手順 1
  通信サービス仕様から，通信システムの初期状態から実際に遷移可能なサービス状態とサービス状態遷移を求める．
- 手順 2
  手順 1 で求めた各サービス状態の遷移に対して状態記述要素の増減を求める．
- 手順 3

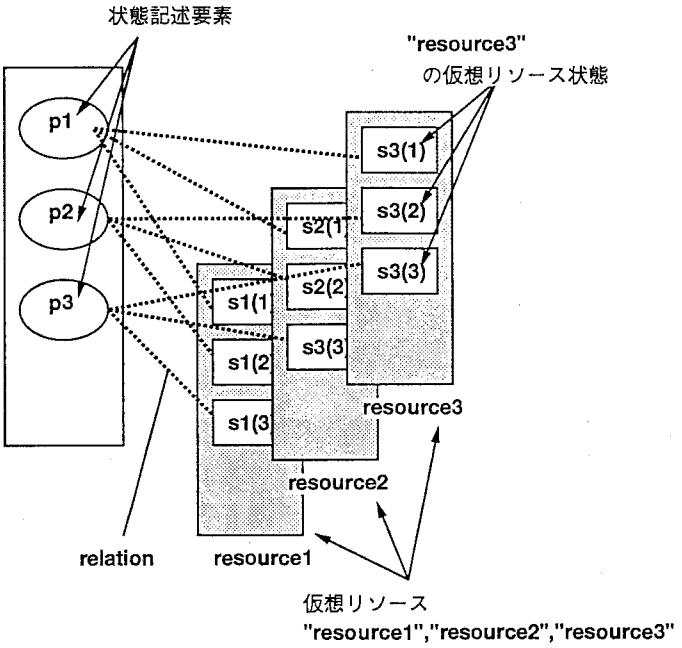


図 8: 状態記述要素と仮想リソース

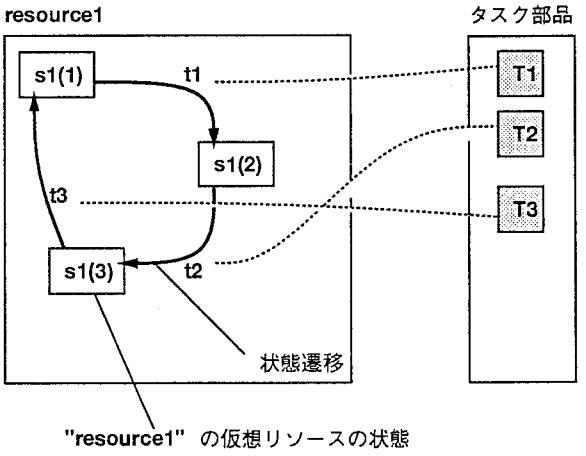


図 9: 仮想リソースとタスク部品

各々のサービス状態遷移を構成する状態記述要素各々に対する仮想リソースを求め，サービス状態遷移に対応する仮想リソース状態の遷移を求める．

- 手順 4
  各々の仮想リソースに対する仮想リソース状態の遷移に対応したタスク部品を，仮想リソース状態遷移－タスク部品対応知識から求めることにより，各サービス状態の遷移に対する各々の仮想リソース毎のタスク部品が求まる．一つのサービス状態遷移に対して対応する全ての仮想リソースのタスク部品を集めてタスクを構成する．

以上の手順で通信サービス仕様に定義されたサービス状態遷移に対するタスクを自動的に導出することができる．

このタスクの自動導出手段を用いると非専門家による通信サービス仕様の定義が行なわれた場合においても，ネットワークが必要とするタスクを求めることができる．しかしながら，この手順においては 2 種類の専門家のノウハウの蓄積として用意した対応知識の欠落のために，知識に合致しない新しいタスク定義が発生する可能性がある．

一つは，サービス定義に使用されている状態記述要素に対応する仮想リソースとリソース状態の対応知識が定義されていない場合であり，もう一つは仮想リソースの状態遷移に対するタスク部品が用意されていない場合である．前者の場合は，検出された状態記述要素を，新規状態記述要素として仮想リソースおよび仮想リソース状態を定義するように専門家に仮想リソース対応知識の不足を通知し知識に追加することができる．また，どの既存の仮想リソース種別にも状態記述要素が対応しない場合には新規の仮想リソースを追加することができる．していないことを専門家に通知し，新たにタスク部品を用意する必要があることを示すことができる．専門家はこの通知を受けて該当する仮想リソース状態の遷移を行なわせるタスク部品を定義し，知識に追加することにより不足は解消される．

## 5 評価

### 5.1 UPT サービスへの適用例

次世代の新しい通信サービスとして UPT(ユニバーサルパーソナル通信) サービスが CCITT 勧告で標準化が進められている [24]．UPT サービスは，UPT ユーザが場所を移動して，移動先のネットワークにおいて発信，着信の通信を行なうことのできるサービスである．UPT ユーザは，ユーザ毎に固有の UPT 番号を持ち，移動先の端末から通信を行うに当たって UPT 番号と認証コードを用いて利用の正当性を確認するための認証手順を行う．認証終了後，UPT ユーザは UPT サービスを利用することが可能となる．IN アーキテクチャにおいては，GFP(General Functional Plane) では分散を意識しないサービス制御ロジックが定義され，DFP(Distributed Functional Plane) では分散を意識した制御が定義される．つまり，サービス制御プログラムは分散を意識せずに作成され，コンパイルまたは分散状態の記述により分散を意識したサービス制御プログラムに変換されて実行される．しかし，機能エンティティ (FE) で分散を意識するとはいっても DFP では物理的な機能の分散が含まれており位置情報が固定で組み込まれている．その意味で IN における機能エンティティの動作は当機能分散モデルにおける物理分散階層に対応する．すなわち物理機能対応定義として DFP における機能配置を用意することによって，IN と同じ通信サービス仕様を実現する各機能エンティティ毎のサービス仕様を導出することができる．

### 5.2 今後の課題

本稿の詳細化手法を用いると，通信サービス仕様はネットワークの内部に依存せずに定義され，詳細化の過程で自動的に内部のプロトコル仕様を生成して，新しいサービスの追加に伴う全てのノードに対するサービス制御ソフトウェアの生成を行なうことができる．非専門家が定義した通信サービス仕様から，サービス仕様の詳細化を自動的におこない，ネットワーク内部の異なる機能から構成される複数の各物理ノードの通信ソフトウェアに自動的に変換することができる．

また，通信サービスを実現するネットワークの機能に着目した機能分散モデルにおいてネットワークの仮想リソース概念を導入し仮想リソースの状態遷移としてネットワーク内の機能を定義し通信サービス仕様からネットワーク内の機能動作を決定するタスクを自動生成することができる．

1. 対応する物理分散形態の拡張
   アーキテクチャの違いを機能として明確に定義する方法については，物理分散階層のノードへの論理機能の配置においてまだ十分表現できない．現在の配置定義方法では，ノードと論理機能，すなわち状態記述要素との対応で配置定義を行なっているため，一つの論理機能を分割して複数ノードに配置する場合を考慮していない．これは物理分散に伴う機能であり，物理分散階層の重要な課題である．

2. 論理機能対応定義の自動生成の可能性検討
   通信サービス仕様で用いられている状態記述要素やイベントに対応する論理機能対応定義は現在人手にて作成していかなければならない．しかしこの定義はサービス仕様の要求理解で用いる状態記述要素やイベントの意味解釈から導かれる機能要素の取り出しを検討し，アーキテクチャと機能要素の対応から配置定義を自動的に導出することができると考えている．これにより，一度与えたアーキテクチャ定義を用いて自動的に機能を配置することも可能となると考えられる．

3. 仮想リソース知識の自動生成の可能性検討
   通信システムの基本機能をインプリメントに依存しない論理的な機能として表現する概念モデルの研究が進められている [25, 26]．この概念モデルを用いるといろいろな人が記述した通信サービス仕様中に表現される仕様記述用語がサービスの一つの基本概念として整理される．この基本概念を用いて仮想リソースが表現する論理的な機能と通信サービス仕様に表現されているサービスの基本機能の対応知識を構築することができれば，通信サービス仕様記述から対応する仮想リソースの状態が求められ，これを用いて自動的にタスク定義を求めることが可能となると考えられる．

   このために必要な課題としては，概念モデルで持つ概念知識と仮想リソースとして表現されたサービスの基本機能との対応関係をどうやって整理するかを検討する必要がある．

## 6 おわりに

本稿では，通信ソフトウェアの自動生成に関する手法として以下の点について議論した．

- 論理的な機能の分散と物理的な機能の分散の二段階の詳細化を行うモデルとして機能分散モデルを用意する．このモデルの論理機能階層ではアーキテクチャに依らない論理的な機能の分散を明らかにすることができる．
- 通信サービス仕様とアーキテクチャ仕様を与えて，分散ノードから成るネットワークの通信ソフトウェア仕様を自動的に生成することができる．
- 通信サービス仕様が持つ論理的な機能を示す仮想リソース知識を用いてサービス仕様の詳細化に必要なタスク定義を自動的に求めることができる．

謝辞 本研究を進めるにあたり，熱心な御討論を頂いたＡＴＲ通信システム研究所の研究員の皆様に深く感謝いたします．

## 参考文献

[1] CCITT Recommendations Z.100: “CCITT Specification and Description Language (SDL)” (1992).

[2] M. Faci, L. Logrippo and B. Stepien: “Formal specification of telephone systems in lotos: the constraint-oriented style approach”, Comput. Networks and ISDN Syst., **21**, pp. 53–67 (1991).

[3] L. Drayton, A. Chetwynd and G. Blair: "Introduction to lotos through a worked example", Computer Communications, **15**, 2, pp. 70–85 (1992).

[4] T. Bolognesi and E. Brinksma: "Introduction to the iso specification language lotos", Computer Networks and ISDN Systems, **14**, pp. 25–59 (1987).

[5] E. Cameron, D. Cohen, T. Guithner, W. Keese, Jr., L. Ness, C. Norman and H. Srinidhi: "The l.0 language and environment for protocol simulation and prototyping", IEEE Trans. on Compu., **40**, 4 (1991).

[6] J. Tsai, T. Weigert and H.-C. Jang: "A hybrid knowledge representation as a basis of requirement specification and specification analysis", IEEE Trans. on Software eng., **18**, 12, pp. 1076–1100 (1992).

[7] Y. Hirakawa and T. Takenaka: "Telecommunication service description using state transition rules", Int. Workshop on Software Specification and Design, pp. 140–147 (1991).

[8] G. v. Bochmann: "A general transition model for protocols and communications service", IEEE Trans. Comm.,COM-28(4), pp. 643–650 (1980).

[9] K. Kawata and Y. Hirakawa: "Efficient process generation from state transition based telecommunication service specifications", Proc. of 5-th JC-CNSS, pp. 3–8 (1992).

[10] 河田, 田倉, 太田：“宣言型通信サービス仕様記述言語からプロセス仕様を生成する手法”, 電子情報通信学会 人工知能と知識処理研究会 AI92-65 (1992).

[11] 河田, 田倉, 太田：“通信ソフトウェア生成システムにおけるネットワーク制御タスクの生成機構”, 電子情報通信学会 交換システム研究会 SSE92-46 (1992).

[12] 田倉, 河田, 太田：“端末動作に着目した通信サービス仕様記述とその実行環境について”, 電子情報通信学会 交換システム研究会 SSE92-137 (1993).

[13] 世良, 田倉, 太田：“端末動作記述によるサービス仕様記述の機能分散モデルへの展開”, 電子情報通信学会 交換システム研究会 SSE93-155, pp. 63–68 (1993).

[14] T. Sera, A. Takura and T. Ohta: "Distributed functional model for telecommunication service specifications described with terminal behaviors", Proc. of 7-th JC-CNSS, pp. 299–304 (1994).

[15] "Special issue: Marching toward the global intelligent network" (1993).

[16] CCITT SG XI Draft Recommendation: "Q.1200 Series Intelligent Network" (1992).

[17] 世良, 田倉, 太田：“機能分散モデルによるサービス機能展開の考察”, 電子情報通信学会 秋季全国大会予稿集 B–539 (1994).

[18] 世良, 田倉, 太田：“機能分散モデルを用いた機能配備法”, 電子情報通信学会交換システム研究会 SSE94-167, pp. 7–12 (1995).

[19] T. Sera, A. Takura and T. Ohta: "Task generation for distributed functional model", International Jounal of Artifical Intelligence Tools (1996).

[20] 世良, 田倉：“プロダクションルールを用いた通信サービス仕様の詳細化”, 情報処理学会 第51回全国大会予稿集5分冊 6M-4, pp. 151–152 (1995).

[21] 世良, 田倉, 太田：“機能分散モデルのアーキテクチャ配置の考察”, 電子情報通信学会 交換システム研究会 SSE95-41, pp. 7–12 (1995).

[22] T. Sera, A. Takura and T. Ohta: "Architecture refinement for distributed functional model", IASTED International Conference : NETWORKS, pp. 173–176 (1996).

[23] A. Takura, T. Sera and T. Ohta: "Protocol synthesis from service specifications described by graph rewriting rules", 京都大学数理解析研究所共同研究集会 書き換えシステムの理論と応用 (1995).

[24] CCITT Draft Recommendation F.851: "Universal Telecommunication (UPT) - Service Description" (1992).

[25] 榎木, 高見, 太田 : "通信サービスの概念モデルの一考察", 電子情報通信学会交換システム研究会 SSE93–77 (1993).

[26] H. Enoki, Y. Kobayashi and T. Ohta: "A method for discriminating ambiguous terms in specifications by using a conceptual model", Proc. of 7-th JC-CNSS, pp. 325–330 (1994).

# 付録

## A　通信サービス機能の分散ソフトウェア生成のアルゴリズム

通信サービスの実現環境として，機能が分散された各論理エンティティのソフトウェア仕様を生成するための基本構成について説明する．

### A.1　論理分散機能

通信サービス仕様から端末対応のプロセス仕様を導出するアルゴリズムでは，STR による通信サービス仕様から変数種別（端末，人間など）毎のプロセス仕様を生成している．しかし，実際の通信ネットワークでは，プロセスはさらに論理的な個々の機能を実現する論理エンティティの協調動作によってサービス状態遷移を実現する．この協調動作に対応するソフトウェア仕様を出力として生成する．たとえば，端末または人間対応のプロセスのソフトウェア仕様を，各々の機能分散した論理エンティティのソフトウェア仕様に分割する．各々の論理エンティティは，異なる物理ノードに配置される．

### A.2　プロセスソフトウェア仕様の分割概要

#### A.2.1　分割単位

論理機能の分割は，STR 記述における状態記述要素の論理エンティティ収容定義により行なう．したがって，端末対応のプロセスの実状態に対する各状態記述要素の論理エンティティ分割を考慮するだけでよい．なおこの論理エンティティ収容定義は，各変数（端末毎，人間毎．．．）に対し実変数毎に行なわれる．

収容定義の例（詳細未定）：エンティティ１エンティティ２エンティティ３．．として各論理エンティティ毎の収容を定義する．

各状態記述要素がどの論理エンティティ収容かを定義する．

```
Primitives:
論理エンティティ１ = {状態記述要素名，状態記述要素名，．．．}
論理エンティティ２ = {状態記述要素名，．．．}
```

イベントについても生起する論理エンティティを定義する．

```
Events:
論理エンティティ１ = {イベント名，イベント名，....}
論理エンティティ２ = {イベント名，．．．}
```

#### A.2.2　状態探索信号

通信サービス状態遷移を実現するプロセスの処理は，２つのフェーズから成る．一つは，状態探索フェーズであり，もう一つは状態遷移確定フェーズである．状態探索フェーズは現状態で適用可能な規則を求めるフェーズで，適用可能な規則が確定できるまで探索を行なう（ここでは，適用規則がない場合も適用可能な規則の確定に含む）．状態遷移確定フェーズでは，適用可能な規則の確定に従って，各々の状態遷移を実行させる遷移通知信号を送信，受信し次のサービス状態に遷移を行なう．

状態探索フェーズでの状態探索のための信号を状態探索信号といい，端末（または人間）対応プロセス間で行なう状態探索信号と，同一端末（または人間）内対応の論理エンティティ間で行なう状態探索信号の２種類がある．状態確定フェーズでの信号は，端末（または人間）対応プロセス間の遷移通知信号を受信後各端末（または人間）内対応論理エンティティ間の遷移通知信号に分配する．

実際には，論理エンティティ配置における通信経路は物理的なノードの接続関係によって制約を受けるが，論理機能階層では物理的な分散に非依存な階層であるためこの制約は考慮する必要はない．

### A.2.3 プロセス間通信と論理エンティティ間通信

端末対応プロセスのソフトウェア仕様は，サービス状態として存在する端末（または人間）間のグラフ同形判定問題としての解法アルゴリズムを用いている [23]. このアルゴリズムで生成されたプロトコルは，各々の端末（または人間）単位内の現状態探索結果に基づいて適用可能な規則を絞り込む. すなわち，端末（または人間）対応のプロセスの単位ではグラフ同形判定の探索信号（REQUEST 信号）による適用規則の絞り込みを行ない，1つの端末（または人間）の実状態は，機能プロセス間の状態探索信号を用いて判定する. このことから，端末（または人間）対応のプロセス間の信号は，共通であり物理配置定義には依存しない.

図10にコンパイラ内の処理関係を示す. 「変換」では，STR 規則から状態遷移片を導出することを主な処理内容とする. 状態遷移片はグローバル記述である STR からローカルな各端末（または人間）対応のサービスに対応した動作の仕様に変換されたものである.

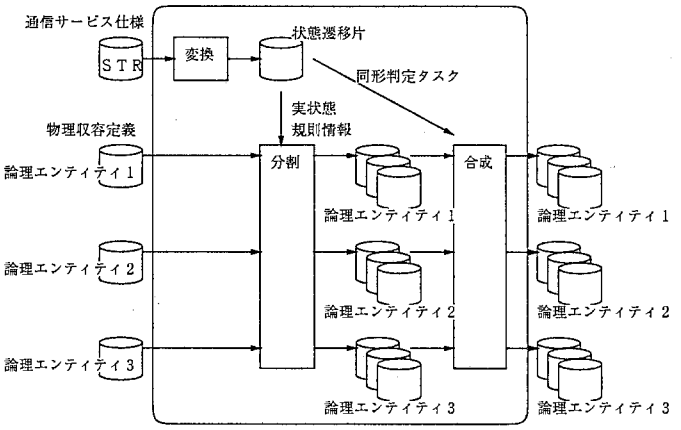


図 10: 論理エンティティ毎ソフトウェア生成処理概念図

### A.2.4 通信経路

同一の端末（または人間）単位内の各論理エンティティ間の通信経路については状態探索の通信経路については，規則の絞り込みを最適に行なう経路（後述，図 11）で通信する. なお，状態遷移通知の通信経路については，端末（または人間）対応プロセス間の通信経路のアルゴリズムと同様に，通信起点論理エンティティ（端末間状態遷移通知信号を受信した論理エンティティ）から端末単位内にブロードキャスト送信（図 12）としている.

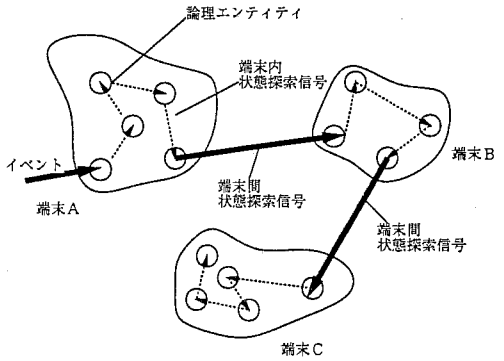


図 11: 状態探索信号（例）

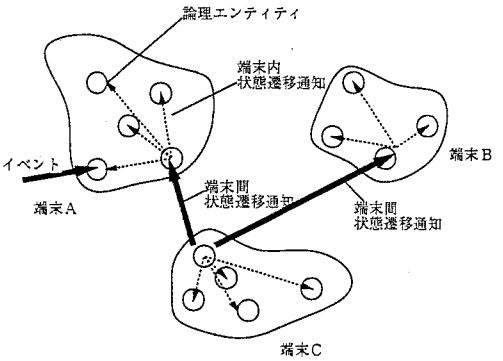


図 12: 状態遷移通知信号（例）

## A.3 論理エンティティ毎動作パターンの導出

### A.3.1 状態遷移片から論理エンティティのSTRへの変換

ソフトウェア仕様に従ってローカルな端末（または人間）対応にある信号またはイベントを受信したときに，現状態，次状態と状態グラフの同型判定，信号の送信を組にしたものを動作パターンと呼ぶ．サービス仕様STRから，規則グラフ，信号確定用重ね合わせグラフから導出される仮の通信経路と信号から，端末（または人間）対応のプロセス毎の動作パターンが求められる．動作パターンは大きくは，探索系と遷移系の2つの状態遷移を示しており受信信号（イベント，端末間状態探索信号，端末間状態遷移信号）に対する状態遷移とタスクに分割できる．探索系は，状態探索のみの動作と，適用規則決定時の探索系から遷移系への切替動作の2種類がある．

- 探索系動作（イベント / 状態探索信号受信）

  状態探索のみ

  - 状態遷移：現状態と次状態が同じ
  - タスク：状態探索信号送信，同形判定条件

  探索系から遷移系への切替

  - 状態遷移：現状態からサービス適用後状態へ遷移
  - タスク：状態遷移信号送信，状態グラフの有向辺の変更，同形判定条件

- 遷移系動作（状態遷移信号受信）

  - 状態遷移：現状態からサービス適用後状態へ遷移
  - タスク：状態グラフの有向辺の変更

以上のことから，求められた動作パターンから状態遷移規則STR（論理エンティティSTR）と，タスク定義（論理エンティティタスク定義）に変換することができる．論理エンティティ収容定義から，各状態記述要素の第1引数は各論理エンティティ名を変数するSTR規則として表現することにより，端末状態のローカルなSTR規則は各論理エンティティのサービス状態遷移を示すSTR（論理エンティティSTR）となる．

### A.3.2 端末内動作パターンの生成

求められた端末状態をグローバル状態とするSTR記述から，端末内の動作パターンを求める．通信経路と論理エンティティ間信号を求めるアルゴリズムは，前に述べた方法で行う．

探索系の論理エンティティSTRから各々の論理エンティティローカルな動作パターンを求める．ただし，この動作パターンでは必ずしも適用可能規則を1つに決めるのが目的ではなく現状態において適用可能な全ての規則候補を求めることである点がネットワークを広域状態とするSTR規則からの展開との大きな違いである．

この適用規則候補は，端末間状態探索信号を送信するタスクに引き継がれる．もし，状態探索の必要な規則が一つもない場合には，状態探索系から遷移系への切替動作を行なう．ただし，全ての未状態探索の規則が適用規則候補にある規則よりも極小な（規則グラフで先頭に近い）場合には探索は終了する（探索を継続しても無意味）．状態遷移系動作は，遷移系の論理エンティティSTRから状態遷移通知を行なう論理エンティティが決定される．状態遷移の際に必要なPID情報は端末間状態遷移通知から引き継がれるものとする．

## B　通信サービス仕様記述の規則の詳細化の例

STR で記述された通信サービス仕様から，本稿でのべた詳細化アルゴリズムで各々の規則が，論理エンティティ毎のソフトウェア仕様に詳細化されることをしめすため，いくつかの規則を例にとり具体的に詳細化の過程を示す.

＜サービス仕様定義 STR 規則＞

```
r1) p1(A,B),p2(B,C),p3(B),p4(A)    ev1(A,B): p1(A,C),p2(A,B),p5(A).
r2) p2(A,B),p3(D),p5(A),p7(A,D)    ev1(A,B): p2(A,D),p4(A),p7(A,B).
r3) p1(A,B),p2(B,E),p2(B,C),p3(B),p4(A) ev1(A,B): p1(A,B),p1(A,C),p2(A,E),p2(B,A),
                                                   p3(A),p3(B).
r4) p4(B),p5(A),p7(A,D),p8(F,E),p9(D,G),p11(A,F) ev1(A,B):
                                                 p1(A,B),p1(B,E),p2(A,D),p7(B,G),p9(A,E).
```

仮通信経路が

```
  → A → F → B → D
  ev1  m1   m2   m3
```

と決定した場合について例示する.
通信経路優先度は entity1,entity2,entity3,entity4 の順になる.
REQUEST,RESPOND 信号の受信は entity1 で行うと仮定する.

＜論理エンティティ収容定義＞

```
entity_def={entity1,entity2,entity3,entity4}
Terminals:A,B

Primitives:
entity1={p4(A),  p5(A),  p3(A)}
entity2={p1(A,B),p7(A,B),p11(A,B)}
entity3={p2(A,B),p9(A,B),p8(A,B)}

Events:
entity1={ev2(A,B)}
entity2={ev1(A,B)}
entity3={ev3(A,B)}

Signals:
entity1={REQUEST,RESPOND}
```

<端末対応内の論理エンティティ STR の導出>

```
Terminals: A,B,C,D,E,F,G
Physical1: entity1
Physical2: entity2
Physical3: entity3
Physical4: entity4

r1-A1) p1(entity2,B),p4(entity1)              ev1(entity2,B):
                    p1(entity2,B),p4(entity1).
r1-A2) p1(entity2,B),p4(entity1)              r1(entity1):
                    p1(entity2,C),p5(entity1),p2(entity3,B).
r2-A1) p7(entity2,D),p5(entity1),p2(entity3,B)  ev1(entity2,B):
                    p7(entity2,D),p5(entity1),p2(entity3,B).
r2-A2) p7(entity2,D),p5(entity1),p2(entity3,B)  r2(entity1):
                    p7(entity2,B),p4(entity1),p2(entity3,D).
r3-A1) p1(entity2,B),p4(entity1)              ev1(entity2,B):
                    p1(entity2,B),p4(entity1).
r3-A2) p1(entity2,B),p4(entity1)              r3(entity1):
                    p1(entity2,B),p1(entity2,C),p3(entity1),p2(entity3,E).
r4-A1) p11(entity2,E),p7(entity2,D),p5(entity1) ev1(entity2,B):
                    p11(entity2,E),p7(entity2,D),p5(entity1).
r4-A2) p11(entity2,E),p7(entity2,D),p5(entity1) r4(entity1):
                    p1(entity2,B),p9(entity3,E),p2(entity3,D).
r1-B2) p3(entity1),p2(entity3,C)              m2(entity1):   empty.
r2-B2) p3(entity1)                          m3(entity1):   empty.
r3-B2) p3(entity1),p2(entity3,C),p2(entity3,E)  m2(entity1):
                    p3(entity1),p7(entity2,A).
r4-F1) p8(entity3,E)                        m1(entity1):   p8(entity3,E).
r4-F2) p8(entity3,E)                        r4(entity1):
                    p1(entity2,B),p9(entity3,E),p2(entity3,D).
r4-B1) p4(entity2)                          m2(entity1):   p4(entity2).
r4-B2) p4(entity2)                          r4(entity1):
                    p1(entity2,E),p7(entity2,G).
r4-D2) p3(entity1),p9(entity3,G)              m3(entity1):   empty.
```

<端末対応内論理エンティティのタスク定義>

```
r1-A1 適用時        send(m2:B)
r1-A2 適用時        change
r1-B2 適用時        send(r1:A);change
r2-A1 適用時        send(m3:D)
r2-A2 適用時        change
```

以下略

## C　禁止宣言の STR 規則化処理の方法

### C.1　禁止宣言の STR 規則への変換

禁止宣言を STR 規則へ変換するアルゴリズムについて説明し，生成する禁止適用規則の生成方法について述べる．禁止宣言を STR 規則に変換することにより，後の段階的詳細化において禁止宣言に対する詳細化を考慮しなくて良いという利点がある．

STR 仕様書に記載されているように，ある規則の適用によって遷移が行なわれると禁止状態に抵触するとき，該イベントが生起した端末のみ状態遷移の付け変えを行なうため，イベント生起端末以外の状態は変化しない．したがって，これらを考慮すると変換アルゴリズムは以下のようになる．

1. 全ての禁止宣言について，禁止状態間での包含関係の極大なものから順に並べ，以下極大なものから極小な禁止状態について STR 規則を生成する．

2. 全ての禁止宣言について，その禁止条件となるための状態記述要素の部分集合を考える．

   ある禁止宣言で，禁止となる状態記述要素の集合を $I_{cond}$ とし，禁止条件が適用された場合のイベント生起プロセスの次状態を $I_{next}$ とするとき，$I_{cond}$ の部分集合のうち空集合でないものを包含関係の極小なものから順に並べたものを $I_i(i = 1..n)$ とする．
   $(I_i \neq \Phi)$．

3. STR 規則に対し，禁止状態 $I_{cond}$ の適用可能性をチェックするため，次状態でに $I_i$ を含む規則を取り出す．

   取り出した STR 規則に対して，次状態で $I_i$ を含み，最も極大な $I_{i,max}$ を求める．

   $P, Q$ を状態記述要素の集合とし，現状態 $P$, イベント $ev$，次状態 $Q$ とするとき，イベント生起端末の状態記述要素を $ev$，それ以外の端末の状態記述要素を $other$ とするサフィックスを付加してあらわす．さらにイベント生起端末の状態記述要素について，規則の適用により変化しない状態記述要素，減少する記述要素，増加する記述要素のサフィックスをそれぞれ $cond, minus, plus$ として表すことにする．元の STR 規則は，

   $$P_{ev,cond}, P_{ev,minus}, P_{other} \quad ev: \quad P_{ev,cond}, Q_{ev,plus}, Q_{other}.$$

   となる．

4. 禁止条件適用時の遷移規則の生成

   元の規則の現状態と次状態に $(I_{cond} - I_{i,max})$ を加え，禁止適用時のイベント生起プロセスの状態遷移を規則に記述する．

   STR の禁止適用時の仕様に従って，イベント生起プロセスの状態は，元の規則における状態記述要素のマイナス差分を除いたものに禁止宣言で定義された次状態を加えたものが次状態となる．したがって，

   $$(I_{cond} - I_{i,max}), P_{ev,cond}, P_{ev,minus}, \ P_{other} \quad ev: \quad (I_{cond} - I_{i,max}), P_{ev,cond}, I_{next}, P_{other}.$$

   となる規則を追加すればよい．

5. 禁止宣言で禁止適用時の次状態 $I_{next}$ が宣言されていない場合は，上記生成規則中 $I_{next} = P_{ev,minus}$ と置き換えて生成する．

6. 禁止宣言で用いられた状態記述要素の変数のユニフィケーションの組合せは全ての組合せを元の規則に当てはめた規則を生成する．

7. 全ての禁止宣言について，同様の処理を行ない STR 規則への変換を行なう．

## C.2 既存規則と生成規則の競合

禁止宣言をSTR規則化したときに新たに発生する問題点を整理する．STR規則化した禁止宣言がもとの禁止動作と等価になるようにするため，この規則化アルゴリズムでは，既存規則の包含関係を参照する必要がある．

もともと禁止宣言は，規則適用があった後にチェックするものであるから，この方式のように適用前にチェックしてしまうと規則の適用関係を乱してしまう恐れがある．これを回避するため以下の場合分けに対して生成規則の追加条件を求める．

**（場合分け１）** 生成される規則が既存規則と包含関係がない場合

たとえば

```
r1) p2,p3  ev:  p5,p6.
r2) p3     ev:  p4.
禁止 {p1,p4}{p6}
```

生成される規則は p4 を部分集合として r2 から

```
r3) p1,p3  ev:  p1,p6.
```

包含関係：（左方向が包含関係の極小方向．以下同様）

```
r2  -+-- r1
     |
     +--( r3 )
```

r1 と r3 の間は包含関係なし．もし実状態で p1,p2,p3 の状態が存在したとき，r1,r3 の両方が適用可能である．本当は，禁止規則 r3 は r2 の適用決定のあとしか判定してはいけないが，非決定として現状どうなるかはわからない．この場合，非決定規則であるが既存の規則のほうが優先度があるように非決定解消規則を生成する．

逆に包含関係がある場合は，その場所によって問題が発生する可能性がある（場合２，３，４）．

解消規則：

```
r3) p1,p2,p3       ev: p1,p5,p6.  (禁止規則と既存規則の和集合→
                                    既存規則適用)
r4) p1,not(p2),p3 ev: p1,p6.      (禁止規則のみ適用可時の規則→
                                    禁止規則適用)
```

包含関係：

```
r2  ---- ( r3 )
```

**（場合分け２）** 既存の規則との包含関係があり，元の規則の次に入る場合

```
r5) p1         ev: p4.
r6) p1,p2,p3   ev: p5,p6,p7.
禁止：{p2,p4}{p9}
```

生成規則は

```
r7) p1,p2      ev: p2,p9.
```

包含関係：

```
r5 ----( r7 )----- r6
```

この場合は，r6 が適用されなければ本来 r4 が適用されて r7 の禁止状態判定が行なわれるので問題はない．r7 の適用は正しく行なわれる．そのまま追加してよい．

**（場合分け３）** 既存の規則との包含関係があり，同一現状態規則がある場合

```
r8) p1          ev: p4.
r9) p1,p5       ev: p5,p6.
禁止 : {p4,p5}{p9}
```

生成規則は

```
r10) p1,p5       ev: p5,p9.
r11) p1,p4,p5    ev: p4,p5,p9. <---- 現状態に既に禁止状態を含むので無効
```

包含関係：

```
r8 ----  r9
       ( r10 )
```

本来，r10 は適用されず，r9 が適用されるはず．r8 が適用可能でかつ r9 が適用不可の場合のみ r10 を適用する．しかしこのような場合には自動的に r9 と同時に r10 も適用されないのでそのまま r8 を適用してよい．このような場合には規則を追加することは行なわない．

**（場合分け４）** 既存の規則との包含関係があり，従来規則が間に入る場合

```
r12 4) p1           ev: p4.
r13 5) p1,p5        ev: p5,p6,p8.
r14 8) p1,p5,p8,p9  ev: p8,p10,p11.
禁止 : {p4,p5,p7}{p9}
```

生成規則は

```
r15 6) p1,p5,p7      ev: p5,p7,p9.
r16 7) p1,p4,p5,p7   ev: p4,p5,p7,p9.<---- 現状態に禁止状態含むので無効
```

包含関係：

```
r12 ----  r13 ------( r15 )----- r14
```

r13 が適用不可ならば r15 は適用されないので禁止にかかることはない．しかし，本来なら，r13 が適用されず r12 が適用可の場合だけ r15 の禁止判定は有効である．r14 が適用されず，r13 が適用される場合には r15 が適用可能であっても適用しないはずだが，現状 r15 が詳細優先で適用されてしまう．r15 が適用可能な時は，r13 も同時に適用可能なはずであるから，r15 の追加は不要である．

## C.3 生成規則追加のための条件

以上のように，禁止宣言と既存規則から禁止に適合する規則を生成したものを実際に追加するとき，以下のような条件で追加するしないを決定する．

1. 生成された規則を追加するとき，生成された規則の現状態中に既に禁止宣言にある状態記述要素の組合せが存在するときその禁止適用ルールは追加しない．

2. 生成された規則を追加するとき，既存規則に生成された規則と同一の現状態を持つ規則が既に存在するとき，その禁止適用ルールは追加しない．（場合分け３）

3. 生成された規則を追加するとき，生成された規則が，既存規則間の包含関係のなかで，生成元となった規則との包含関係の間に，他の規則が存在する場合には，その禁止適用ルールは追加しない．（場合分け４）

4. 生成された規則が既存規則との間の包含関係の中で非決定を発生する場合には，非決定解消規則を生成し追加する．（場合分け１）

   解消規則は，競合する両規則が同時に適用可能な場合（既存規則を適用），既存規則が適用不可で生成した禁止適用規則が適用可能な場合（生成禁止適用規則を適用）について生成する．なお，これにより生成した規則がさらに他の規則との間に競合を発生する場合は同様に解消規則を生成しなければならない．

## C.4 EXAMPLE

### C.4.1 EXAMPLE1: イベント生起端末と禁止宣言端末が同一

規則 $r1) p1(A,B), p4(A,C)\ ev(A) :\ p1(A,B), p2(A,C).$
★禁止宣言
$p1(A,B), p2(A,C), p3(A,D) \rightarrow p5(\$, B)$
★ $I_{cond}$ の部分集合は,
$I_0 = p1(A,B), I_1 = p2(A,C), I_2 = p3(A,D),$
$I_3 = p1(A,B), p2(A,C), I_4 = P1(A,B), p3(A,D), I_5 = p2(A,C), p3(A,D)$
$I_6 = p1(A,B), p2(A,C), p3(A,D)$
$I_{next} = p5(A,B)$
★規則 r1 において, $I_{i,max} = p1(A,B), p2(A,C)$
r1 より,
$P_{ev,cond} = p1(A,B), P_{ev,minus} = p4(A,C), Q_{ev,plus} = p2(A,C), P_{other} = \phi$
$I_{cond} - I_{i,max} = p3(A,D)$
★したがって生成する規則は,
$p3(A,D), p1(A,B), p4(A,C) ev(A) : p3(A,D), p1(A,B), p5(A,B).$
となる.

### C.4.2 EXAMPLE2: イベント生起端末と禁止宣言端末が異なる

規則 $r2) p1(A,B), p4(A,C), p6(B,A) ev(B,A) : p1(A,B), p2(A,C), p7(B.C).$
★禁止宣言
$p1(A,B), p2(A,C), p3(A,D) \rightarrow p5(\$, C)$
★ $I_{cond}$ の部分集合は,
$I_0 = p1(A,B), I_1 = p2(A,C), I_2 = p3(A,D),$
$I_3 = p1(A,B), p2(A,C), I_4 = P1(A,B), p3(A,D), I_5 = p2(A,C), p3(A,D)$
$I_6 = p1(A,B), p2(A,C), p3(A,D)$
$I_{next} = p5(A,B)$
★規則 r2 において, $I_{i,max} = p1(A,B), p2(A,C)$
r2 より,
$P_{ev,cond} = \phi, P_{ev,minus} = p6(B,A), Q_{ev,plus} = p7(B,C), P_{other} = p1(A,B), p4(A,C)$
$I_{cond} - I_{i,max} = p3(A,D)$
★したがって生成する規則は,
$p3(A,D), p6(B,A), p1(A,B), p4(A,C) ev(B,A) : p3(A,D), p5(B,C), p1(A,B), p4(A,C).$
となる.

### C.4.3 EXAMPLE3: 禁止宣言で次状態記述のない場合

規則 $r3) p1(A,B), p4(A,C), p6(B,A) ev(B,A) : p1(A,B), p2(A,C), p7(B.C).$
★禁止宣言
$p1(A,B), p2(A,C), p3(A,D) \rightarrow \phi$
★ $I_{cond}$ の部分集合は,
$I_0 = p1(A,B), I_1 = p2(A,C), I_2 = p3(A,D),$
$I_3 = p1(A,B), p2(A,C), I_4 = P1(A,B), p3(A,D), I_5 = p2(A,C), p3(A,D)$
$I_6 = p1(A,B), p2(A,C), p3(A,D)$
$I_{next} = I_{ev,minus}$
★規則 r3 において, $I_{i,max} = p1(A,B), p2(A,C)$
r3 より,

$P_{ev,cond} = \phi, P_{ev,minus} = p6(B,A), Q_{ev,plus} = p7(B,C), P_{other} = p1(A,B), p4(A,C)$
$I_{cond} - I_{i,max} = p3(A,D), I_{next} = p6(B,A)$
★したがって生成する規則は,
$p3(A,D), p6(B,A), p1(A,B), p4(A,C) ev(B,A) : p3(A,D), p6(B,A), p1(A,B), p4(A,C).$
となる.