〔公　開〕

ＴＲ－Ｃ－０１２８

通信サービス仕様に関する要求理解とソフトウェアへの変換に関する研究

田倉　昭
Akira TAKURA

１９９６　２．２９

ＡＴＲ通信システム研究所

目次

第 I 部
はじめに

1 概要

ソフトウェア自動作成は信頼性のあるソフトウェアを効率的に開発する有望な方法である．本研究は，通信サービスを対象として，形式的に記述された要求仕様から通信ソフトウェアを自動作成することを目標としている．通信ソフトウェアは複雑で大規模なソフトウェアの一つである．通信サービスを提供する通信システムに対しては，サービスの追加が頻繁に行われる．このため，通信ソフトウェア開発の効率化が切望されている．また，通信サービスは今まではすべてのユーザに等しく提供されてきた．今後は，個々のユーザ毎に個別のサービスを提供する時代に移りつつある．このため，ユーザ自身が自分でサービスを開発することが可能となる通信ネットワークのオープン化が進められつつある．このような状況において，通信ネットワークや通信ソフトウェアの詳細を知らないユーザでも，新たな通信サービスが開発できるような技術が必要である．本論分では，形式的に記述されたサービス仕様からソフトウェアの自動作成を可能にすることで効率的な通信ソフトウェア開発を実現することを目指す．本論文で述べるソフトウェア自動作成は以下の特徴を有する．

- 通信サービスにおけるユーザが把握可能な仕様を記述するだけでソフトウェアを作成することができる．
- 既存ソフトウェアの詳細を知らなくても新サービスの追加が可能である．
- 誤りのない信頼性のあるソフトウェアを作成する．

上記目的を達成するため，ユーザの記述した通信サービスに対する初期要求をもとに，通信サービスとしての条件を満足する仕様を導出するための要求理解手法の確立および導出された通信サービス仕様を満足するソフトウェアを自動作成する手法を述べる．本研究では，通信サービスに直接関係するソフトウェアを自動作成の対象とする．すなわち，サービス追加に際して新たに開発する必要があるソフトウェアを自動作成の対象とする．オペレーティングシステムや保守運用などのサービスの追加にともなって機能追加が行われない部分は対象外とする．

サービスの設計者は，通信サービスに対する要求仕様を仕様記述言語 STR [1] を用いて記述する．この初期要求は，不完全で矛盾を含んでいることがある．この要求仕様をもとにユーザとのインタラクションを介して通信サービスとしての条件を満足する仕様に変換する．これを要求理解と呼ぶ．

通信サービスは既存サービスに新たなサービスを追加する開発方式をとる場合がほとんどである．従って，既存サービスとの競合を仕様記述段階で検出し解消することが重要な課題である．STR で記述された仕様については，競合の検証及び解消方式が既に提案されている．この仕様検証については，本研究の対象外である．

獲得された“完全な”サービス仕様からプロトコルを合成する．プロトコルアーキテクチャとして OSI で代表される階層モデルを用いる．通信サービスは交換機や電話などの機器を通してユーザに提供される．ところが，合成されたプロトコルはプロトコルエンティティ間の通信を規定するだけであって，通信システムを制御する仕様を含んでいない．そこで，予め用意された通信システム制御知識を用いてソフトウェア仕様へ変換する．このソフトウェア仕様は階層モデルでアーキテクチャが規定される通信システムにインストールすることにより，通信ソフトウェア仕様として使うことができる．

通信サービスを実現するアーキテクチャとして機能モデルが標準化されている．これは機能ごとに一つのプロトコルエンティティが対応する分散システムのアーキテクチャである．このアーキテクチャに整合するソフトウェア仕様を段階的詳細化により求めることができる．

以上の項目についての自動化方式を述べる．本論文で自動化の対象としている範囲を一貫して自動化の対象としている従来の研究はない．すなわち，通信ソフトウェアの詳細知識を必要とせずに記述可能な通信サービス仕様からプロトコル合成を経由して，通信ソフトウェアを自動作成し，それを用いて実際に通信サービスを動作させたはじめての研究である．従来は，通信システムをホワイトボックスとしてとらえ，その内部動作に着目して仕様記述を行っている．また，通信システムをブラックボックスとして通信サービス仕様を記述し，その仕様からプロトコルを合成する研究では，最終的に通信システムを制御する通信ソフトウェアへの変換までは行っていない．本研究は，通信サービスを対象として不完全な要求仕様から出発して，通信サービスとして意味のある仕様を求め，最終的に通信ソフトウェアへの自動変換を行っている．

本研究で提案する手法は図 1に示す通信ソフトウェア自動作成システム [2] の一部として実現されている．このシステムは次の特徴を有する．

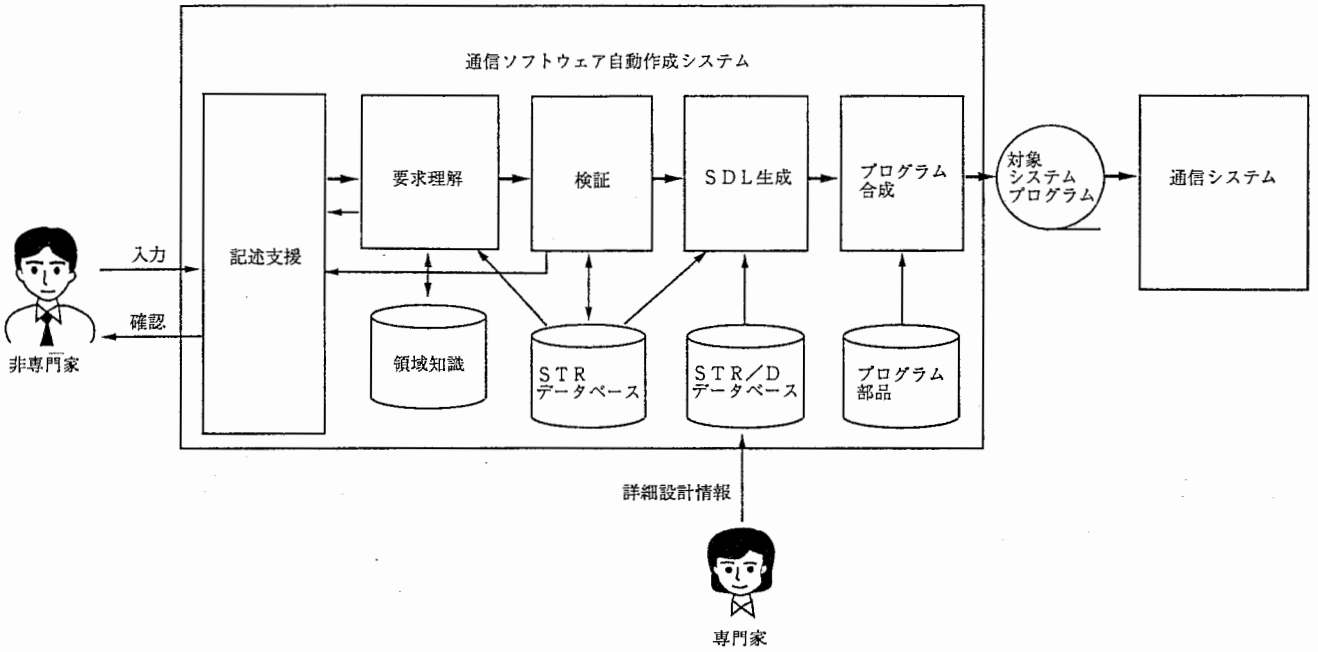

図 1: 通信ソフトウェア自動作成システム

1. 通信システム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通信サービス仕様が記述する.
2. 自然言語や図形を用いた仕様の記述支援機能を有する.
3. 個々の規則に含まれる誤り，不足する規則の検出と補完する機能を有する.
4. 仕様の確認および競合検証機能を有する.
5. サービス仕様を実現するプロトコルを合成する.
6. 詳細仕様を知識として記述する.
7. 合成されたプロトコルから詳細仕様を用いてソフトウェア仕様へ変換することができる.

これらの特徴を有する通信ソフトウェア自動作成により期待できる効果を述べる.

1. 不完全なサービス仕様から出発して，通信ソフトウェアを作成することが可能である．従って，ユーザのもつ曖昧で断片的な要求仕様から通信サービス仕様を作成することが可能である.
2. ユーザの意図通りにソフトウェアが作成可能である．すなわち，ユーザの意図と作成されたソフトウェアの動作が一致する.
3. 仕様誤りの検出と解消が可能である.
4. 非専門家による通信サービス仕様記述が可能である．非専門家とは，通信ソフトウェアの構造，プロトコル，通信システムの制御法に関する知識を持たない人と定義する.
5. ユーザの要求仕様から通信ソフトウェアを自動作成するのに専門家が関与する割合は小さい.
6. 仕様の記述量が作成されるソフトウェアと比較して少ない．この結果，サービス開発時間が短いと期待できる.

性質 1は第 II部でその方式と限界を述べる．性質 2は，ユーザが記述した仕様を確認した後，その仕様を満足する通信ソフトウェアを自動作成するので自然に達成される．性質 3は II章で述べる個別サービスに関する検証，及び複数の通信サービス仕様間で発生する可能性のある競合の検出と解消により実現される．但し，競合検証は本研究の対

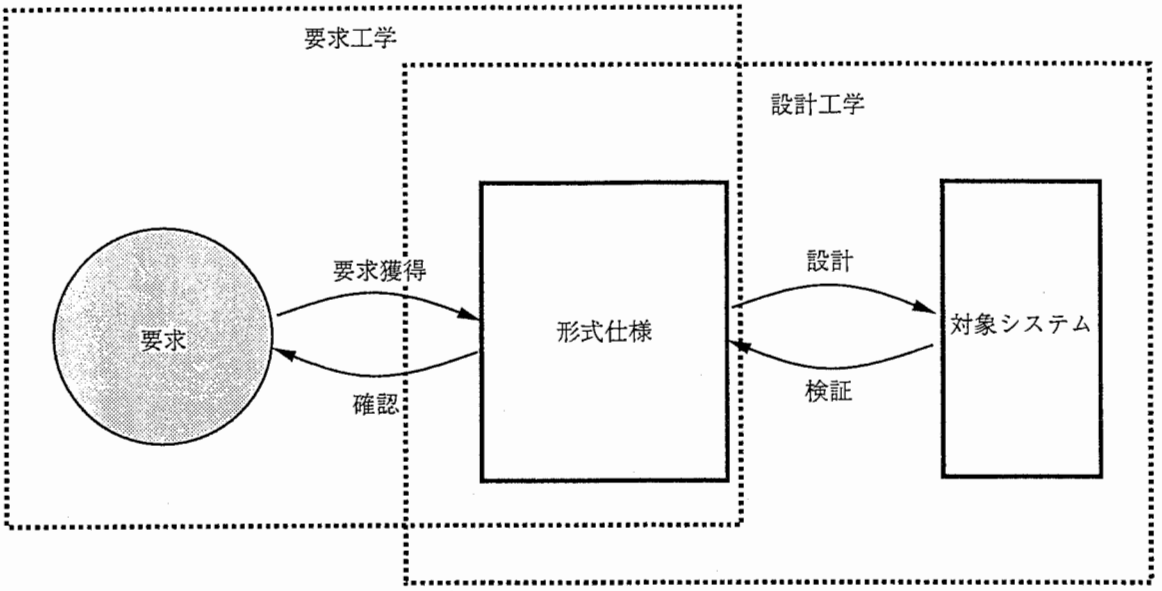

図 2: 要求工学と設計工学

象範囲外である．性質 4, 5, 6については10章で実験結果によって，本論文で述べる通信ソフトウェア自動作成が有する特徴であることを示す．

以上の項目についての自動化方式を本論文で提案する．通信ソフトウェアの自動作成のためには，通信サービスの形式的な仕様記述が前提となる．本論文では，仕様記述言語としてSTRを用いる．3章で仕様記述言語としてSTRを用いる理由，及びソフトウェア自動作成を行う上で仮定するソフトウェアアーキテクチャについて述べる．

2 関連研究

2.1 要求理解

ソフトウェア開発において，上流工程のエラーほどその修正にかかるコストは大きい [3]．それにもかかわらず，現状ではソフトウェア開発における上流工程である要求理解に関する自動化が最も遅れている．ユーザは，はじめから明確な要求をもっているわけではなく，要求には曖昧性，不完全性，矛盾が含まれるのは避けられない [4]．通信ソフトウェアや通信ネットワークの詳細を知らないユーザによる通信サービス定義を実現するためには，このような不完全な要求から通信サービスとしての性質を満足する仕様を導出する機能の自動化が必須である．

従来，多くの研究が行われている仕様記述の研究では，ユーザが要求を計算機処理可能な形式言語で明確に定義することを前提としている．すなわち，形式言語に関する研究は，形式言語で記述されたユーザ要求から対象システムのソフトウェアを設計する過程を計算機支援により自動化することが課題である．この自動化項目の中には，対象システムから来る制約を要求が満足しているかどうかを検証 (verification) することも含まれる．形式言語を出発点とする自動化技術に関する研究は設計工学と呼ばれる [5]．これに対して，そもそもユーザから要求を獲得することが困難であるという認識に立ち，要求仕様を獲得する部分を要求獲得と呼ぶ．要求獲得のためには，要求獲得と獲得した要求がユーザの望む仕様であるかどうかの確認 (validation) が必要である．この要求獲得と確認に関する研究を合わせて，要求工学と呼ぶ．要求工学と設計工学は，一般に図2で示される関係にある [5, 6]．これらは最終生産物である仕様に関して，次のような特徴付けが可能である．要求仕様は対象となるソフトウェアシステムとそれを取り巻く環境との関係に着目した仕様であるのに対して，設計で得られるシステム仕様は機能や構造に着目した仕様である．

Requirements Apprentice [4] や仕様記述言語 FRORL [7] に基づくソフトウェア開発環境などの要求理解システムにおいては，要求を出すユーザと要求を形式言語で記述する要求分析者を分離することを仮定している．この結果，これらのシステムでは要求は形式言語あるいは仕様記述言語とプログラミング言語の中間に位置するかなり限定的なコマンド言語で入力される．本研究では，通信サービスのユーザすなわち通信システムの非専門家がサービスを記述できるようにすることを目指している．そこで，形式言語を用いてユーザが直接要求を記述することを前提とする要求理解を対象とする．

要求仕様の記述に非形式言語を用いる場合には，要求を出す人は要求仕様に基づき作成されるシステムのユーザ

であるので，ユーザに要求理解システムが提供する形式言語を覚えさせることには無理があるという立場に立っている．

要求理解における領域知識の重要性が認識されている．領域知識を要求理解に使う場合，どういう知識をどう表現し，どう使うかという問題と新しい知識をどう獲得するかという問題がある．領域知識には，対象となるソフトウェアシステムの環境に関する一般概念や過去に作成した仕様に関する知識や分析，設計の方法に関する知識が考えられる．本論文では，ユーザの要求の欠落部分を補うために領域知識を用いる．従って，領域知識として通信サービスに関する知識を必要とする．また，領域知識は通信サービスをよく知っている専門家があらかじめ用意することを前提とする．

形式言語で記述された要求仕様は，曖昧性や誤りを含んだり，あるいは仕様に不足が生じることがある．このような不完全な要求仕様から通信サービスとして必要な性質を満たす仕様を導出する過程を本研究の対象とする．不完全な要求から通信サービスとして意味のある仕様を導出するためには，誤りや仕様が断片的であることを検出し，望ましい仕様にするための仕様の変更や補完が必要である．このような支援を行う上で重要な役割を果たすのが，領域知識である．領域知識には，対象となるソフトウェアシステムの環境に関する一般概念や過去に作成した仕様に関する知識や分析，設計の方法に関する知識がある．

通常使われている領域知識には，特定の対象領域において共通的に存在する構造あるいは性質に関する知識と仕様の決定順序に関する設計知識がある．領域知識のうち対象システムに関する知識は，満たさなければならない性質とあってはならない制約の二種類に分類することができる．ASPIS[8] では，分析，設計方法に関する知識と対象領域に関する知識を使っている．前者の知識は，設計者が設計途中で何を行なわなければいけないのか，あるいはこれから何を行なわなければいけないのかが分かるようにするための知識である．後者の知識は，分析結果が対象領域に関する一般概念に反しないかの検証や同意語の認識に使われる．

WATSON[9] は有限状態システムを適用領域とするシステムであり，通信サービスに対する要求理解を適用例の一つとして取り上げている．WATSON における領域知識には，電話機，ネットワークプロトコル，利用者エチケット，例外処理，タイムアウト，典型的な制御のひな型がある．これらの知識を時制論理を用いて表現する．WATSON では，要求の入力を自然言語 (英語) で行なう．要求は使用例で記述され，その要求を一般化し，時制論理の式として表現する．WATSON におけるもう一つの要求入力手段として，グラフィックアニメーションツールを用いた入力方法がある．時制論理で表現された要求は論理式に関する推論機構を用いることにより状態遷移規則に変換し，最終的に有限状態遷移機械が得られる．

領域知識の獲得法に関して，BLIP[10] では，sloppy modelling (おおざっぱなモデル化) と呼ばれるユーザとシステムの協調作業で知識獲得を行う方法を提案している．この結果，領域知識を予め完全で構造化された形でシステムに入れておく必要がなくなる．ユーザは，要求を変更することがしばしばある．ARIES[11] では，要求理解の途中で各種の表現方法を用いて要求をユーザに提示するとともに，要求の変更に対して，システムがそれまでに獲得した知識を矛盾なく更新する支援を行っている．

本論文では，ユーザの要求に不足している仕様を補うために領域知識を使うため，通信サービスの抽象モデルを領域知識として扱う．この結果，ユーザが記述した要求仕様に欠落している部分を補う仕様を提案することが可能となる．

2.2 プロトコル合成

通信サービスは複数のユーザ間で行われる情報交換機能を提供する．この情報交換を円滑に行うための通信主体 (エンティティ) 間の相互作用に関する規則をプロトコルと呼ぶ．通信ソフトウェアにおいて，プロトコルは通信サービス制御の流れの骨格を形づくる．そこで，ソフトウェア自動作成の一段階としてサービス仕様からのプロトコル合成を行う．

要求理解の段階で獲得された通信サービスとしての条件を満足する仕様からプロトコルを合成し，更に通信サービスを制御するソフトウェアを自動作成する．通信ソフトウェアを対象とした仕様記述言語に基づくソフトウェア自動作成に関する関連研究について述べる．

プロトコル合成を行うには予めプロトコルアーキテクチャを決めておかなければならない．プロトコルに関するアーキテクチャには，OSI 参照モデルや No.7 信号方式で用いられているアーキテクチャなどがある．いずれのアーキテクチャにおいても，通信機能を階層分けしている．各階層は分散配置されたプロトコルエンティティから構成されている．図 3に示すように，レイヤ (N) はレイヤ (N-1) から提供される (N-1) サービスを利用し，通信相手のレイヤ (N) と相互通信を行って，レイヤ (N+1) に提供する (N) サービスを実現する．レイヤ (N) 内のエンティティ間で通信を行うための通信規約を (N) プロトコルと呼ぶ．レイヤ (N + 1) がレイヤ (N) サービスにアクセスする点を (N) サービスアクセス点 (SAP： Service Access Point) と呼ぶ．(N+1) エンティティは，(N) サービスのユーザと

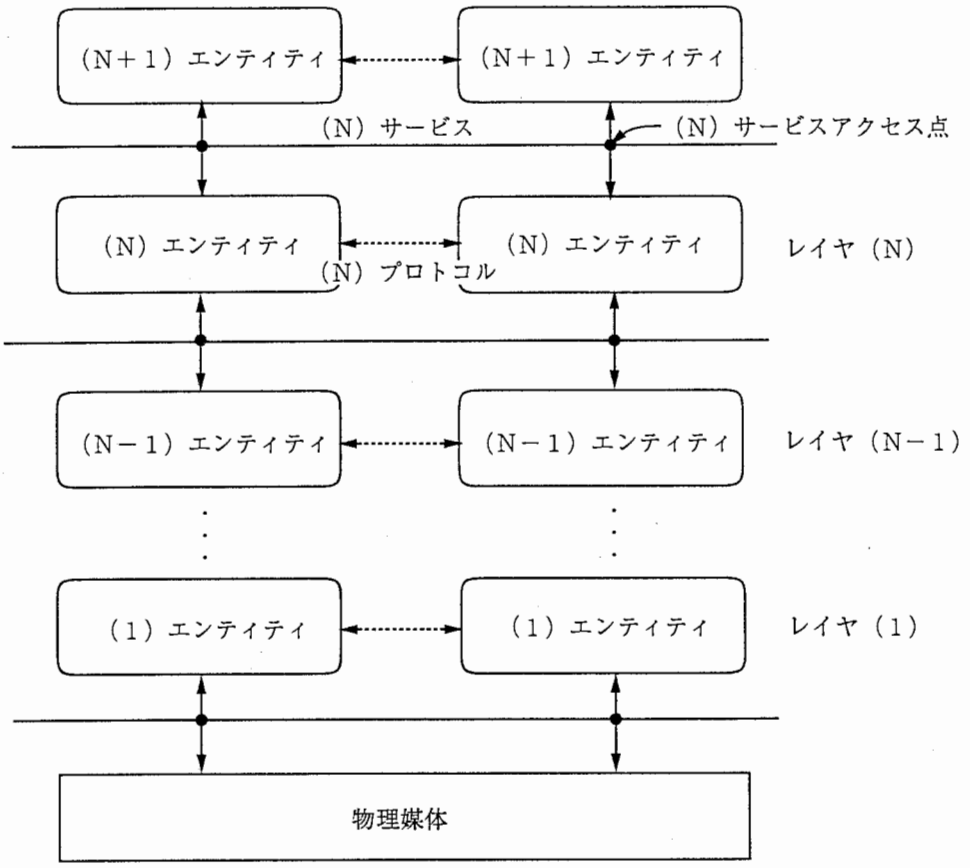

図 3: 階層型プロトコルアーキテクチャ

なる．

前記の要求理解手法により獲得された通信サービス仕様から階層型アーキテクチャに適合するプロトコルを合成する手法を示す．プロトコル合成は，誤りのない信頼性のある通信ソフトウェアを開発する上での重要な要素である．プロトコルの合成は，プロトコルエンティティの部分的な仕様から出発する方式とサービス仕様から出発する二つの型に大きく分類することができる [12, 13]．前者の方式は，n 個のプロトコルエンティティの仕様を求める際に，n-1 個のプロトコルエンティティの仕様から残りの一つのプロトコルエンティティの仕様を求める方式 [14]，プロトコルエラーを含む不完全なプロトコル仕様からプロトコルエラーのないプロトコル仕様を合成する方式 [15, 16, 17, 18, 19]，複数の部分的なプロトコル仕様からエラーのないプロトコル仕様を合成する方式 [20, 21] に分類することが可能である．後者の方式は，サービス仕様の記述方式に関して LOTOS-based [22, 23] と FSM-based [24, 25, 26] の二種類がある．

本研究で述べるプロトコル合成方式は，FSM-based なサービス仕様の記述方式を用いている．従来のプロトコル合成方式と異なる点を述べる．従来方式では，サービスアクセス点におけるイベントの実行順序を指定したサービス仕様記述を行っている．これに対して，本論文で述べる方式では，規則形式の FSM-based な仕様記述を用いている．この記述方式では，サービスアクセス点を変数を用いて記述するとともに，通信システム内で規則の適用条件を満足する部分状態があれば，その部分状態に対して規則が適用される．このため，適用条件に出現する端末数が有限ならば，サービスに参加する端末数は有界である必要はない．従って，不特定多数の参加が可能な会議サービスのようなサービスに対する仕様を記述することが可能である．

通信サービスは複数の端末の複合動作として規定されるので，サービス仕様は複数端末のグローバルな状態遷移記述となる．通信システムは地理的に分散したシステムであるので，与えられたサービスに関するグローバルな状態遷移記述から各端末毎のローカルな状態に関する情報に基づいて必要な通信を行う．合成されるプロトコルは，ユーザの端末操作に対応するイベントが生起したときにどの規則を適用すべきかを分散システムである通信システム上で通信を行うことにより決定する．一つの規則の適用条件はイベント生起点からすべての頂点へ至る経路が存在する連結な有向グラフとして表現される．従って，通信システム全体の状態を表すグラフにどの規則の適用条件に対応する

グラフが含まれるかを探索することにより，適用規則を確定することが可能である．この探索のための通信の方法として，逐次的に通信を行うプロトコルを合成するアルゴリズムを示す．

一つのSTR規則はラベル付き有向グラフの書き換え規則とみることができる．同様に，通信システム全体の状態もラベル付き有向グラフとして表現することができる．このとき，合成されるプロトコルは通信システムの状態を表すグラフの中から書き換え対象となるグラフを探索するグラフの同型判定問題を解く分散アルゴリズムを実現する．グラフの同型判定問題はNP完全問題[27]である．通常の通信サービスにおいては，同型グラフの探索範囲となる通信システム内の連結な部分グラフはそれほど大きくはない．このため，合成されるプロトコルは実用上は問題のない範囲で動作する[28]．

2.3 プロトコル仕様からソフトウェア仕様への変換

上記のプロトコル合成法で合成されるプロトコルは，図3に示す階層型アーキテクチャに適合する．PBXのように機能分散に関する制約がない場合には，階層型アーキテクチャをプロトコルアーキテクチャとするソフトウェア自動作成が有効である．階層型アーキテクチャをプロトコルアーキテクチャとして仮定した場合にサービス仕様から作成されるソフトウェアを用いてPBX上で実際に代表的な通信サービスを実現することにより，本研究で示したソフトウェア自動作成法の有効性を示す．

プロトコル仕様では，プロトコルエンティティ間の通信を規定するのみであり，プロトコル仕様だけでは実際の通信システムを制御する仕様すべてを含まない．通信システム上で動作可能な通信ソフトウェアを実現するために，プロトコル仕様の詳細化が必要となる．従来，この詳細化は手動で行われていた[29, 7, 30]．この詳細化に当たっては通信システムや通信ソフトウェアの詳細知識が必要となる．本研究では，このような詳細知識を使わずに通信ソフトウェアを自動作成することが目標である．そこで，詳細化に関する仕様を予め知識として記述する手法を示す．このため，通信システムの制御仕様を知識として記述可能な詳細仕様記述言語を定義する．予め用意された知識を使うだけでは詳細化ができない場合には，詳細知識を有する専門家の介在が必要となる．以上のプログラム作成手法を用いて実際のPBXの通信ソフトウェアを開発し，代表的な通信サービスを実現することによりその有効性を示す．

自動作成ソフトウェアを用いて2台の異なるPBX上で代表的な通信サービスを実現した結果について述べる[28, 31]．サービス仕様から4章で示したアルゴリズムを用いてプロトコルを作成し，更に6章で述べた詳細化によりソフトウェア仕様を作成した．PBXを初めとする通信システムは，ベンダ毎あるいは機種毎に制御インタフェースは異なる．異なるインタフェースを持つ通信システム毎にソフトウェアを自動作成する代わりに，通信システムを制御する論理インタフェースを設定し，その論理インタフェースに合わせてソフトウェアの自動作成を行った．ソフトウェアの自動作成を目的としたものではないが，異機種通信システム間で共通のアプリケーションプログラムを動作させることを目的として，CTRON(Central and Communication The Realtime Operating System Nucleus)が提案されている．このようなベンダ間で共通するインタフェースが設定されていれば，そのインタフェースに合わせてソフトウェアを自動作成すればよい．本稿では，STRで用いられている状態記述要素の通信システム上での意味を規定できるインタフェースを設定し，通信サービスを動作させた．このようなインタフェースの設定を可能とするソフトウェア構成及び設定したインタフェースを述べるとともに，通信サービスの動作結果を示す．

3 準備

3.1 仕様記述言語

要求理解およびソフトウェア自動作成を行う上で，仕様を表現するための仕様記述言語が必要である．上に述べた目的を達成する上で考慮すべき仕様記述言語に対する要求条件を明らかにする．

- インプリメンテーション非依存な仕様記述

  通信サービスに対する要求仕様は，インプリメンテーションに依存する仕様である対象システムの機能やソフトウェア構造に関する仕様を含むべきではない．ユーザが通信サービスを使うときに，通信システムとそれを取り巻く環境の中で行われる動作が通信サービスに対する仕様となる．

- 既存仕様の詳細知識を必要としない新規サービス追加

  通信サービスは，既存サービスに新たなサービスが追加される形で開発されるのが通例である．既存サービス仕様を知ることなく新サービスの仕様を記述し，追加できるのが望ましい．規則形式の仕様記述を用いると新たなサービスに対する仕様のみの記述で新サービスが追加できる可能性がある．

- 競合の自動検出

  仕様には矛盾が含まれたり既存仕様との間で競合が生じたりすることがある．これらの望ましくない現象を検出することができる．

- 仕様からプログラムへの段階的詳細化

  通信サービス仕様から段階的詳細化によりソフトウェアに変換することができる．

通信ソフトウェアやプロトコルの仕様を記述するために，SDL [32] やLOTOS [33] をはじめとする多くの仕様記述言語が提案されている．SDLは通信システムのソフトウェアやプロトコルの開発において最も広く使われている [34]．LOTOSもまた勢力的に多くの研究が行われているとともに，通信サービス仕様の形式的な記述言語として使われている [35, 36]．これらの言語では抽象化のレベルによりインプリメンテーション非依存な仕様記述が可能である．SDLやLOTOSでは，イベントの実行順序やプロセス間の同期といった概念が入っている．このため，仕様の合成に当たり既存の仕様との整合が必要である．これに対して，規則を用いて仕様を記述すると，仕様の合成は単にlogical conjunctionで行うことが可能である．すなわち，既存仕様に手を加えることなく，新たな仕様を追加するだけで新たなサービスの仕様追加が可能である．従って，規則形式の仕様記述言語を用いると，既存仕様の詳細知識が不要な新規サービス追加が可能となる．

規則に基づく仕様記述言語として，FRORL, L.0, STRがある．本論文では，仕様記述言語としてSTRを用いる．STRを用いると本論文で提案するソフトウェアへの自動変換手法と既に提案されている検証技術を組み合わせることにより，上記の仕様記述言語に対する要求条件を満足することが可能である．FRORL，L.0では仕様を段階的に詳細化することによりソフトウェアを得ている．通信ソフトウェアにおいて，プロトコルはソフトウェアにおける制御の流れを決定する基本的な役割を果たす．このプロトコルをFRORLやL.0では合成することを行っていない．本論文では，STRで記述されたサービス仕様からプロトコルを合成することにより [37, 38]，ソフトウェアの自動作成を行う．この点がSTRとFRORL，L.0との違いである．

仕様記述言語を通信ソフトウェア開発に適用する場合，仕様からソフトウェアを得るために段階的な詳細化が必要である [29, 7, 30]．例えば，Cameron他 [29] は，規則ベースの仕様記述言語L.0を用いて実際のプロトコルを実現している．Tsai他 [7] は，フレームと規則ベースの要求記述言語FRORLを用いている．これらの方法では，仕様を段階的に詳細化してソフトウェアを導出している．この過程で，プログラム内部の動作を理解する必要がある．サービス仕様からソフトウェアを自動的に作成することができれば，段階的詳細化を行うことなく，誤りのないソフトウェアを自動作成することが可能である．

これらの要求条件を満足する仕様記述言語の一つとしてSTR [1] がある．STRでは，通信システムの外部から認識可能な端末の動作あるいは状態変化を規則で表すことによりサービス仕様を記述する．これは対象システムやインプリメント法に依存する詳細仕様を使わずにサービス仕様が可能であることを意味する．一つの規則は有向グラフの書き換え規則と見ることが可能である．本論文では，サービス仕様の記述言語としてSTRを採用するが，本論文で得られる結果はSTRに固有ではなく，規則を用いた仕様記述あるいは有向グラフの書き換え規則として仕様が表現可能な言語一般に対して有効である．

サービス仕様は通信サービスが満足しなければならない要求仕様のみを記述すべきであり，それ以外の仕様を記述すべきではない．通信サービスでは，端末の状態変化に関する仕様がこれに当たる．STRでは，ユーザが端末を操作したときに，その操作端末を含めた関連端末の状態変化を一つの規則として記述する．通信サービス仕様は，このような規則の集合として記述する．

サービスはSTR規則の集合として定義する．一つのSTR規則は現状態，イベント，次状態の3要素からなり，次の形式をしている．

現状態　イベント ：　次状態.

現状態と次状態は複数端末のグローバル状態を表す．グローバル状態はローカル状態の集合として表現される．一つのローカル状態はある一つの端末の状態を表す状態記述要素の集合として定義される．一つの状態記述要素は，端末を表す一つあるいは二つの変数を引数としてもつ．第二引数が記述されている場合には，第一引数で指定される端末は第二引数で指定される端末との間でその状態記述要素で規定される状態にあることを表す．従って，一つの端末のローカル状態は，その端末を第一引数で指定している状態記述要素の集合として定義される．一つの状態記述要素は通信システムの外部から認識可能な端末の状態を表す．

イベントも一つあるいは二つの端末変数をもつ．イベントは第一引数で指定される端末への論理的な入力を表す．イベントに第二引数が記述されている場合には，そのイベントによって入力された端末識別子を表す．

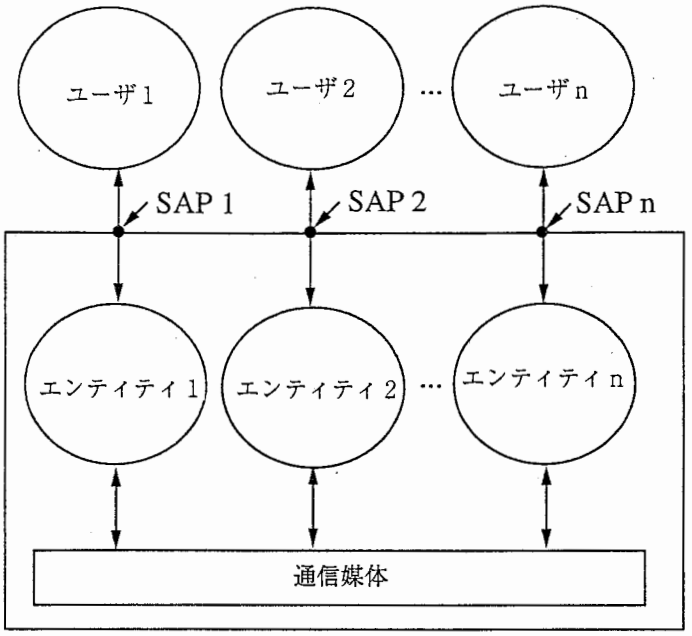

図 4: 階層型プロトコルのアーキテクチャモデル

一つの規則が端末の集合 $t_1, \cdots, t_n$ に適用される条件は，その規則のイベントがこれらの端末の一つで起こり，更にこれらの端末が規則の現状態で指定される状態記述要素をもつ場合である．二つの規則 r1, r2 が，この規則適用条件を満足し，これらの規則の現状態で指定される状態記述要素の二つの集合の間に包含関係があるとき，より大きな集合の規則が適用される．すなわち，より詳細に適用条件が規定されている規則を優先して適用する．このようにしても，複数の規則が適用される可能性が残っている．その場合には，任意の一つの規則が適用される．ある端末の集合と適用すべき規則 r が決定したとき，r の始状態で規定されている状態記述要素を r の次状態で規定されている状態記述要素で置き換えることを r の適用と定義する．

3.2 ネットワークアーキテクチャ

通信サービスを通信システム上で実現するためのアーキテクチャには，OSI 参照モデルに代表される階層化アーキテクチャと通信網上での機能の分散を表す機能分散モデルがある．それぞれの定義とソフトウェア自動作成における特徴を述べる．通信ソフトウェアに対する階層化モデルを図 4に示す．

階層化モデルでは，ユーザはサービスアクセスポイントを通してサービスの提供を受ける．一つの SAP に対して一つのプロトコルエンティティが対応している．ユーザからの要求は SAP を通してプロトコルエンティティが受け取り，必要に応じて他のプロトコルエンティティと通信を行うことにより，ユーザに通信サービスを提供する．通信システムでは，電話をはじめとする端末が SAP に対応する．プロトコルエンティティ間の通信は，信頼性のある通信媒体によりエラーのない通信が保証される．このモデルでは，サービス仕様とプロトコル仕様は次のように定義される [39]:

- サービス仕様は，下位のプロトコルレイヤのプロトコルエンティティが上位のプロトコルレイヤのユーザに提供するサービスを規定する．下位レイヤによって提供されるサービスは，サービスアクセス点での動作を規定するサービスプリミティブの集合を用いて定義される．

- プロトコル仕様は，下位プロトコルレイヤ間のインタラクションを規定する．このインタラクションは上位プロトコルレイヤに提供されるサービスと下位の通信媒体から提供されるサービスによって定義される．

IN サービスは図 5に示す IN CS-1 機能モデルにより適用される．階層化モデルでは，機能が階層化により抽象化されている．これに対して，機能分散モデルでは，機能毎に機能エンティティが割り付けられ，ネットワーク内に分散配置されたアーキテクチャとなっている．図 5において，端末は CCAF に接続されているが，一般の電話サービスから UPT サービスに制御が移った時点で，SCF が制御を担当する．更に，端末からの入力の一部は SRF が直接受

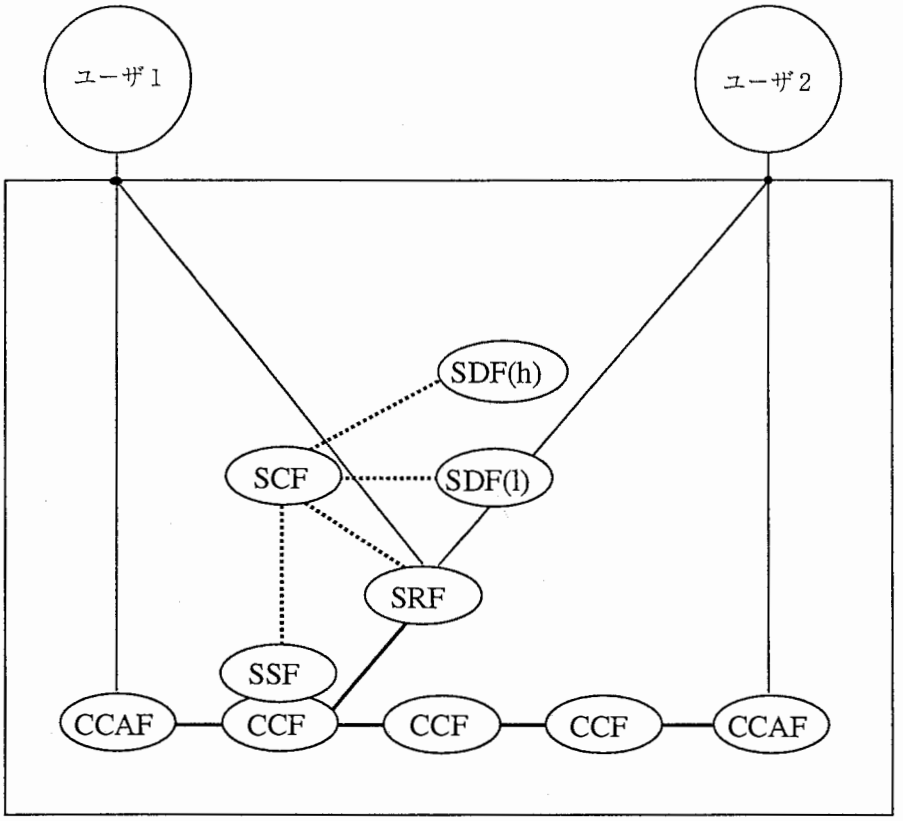

図 5: 機能モデル

け取り，制御を行うことがある．階層化モデルと比較して，SAP とプロトコルエンティティの関係について，次の二つの特徴を有する．

- 一つの SAP に複数のプロトコルエンティティが対応する．
- SAP に対応しないプロトコルエンティティが存在する．

機能エンティティ毎に論理的に一つの SAP が対応するという見方もある．ところが，ネットワークアーキテクチャに関する知識を一切使わずに仕様を記述することを目指しているので，機能分散モデルを構成する機能エンティティ毎の SAP を区別する仕様を記述することはできない．

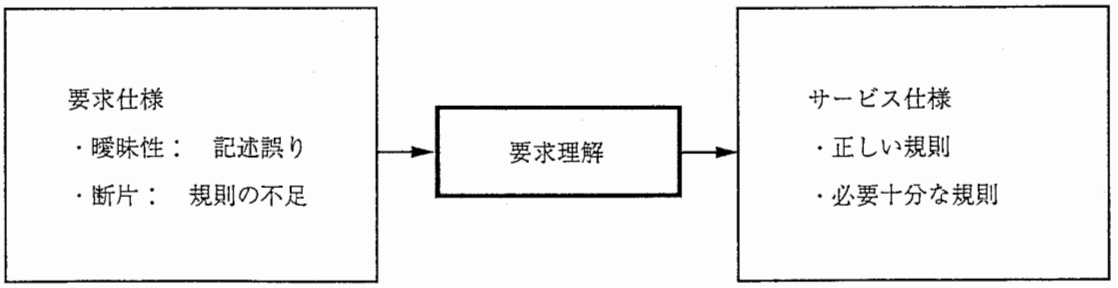

図 6: 要求理解

第 II 部
要求理解

4 要求理解の定義

4.1 要求理解の課題

要求仕様には誤り，曖昧性，不完全性が含まれることがある．要求仕様作成段階で仕様に含まれるこれらの不正確さを取り除くことはソフトウェア開発を効率化する上でも重要である．ソフトウェア開発において，この段階の支援が最も遅れている．ここでは，このような不正確な仕様を通信サービスとして意味のある仕様に変換する手法について述べる [40, 41, 42].

本研究では，要求理解を形式的に記述された要求仕様から通信サービス仕様を導出することであると定義する．要求仕様は必ずしも十分な仕様を含んでいるとは限らない．要求仕様には曖昧性，仕様の不足あるいは矛盾が含まれることがある．STR で記述された通信サービス仕様においては，これらの仕様誤りは，規則に含まれる状態記述要素の過不足，誤った状態記述要素の記述，規則の不足と解釈することができる．図 6に STR における要求理解の概要を示す．

領域知識として既存サービスで使われている規則と通信サービスモデルを用いる．これらは，新サービスを設計する段階において，予め与えられているものとする．これらの領域知識が予め与えられている状況において，不完全な要求から完全なサービス仕様を導出する要求理解が本研究での主要課題である．通信サービスモデルの拡張を含む領域知識獲得方法は今後の課題である．要求獲得により得られたサービス仕様が設計者の意図通りの仕様であるかどうか確認することで要求獲得が完了したと定義する．この確認はアニメーションを用いたシミュレーション [43] により行う．

通信サービス開発では，既にあるサービスに追加する形で新サービスが開発される．この開発形態を仮定する場合，要求から得られた新規サービス仕様についての設計者による確認をもって要求獲得が完了したと考えて構わない．なぜならば，既存サービスと新規サービスを組み合わせた全体サービス仕様の正当性は競合検証 [44] 及びその解消により行うことができるからである．

4.2 要求仕様とサービス仕様

STR におけるサービス仕様の定義を行う．S を STR 規則の集合とする．S に対して，次の 3 条件を満足する端末の集合 T が存在するとき，S はサービスであると定義する．

α を T に含まれるすべての端末が初期状態（空き状態）にあるグローバル状態と定義する．β を S に含まれる規則を使って α から到達可能な T のグローバル状態とする．ある規則の集合が次の 3 条件を満足する場合に，S はサービスであると定義する．

1. α は β から到達可能である．
2. β で適用可能な規則がただ一つある．
3. すべての規則 r に対して，α から到達可能であり，r が適用可能なグローバル状態 β がある．

第1条件はデッドロック，ライブロック，規則適用の競合が発生する状況に陥らないことを保証する．第3条件は，不要な規則がないことを保証する．Sが通信サービスとして意味のある仕様になっているとは限らない．Sが通信サービスとして意味のある仕様になっているかどうかは，アニメーションを使ってサービス設計者が確認することを前提とする．上記の定義は規則の集合が与えられたとき，規則適用の観点から不都合がないことのみを保証する．

5 既存規則の範囲内での要求理解

5.1 仕様誤りの検出と解消

要求仕様に含まれる誤りは，規則に含まれる記述誤り，規則間の矛盾，規則の不足に分類することができる．要求理解の目的は，これらの誤りを含む要求仕様からサービス仕様を求めることである．これらの誤りを検出して，除く手法を示す．手法の概略を図7に示す．

Step 1 個々の規則の誤りを検出し，サービス設計者とインタラクションをとりながら誤りを訂正する．

Step 2 不足規則を検出する．これは，初期状態を始めて再び初期状態に戻ることができるかどうかの到達可能性解析で行う．デッドロック状態が検出されたら，不足規則があると判定する．

Step 3 検出されたデッドロック状態からどの状態に遷移させたいかを追加要求としてユーザから入力してもらう．

Step 4 既存サービスを定義する規則を利用することにより不足仕様を補えるかどうか調べる．仕様を補完するための規則を探索するのに仮説推論を用いる．

Step 5 既存規則では不足する仕様を補えない場合には，通信サービスの抽象モデルであるドメインモデルを用いて不足する規則の新規生成を行う．

Step 6 入力されたすべての規則が使われているかどうか調べる．残っている規則がある場合には，step 2 に戻り，要求の追加を依頼する．

Step 7 アニメーションを用いた獲得された仕様のシミュレーションにより，得られたサービス仕様がユーザの要求を満足する仕様になっているかどうか確認する．満足する仕様になっていない場合には，初めからやり直す．

ステップ2～5について，続く節で詳細を示す．

5.2 規則誤りの検出と解消

規則の記述を正しく直さなければならない記述誤りは，次の二つに分類することができる．一つは規則に記述抜けや誤った記述を含む場合である．もう一つは規則自体の記述は正しいが，他の規則との関係により矛盾がある場合がある．これらの誤りを，非決定変数，自由変数，非決定規則の3種類の誤りとして検出する [45]．はじめにこれらを定義する．

非決定変数 ある規則を適用すると，次の状態が複数取り得る場合の規則をさす．

自由変数 規則に現れる変数の中にイベント生起点から直接あるいは間接的にもたどれない変数をさす．

非決定規則 二つの規則の間で，始状態，イベントが同一であるにもかかわらず，次状態が異なる関係にある規則をさす．

これら3種類の誤りの例を示す．初めに，非決定変数を含む規則の誤りの例を示す．

path(A,B),path(B,A),path(A,C),path(C,A)
flash(A):
path(A,B),path(B,A),hold(A,C),hold(C,A)

上の規則では，AにとってBとCが対象な関係にある．a, b, c を端末とするとき，実際の状態
“path(a,b),path(b,a),path(a,c),path(c,a)”
に上記規則の適用を行うと，次の二つのどちらの状態に遷移したらよいのか決定できない．
“path(a,b),path(b,a),hold(a,c),hold(c,a)” and

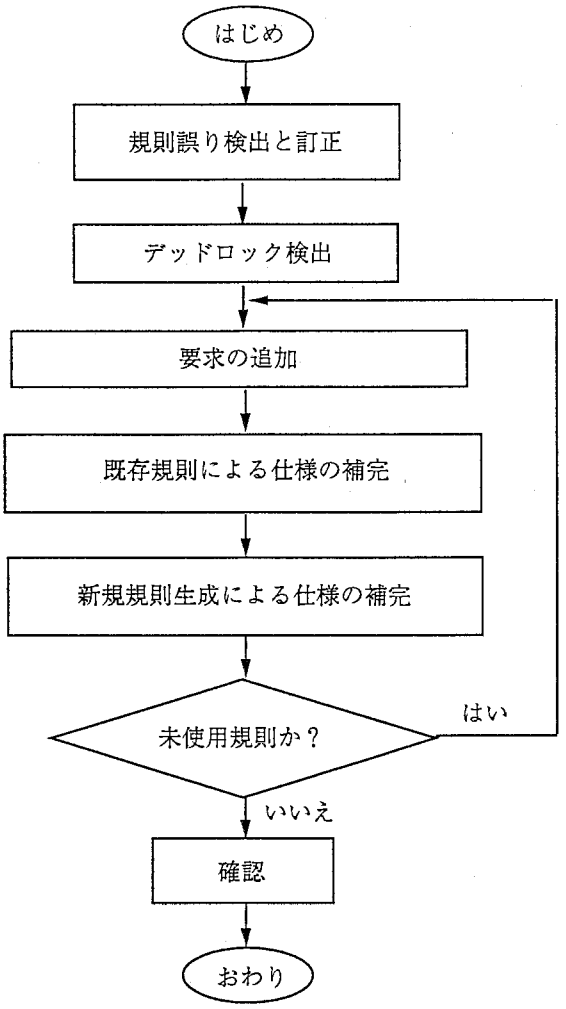
はじめ
規則誤り検出と訂正
デッドロック検出
要求の追加
既存規則による仕様の補完
新規規則生成による仕様の補完
未使用規則か？
はい
いいえ
確認
おわり

図 7: 要求理解流れ図

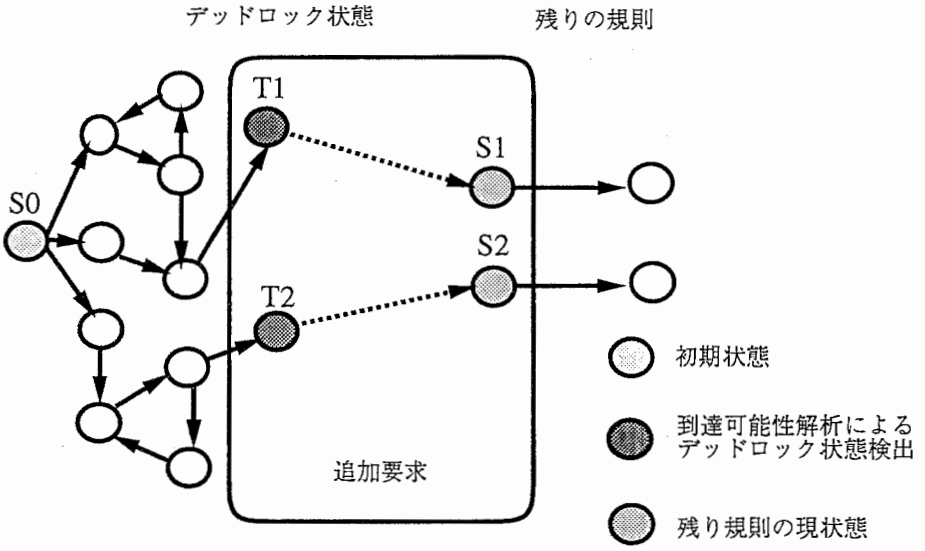

図 8: 到達可能性解析による不足規則検出

"path(a,c),path(c,a),hold(a,b),hold(b,a)".

次に，自由変数を含む規則の誤りの例を示す．次の規則では，現状態及びイベントに含まれる情報から C を特定することができない．

path(A,B),path(B,A),idle(C)
flash(A):
hold(A,B),hold(B,A),ring-back(A,C),ringing(C,A)

最後に，非決定規則の誤りの例を示す．一つめの規則はホットラインにおいて，ダイヤル先の電話に直接接続される規則であり，二つめは通常の空いて呼びだしを行う規則である．

dial-tone(A),idle(B) dial(A,B): path(A,B),path(B,A)
dial-tone(A),idle(B) dial(A,B): ring-back(A,B),ringing(B,A)

これらの規則誤りの検出は自動的に行うことができるが，誤りの修正はユーザとのインタラクションを通して修正を行う．

上記の 3 種類の記述誤り以外に，規則は正しい記述となっているが，サービスを構成する上で誤った記述になっている規則がある．これは，次節以降における要求仕様がサービスとしての性質を満足するかどうかの判定で検出し，正しい規則と置き換える．

5.3 不足規則の検出

要求仕様において，サービス仕様として不足する規則の検出法を示す．規則の不足がある場合には，初期状態からはじめて到達可能性解析により起こり得る動作を求めるとデッドロック状態に至る．

到達可能性解析の結果デッドロック状態が検出されたら，要求理解システムはサービス設計者に新たな要求の追加を依頼する．追加する要求は，検出されたデッドロック状態を出て最終的に到達して欲しいとサービス設計者が望む状態を指定する．この状況を図 8に示す．デッドロック状態 T1，T2 が検出され，T1 から S1 に，T2 から S2 に到達して欲しいという要求が入力された状況を示す．目標状態 S1 と S2 はサービス設計者により記述されるか，それまでに使われていない規則の現状態から選択される．

要求仕様のシミュレーションを行うとき，用いる端末の数を予め決定しておく．従って，用意した端末の数を越える範囲ではじめて起こるデッドロックについては検出することができないという限界がある．ある状態が起こるのに必要な端末の数については予め決定することが可能である [46, 47].

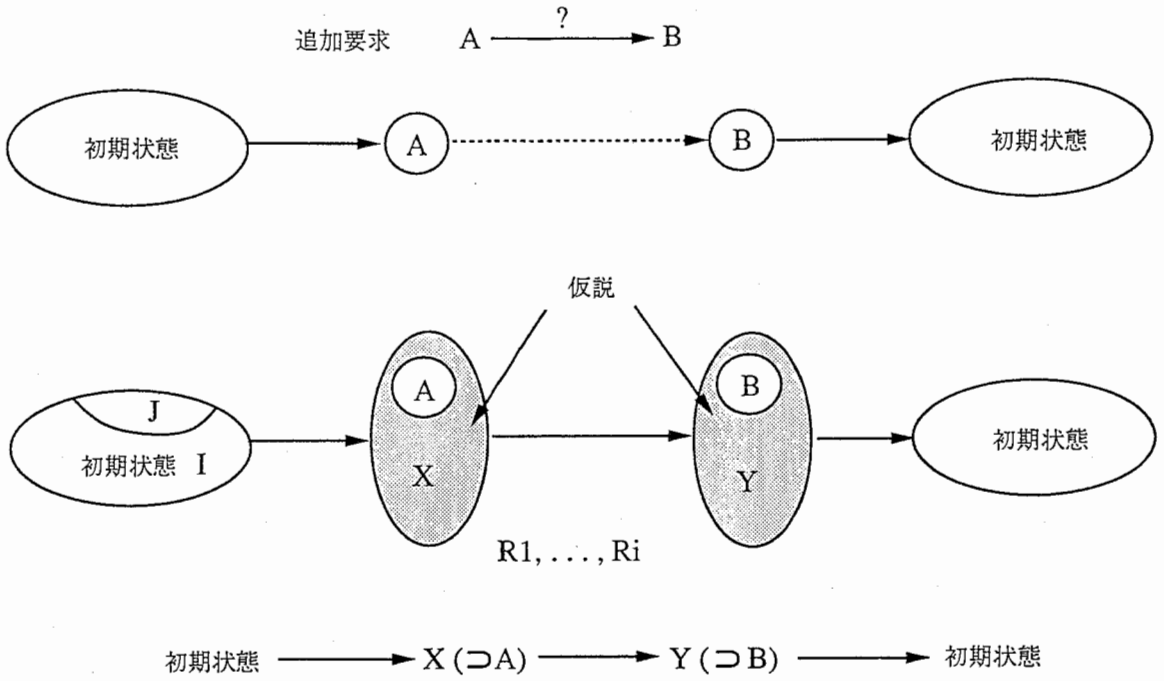

図 9: 仮説推論

5.4 不足規則の補完

図9に，デッドロック状態としてAが検出され，AからBに遷移するという新たな要求が追加されたときの仮説推論の様子を示す．システムは状態X（⊃A）から状態Y（⊃B）への遷移を可能にする規則の集合$\{R_1, \cdots, R_i\}$で要求を補完する．仮説には，次の二つの制約がある．

- 仮説X-Aは空集合であるか，X-Aだけからなる状態に初期状態の一部Iから遷移可能であり，この遷移は初期状態の残りJからAへの遷移とは独立でなければならない（図9）．

  これは，初期状態からAへの遷移に何の影響を与えることもなく，X-Aが初期状態から到達可能でなければならないことを意味する．この制約により，デッドロック状態Aの解消が可能となる．

- X-A及びY-Bは，それぞれA及びBに矛盾してはならない．

  A（B）に状態記述要素pが含まれているとき，¬pを仮説としてX-A（Y-B）に付加してはならない．

不足規則を補うために用いる仮説推論による到達可能性解析の例について説明する．要求として，次が入力された場合を考える．

A: dial-tone(a)
B: path(a,b)

このとき，後向き仮説推論を行うことによって，状態Aから状態Bへ至るのに必要となる規則を探索する．このとき，状態AからBへはそれぞれそれらを含む状態X，Yとなるように仮説を追加することにより到達する．

X: dial-tone(a),idle(b),m-cfv(c,b),¬(idle(c))
Y: path(a,b),path(b,a),m-cfv(c,b),¬(idle(c))

状態X，Yに含まれる¬(idle(c))は状態に対する制約であり，状態の一部ではない．図10に仮説推論がどのように行われるかを示す．

太枠で囲んだ1の状態を元の指定状態とする．仮説“path(b,a)”を付加して規則pots-5を適用することにより，状態2“ring-back(a,b), ringing(b,a)”が導出される．このとき，状態1は状態4となる．更に，状態2に仮説“m-cfv(c,b), ¬(idle(c))”を付加して規則cfv-10を適用することにより，状態3が導かれる．このとき，状態4は状態6となる．仮説を付加したときに得られる状態には矛盾がないことを調べる．

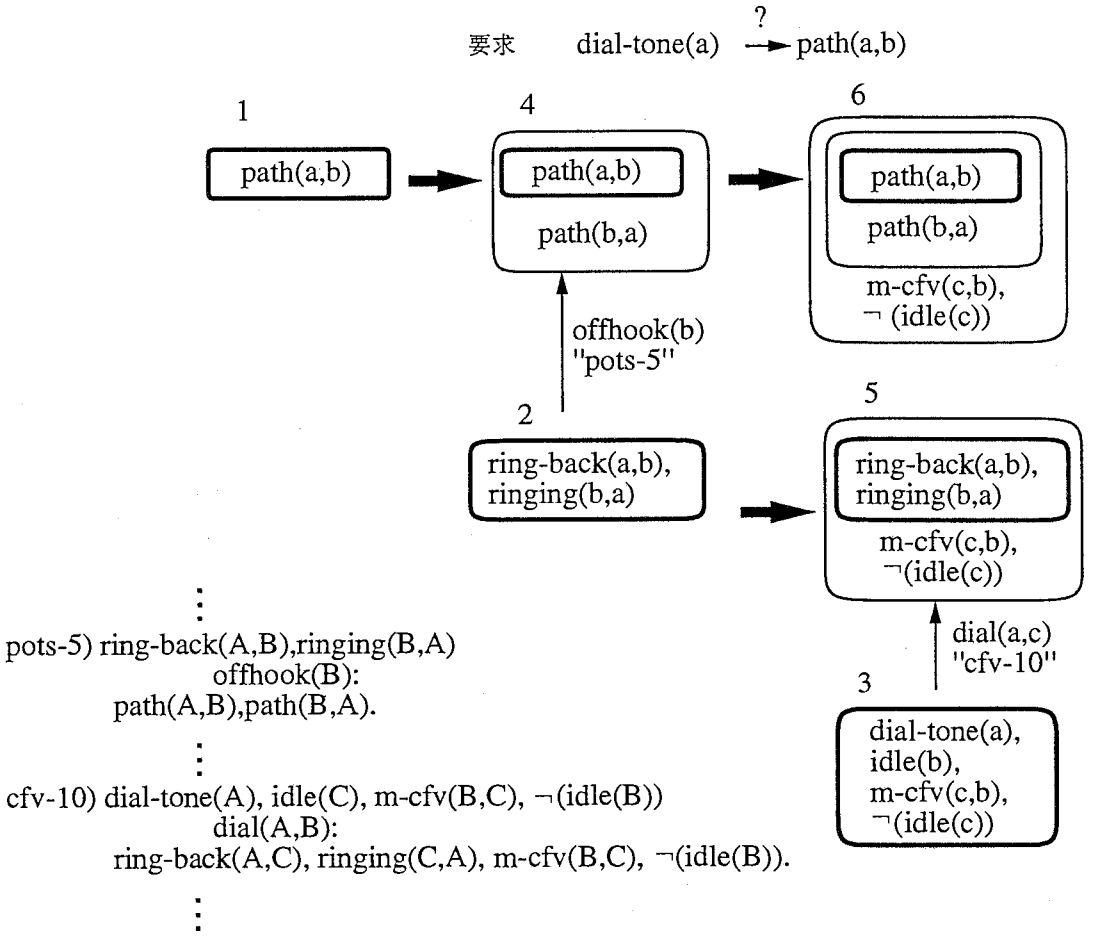
要求
dial-tone(a)
?
path(a,b)
1
path(a,b)
4
path(a,b)
path(b,a)
6
path(a,b)
path(b,a)
m-cfv(c,b),
¬ (idle(c))
offhook(b)
"pots-5"
2
ring-back(a,b),
ringing(b,a)
5
ring-back(a,b),
ringing(b,a)
m-cfv(c,b),
¬(idle(c))
dial(a,c)
"cfv-10"
3
dial-tone(a),
idle(b),
m-cfv(c,b),
¬(idle(c))
pots-5) ring-back(A,B),ringing(B,A)
offhook(B):
path(A,B),path(B,A).
cfv-10) dial-tone(A), idle(C), m-cfv(B,C), ¬(idle(B))
dial(A,B):
ring-back(A,C), ringing(C,A), m-cfv(B,C), ¬(idle(B)).

図 10: 仮説推論の例

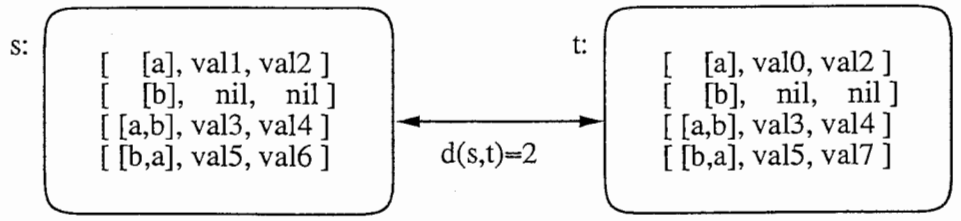

図 11: モデル推論

この例において，X-A = “idle(b),m-cfv(c,b),¬(idle(c))” は，dial-tone(a) とは独立に到達可能な状態であると仮定する．従って，仮説推論により得られた規則 pots-5, cfv-10 が所望の規則となる．この例において，“path(a,b)” は二端末間の状態の一部として現れるので，A，B それぞれの状態を補う仮説推論を行わない限り，不足規則を求めることはできない．

6 新規規則の生成が必要な要求理解

6.1 ドメインモデル

既存サービスに使われている規則を利用するだけでは，不足規則を補うことができないときがある．この場合，新規規則を生成することが必要である．不足する新規規則を生成するために，通信サービスを抽象化したドメインモデルを領域知識として用いる．

ドメインモデルは，ドメイン知識とドメイン辞書からなる．ドメイン知識は，通信サービスを抽象化した表現である通信サービスモデルを表す．通信サービスモデルは，通信サービスを特徴付ける属性と属性の値を判定し，変更する属性操作からなる．ドメイン辞書は，STR で記述されたサービス仕様と通信サービスモデルとの相互変換機能を有する．

ドメインモデルを用いた新規規則生成は次の 3 ステップから構成される．この概略図を図 11に示す．

Step 1 検出されたデッドロック状態とそこからの遷移先として指定された状態をドメイン辞書を使って通信サービスモデルに写像する．

Step 2 通信サービスモデル上で推論を行いデッドロック状態，遷移先状態に対応する通信サービスモデル上の状態間の遷移を実現する属性操作系列を求める．

Step 3 得られた属性操作系列をドメイン辞書を使って通信サービス仕様に変換する．

通信サービスモデルの構成要素である属性及び属性操作の定義を行う．任意の通信サービスを基本的な属性及び属性操作を用いて構成的に定義するために，呼要素を導入する．呼要素は，一端末の属性を表す呼要素と二端末間の関係を表す呼要素の二種類がある．前者をタイプ 1 呼要素，後者をタイプ 2 呼要素と呼ぶ．n 個の端末の状態は，n 個の端末それぞれに対するタイプ 1 呼要素 n 個とある端末と他の n-1 個の端末との関係を表す n × (n-1) 個のタイプ 2 呼要素により表現する．通信サービスに参加する端末の数は，一般に不特定多数である．呼要素の組合せとして状態を定義することにより，サービス参加者数が固定していない通信サービスを通信サービスモデルに対応付けることが可能になる．

STR で表現された通信サービスの状態をサービス状態，属性の値として表現された通信サービスモデル上の状態をモデル状態と呼ぶことにする．モデル状態は，呼要素の集合で定義され，呼要素はタイプ別に定義された属性に対する値から構成される．すべての呼要素はどの端末変数に対応する呼要素であるかを表すアドレス属性を持つ．アドレス属性には，発アドレスと着アドレスがある．発アドレスは，呼要素の帰属先を表す．着アドレスは，タイプ 2 呼要素に対して，発アドレスで指定される端末がどの端末との間の関係を表す呼要素であるかを指定する．あるサービス状態が与えられると，対応するモデル状態を定義する呼要素の集合は自動的に決まる．サービス状態とモデル状態の相互変換は，ドメイン辞書に定義されている部分的なサービス状態，部分的なモデル状態から全体のサービス状態，モデル状態を合成することにより行う．これらの関係の例を図 12に示す．

サービス状態

“p1(a),p2(a,b),p3(b,a)”

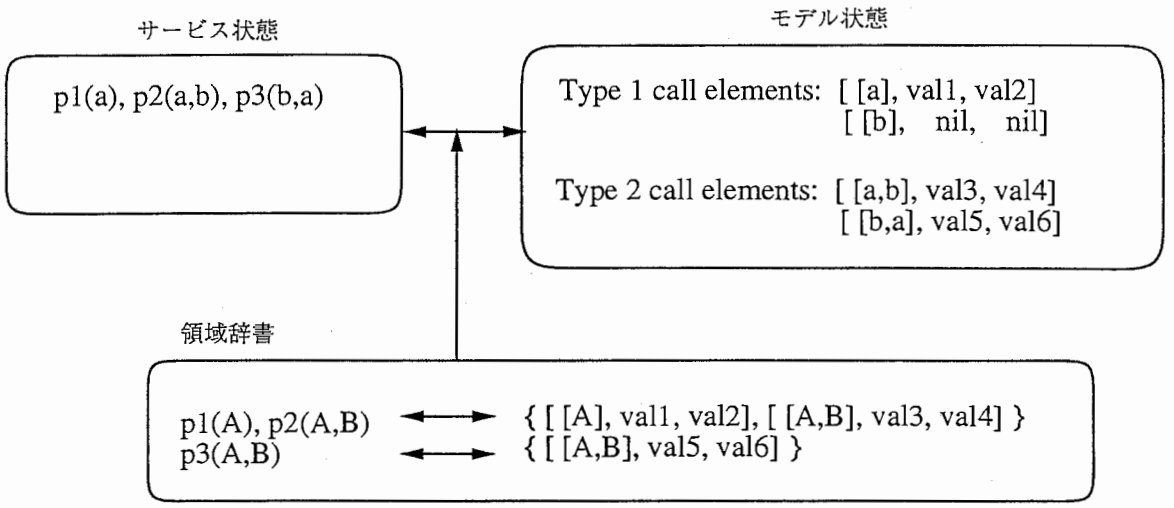

図 12: サービス状態とモデル状態の相互変換

にモデル状態として,

```
1型呼要素 : {  [    [a],   val1,   val2   ],
               [    [b],    nil,    nil   ] }
2型呼要素 : {  [   [a,b],   val3,   val4   ],
               [   [b,a],   val5,   val6   ] }
```

が対応する．この対応関係は，ドメイン辞書に定義されている関係

```
p1(A),p2(A,B)   ↔   { [[A], val1, val2], [[A,B], val3, val4]},
p3(A,B)         ↔   { [[A,B], val5, val6]}
```

により求められる．

モデル状態は属性操作の適用により属性値の変更が行われ，別のモデル状態に遷移する．属性操作は，主属性操作と従属性操作の二種類がある．主属性操作はイベントに現れる引数を決定するイベント引数決定条件が規定されているところが従属性操作と異なる．主属性操作は，適用条件，操作条件，操作，イベント引数決定条件から構成され，従属性操作は，適用条件，操作条件，操作から構成される．

適用条件は，対象となるモデル状態に対して適用可能な属性操作を限定するための条件を表す．一つの適用条件は，一つの呼要素識別子（CE）と 0 個以上の属性識別子（$IF）から構成される．and 条件を表す “,” と or 条件を表す “—”，否定条件を表す “not” を用いることにより複合適用条件を表す．

呼要素識別子は，属性識別子の前に記述して，属性識別を行う呼要素を限定する．

シンタクス:　$CE(call element type, address 1, address 2)

呼要素識別子は，呼要素タイプ，アドレス属性 1，アドレス属性 2 の 3 引数をとる．呼要素タイプは，タイプ 1 呼要素を 1，タイプ 2 呼要素を 2，タイプを指定しない場合は “-” で表す．アドレス属性 1 はつねに発アドレスを表す．アドレス属性 2 は，タイプ 1 呼要素ではつねに “-” であり，タイプ 2 呼要素では着アドレスを表す．任意の発アドレス，着アドレスを表す場合には “-” を用いる．対象としている呼要素の発アドレスを表す変数として oa，着アドレスを表す変数として ta を用いる．

属性識別子は，具体的な属性及び属性値により，呼要素識別子により限定された呼要素を更に限定するために用いる．

シンタクス:　$IF(属性名, 属性値)
$IF(属性名, -)
$IF(属性名)
$IF(属性名, 変数)

$IF(属性名, 属性値): 属性名で指定された属性の値が属性値で指定された値のときに真となる.

$IF(属性名, -): 属性名で指定された属性の値が未設定であるとき真となる.

$IF(属性名): 属性名で指定された属性の値が設定されているときに真となる.

$IF(属性名, 変数): 属性操作内で設定されている他の同一の名前をもつ変数のさす値と同一であるときに真となる.

操作条件は, 適用条件で指定された呼要素のうち, 操作を適用する呼要素を限定するために用いる. 操作条件は, 呼要素識別子, 属性識別子で構成される. 記述形式は, or 記述ができないことを除いて, 適用条件の記述形式と同一である.

操作条件で限定された呼要素に対して操作を行う.

シンタクス: $VALUE(属性名, 属性値)
$VALUE(属性名, -)
$VALUE(属性名)

$VALUE(属性名, 属性値): 属性名で指定された属性の値を属性値で指定された値に書き換える.

$VALUE(属性名, -): 属性名で指定された属性の値を未設定にする.

$VALUE(属性名): 属性名で指定された属性の値にその属性に対する初期値を設定する.

イベント引数決定条件は, イベントの引数を決定する呼要素を指定する. 記述形式は適用条件と同一であるが, or 記述はできない.

6.2 ドメインモデルを用いた推論

規則生成手順

開始状態と目標状態が与えられたとき, 開始状態から目標状態に至る STR 規則を求める手法を述べる. 次に示す手順で不足規則の生成を行う.

ステップ1 目標状態の呼要素の集合が開始状態の呼要素の集合を含むように, 開始状態に状態記述要素の補完を行う. 以降では, 補完された開始状態を単に開始状態と呼ぶ.

ステップ2 開始状態, 目標状態をドメイン辞書を使って呼要素の集合に変換する. 開始状態に対応するすべての呼要素に対する属性値の集合を開始モデル状態, 目標状態に対応するすべての呼要素に対する属性値の集合を目標モデル状態と呼ぶ.

ステップ3 開始モデル状態から始めて適用条件を満足する属性操作を行うことにより, 目標モデル状態を含む状態に至る属性操作系列を求める. ここでモデル状態の包含関係は呼要素の集合の包含関係として定義する. 属性操作系列を求めるときに, 後述の二つのモデル状態間に定義される距離を用いる.

ステップ4 得られた属性操作系列をドメイン辞書を使って STR 規則の集合に変換する.

距離

二つのモデル状態間の距離を定義する. s, t を二つのモデル状態とする. s,t に共通する呼要素間で属性の値が異なるものの総数を s と t の間の距離と定義する. s と t の間の距離を d(s,t) と書く. 図 13に距離が 2 である二つのモデル状態の例を示す.

モデル推論

開始モデル状態から目標モデル状態へ至る属性操作系列の求め方を述べる. 求める属性操作系列は主属性操作の後に 0 個以上の従属性操作が続く系列が一つ以上つながる属性操作系列である. 属性操作は, 操作適用後に得られるモデル状態と目標モデル状態との間の距離が短くなるものをはじめに選ぶ. これを直接推論と呼ぶ. 距離が短くなる属性操作がない場合には, 距離が変わらない属性操作を次の候補として選ぶ. これを迂回推論と呼ぶ. 迂回推論ができない場合には, 距離が遠くなる属性操作を選ぶ. これを遠回り迂回推論と呼ぶ. 直接推論, 迂回推論, 遠回り迂回推論の例を図 14に示す.

6.3 例

モデル推論を用いて断片的な要求から完全なサービス仕様を求める例を示す. 転送とキャッチホンサービスを組み合わせた新しいサービスに対する部分的な仕様が要求として与えられた場合を考える. 通常の転送では, 単に着信

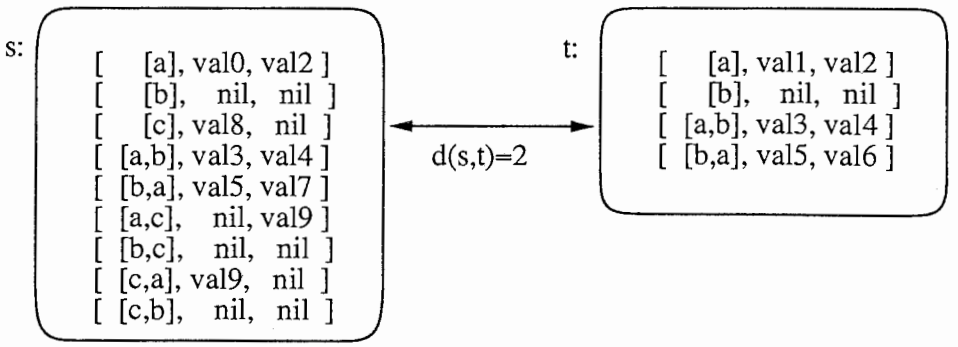

図 13: モデル状態間の距離

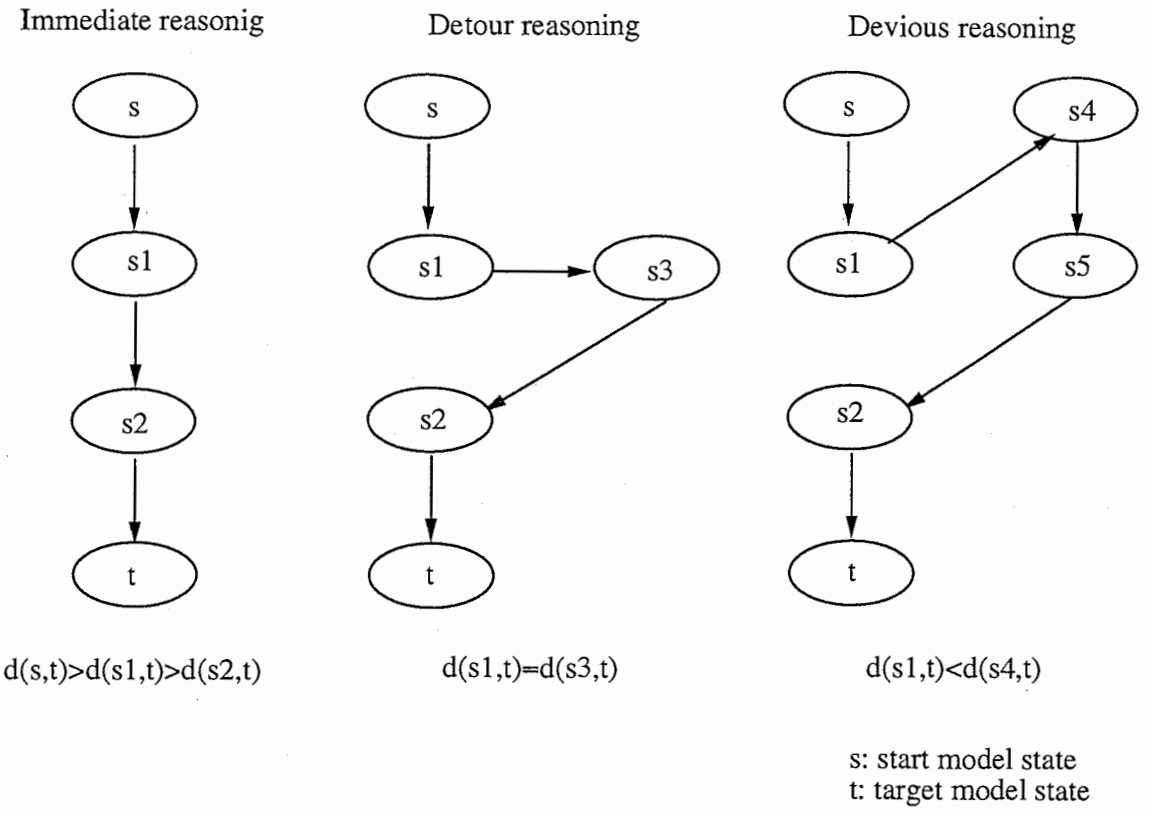

図 14: モデル上の種々の推論

呼だけを転送するが，このサービスでは転送元で持っている機能も転送する．そこで，キャッチホン加入者（A）が着信呼の転送を設定しておくと，その電話（A）への着信は転送先（B）に転送される．転送先が他の電話（C）と通話中のとき，転送先ではキャッチホンでの着信となる．Bでフラッシュすると，BC間の通話は保留され，CはAと通話状態となる．

要求仕様として，次の二つの状態が与えられたとする．

開始状態: dial-tone(A),idle(B),m-cw(B),m-trans(B,C),path(C,D),path(D,C),

目標状態: idle(B),m-cw(B),m-trans(B,C),
m-cw-transed(C),path(A,C),path(C,A),hold(C,D),hold(D,C).

通信サービスモデルには，キャッチホン，通常の転送，任意の機能を他に転送する機能があるとする．これらのサービスを表現するために，タイプ1，2呼要素に対して，それぞれ次の属性を用意する．実際の通信サービスに対してはこれ以外の属性も必要となるが，ここでは最低限必要となる属性を示す．

Type 1: [[orig], handset, tone, cw, ftrans, ftransed]
Type 2: [[orig, term], path, ring, trans]

各属性は以下の値を取る．

1型呼要素：

orig 呼要素の属する先 originating address を表す．

handset 受話器の使用状態 on, off を表す．

tone ベルの状態 in, out, no を表す． in は受話器から， out は電話機から， no は無鳴動状態を表す．

cw キャッチホンの登録状態 on, off を表す．

ftrans orig が持っている機能の転送可否 on, off を表す．

ftransed orig が持っている機能の転送状態 on, off を表す．

2型呼要素：

orig と term タイプ2呼要素の属する先 originating and terminating addresses を表す．

path terminating address で指定された端末との通話状態を表す．

ring terminating address で指定された端末との間における音の状態 on, off を表す．

trans 機能の転送登録状態 on, off を表す．

ドメイン辞書に登録されている属性操作を以下に示す． MAIN で始まるラベルをもつ属性操作は主属性操作を表し，SUB で始まるラベルをもつ属性操作は従属性操作を表す．

```
PRIM1:
  Apply_Condition:
    $CE(1,ad2,-):$IF(ftrans,on),$CE(2,ad2,ad3):$IF(trans,on);
  Operation:
    $CE(1,ad3,-):$IF(cw,off),$VALUE(cw,on),$VALUE(ftransed,on);
  Event_Condition:
    $CE(1,ad1,-):$IF(tone,dial-tone),$CE(2,ad2,ad3):$IF(trans,on),$CE(2,ad1,ad3);

SUB1:
  Apply_Condition:
    $CE(1,ad1,-):$IF(tone,dial-tone),$CE(2,ad2,ad3):$IF(trans,on);
  Operation:
    $CE(1,ad1,-):$VALUE(tone,off);
    $CE(2,ad1,ad3):$VALUE(ring,cw-rbt);
    $CE(2,ad3,ad1):$VALUE(ring,cw-rgt);
```

```
PRIM2:
  Apply_Condition:
    $CE(2,ad1,ad3):$IF(ring,cw-rbt),$CE(2,ad3,ad1):$IF(ring,cw-rgt),
    $CE(2,ad3,ad4):$IF(path,conn),$CE(2,ad4,ad3):$IF(path,conn);
  Operation:
    $CE(2,ad1,ad3):$IF(ring,cw-rbt),$VALUE(path,conn),$VALUE(ring,off);
    $CE(2,ad3,ad1):$IF(ring,cw-rgt),$VALUE(path,conn),$VALUE(ring,off);
    $CE(2,ad3,ad4):$IF(path,conn),$VALUE(path,hold);
    $CE(2,ad4,ad3):$IF(path,conn),$VALUE(path,hold);
  Event_Condition:
    $CE(2,ad3,ad4):$IF(path,conn),
    $CE(2,ad4,ad3):$IF(path,conn),
    $CE(1,ad3);
```

与えられた要求に関係する部分のドメイン辞書を示す.

Service state	Model state
dial-tone(T1)	[[T1],off,dial-tone,-,-,-]
m-cw(T1)	[[T1],-,-,on,-,-]
m-trans(T1,T2)	[[T1,T2],-,-,on]
path(T1,T2)	[[T1],off,-,-,-], [[T1,T2],conn,-,-]
cw-ringing(T1,T2)	[[T1],off,off,-,-,-], [[T1,T2],-,cw-rgt,-]
cw-ring-back(T1,T2)	[[T1],off,off,-,-,-], [[T1,T2],-,cw-rbt,-]
hold(T1,T2)	[[T1],off,-,-,-], [[T1,T2],hold,-,-]
idle(T1)	[[T1],on,off,-,off,-]
m-cw-transed(T1)	[[T1],-,-,on,-,on]

Event	Attribute operation
PRIM1,SUB1	dial
PRIM2	flash

以上の定義に基づいて行ったサービスモデル上での推論結果を以下に示す. 変更のあった属性要素は値にアンダーラインを付けて示してある.

開始状態： dial-tone(A),idle(B),m-cw(B),m-trans(B,C),path(C,D),path(D,C)

{ [[A],off,dial-tone,off,off,off],[[A,B],disc,off,off],[[A,C], disc,off,off],[[A,D],disc,off,off],
[[B],on,off,on,on,off],[[B,A],disc,off,off],[[B,C],disc, off,on],[[B,D],disc,off,off],
[[C],off,off,off,off,off],[[C,A],disc,off,off],[[C,B],disc, off,off],[[C,D],conn,off,off],
[[D],off,off,off,off,off],[[D,A],disc,off,off],[[D,B], disc,off,off],[[D,C],conn,off,off]}

推論途中状態：

{ [[A],off,dial-tone,off,off,off],[[A,B],disc,off,off],[[A,C], disc,off,off],[[A,D],disc,off,off],
[[B],on,off,on,on,off],[[B,A],disc,off,off],[[B,C],disc, off,on],[[B,D],disc,off,off],
[[C],off,off,<u>on</u>,off,<u>on</u>],[[C,A],disc, off,off],[[C,B],disc,off,off],[[C,D],conn,off,off],
[[D],off,off,off,off,off],[[D,A],disc,off,off],[[D,B],disc,off,off],[[D,C],conn,off,off]}

中間状態： idle(B),m-cw-transed(C),m-cw(B),m-trans(B,C),
path(C,D),path(D,C),cw-ring-back(A,C),cw-ringing(C,A)

{ [[A],off,<u>off</u>,off,off,off],[[A,B],disc,off,off],[[A,C], disc,<u>cw-rbt</u>,off],[[A,D],disc,off,off],
[[B],on,off,on,on,off],[[B,A],disc,off,off],[[B,C],disc, off,on],[[B,D],disc,off,off],
[[C],off,off,on,off,on],[[C,A],disc,<u>cw-rgt</u>, off],[[C,B],disc,off,off],[[C,D],conn,off,off],
[[D],off,off,off,off,off],[[D,A],disc,off,off],[[D,B],disc,off,off],[[D,C],conn,off,off]}

目標状態： idle(B),m-cw-transed(C),m-cw(B),m-trans(B,C),
path(A,C),path(C,A),hold(C,D),hold(D,C)

{ [[A],off,off,off,off,off],[[A,B],disc,off,off],[[A,C], conn,off,off],[[A,D],disc,off,off],
[[B],on,off,on,on,off],[[B,A],disc,off,off],[[B,C],disc, off,on],[[B,D],disc,off,off],
[[C],off,off,on,off,on],[[C,A],conn, off,off],[[C,B],disc,off,off],[[C,D], hold,off,off],
[[D],off,off,off,off,off],[[D,A],disc,off,off],[[D,B],disc,off,off],[[D,C],hold,off,off]}

次のSTR規則r1, r2が生成される.
dial-tone(A), idle(B), m-cw(B), m-trans(B,C), path(C,D), path(D,C)
dial(A,B):
idle(B), m-cw-transed(C), m-cw(B), m-trans(B,C), cw-ring-back(A,C), cw-ringing(C,A),
path(C,D), path(D,C).

cw-ring-back(A,C), cw-ringing(C,A), m-cw-transed(C), path(C,D), path(D,C)
flash(C):
path(A,C), path(C,A), hold(C,D), hold(D,C), m-cw-transed(C)

この例では，ドメイン辞書を完全な形で定義してある．実際には，相互に変換できない部分がある場合がある．その場合には，サービス設計者との会話を通して変換を行う．

7 評価と今後の課題

本研究で提案する要求理解の有効性と限界及び今後の課題について述べる．STRで記述された要求仕様から通信サービスとして意味があり，サービス設計者の意図を満足するサービス仕様に変換する要求理解手法について述べた．この要求理解手法は，既存規則を利用して変換を行う手法と新規規則を生成して変換を行う手法から構成される．既存規則を利用する要求理解に関しては，仮説推論を組み込んだことにより十分な要求理解に関する機能を有していると評価できる．新規規則の生成が必要な要求理解に関しては，ドメインモデルの一般性とモデル上での推論能力により，要求理解機能が評価できる．ドメインモデルに関しては，発アドレスと着アドレスからなる二つのアドレス属性と二つのアドレスで指定される端末間の接続状態に関する属性からなる呼要素を基本単位として，任意個数の端末が関係する任意の通信サービスをモデル化することができる手法を示した．モデル上の推論能力に関しては，異なる属性の数として距離を定義し，その距離を縮める推論手法を採用している．この距離に関して，重み付けを行えるように機能拡張することが今後の課題として残っている．ドメインモデルで標準として用意すべき属性に関するデータの充実は今後の課題である．通信サービスでは，切断処理のようにどのような場合でも用意しておかないといけない処理がある．このような処理をつねにサービス設計者が入力するとは限らない．このような処理の補完は，ドメインモデル上に定義することにより可能となる．但し，切断処理のきっかけとなる入力を受け付ける部分の処理は補うことは可能であるが，その後でどのようなサービス仕様を付加するかはユーザとのインタラクションにより補完しなければならない．

第 III 部
通信ソフトウェアへの変換とソフトウェア開発への適用

8 プロトコル合成法

8.1 規則のグラフ表現

通信システムに属するすべての端末の状態を次の対応関係により一つの有向グラフ (連結とは限らない) として表現することができる．このグラフをシステムグラフと呼ぶ．

端末を頂点，1引数の状態記述要素及びイベントを頂点のラベル，2引数の状態記述要素及びイベントを第一引数から第二引数へのラベル付き有向辺として表現する．イベントの第一引数に対応する頂点をイベント生起点と呼ぶ．システムグラフ内のどの頂点も唯一の識別子としてプロセス識別子を持つ．ある頂点 v に対して，v 及び v を始点とする有効辺，及びそれらのラベルからなる部分グラフを局所状態と呼ぶ．

上の対応関係を用いると，一つの規則は，現状態とイベントのグラフ表現である始グラフと次状態のグラフ表現である次グラフからなる二つの有向グラフの組として表現することができる．このグラフ表現を用いると規則の適用は，条件を満足する部分グラフを探索し，適用すべき規則の次グラフで書き換える問題として定式化することができる．

図15は，次の STR 規則のグラフ表現である．

dial-tone(A),idle(B),m-cfv(B,C),idle(C) dial(A,B): ringback(A,C),ping(B,A),m-cfv(B,C),ringing(C,A).

この規則はダイヤルトーン受信端末 A(dial-tone(A)) のユーザが端末 C に転送を設定してある端末 B(m-cfv(B,C)) にダイヤル (dial(A,B)) したときに，端末 C が空き状態 (idle(C)) ならば，端末 B への着信は端末 C に転送され，端末 A は ring-back(A,C) に，端末 B は ping(B,A),m-cfv(B,C) に，端末 C は ringing(C,A) に状態遷移することを表している．

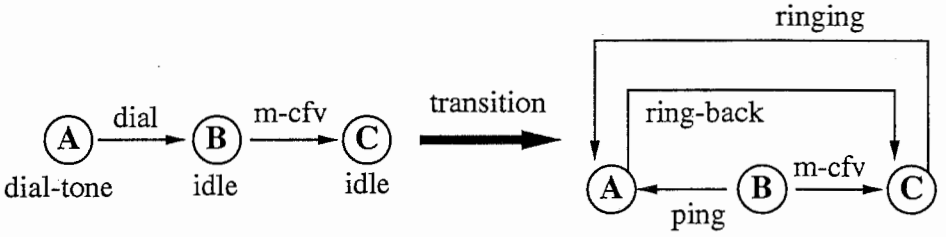

図 15: Example of an STR rule

8.2 問題

$g = (V, E, v^0)$ を，頂点の集合を V，辺の集合を E，根を v^0 とするラベル付き根付き有向グラフとする．本研究では，グラフ g 内のすべての頂点に対して，根 v^0 からその頂点に至る有向路があるグラフのみを対象とする．グラフ g の頂点の集合を $V(g)$，辺の集合を $E(g)$，根を $root(g)$ と表す．頂点の集合 V，辺の集合 E からなるラベル付き有向グラフを $g = (V, E)$ と表す．

定義 1 (部分グラフ同型)
$g = (V, E)$ をラベル付き有向グラフ，$g_1 = (V_1, E_1, v_1^0)$ をラベル付き根付き有向グラフとする．グラフ g が g_1 と同型な部分グラフ $g' = (V', E', v^0)$ をもつための必要十分条件は次の条件を満足する部分集合 $V' \subset V$ と $E' \subset E$ が存在することである．
1. $v^0 \in V'$, $|V'| = |V_1|$, $|E'| = |E_1|$,

2. 次の条件を満たす一対一写像 $f : V_1 \longrightarrow V'$ がある：

$$
\begin{aligned}
&f(v_1^0) = v^0 \\
&(x,y) \in E_1 \Longrightarrow (f(x), f(y)) \in E' \\
&(f(x), f(y)) \in E' \Longrightarrow (x,y) \in E_1 \\
&\alpha(x) \subset \alpha(f(x)) \\
&\beta((x,y)) \subset \beta((f(x), f(y)))
\end{aligned}
$$

ここで，α は頂点に付いているラベルの集合を得る関数であり，β は辺に付いたラベルの集合を得る関数である．□

g が g_1 と同型な部分グラフを持つとき，$g_1 \sqsubset g$ と書く．

定義 **2** (頂点被覆路)
ラベル付き根付き有向グラフ g に対して，頂点被覆路 $sp(g)$ は次の条件を満足する経路として定義する．

1. $sp(g)$ の頂点の集合 g はに含まれるラベル付き頂点の集合と一致する．
2. $sp(g)$ の始点は $root(g)$ である．
3. (u,v) が $sp(g)$ の辺ならば，$root(g)$ と u の間と $root(g)$ と v の間のすべての頂点は $sp(g)$ において u 以前に現れる．

解くべき問題は次の通り形式的に定義される．

定義 **3** (問題)
$R = \{r_1, ..., r_n\}$ をＳＴＲ規則の集合，$G = \{g_1, ..., g_n\}$ を R の始グラフの集合，$G' = \{g'_1, ..., g'_n\}$ を R の次グラフの集合，$g = (V,E)$ をシステムグラフとする．このとき，はじめに次の二つの条件を満足するグラフの組 $(sg(v^0, g), g_i)$ を探す．

1. $sg(v^0, g)$ は $g_i \in G$ と同型である．
2. $g_i \sqsubset g_k$ であるような g_k と同型な部分グラフが g に存在しない．

次に，$sg(v^0, g)$ を $g'_i \in G'$ と同型になるように変換する．□

次の補題が部分グラフ同型に対して成り立つ．この補題を使って，上記問題を解く分散アルゴリズムを構築する．
補題
$g = (V,E)$ をラベル付き有向グラフ，$g_1 = (V_1, E_1, v_1^0)$ をラベル付き根付き有向グラフとする．p を，辺の集合を $\{(u_1, u_2), \cdots, (u_{m-1}, u_m)\}$ $(u_1 = v_1)$ とする g_1 の頂点被覆路，$V_1 = \{u_1, \cdots, u_m, u_{m+1}, \cdots, u_n\}$ とする．このとき，グラフ g が g_1 と同型な部分グラフ $g' = (V', E', v^0)$ を含むことと，$|V'| = |V_1|$ かつ $|E'| = |E_1|$ であり，次の条件を満足する写像 $f : V_1 \longrightarrow V'$ があることとは同値である：
任意のラベル付き頂点 u_i, u_j in V_1 とラベルなし頂点 u_k, u_l in V_1 に対して，次の式が成り立つ．

$$
\begin{aligned}
&f(u_1^0) = v^0, \\
&u_i \neq u_j \Longrightarrow f(u_i) \neq f(u_j), \\
&f(u_i) \neq f(u_k), \\
&u_k \neq u_l \Longrightarrow f(u_k) \neq f(u_l), \\
&(u_i, u_j) \in E_1 \Longleftrightarrow (f(u_i), f(u_j)) \in E', \\
&\alpha(u_i) \subset \alpha(f(u_i)) \\
&\beta((u_i, u_j)) \subset \beta((f(u_i), f(u_j))) \\
&\beta((u_i, u_k)) \subset \beta((f(u_i), f(u_k)))
\end{aligned}
$$

□

この補題は，頂点被覆路に沿った通信により規則の適用可能性を決定することが可能であることを意味する．

次に同じイベントをもつ規則が複数ある場合に，それらの規則間で共通する部分がシステムグラフ中に存在するかいなかを一度に判定するための最適化を可能にする頂点に対する番号付けの方法を示す．この番号付けは，各始グラフの頂点被覆路を与える．

イベント e をもつ規則の始グラフを $g(r_1), \cdots, g(r_m)$ とする．このとき，$\mathrm{V} = \mathrm{V}(g(r_1)) \cup \cdots \cup \mathrm{V}(g(r_m))$ に対して，n をある自然数とするとき，次の 6 条件を満足する番号付け f: $\mathrm{V} \longrightarrow \{1, \cdots, n\} \cup \{\lambda\}$ を考える．

1. 頂点 $v(\in \mathrm{V})$ がラベル付き頂点ならば $f(v) \in \{1, \cdots, n\}$ であり，v がラベルなし頂点ならば $f(v) = \lambda$ である．

2. $f(root(g(r_1)))=\cdots=f(root(g(r_m)))=1$. $root(g(r_i))$ は $g(r_i)$ の根を表す.

3. 二つの頂点 $u,v(\in V(g(r_i)))$ に対して, $u\neq v$ ならば $f(u)\neq f(v)$.

4. $u\in V(g(r_i))$, $v\in V(g(r_j))$ のとき, $f(u)=f(v)$ ならば, $path(root(g(r_i)),u)$ と $path(root(g(r_j)),v)$ 上の頂点の集合をそれぞれ V_1, V_2 とすると, $f(V_1)=f(V_2)$. ここで, $path(root(g(r_k)),w)$ は $g(r_k)$ における $root(g(r_k))$ から w への有向辺を表す.

5. $u,v\in V(g(r_i))$, $w,x\in V(g(r_j))$, $uv\in E(g(r_i))$, $wx\in E(g(r_j))$ のとき, $f(u)=f(v)$, $f(v)=f(x)$ ならば, $\beta(uv)\subseteq\beta(wx$ または $\beta(wx)\subseteq\beta(uv)$. ここで, $\beta(yz)$ は辺 yz のラベルを表す.

6. $f(u)\neq f(v)$ のとき, x を u,v と異なる頂点とするとき, $g(u)=g(v),g(x)=f(x)$ である任意の関数 g において, 3 が成り立たない.

図 16に 4 つの始グラフ r1, ⋯, r4 の頂点に対する番号付けの例を示す.

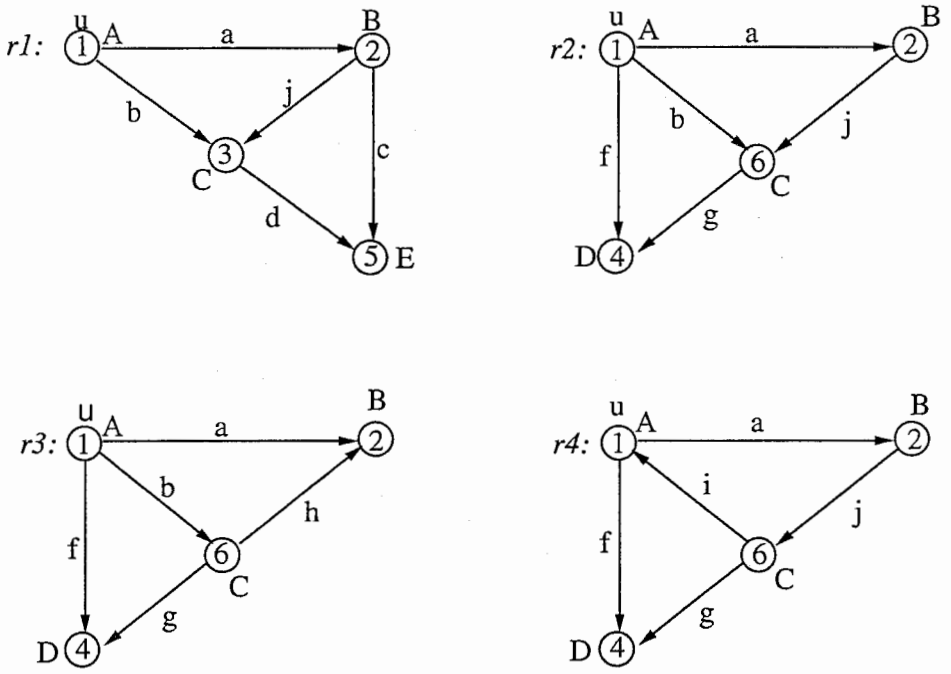

図 16: 頂点に対する番号付けの例

上記の番号付けに関して次の性質が成り立つ.

性質 1 ある始グラフ r がシステムグラフ G に含まれるとき, r の頂点に対する上記の番号付けの番号順に頂点間の通信を行えば, r が G の部分グラフであることを判定することが可能である.

性質 2 あるイベントをもつ始グラフの集合を R とする. 二つの始グラフ $r_1,r_2\in R$ において r_1 が r_2 の部分グラフであるとき r_1 の頂点に対する番号付けが r_2 の頂点に対する番号付けの部分列となるような番号付けが存在する.

8.3 状態遷移片

あるイベント e とそのイベントを持つ始グラフに対する番号付けで用いられている数 $i(\neq 1)$ の組み合わせに対して, 一つの信号名 (e,i) を用意する. この信号は規則の集合を内部データとして持つことができる. この信号を探索信号と呼ぶ. これ以外に, 規則毎に応答信号を用意する. この信号を使って, 次の 3 種類の状態遷移片を定義する.

タイプ 1 番号 1 をもつ局所状態に対しては, イベントを受信した後, 何もしないか, 探索信号を送信した後で, 元の局所状態に遷移する状態遷移片を作成する. ある始グラフに含まれる局所状態が一つしか含まれない場合には何もせず, 複数の局所状態が含まれる場合には番号 2 の局所状態を保持する頂点への探索信号を送る. この探索信号にはその規則が内部データとして含まれる.

タイプ2 番号が2以上の局所状態に対しては，探索信号受信後に，次の番号の局所状態に探索信号を送るか，次の番号の局所状態がない場合にはイベント生起後に通過してきた頂点すべてに応答信号を送った後で，元の局所状態に遷移する．

タイプ3 ある始グラフにおいて番号が最後でない局所状態に対しては，その規則に対応する応答信号を受けとった後で，次グラフの対応する頂点の局所状態に遷移する．

図16の4つの始グラフにおける番号2の頂点に対して作成した状態遷移片を図17に示す．但し，信号名はm1と表記してある．

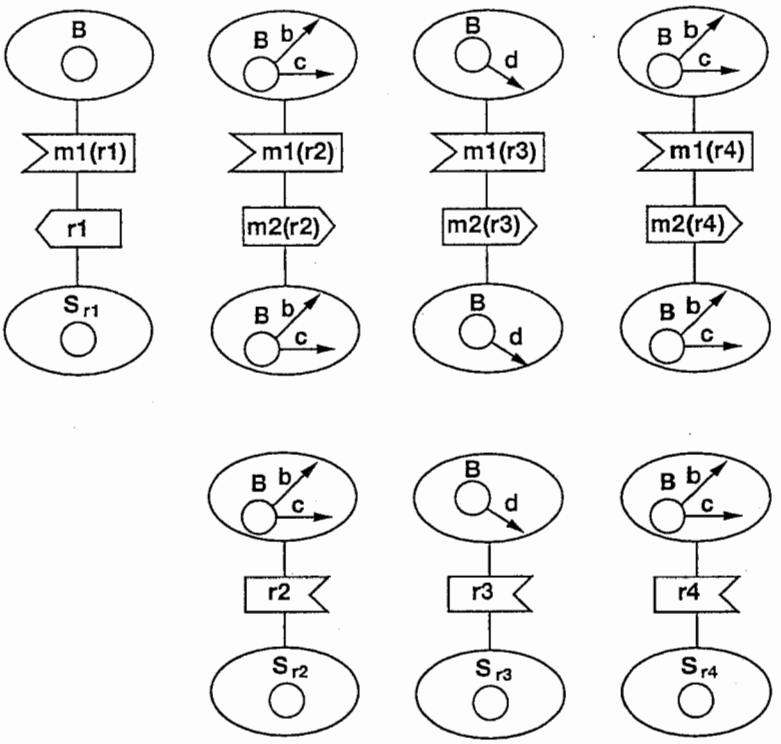

図 17: 状態遷移片の例

8.4 プロトコル合成アルゴリズム

サービス仕様からプロトコルを合成するアルゴリズムを示す．以下に述べるプロトコルの合成は，複数の規則間での適用条件判定のための通信を共通化する方針をとる．

(1) 始グラフの頂点に対する番号付け

すべての始グラフに対して，頂点の番号付けを行う．

(2) 状態遷移片の作成

すべての局所状態に対して状態遷移片を作成する．

(3) プロセス仕様の合成

初期状態から順に状態遷移片を合成し，新たな状態が生成されなくなるまで，合成を繰り返し，最終的に全体のプロセス仕様を求める．合成の過程で，ある新しい状態 s が合成されたとする．このとき，s の部分グラフとして始状態が含まれる状態遷移片をすべて集める．集めた状態遷移片に対して次の方法で合成を行う．

合成されたプロセス仕様は，通信が進むにつれて適用すべき規則が絞られていき，最終的に適用すべき規則が決まる．適用すべき規則が決定した時点で，該当するプロセスがイベント生起後，通過してきたプロセスに応答信号を送る．応答信号には，遷移指示信号と非遷移指示信号がある．遷移指示信号を受けとったプロセスは，状態遷移片で指定された状態に遷移する．非遷移指示信号を受けとったプロセスは，探索信号受信前の状態に遷移する．

8.5 例

プロトコル仕様の合成例を示す．図18にサービス仕様を示す．このサービスでは，データ送信端末とデータ受信端末の二種類の端末を用意する．データ送信端末はイベント “start” により，そのイベントで指定される二つの端末が “idle” のときにデータ送信を開始する．データ送信端末はイベント “stop” によりいつでもデータ送信を終了することが可能である．データ受信端末はイベント “pause” によりデータ送信の休止を，イベント “resume” によりデータ送信の再開を送信端末に要求することが可能である．図19はこのデータ送信プロトコルに対する状態遷移図を表す．

```
rule 1) idle(A),idle(B) start(A,B):
          sending(A,B),receiving(B,A).
rule 2) sending(A,B),receiving(B,A) stop(A):
          idle(A),idle(B).
rule 3) receiving(A,B),sending(B,A) pause(A):
          r-wait(A,B),s-wait(B,A).
rule 4) r-wait(A,B),s-wait(B,A) resume(A):
          receiving(A,B),sending(B,A).
```

図 18: データ送信サービスの STR 記述

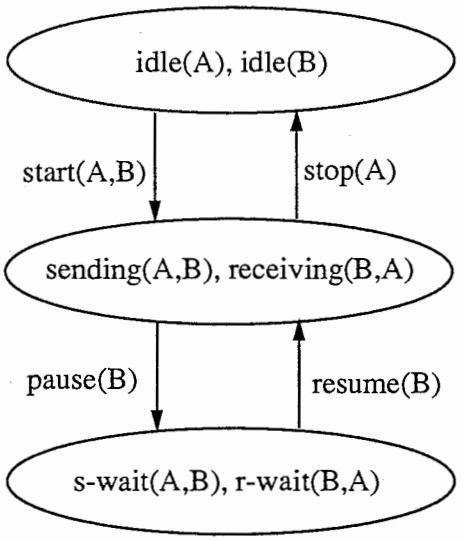

図 19: データ送信サービスの状態遷移図

図18のサービス仕様対するグラフ表現を図20に示す．この例から得られる状態遷移片を図21に示す．図22は図21に示す状態遷移片から合成される一つのプロトコルエンティティの仕様を示す．図において，メッセージ “m1”, “m2”, “m3”, “m4” は探索信号を表し，“r1”, “r2”, “r3”, “r4” は応答信号を表す．この例では，適用する規則が何もないことを表す “norule” の送信と受信動作は省略してある．更に，この合成されたプロトコルでは，予期しない探索信号を受けとったら，その探索信号の送信元に “norule” を送り返すことを仮定している．

9 プロトコル仕様からソフトウェア仕様への変換

対象となる通信システムを制御するための仕様をサービス仕様とは別に記述する．これを知識として蓄積することにより，サービス仕様だけを記述したらソフトウェアが生成できるようにする．

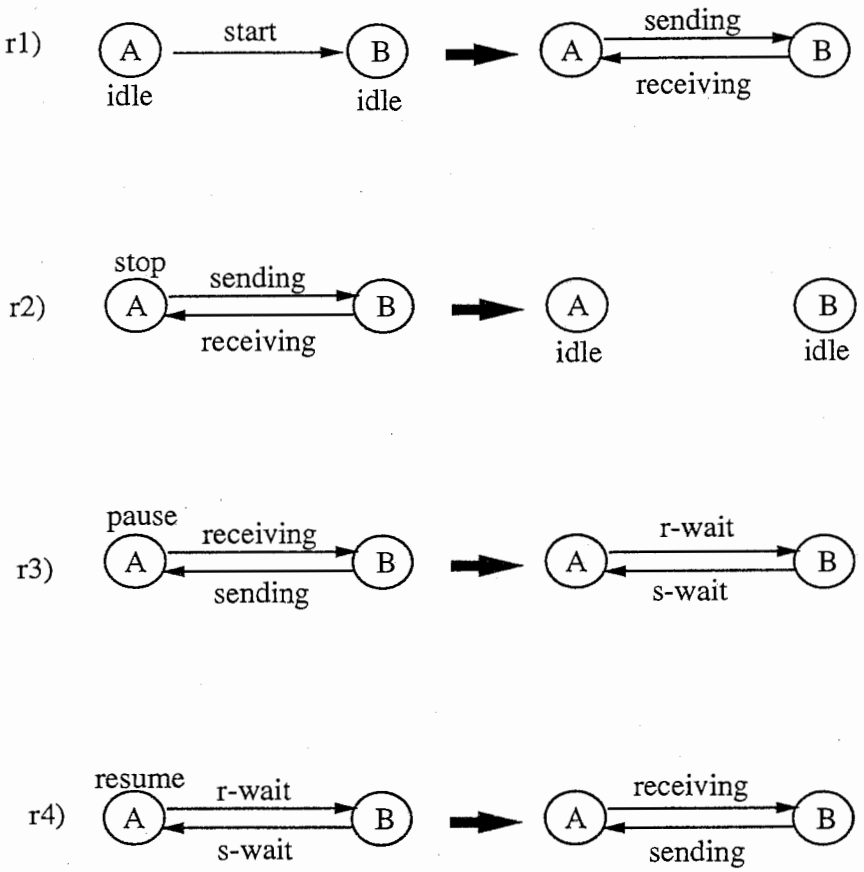

図 20: データ送信サービスに対する STR 記述のグラフ表現

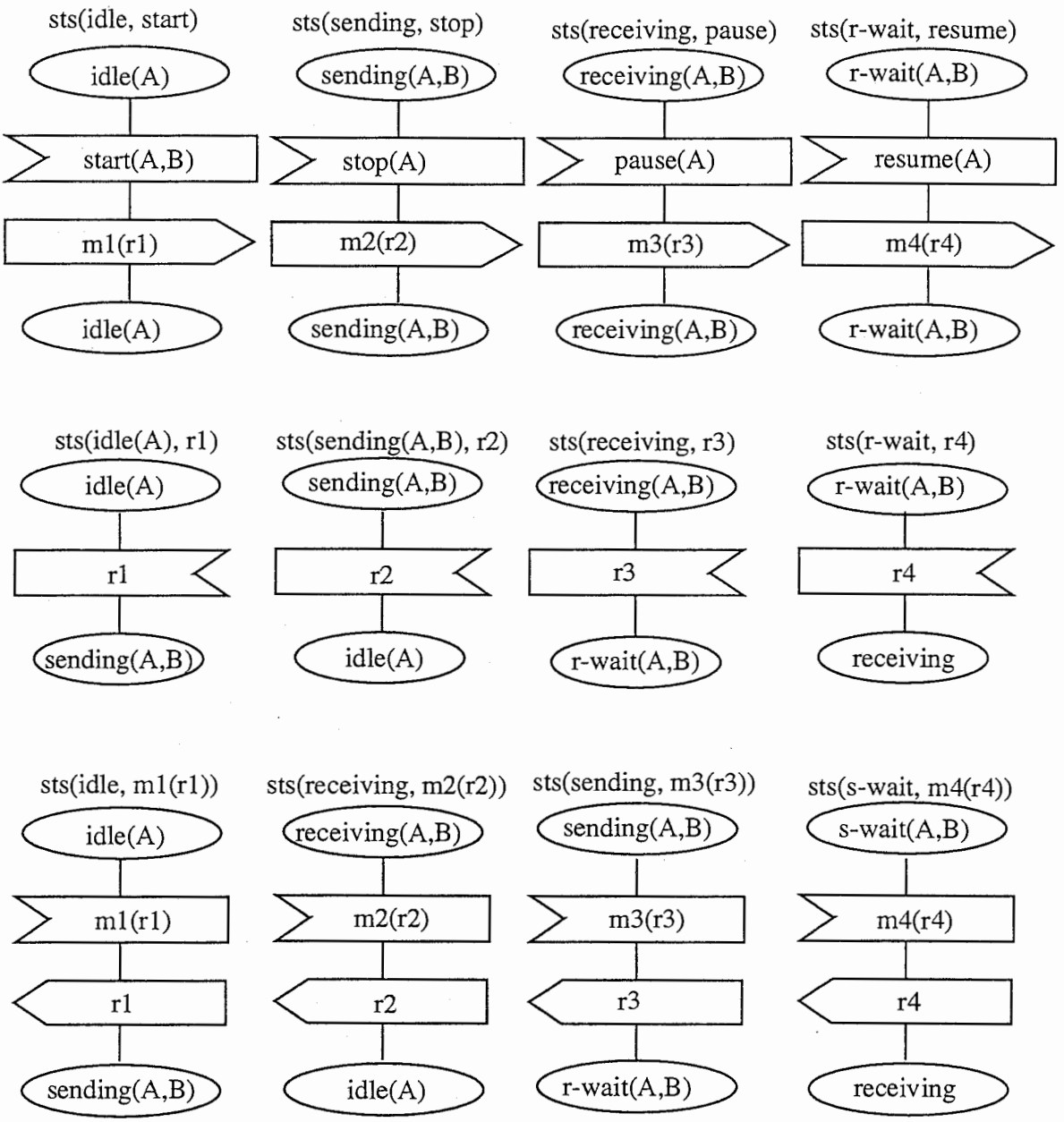

図 21: データ送信サービスに対する状態遷移片

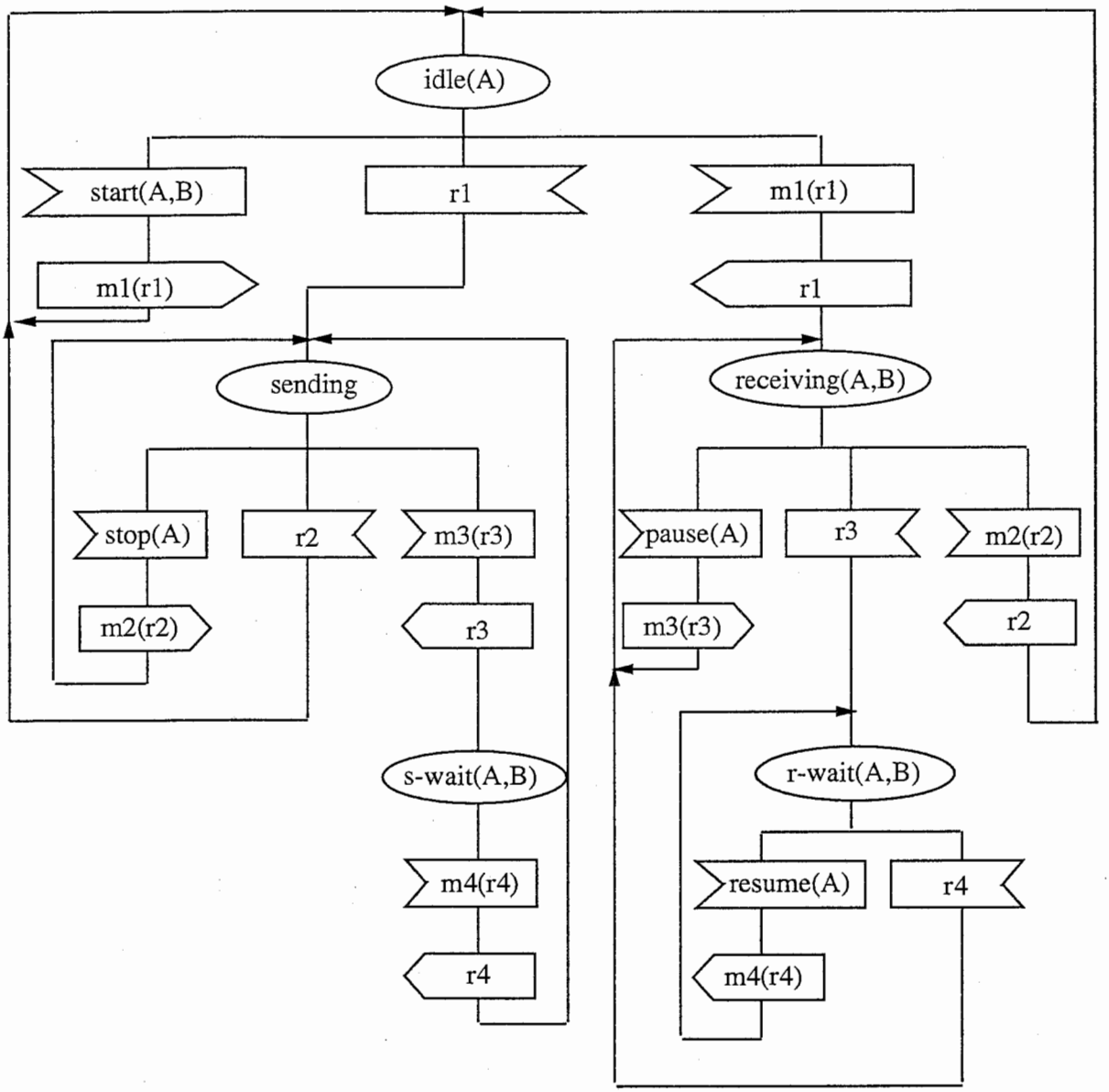

図 22: データ送信サービスを実現する合成プロトコル

9.1 詳細仕様記述言語 STR/D

STR/D[48] は STR で記述されたサービス仕様から合成されたプロトコル仕様をもとに，実際に通信システム上で通信サービスを実現可能なソフトウェア仕様に変換するための付加仕様を規定する言語である．STR/D 仕様は STR/D 規則の集合から構成される．一つの STR/D 規則は STR 規則で記述された状態遷移時に実行すべきタスクを規定する．一つの STR/D 規則は次のシンタクスで記述される：

位置指定 ｛タスク指定｝

この規則は位置指定の条件を満足する状態遷移時にタスク指定に記述されたタスクを実行することを表す．

9.1.1 位置指定

一つの合成プロトコルは，局所状態，入力とその他の要素から構成される．合成プロトコル上の位置を指定するために，局所状態と入力を使う．局所状態と入力の構成要素は，STR 規則における状態記述要素とイベントからなる．このことから，STR/D 規則は状態記述要素と入力が分かれば記述することが可能であると言える．STR/D 規則を個々の合成プロトコル仕様に依存せず，状態記述要素とイベントに関する知識だけを使って記述できるならば，STR/D 規則はこれらの STR 規則構成要素の意味を定義することになる．新しい状態記述要素やイベントを使ってサービスが記述されている場合には，それらの意味を規定する新たな STR/D 規則の追加が必要となる．以上の考察より，位置指定は状態と入力の組み合わせで記述する．

状態指定

状態を *initial*, *terminal*, *state*, *next-state*, *primitive*, *next-primitive* の 6 種類で指定する．

initial サービスの開始時点で実行すべきタスクを規定するための初期状態を指定する．図 23における位置 1 に対応する．

terminal 初期状態に戻る直前に実行すべきタスクを規定するための最終状態を指定する．図 23における位置 6 に対応する．

state 指定された状態で入力があった直後の位置を指定する．図 23における位置 2，3，4 が対応する．

next-state 指定された状態の直前の位置を指定る．図 23における位置 5 が対応する．

primitive 指定された状態記述要素をもつ状態での入力の直後の位置を指定する．図 23における位置 2，3，4 が対応する．*state* と *primitive* の相違は状態が完全に指定されるか部分的に指定されるかである．

next-primitive 指定された状態記述要素と一致する状態の直後の位置を指定する．図 23における，位置 5 が対応する．

これらの状態指定のうち，*state* と *primitive* は *next-state* と *next-primitive* との組合せが可能である．そのような組み合わせは図 23における位置 5 に対応する．

入力指定

入力指定は指定されたメッセージを受信した直後の位置を指定する．指定法として，*event*, *request*, *respond*, *norule* の 4 種類がある．入力メッセージの型のみが指定された場合には，その型のメッセージを受けとった直後の位置すべてを表す．*evet* に関しては，個別のイベント名を指定することが可能である．入力指定は状態指定との組み合わせが可能である．図 23における位置 2，3，4 は *input* により指定される．

状態遷移指定

状態遷移辺は現状態と次状態に含まれる状態記述要素の差分として指定する．シンタクスは次の通りである．

transition (< 状態遷移式 >)

この表現における，状態遷移式は次の二つのシンタクス（1），（2）により記述する，

（1） ＋ (状態記述要素の集合)
（2） － (状態記述要素の集合)

シンタクス（1）は現状態には含まれていないが，次状態には含まれている状態記述要素を規定する．シンタクス（2）は逆に現状態には含まれているが，次状態には含まれていない状態を規定する．図 23における位置 5 が *transition* で指定可能である．

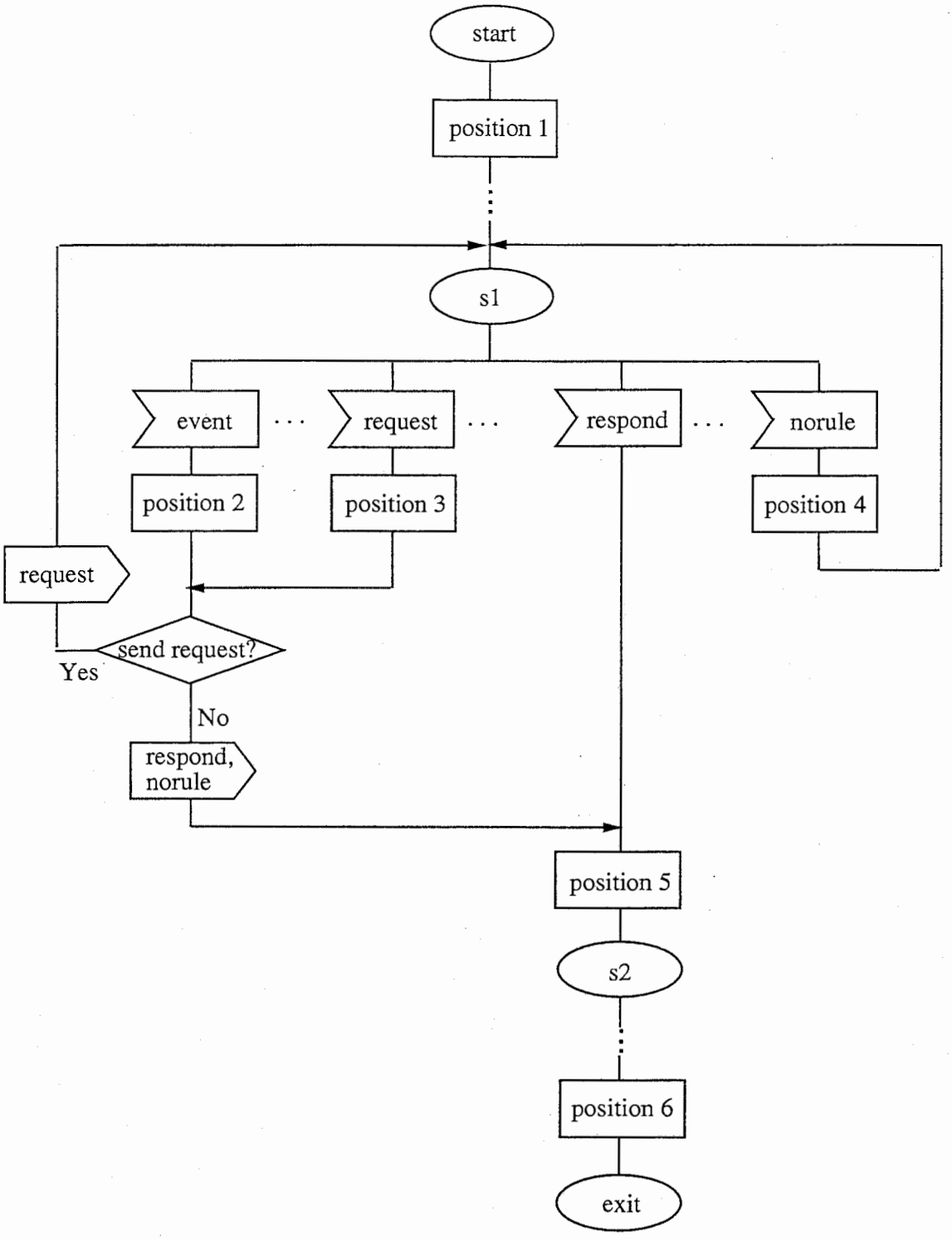
start
position 1
s1
event
request
respond
norule
position 2
position 3
position 4
request
send request?
Yes
No
respond,
norule
position 5
s2
position 6
exit

図 23: タスク挿入位置

9.1.2 タスク指定

“タスク指定” にはC言語の式で記述する．複数のタスクを一つの “タスク指定” に記述する場合には，タスクをセミコロンで区切って記述する．更に，タスク指定に条件文を記述することが可能である．条件文はタスクの実行結果により次に遷移する状態を変えることが可能である．条件文は次のシンタクスで記述する．

$$unless(< \text{C の式} >)\ \{< \text{C の文} >\}$$
$$< \text{状態指定} >;$$

この文は条件が満たされなかったときの仕様を規定する．Cの式が満たされなかったときには，この文に記述されているCの文が実行され，状態指定に記述された状態に遷移する．

10 PBXソフトウェア開発への適用

10.1 実験の説明

通信ソフトウェア開発は，既存のサービスに新たなサービスが追加される形が通常である．そこで，以下に述べるサービスPOTS (Plain Old Telephone Service), CCBS (Call Completion for Busy Subscriber), CW (Call Waiting), CFV(Call Forward Variable), TWC (Three Way Calling), UPT (Universal Personal Telecommunication), TCS (Terminated Call Screening) を順に追加した場合のサービス仕様，交換機上で動作させるために必要となる仕様及びプログラムの追加に関する量を求めた．はじめに，この実験で用いた7種類の通信サービスの機能を概説する．

POTS 二端末間の通常の通話である．

CCBS 発信者Aが電話番号を回した相手の端末Bが使用中のときに，CCBSサービスを行うための操作をAで行うと，相手端末Bが空き状態になったら，Aを呼びだし，Aが受話器をあげるとBを自動的に呼び出すサービスである．

CW 端末Bと通話中の端末Aに，他の端末Cから着信があったら，Aに着信があったことを伝え，フックスイッチの操作により，Aは通話相手，通話保留相手を交互に切替えることが可能である．

CFV 着信があったら，その呼を予め設定してある転送先に転送することが可能なサービス．転送先の変更を端末から行うことが可能である．

TWC 二者通話中に，フックスイッチをフラッシュすると，通話相手を保留状態にしてから，第三者を呼び出して三者通話に移行するサービス．

UPT ユーザがUPT番号と呼ばれる端末に固有ではない番号を持ち，あるユーザがUPT番号をどこかの端末に登録したら，その端末がそのユーザの電話となるサービスである．

TCS 時刻，発信番号により，起動されるサービスが変化する．ここでは，起動サービスとして，通常接続，メッセージ接続，転送を使用．

10.2 実験結果

表1に，各サービスを順に組み合わせて得られる7つのサービスS1，…，S7のSTR規則，交換機上で動作させるのに必要なSTR/D規則，STR/D規則の追加に当たり必要となった部品の個数を示す．括弧内の数値は，一つ前のサービスと比較して新たに追加されたものの数を表す．

上記サービスを実現するのに必要となった部品のC言語での行数は約2 Kである．部品はすべて人手で作成しなければならないが，作成量はそれほど多くはない．但し，部品は予めプラットフォームとして提供された通信システム制御のための基本部品を用いて作成した．

表2は，上で得られた追加部分についての割合を百分率で表したものである．各サービスをにおける通信回数，通信時間に関する複数の規則間における通信の共通化を行わない場合と比較した最適化率を，表3に示す．

10.3 評価

表2より，STR，STR/D，部品の平均追加率はそれぞれ３６％，２１％，７％である．部品は交換機のリソース制御を行う処理であるので，リソースの追加がない範囲では，新たな追加は不要あるいは非常に少ないと期待できる．また，STR/Dの追加に関しては，次に述べる自動化が今後の課題である．STRで使用されるプリミティブの意味を通信サービスを抽象化した通信サービスの概念モデルに対応させることにより，STR/D規則の半自動作成を行う．非専門家が通信サービスを開発する場合には，OS等のプラットホームに相当する部分は既に用意されてい

表 1: 規則と部品の数

サービス名	STR	STR/D	部品
S1(POTS)	23	14	10
S2(S1+CCBS)	46 (+23)	16 (+2)	10 (+0)
S3(S2+CW)	80 (+34)	20 (+4)	12 (+2)
S4(S3+CFV)	129 (+49)	26 (+6)	12 (+0)
S5(S4+TWC)	207 (+78)	26 (+0)	12 (+0)
S6(S5+UPT)	348 (+141)	72 (+46)	16 (+4)
S7(S6+TCS)	379 (+31)	76 (+4)	16 (+0)

表 2: 追加規則と部品の割合

追加サービス	STR(%)	STR/D(%)	部品 (%)
S1 → S2	50	13	0
S2 → S3	43	20	17
S3 → S4	38	23	0
S4 → S5	38	0	0
S5 → S6	41	64	25
S6 → S7	8	5	0
平均	36	21	7

表 3: 通信回数及び通信時間の最適化率

サービス名	通信回数比 (%)	通信時間比 (%)
S1	57	57
S2	53	49
S3	56	51
S4	43	37
S5	42	34
S6	36	28
S7	33	25
平均	46	40

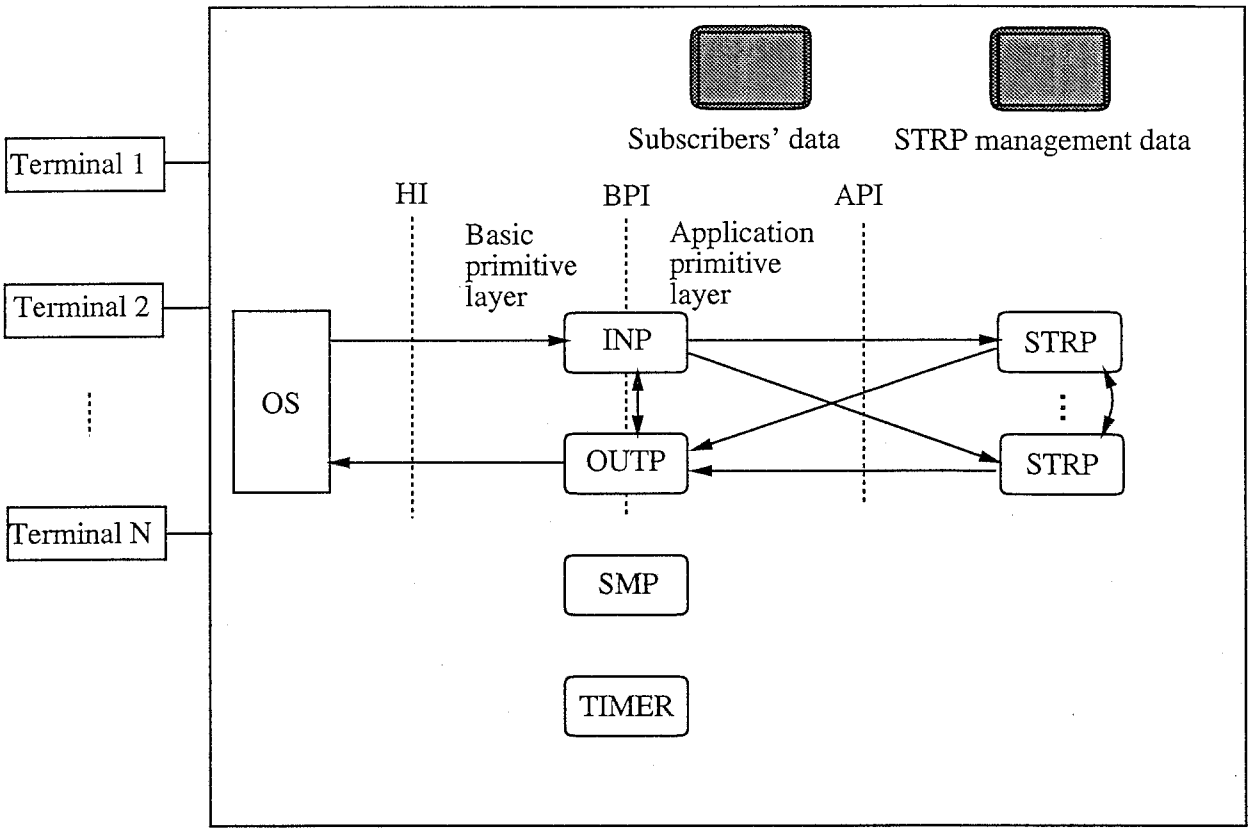

図 24: システムアーキテクチャ

ると仮定できる．従って，非専門家を対象とするサービスの開発においては，ほとんどが自動作成の対象となっている．

次に，自動作成ソフトウェアにおいて，本論文で述べる通信に関する最適化がどの程度効果があるかの評価を行う．表3より，後からサービスを追加するときほど，通信に関する回数及び時間ともに最適化の割合が大きくなる傾向がある．非専門家が通信ソフトウェアを開発する場合には，ある程度サービスがそろっている状態を仮定できるので，通信回数，通信時間は素朴に一つづつ規則の適用判定を行う場合と比較して，それぞれ３３％，２５％以下に押えられると期待できる．

10.4 ソフトウェアアーキテクチャ

自動作成されたプログラムを用いて通信サービスを制御するソフトウェアアーキテクチャを示す．このアーキテクチャに基づき二つのPBX上に自動作成ソフトウェアを用いた通信サービス動作環境を開発した．インタフェースは，対象ハードウェア依存部分をなるべく少なくすることを目的としている．このインタフェースは，複数ノードからなる分散システムに容易に適用可能である．

図24に示す5種類のプロセスSTRP，INP，OUTP，SMP，TIMERを設定した．以下にこれらのプロセスとその役割を説明する．

STRP は，STR と STR/D から自動的に作成されるプログラムである．一つの STRP プロセスは，発呼があったとき，あるいは空き端末に着信があったときに起動される．

INP は，端末からの入力を STR 規則に記述されているイベントに変換する．INP は，OS を通した端末からの入力をバッファリングし，変換を行う．また STRP プロセスを起動する機能を有する．一つのイベントは複数の入力列から作成されるか，一つの入力から複数のイベントが作成されるかのいずれかである．すべての入力は INP を通るので，INP はセマフォを用いた規則適用の排他制御機能を有する．

OUTP は，STRP からの出力を PBX 機器を制御する PBX 制御プリミティブに変換する．OUTP は OS への出

表 4: 性能測定

	実行時間	C ダイナミックステップ
STRP/MMP	1.1	1.2

力を変換し，バッファリングする機能を有する．

SMP は，イベントとプロセス間通信を記録する．

TIMER はタイマである．

INP と OUT は二階層からなり，新規インタフェースを容易に作成可能とするためのカスタム化用基本プリミティブ階層と STRP プロセスの論理インタフェースを提供するためのアプリケーションプリミティブ階層からなる．

10.5 論理インタフェース

PBX 制御プリミティブに関して，基本プリミティブ階層とアプリケーションプリミティブ階層の二階層がある．これらは INP と OUTP 内部に実現されている．OS と INP または OUTP との間のインタフェースをハードウェアインタフェース (HI) と呼ぶ．基本プリミティブとアプリケーションプリミティブの間のインタフェースを BPI，アプリケーションプリミティブと STRP との間のインタフェースを API と呼ぶ．アプリケーションプリミティブは基本プリミティブから作成される．BPI は特定のハードウェアには依存しない．従って，ハードウェア知識を使わずに BPI を使って API のカスタマイズが可能である．API と BPI の大きな差は端末ん特定法である．BPI では端末は物理アドレスで表現され，API では論理アドレス（電話番号）で端末の識別を行う．

基本プリミティブインタフェース

基本プリミティブインタフェースは，ハードウェア制御プリミティブを使って新しいイベントと新しいタスク追加するために設定してある．インタフェースは次の特徴を有する．ハードウェアあるいはタイマからの入力は基本的には一つのイベントに対応する．電話番号を特定するダイヤルについては，端末を特定する番号のときに意味をもつ．STR においてイベントは実際の入力ではなく論理的な入力として定義される．

アプリケーションプリミティブインタフェース

API は STR と STR/D で使われているイベントとタスクのインタフェースである．アプリケーションプリミティブインタフェースは基本プリミティブインタフェースを用いて定義される．

10.6 結果

自動作成されたプログラムを用いて開発された PBX を使って実行時間の計測を行った．計測を行ったプログラムは STRP と人手で作成した STRP より最適化された MMP である．STRP と MMP の実行時間の比較には，簡単化した POTS を用いた．STRP と MMP は同じアーキテクチャを用いて実行した．各端末は一つのプロセスにより制御され，プロセスは互いに他プロセスの状態を知るために通信を行う．MMP は POTS を制御するのに専用化したプログラムである．表 4に計測結果を示す．実行時間はプロセスが消費した時間を表し，C ダイナミックステップは C ソースプログラムのステップ数を表す．性能については人手で作成したプログラムに比較して 10 %の劣化が見られた．これは今後のプロセッサの高速化を考えると実用上問題にならない．

11 機能モデルに適合するソフトウェア作成

今までに述べてきたプロトコル合成は階層型アーキテクチャを仮定している．ユニバーサルパーソナル通信 [49] は図 5に示す機能モデルを用いて提供されることが標準化により決定している．機能モデルにおいては機能が階層化されているのではなく分散配置されている．この機能は機能エンティティに分散配置されている，本章では，この機能モデルに適合するソフトウェア作成法を示す [50, 51].

11.1 段階的詳細化

STR 規則と STR/D 規則から任意の機能モデルに適合するソフトウェア作成法を示す．はじめに，STR 規則と STR/D 規則から各機能エンティティ毎のサービス仕様を求める．次に，得られた機能エンティティのサービス仕様

から状態遷移機械を合成する．この手法では中間言語としてSTR(L)とSTR/D(L)を用いる．これらはSTR，STR/Dと同一シンタクスを有するが，一つの機能エンティティの局所的な仕様を記述する．

機能エンティティのソフトウェア仕様作成は次の段階からなる：

ステップ1 STR規則に現れるイベントを実際に生起する機能エンティティに対応づける．

ステップ2 元のSTR，STR/D規則と機能エンティティへのイベント割り付けからSTR(L)とSTR/D(L)で記述された仕様を求める．

はじめに，STR規則に記述されている現状態，イベント，次状態とSTR/D規則における位置指定とを比較する．STR規則の条件がSTR/D規則の位置指定と一致する場合には，これらの規則を結合して新たなグローバルな状態遷移規則を作る．得られるグローバル状態遷移規則は，変更された現状態，イベント，タスク指定，変更された次状態からなる．

次に，得られたグローバル状態遷移規則をタスク指定に含まれる各タスクを実行するのに使われる機能エンティティに対するローカルな状態遷移規則に分割する．但し，この分割により得られるローカルな状態遷移規則における状態は元のSTRとSTR/D規則にあるグローバルな状態のまま記述しておく．この分割において，通信に関する動作は，送信動作と受信動作の二種類に分類される．ステップ1である機能エンティティに割り付けられたイベントは，その機能エンティティの仕様に記述される．

最後に，得られたローカルな状態遷移規則からSTR(L)とSTR/D(L)規則を作成する．この作成において，送信動作は対応する機能エンティティに対するSTR/D(L)規則を生成し，受信動作は対応する機能エンティティに対するSTR(L)規則を生成する．

ステップ3 機能エンティティ仕様の生成を行う．

得られた書く機能エンティティ毎のSTR(L)とSTR/D(L)規則から，FSM形式の仕様が階層型アーキテクチャに対するプロトコル合成法で用いられたプロセス仕様の合成法で用いられたのと同じ方法で合成される．

この機能エンティティ仕様の生成は基本的な送信と受信動作が機能エンティティで実行されるタスクから抽出可能な場合に適用可能である．

11.2 ユニバーサルパーソナル通信

UPT(ユニバーサルパーソナル通信)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，着信の通信を行うことができるサービスである．UPTユーザは，ユーザ毎に固有のUPT番号を持っており，移動先の端末から通信を行うに当たり，UPT番号と認証コードによる利用の正当性を確認するための認証を行う．ユーザの認証終了後，UPTサービスを使うことが可能となる．

UPTサービスは図5により実現される．図reffig:dfmにおいて各機能エンティティは次の意味を持つ．

FE1 発CCAF
FE2 SSFに関係する発CCF
FE3 中継CCF
FE4 着CCF
FE5 着CCAF
FE6 SCF
FE7 SDF(l) (行った先のネットワークにおけるSDF)
FE8 SRF
FE9 SDF(h) (ホームネットワークのSDF)

ここで，略語は次の機能を表す．

SSF 交換機能 (Service Switching Function)
SRF 特殊リソース (Specialized Resource Function)
CCF 呼制御 (Call Control Function)
CCAF 呼制御エージェント機能 (Call Control Agent Function)
SCF サービス制御機能 (Service Control Function)
SDF サービスデータ機能 (Service Data Function)

次に，UPTユーザがUPTサービスにアクセスし，ユーザの特定と認証を行う部分についての手続きの概略を示す．

1. UPT ユーザがアクセスコードを入力する．

2. アクセスコードを認識し，CCF における呼制御を中断するとともに，SRF に一時的な接続を行う．

3. プロンプトを出し，UPT 番号の入力後，応答を返す．

4. プロンプトを出し，認証コードの入力後，応答を返す．

5. UPT ユーザのサービス提供者が認証検査を行い，結果を返す．

6. 次の判断を行う．

   - 認証が成功ならば，その後で行う処理手続き特定に進む．
   - 失敗の場合には，3 回を上限に再試行を許す．
   - 3 回失敗したら，その呼を解放する．

UPT サービスの認証を行う部分のシーケンスを図 25に示す．この他に，誤った認証コードを入力した場合に再試行するシーケンス，認証の失敗回数が規定回数を超えた場合に認証を中止するシーケンスがある．

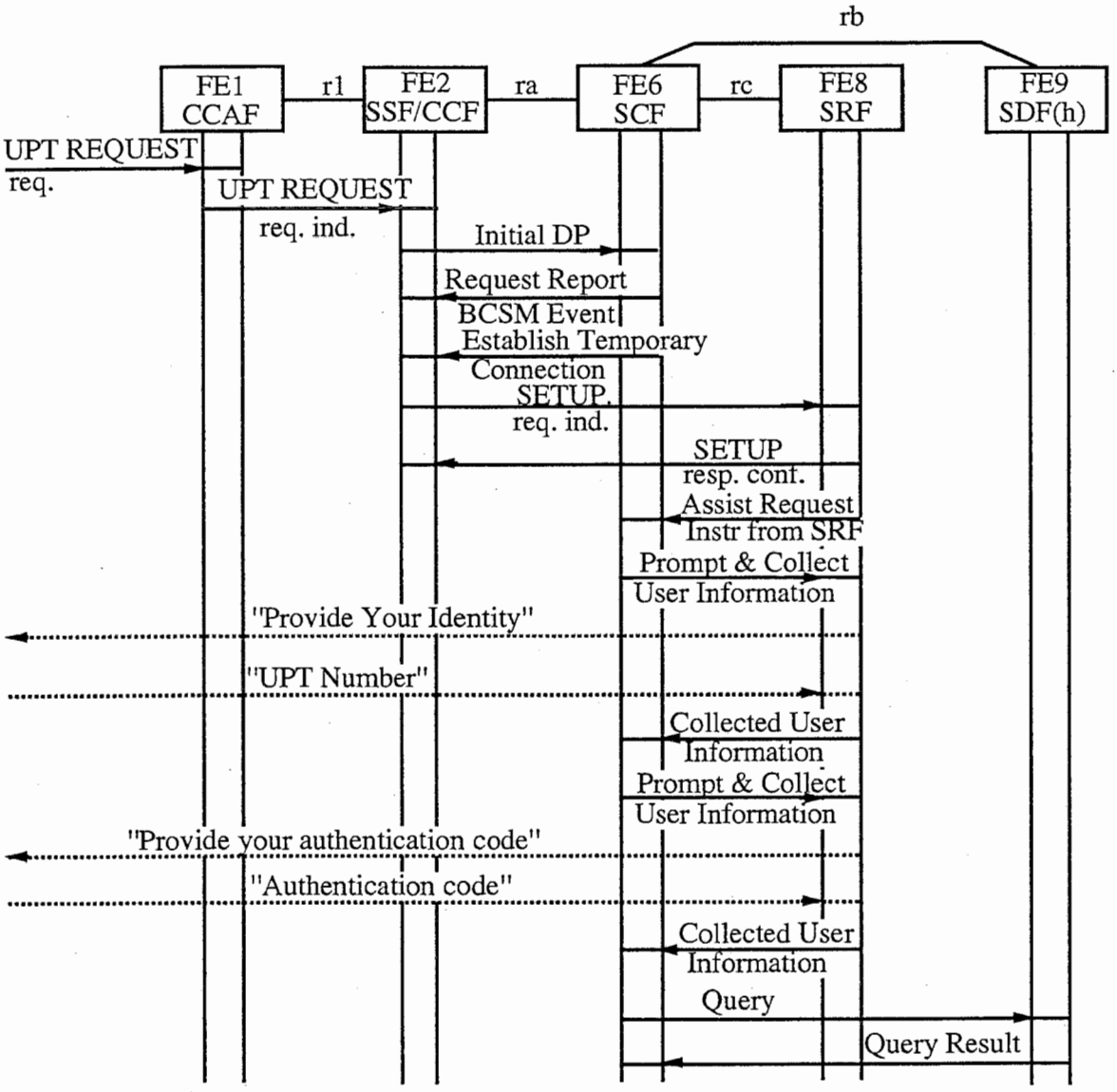

図 25: ユーザ認証シーケンス

Terminals A;
Users U, V;

r1) dial-tone(A)	uptreq(A):	ident(A).
r2) ident(A)	idnumber(A,U):	auth(A,U).
r3) auth(A,U)	acode(A,U):	success(A,U).
r4) auth(A,U)	acode(A,V):	ident(A).
r5) auth(A,U), m_limit(A)	acode(A,V):	fail(A).

図 26: ユーザ認証の STR 記述

11.3 STR Description of UPT

上記の UPT サービスにおけるユーザ認証シーケンスを
STR で記述した例を図 26に示す．Terminals で宣言された変数 A は端末を表し，Users で宣言された変数 U，V は，その変数で表される UPT 番号をもつ UPT ユーザを表す．

STR 規則 **r1**, **r2**, **r3** が図 25のシーケンスに対応する STR 記述である．**r4** は誤った認証コードを入力したために再試行するシーケンスに対応する STR 記述である．**r5** は認証の失敗回数が規定回数を超えた場合に認証を中止するシーケンスに対応する STR 記述である．

図 26で用いられている状態記述要素，イベントの意味を述べる．

状態記述要素

dialtone(A) UPT サービスの開始要求を受け付ける状態．

ident(A) UPT 番号を受け付ける状態．

auth(A,U) 端末 A で入力された UPT 番号 U に対する認証コードを受け付ける状態．

success(A) 認証が成功した状態．

fail(A) 認証の再試行が制限回数に達したため，認証が失敗した状態．

m_upt(A,U,C) 端末 A から入力された UPT 番号 U に対して認証コード C が正しい場合に成立する条件判定．not(m_upt(A
は m_upt(A,U,C) の否定条件を表す．m_で始まる状態記述要素が現状態に記述されている場合は，イベント
が発生した直後に，その条件判定が実行される．従って，not(m_upt(A,U,C)) は
acode(A,C) により入力された認証コード C が UPT 番号 U に対する認証コードでないときに成立する．

m_inlimit(A) 端末 A からの失敗が制限回数以内の場合に成立する条件判定．

イベント

uptreq(A) UPT サービスの開始要求．

number(A,U) UPT 番号の入力を表すイベントであり，U が入力された番号を表す．

acode(A,C) 認証コードの入力を表すイベントであり，C が入力されたコードを表す．

11.4 STR/D 記述

図 25の STR 記述を図 5の機能モデル上で実現するために必要な詳細仕様を図 27に示す．図の宣言部において，STR 記述で用いた宣言に加えて，図 5で定義されている機能エンティティを Entity で宣言する．タスク指定に記述されている個別のタスクは予め部品として用意しておく．これらの部品は標準化により動作が決まっている．

タスクの引数のうち，Entity 宣言で宣言された引数が表すエンティティ間では，そのタスクを実行するときに通信があることを表す．一つのタスク指定内に複数のタスクが記述されている場合には，記述された順にタスクを実行する．この結果，一つのタスク指定内に記述されるタスクにおいて行われる通信の順番は保証される．

```
Terminal A;
User U;
Entity CCAF, SSF/CCF, SCF, SRF;

transition(-(dial-tone(A)) +(ident(A)))
{ UptReq(CCAF, SSF/CCF);
  InitialDP(SSF/CCF, SCF);
  ReqReport(SCF, SSF/CCF);
  TempConnect(SCF, SSF/SCF);
  SetupReqInd(SSF/CCF, SRF);
  SetupRespConf(SRF, SCF);
  AssistReq(SRF, SCF);
  PromptCollect(SCF, SRF, "Provide your identity"); }
input(event number(A, U))
{ CollectedUserInf(SRF, SCF);
  PromptCollect(SCF, SRF, "Provide your authentication code"); }
transition(-(auth(A, U)) +(success(A)))
{ CollectedUserInf(SRF, SCF); }
transition(-(auth(A, U)) +(ident(A)))
{ PromptCollect("Wrong authentication, please retry. Provide your authentication code"); }
transition(-(auth(A, U)) +(fail(A)))
{ PromptCollect("Retry limit exceeded. Your line is now blocked. Please hang up."); }
```

図 27: ユーザ認証の STR/D 記述

11.5 UPT サービス仕様に対する段階的詳細化

図 25，図 27に記述された仕様に対して，上記プロセス仕様生成法を適用し，機能エンティティの STR 記述，STR/D 記述を導出する．

ステップ 1 図 25の STR 規則に現れるイベントを実際に生起する機能エンティティに対応づける．

```
uptreq(A)       :  CCAF
idnumber(A,U)   :  SRF
acode(A,C)      :  SRF
```

ステップ 2 ステップ 1 での割り当てに従って新しく STR(L) と STR/D(L) 規則を生成する．"s1(A)" と "s2(A)" は新しく生成された状態記述要素を表す．次の規則は SCF に対する STR 規則と STR/D 規則である．生成された各 STR 規則は一つの機能エンティティのローカルな状態遷移を規定する．

```
dial-tone(A) InitialDP(A): s1(A).
s1(A) AssistReq(A): s2(A).
s2(A) CollectedUserInf(A): ident(A).

transition(-(dial-tone(A)) +(s1(A)))
{ send(SSF/CCF,ReqReport);
  send(SSF/CCF,TempConnect); }
transition(-(s1(A)) +(s2(A)))
{ send(SRF,PromptCollect,
      "Provide your identity"); }
transition(-(s2(A)) +(ident(A)))
{ send(SRF,PromptCollect,
      "Provide your authentication code"); }
```

ステップ 3 最後に，エンティティ別の STR 規則，STR/D 規則から各エンティティ毎の制御プログラムを自動生成する．

12 評価と今後の課題

STRで記述された任意のサービス仕様からプロトコルを合成し，ソフトウェア仕様へ変換する手法を示した．サービス仕様からのプロトコル合成は完全に自動的に行うことが可能である．合成プロトコルからソフトウェア仕様への変換には，予め知識として専門家により記述された詳細化仕様を用いる．詳細化仕様の自動作成については，通信サービスで用いるリソースを論理化することにより自動化する方式を考案している [52]．機能モデルに適合するソフトウェア仕様の生成については，STRで使われているイベントとプリミティブの機能モデル上での意味を与えるだけでよい手法が別途考案されている [53]．

10章で使用したサービス仕様を順に追加して得られるサービス仕様のうち，実際にソフトウェア仕様まで変換できるのはPOTSにCCBSを追加して得られるサービス仕様までである．これより大きいサイズのサービス仕様は計算機のメモリの制約からソフトウェア仕様に変換することができない．これは次の理由による．生成される状態一つの記憶に４，０００バイト程度使用している．従って，１６０MBのメモリをもつ計算機上で記憶できる状態数は４０，０００個程度である．現在のプロトコルの合成方式では，メモリ状態を表現する状態記述要素がある状態で実際に真になる偽になるかに関してすべての組み合わせを考えて中間状態を生成している．すなわち，メモリ状態を表現する状態記述要素が１０個一つのサービスで使われている場合には，そのサービスから生成されるプロトコルの中間生成物を計算機上で表現することができない．この制約は，中間状態を生成するたびに，最終的に出力するプロトコルの状態の表現に変換することで解消することが可能である．この最適化は今後の課題である．

参考文献

[1] Hirakawa, Y. and Takenaka, T., "Telecommunication Service Description Using State Transition Rules", in *Proc. Sixth Int. Workshop on Software Specification and Design* (Como, Italy), pp. 140–147, Oct. 1991.

[2] Ohta, T., Takami, K. and Takura, A., "Acquisition of Service Specifications in Two Stages and Detection/Resolution of Feature Interactions," in *The Fourth Telecommunications Information Networking Architecture Workshop* (L'Aquila, Italy), vol. 2, pp. II-173–II-187, Sep. 1993.

[3] Boehm, B. W., "Software Engineering Economics," *IEEE Trans. Software Eng.*, vol. SE-10, no. 1, Jan. 1984.

[4] Reubenstein, H. B. and Waters, R. C., "The Requirements Apprentice: Automated Assistance for Requirements Acquisition," *IEEE Trans. Software Eng.*, vol. SE-17, no. 3, Mar. 1991.

[5] Jarke, M., Bubenko, J., Rolland, C., Sutcliffe, A. and Vassilou, Y., "Theories Underlying Requirements Engineering: An Overview of NATURE at Genesis," *IEEE Int. Symp. on Requirements Engineering* (San Diego, California), pp. 19–31, Jan. 1992.

[6] Rolland, C. and Proix, C., "A Natural Language Approach for Requirements Engineering," *4th Int. CAiSE Conference*, vol. 593, pp. 257–277, 1992.

[7] Tsai, J. J. P., Weigert, T. and Jang, H.-C., "A Hybrid Knowledge Representation as a Basis of Requirement Specification and Specification Analysis," *IEEE Trans. on Software Eng.*, vol. 18, No. 12, pp. 1076–1100, Dec. 1992.

[8] Puncello, P. P., Torrigiani, P., Pietri, F., Burlon, R., Cardile, B. and Conti, M., "ASPIS: A Knowledge-Based CASE Environment," *IEEE Software*, pp. 58–65, Mar. 1988.

[9] Kelly, V. E. and Nonnenmann, U., "Reducing the Complexity of Formal Specification Acquisition, In Automating Software Design," *Automating Software Design*, ed. Lowry, M. R. and McCartney, R. D., AAAI Press, pp. 41–64, 1991.

[10] Wrobel, S., "Design Goals for Sloppy Modelling Systems," *Int. J. Man-Machine Studies*, pp. 461–477, vol. 29, no. 4, pp. 461 - 482, Oct. 1988.

[11] W. Lewis Johnson, Martin S. Feather, and David R. Harris, "Representation and Presentation of Requirements Knowledge," IEEE Trans. on Software Engineering, vol. 18, no. 10, Oct. 1992.

[12] Probert, R. L. and Saleh, K., "Synthesis of Communication Protocols: Survey and Assessment," *IEEE Trans. Comput.*, vol. 40, no. 4, pp. 468–476, Apr. 1991.

[13] Ichikawa, H. and Takami, K., "Automatic Generation of Communications Software," *The Journal of IEICE*, pp. 522–530, vol. 77, no. 5, May 1994.

[14] Merlin, P. and Bochmann, G. V., "On the Construction of Submodule Specifications and Communication Protocols," *ACM Trans. Programming Languages and Syst.*, vol. 5, no. 1, pp. 1–25, Jan. 1983.

[15] Zafiropulo, P., West, C. H., Rudin, H., Cowan, D. D., and Brand, D., "Towards Analyzing and Synthesizing Protocols," *IEEE Trans. Commun*, vol. Com-28, no. 4, pp. 651–661, Apr. 1980.

[16] Sidhu, D. P., "Protocol Design Rules," in *Protocol Specification, Testing, and Verification*, pp. 283–300, 1982.

[17] Kakuda, Y. and Wakahara, Y., "Component-Based Synthesis of Protocols for Unlimited Number of Processes," in *Proc. IEEE COMPSAC'87*, pp. 721–730, 1987.

[18] Takura, A. and Ichikawa, H., "Completing Protocols According to Requirements," in *Proc. ICCC Symp.* (Beijing, China), pp. 116–119, Sep. 1989.

[19] Takura, A. and Kanai, A., "Completing Protocols Synthesized from Service Specifications," submitted to *IEICE Trans. Commun.*.

[20] Ichikawa, H., Itoh, M., Kato, J., Takura, A., and Shibasaki, M., "SDE: Incremental Specification and Development of Communications Software," *IEEE Trans. Comput.*, pp. 553–561, Apr. 1991.

[21] Genji, K., Takami, K. and Takenaka, T., "Telecommunication Service Design Support System Using Message Sequence Rules," *IEICE Trans. Commun.*, vol. E75-B, no. 8, pp. 723–732, Aug. 1992.

[22] Bochmann, G. V. and Gotzhein, R., "Deriving protocol specifications from service specifications," in *Communications, Architectures & Protocols, Proc. of the ACM SIGCOMM '86 Symp.*, pp. 148–156, 1986.

[23] Gotzhein, R. and Bochmann, G.V., "Deriving Protocol Specifications from Service Specifications Including Parameters," *ACM Trans. Computer Syst.*, vol. 8, No. 4, pp. 255–283, Nov. 1990.

[24] Chu, P. M. and Liu, M. T., "Synthesizing Protocol Specifications from Service Specifications in FSM Model," *Proc. Comput. Networking Symp.*, pp. 173–182, Apr. 1988.

[25] Saleh, K. and Probert, R., "A Service-Based Method for The Synthesis of Communications Protocols," *Int. J. Mini and Microcomputers*, vol. 12, no. 3, pp. 97–103, 1990.

[26] Kakuda, Y., Nakamura, M. and Kikuno, T., "Automated synthesis of protocol specifications from service specifications with parallelly executable multiple primitives," *IEICE Trans. Fundamentals*, vol. E77-A, no. 10, pp. 1634–1645, Oct. 1994.

[27] Garey, M. R. and Johnson, D. S., "Computers and Intractability, A Guide to the Theory of NP-Completeness," NY: W. H. Freeman and Co., 1979.

[28] Takura, A., Kawata, K., Ohta, T. and Terashima, N., "Communication Software Generation Based on Two-Layered Specifications and Execution Environment," in *IEEE GLOBECOM'93* (Houston, USA), pp. 362–368, Dec. 1993.

[29] Cameron, E. J., Cohen, D. M., Guithner, T. M., Keese Jr., W. M., Ness, L. A., Norman, C. and Srinidhi, H. N., "The L.0 Language and Environment for Protocol Simulation and Prototyping," *IEEE Trans. Comput.*, vol. 40, no. 4, pp. 562–571, Apr. 1991.

[30] Dendorfer, C. and Weber, R., "From service specification to protocol entity implementation - An exercise in formal protocol development," in *Protocol Specification, Testing and Verification, XII*pp. 163–177, 1992.

[31] Takura, A. and Ohta, T, "Two-Layered Communications Service Specification Description and Program Specification Generation," *Trans. IPS Japan*, vol. 35, no. 5, pp. 1104–1113. May 1995.

[32] Færgemand, O. and Olsen, A., "Introduction to SDL-92," *Computer Networks and ISDN Systems* vol. 29, no. 9, pp. 1143–1167, May 1994.

[33] Bolognesi, T. and Brinksma, E., "Introduction to the ISO Specification Language LOTOS," *Computer Networks and ISDN Systems* vol. 24, pp. 25–59, 1987.

[34] Belina, F. and Hogrefe, D., "The CCITT-Specification and Description Language SDL," *Computer Networks and ISDN Systems*, vol. 16, no. 4, pp. 311–341, 1989.

[35] Faci, M., Logrippo, L. and Stépien, B., "Formal Specification of Telephone Systems in LOTOS: the Constraint-Oriented Style Approach," *Computer Networks and ISDN Systems*, vol. 21, pp. 53–67, 1991.

[36] Drayton, L., Chetwynd, A. and Blair, G., "Introduction to LOTOS through a Worked Example," *computer communications*, vol. 15, no. 2, pp. 71–85, Mar. 1992.

[37] Takura, A., Sera, T., Ohta, T., "Protocol Synthesis from Service Specifications Described by Graph Rewriting Rules," *RIMS workshop*, Jul. 1995.

[38] Takura, Ohta, T. and Kawata, K., "Process Specification Generation from Communications Service Specifications," *Automated Software Eng.*, no. 2, pp. 167–182, 1995.

[39] Liu, M. T., "Protocol Engineering," it Advances in Computers, vol. 29, pp. 79–195, 1989.

[40] Nakamura, M., Takura, A. and Ohta, T., "A Method for Requirements Elicitation Using a Domain Model," in *Task Force on the Engineering of Computer Based Systems*, May 1994.

[41] Ohta, T. and Takura, A., "The Automated Acquisition of Requirements Specifications for Communications Software," in *Proc. of Seventh Int. Conf. Computing and Information (ICCI '95)*, pp. 1009–1021, 1995.

[42] Takura, A, Ueda, Y., Haizuka, T and Ohta, T., "Requirements Acquisition of Communications Services," to appear in *International Communications Conference (ICC '96)*, Jun. 1996.

[43] Takami, K., Harada, Y., Ohta, T. and Terashima, N., "A Visual Design Support System for Telecommunications Service," in *IEEE Phoenix Conf. Computers and Communications*, pp. 593–599, Mar. 1993.

[44] Harada, Y., Hirakawa, Y., Takenaka, T. and Terashima, N., "A Conflict Detection Support Method for Telecommunication Service Descriptions", IEICE Trans. Commun., vol. E75-B, no. 10, pp. 986–997, Oct. 1992.

[45] Ueda, Y., Takura, A. and Ohta, T., "A Verification method of Communications System Service Specifications," *Technical Report of IEICE*, KBSE94-44, pp. 25–32, Nov. 1994.

[46] Shibata, K., Hirakawa, Y., Takura, A. and Ohta, T., "Reachability Analysis for Specified Processes in a Behavior Description," *IEICE Trans. Commun.*, pp. 1373–1380, vol. E76-B, no. 11, Nov. 1993.

[47] Sato, M., "Reachability Analysis for Communication Service Specification Descriptions in Global State Rules," *IEICE Trans. Commun.*, pp. 245–251, vol. J78-B-I, Jun. 1995.

[48] Takura, A., Ohta, T. and Kawata, K., "Task Generation Mechanisms in a Communication Software Generation Systems," in *1993 Asia-Pacific Conf. on Communications*, 2E.2, Aug. 1993.

[49] CCITT, "Draft Recommendation F.851, Universal Personal Telecommunication (UPT) - Service Description," Oct. 1992.

[50] Takura, A. and Ohta, T., "Stepwise Telecommunication Software Generation from Service Specifications in State Transition Model," in *1993 Int. Conf. Network Protocols* (Boston, USA), pp. 135–142, Oct. 1994.

[51] Takura, A. and Ohta, T., "Stepwise Refinement of Communications Service Specifications for Conforming to a Functional Model," *IEICE Trans. on Communications*, vol. E77-B, no. 11, pp. 1322–13 31, Nov. 1994.

[52] Sera, T. and Takura, A, "Task Generation for Distributed Functional Model," to appear in *Int'l J. on Artificial Intelligence Tools.*

[53] Sera, T, Takura, A. and Ohta, T., "Architecture Refinement for a Distributed Functional Model," in *Proc. of the IASTED Int. Conf.* (Orland, Florida), pp. 173 – 176, Jan. 1996.