〔公　開〕

ＴＲ－Ｃ－０１１７

ドメインモデルによる要求理解について

灰塚　凡樹
Tsuneki　HAIZUKA

１９９５　４．７

ＡＴＲ通信システム研究所

テクニカルレポート：

ドメインモデルによる要求理解について

平成　7年　4月　7日

灰　塚　　凡　樹

目　　次

1．まえがき

本稿では、ソフトウェア要求仕様に既存の要求仕様になかった、新たな要求が含まれていても、ソフトウェアが正しく動作できる要求仕様を自動的に獲得する手法を提案する。

欠落情報や曖昧性を含む要求仕様からソフトウェアが正しく動作できる要求仕様を獲得する事を要求理解という。しかし、正しい要求仕様を記述する事は大変難しい。ソフトウェアを必要とする人、即ち、ユーザが頭に描く要求をソフトウェア開発担当者に正確に伝え、仕様とする事は両者の協調作業が必要となり、認知科学の観点からも大変難しい事が知られている。また、ユーザ自身に要求仕様を記述させたとしても、要求の欠落のない、矛盾のない正しい仕様を記述する事を期待する事はできない。更に、ソフトウェア開発の専門家ですら、完全な仕様を記述できる訳ではない。

これまでの要求理解の研究においては、エンドユーザ（以降は非専門家と記す）と要求分析者の間のコミュニケーションを支援する事により、非専門家の曖昧な要求から正確な仕様を抽出する方法を提案したものがあるが、両者の間の認識のズレを完全に無くす事は困難であり、結果として得られる要求仕様が完全なものである事を期待する事は無理がある。また、既存のソフトウェア要求仕様の組合せによる要求仕様の生成を試みた例もあるが、新規に開発するソフトウェアの要求仕様が入力の場合、既存の要求仕様の組合せにより、正しい要求仕様を構成できるという保証はできない。

一般に、ソフトウェアの開発工程が後になる程、バグの修正に要する手間が大きくなる事が知られているが、要求仕様が正しいものであるならば、この要求仕様に基づき、自動的にソフトウェアを生成する事により、正しく動作するソフトウェアを開発する事は可能となる。

近年、通信サービスの分野においては、通信サービスの社会に対する役割は増大しつづけており、これに伴い今後益々、多種多様な通信サービスの早期な開発が必要となってくる。これは、専門開発要員の不足が生じる事を予想させる。そこで、我々は非専門家が記述した要求仕様を対象とした要求理解を実現する事により、非専門家の通信サービス開発への参画を可能にする事、及び、信頼性の高いソフトウェアの自動生成を効率良く実現する事を目的として、研究している。

我々の研究課題は多々あるが、ここでは、最も重要な課題である要求理解、即ち、ユーザの不完全な要求仕様から自動的に正確な要求仕様を生成する方法について示している。

我々が考案した方法は、ソフトウェアが実現すべき機能を適用する実世界を1つの領域として、その領域固有の概念に基づき抽象化した世界（ドメインモデル）とユーザの要求仕様を対照し、正しい要求となる様に補完する事を特徴としている。

2．要求理解について

2．1　要求理解の定義

欠落情報や曖昧性を含む要求仕様からソフトウェアが正しく動作できる要求仕様を獲得する事を要求理解という。

一般に、ソフトウェアを必要とする人、即ち、ユーザが望む通りの仕様を記述する事は、ユーザ自身が明確な要求を持っていない場合が多いため、ユーザ自身が正しい要求仕様を記述する事は大変難しい。そこで、正しい要求仕様を得るための対策として、要求分析者により、ユーザの要求を分析して正しい要求仕様を得る方法が考えられるが、要求分析者に要求を正確に伝え、仕様を作成させる事は両者の協調作業が必要となる。これは、認知科学の観点から、ユーザと要求分析者の理解が同じ立場、同じレベルである事は大変困難である事が知られている。そもそも、ソフトウェア開発の専門家ですら、人間である以上は完全な仕様を記述できる訳ではない。

そこで、これらの問題を前提として、その有力な解決手法を開発するために人工知能技術を適用した各種の要求理解方法が研究されている。

2．2　要求理解の研究動向

(1)　非専門家と専門家の協調作業としての要求理解

Reubenstein 等[1] は要求を出す人と要求仕様を分析／形式化する人が存在し、両者が協調して要求仕様を記述する事を支援する方法について研究し、ＲＡ（Requirement Apprentice）を試作している。

ＲＡは非形式的仕様と形式的仕様の間に橋をかける形式化フェーズに焦点をあて、話す人と聞き手の間の非形式的コミュニケーションを性格づける一般的特徴（略語，曖昧，順序性の欠如，矛盾，不完全性，不正確さ）を分析し、専門家とＲＡの対話的インタフェースにより、非専門家からの要求を取り出す事を試みた。しかし、要求仕様を記述する際に、専門家ですら誤りを犯す事があり、しかも、２者の協調作業が必要である点は、方式上の弱点である。

(2)　既存の要求仕様の利用

吉村等[2] は通信サービスの再利用性に着目して、事例ベース推論による要求仕様獲得を研究している。しかし、非専門家の要求仕様では既存のサービスや既存の設計概念と異なる要素を含む新規サービスへの対応が必要であるため、事例ベース推論における類似性判定や事例抽出によって、新規サービスへ対応する事は困難である。

(3)　新規な要求仕様への対応

Maiden, Sutcliffe 等[3] はReubenstein の研究成果の拡張を行い、既存のものにない要求仕様を対象とした要求理解の方法について研究した。基本的構想は、新規要求を作る時、ＳＥは心の中にある知識構造を再利用すると考えている。Reubenstein 等と異なる点は、より複雑なソフトウェアライブラリの適用を可能にする事により、新規な要求を既存のものから、生成しようとした点である。しかし、非専門家と専門家（critics）の協調作業という点は、Reubenstein の場合と同じ課題が残っている。

２．３　未解決な課題

Reubenstein やMaiden, Sutcliffe 等は要求理解を非専門家と専門家の協調作業と考えており、その協調作業の支援方法について研究を深めている。しかし、認知科学の面から、この手法を評価すると、専門家が介在しなければならない点は、要求者と要求理解システムだけの構成に対して、完全な仕様を抽出する事を困難にしている面があり、専門家を必要としない手法の研究が未解決な課題となる。

また、既存のソフトウェア要求仕様、ソフトウェア構造を利用する方法では、既存のものにない要求仕様が入力された場合、新規な情報の定義が必要である事は検出できるが、外部から情報を取り込まなければ要求理解はできない点が未解決な問題となる。

3．解決手法

3．1　要求理解の全体構成

要求理解部の入力情報は要求仕様記述言語により表現される状態遷移規則（現状態、イベント：次状態）の集合として構成される事を仮定している。

また、この段階では、要求仕様記述言語としての文法誤りはないが、曖昧な記述、情報の欠落が含まれる事を仮定している。

要求理解により、この不完全な要求仕様から正しい要求仕様を獲得する方法を以下に示す。

(1)　欠落情報検出

要求仕様に記述された状態遷移が途中で終わってしまっている事を検出する。これは、通信システムの性質として、初期状態から出発して、どの様な状態に遷移しても、最終的には初期状態に戻る事が知られており、各状態の遷移のルートを辿る事により、欠落情報の存在を検出する。

欠落情報には、既存の通信サービス要求仕様に含まれるものと、新規の規則を要求するものとがある。既存の通信サービスに含まれる欠落情報は既存の通信サービスの仕様と比較する事により、欠落情報を補完する事ができる。新規の規則を持つ通信サービスは、通信サービスとして正しい状態遷移方法に従う事によって欠落情報を補う事ができる。

本研究では、新規通信サービスの欠落情報補完のために、通信サービスをドメインとするドメインモデルを用いる事とした。

(2)　残規則検出

要求仕様に記述された各規則の中に、状態遷移ルートの中に1度も出現しない規則を検出する。この規則には、他の規則が不足しているために状態遷移ルートに適用されないか、または、冗長な情報であるかの2通りの解釈が成り立つ。不足している規則を補う必要があるのか、残規則を破棄するかは要求提供者の判断に委ねる事となる。

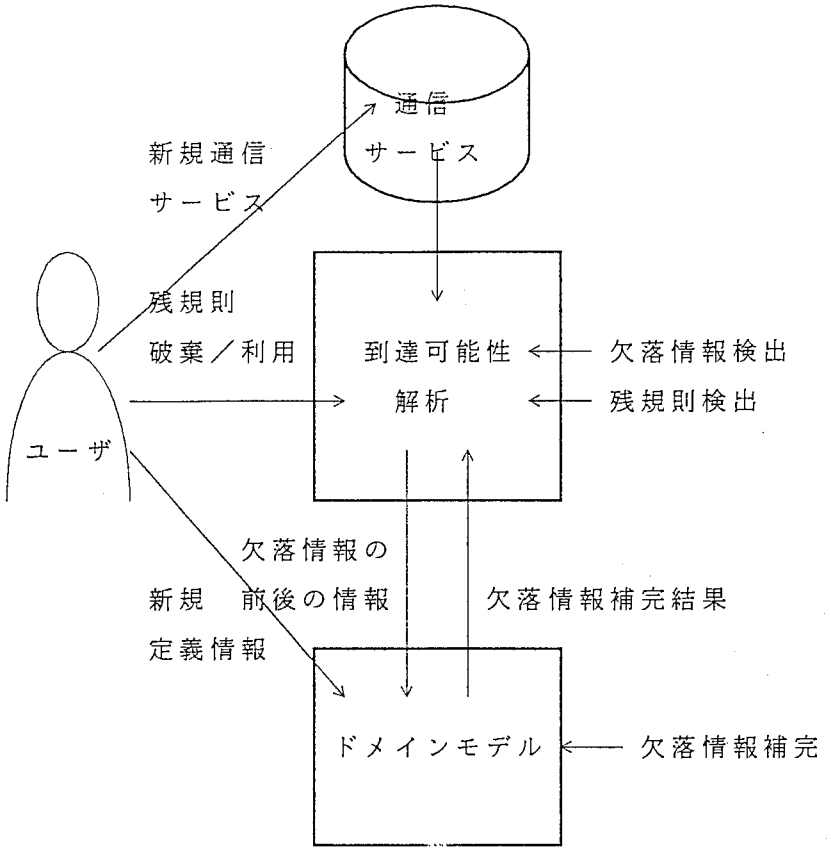


図３－１　要求理解の全体像

３．２　ドメインモデルの概要

３．２．１　ドメインモデルの目的

ドメインモデルの目的は、２．３節に示した要求理解における未解決な課題のうち、非専門家がドメインモデルに入力する新規の要求仕様が持つ欠落情報を補完する事である。

３．２．２　ドメインモデルの概要

ドメインモデルは、ドメイン知識、仕様抽出機構、辞書から構成される。
ドメイン知識と辞書は入力情報によっては、追加／変更が必要となるものである。
ドメイン知識は、対象ドメイン特有の知識であり、対象ドメインを熟知した専門家が与える情報である。本研究では、専門家の知識をいかにして獲得するかは対象外である。

また、辞書は、専門家が与える事もあるがドメインモデルによる要求仕様の補完を行いたいユーザが追加／変更を行う事もある。

以下に各々の構成要素について説明する。

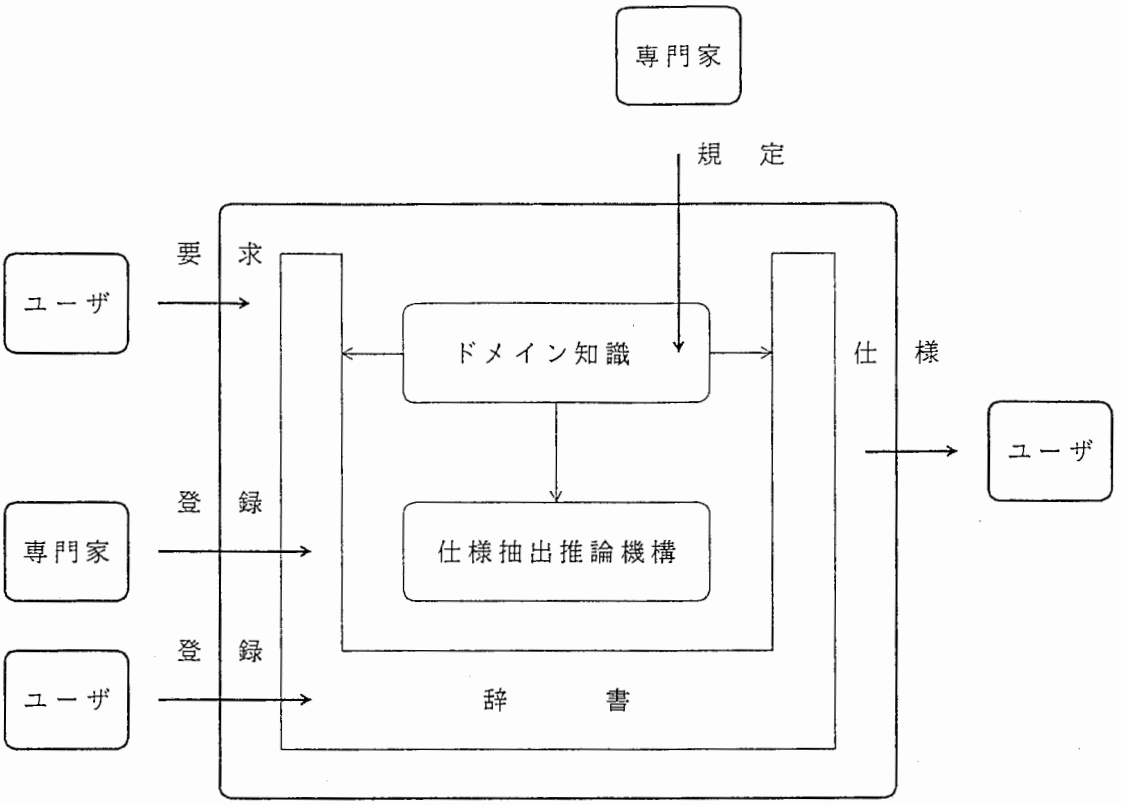


図３－２　ドメインモデルの構成

(1)　ドメイン知識

表１に、ドメインモデルにおける知識の種類を示す。

表３－１　ドメイン知識の種類

種別	内　　容
属性	対象ドメイン固有の属性（概念）、属性間の関係を記述。 何に着目してモデル化するかに依存する。
属性の操作方法	各属性に対する、値の設定／削除方法を記述。 他の属性値から自動的に値を算出するような、属性の関係に関する知識も含む。到達仕様抽出時の推論の単位となる。
制約条件	属性間の関係および属性と操作の組み合わせに関する制約条件を記述。
必須条件	特定の属性において、必ず付加すべき操作法を記述。

(2)　仕様抽出機構

仕様抽出の手順について概要を示す。

①要求仕様は後述する辞書により、補完が必要な部分の前、後の状態をドメインモデル内部の情報（抽象状態）に変換する。

②属性操作を適用する事により、システムとして正しい状態遷移系列を抽出する。

③抽出した抽象状態遷移系列の中から、制約条件に示されるシステムとして許されない状態遷移を行うものがあれば削除する。

④残った抽象状態遷移系列に必須条件に示されるシステムとして正しく動作するために必要な状態遷移を付加して、正しい抽象状態遷移系列の抽出を終える。

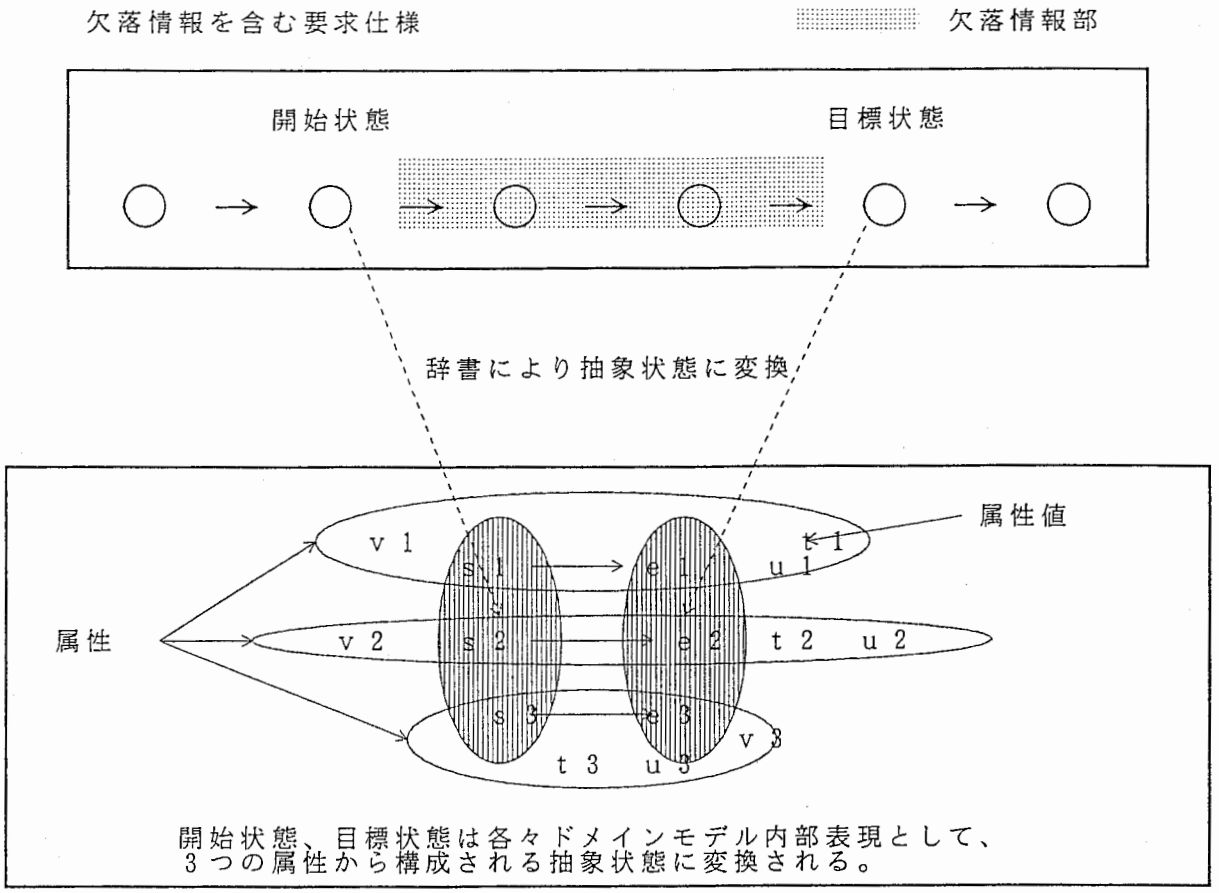
⑤辞書により、抽象状態遷移を系列ごとに要求仕様記述情報に変換する。

(3) 辞書

要求仕様上の情報とドメインモデル内部での抽象化された情報との対応関係を持つ。

要求仕様の補完は、仕様として正しい状態遷移を行う様に、この抽象状態の遷移を起こす遷移系列を抽出する事である。

抽出された抽象状態遷移情報は辞書により、要求仕様情報に変換され、ユーザに示される事になる。



属性操作Ｗ１，Ｗ２，Ｗ３を適用する事により開始状態から目標状態に至る抽象状態遷移の抽出成功。

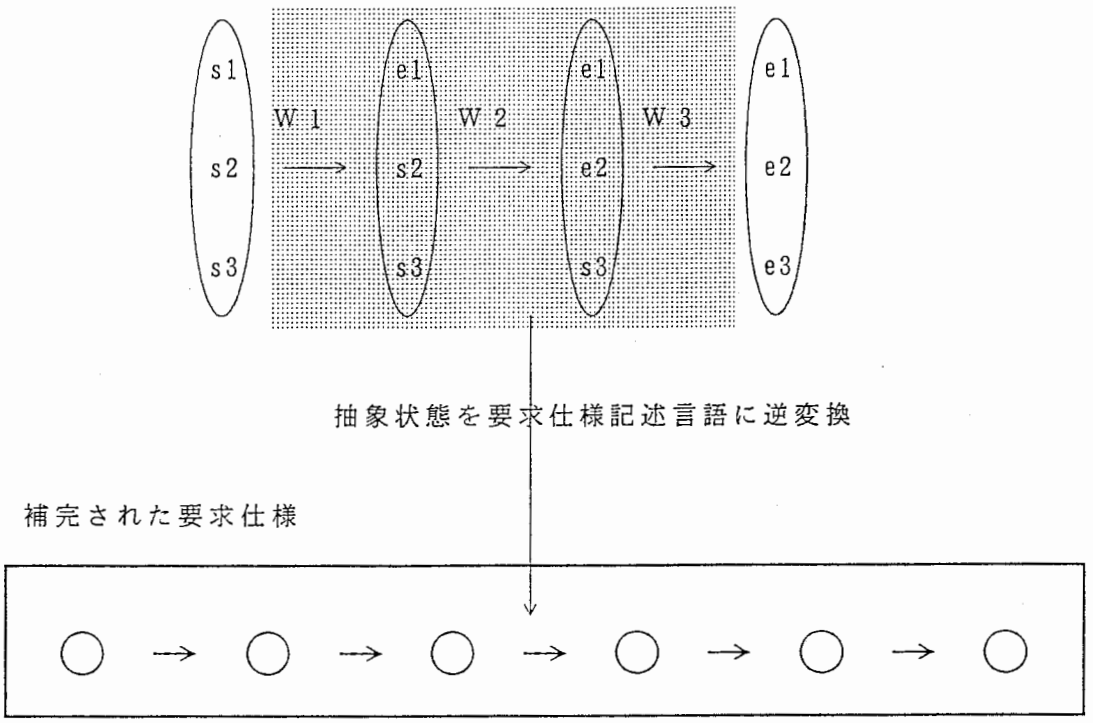


図３―３　欠落情報の補完方法

3．3　ドメインモデルの拡張

前述のドメインモデルにおいて、属性操作、辞書におけるイベントと属性操作の対応関係を変更する事により、従来、要求仕様の補完の対象外であった3コ以上の端末が関係する通信サービスを要求理解の対象に加える事ができた。

以下に変更内容を示す。

(1)　呼の定義

呼は1つ，或いは2つの識別子（端末または人間）の関係を表す状態あるいは呼要素の集合を示す。

(2)　呼要素

呼要素は要求仕様上の状態と対応付けられる最小単位であり、ドメインモデル内部では属性（値）の集合として表される。

以下の3タイプがある。

ａ）タイプ1：対応する要求仕様上の状態の引数が1つ。

ｂ）タイプ2：対応する要求仕様上の状態の引数が2つであり、かつ同一。

ｃ）タイプ3：対応する要求仕様上の状態の引数が2つであり、かつ異なる。

(3)　距離

ドメインモデル内部において、呼を構成する呼要素に対応する全ての属性の中で値の異なるものを数え挙げたものを距離と呼ぶ。

距離＝6（下線部の属性値が開始状態と目標状態で異なる）

呼要素	開始状態	目標状態
A	[nil , nil , nil]	[nil , nil , nil]
B	[nil , nil , nil]	[nil , nil , nil]
C	[nil , nil , nil]	[nil , nil , nil]
A－A	[nil , nil , nil]	[nil , nil , nil]
B－B	[nil , nil , nil]	[nil , nil , nil]
C－C	[nil , nil , nil]	[nil , nil , nil]
A－B	[disconnect , nil , nil]	[connect , nil , nil]
A－C	[initial , nil , nil]	[connect , nil , nil]
B－A	[disconnect , nil , nil]	[connect , nil , nil]
B－C	[nil , nil , nil]	[connect , nil , nil]
C－A	[initial , nil , nil]	[connect , nil , nil]
C－B	[nil , nil , nil]	[connect , nil , nil]

図3－4　距離の例

(4) 初期状態

開始状態、目標状態をドメインモデル内部の情報に変換する前に、開始状態、目標状態の第一引数として現れない、全ての端末変数について要求仕様上の初期状態の定義に従い付加する。

開始状態　Ｐ１（Ａ，Ｂ），Ｐ２（Ｃ）
目標状態　Ｐ３（Ａ，Ｃ）

初期状態　ｉｄｌｅ（Ｘ）

初期状態を付加した開始状態　Ｐ１（Ａ，Ｂ），Ｐ２（Ｃ），ｉｄｌｅ（Ｂ）
初期状態を付加した目標状態　Ｐ３（Ｂ，Ｃ），ｉｄｌｅ（Ａ），ｉｄｌｅ（Ｃ）

図３－５　初期状態付加の例

(5) 属性操作

属性操作を主属性操作と従属性操作に分ける。

主属性操作は、更新条件の規定を含め、属性値の更新方法を定める。

$VALUE ,$CLEAR, $REMOVE,$IF-VALUE がある。

従属性操作は、属性値の更新方法を定める。主属性操作適用した後で適用する。

$IF-ADDED,$IF-CLEAR,$IF-REMOVED,$IF-VALUE がある。

各属性操作の内容は以下の３通りに大別される。

①属性個別の操作種別

(a) $ VALUE

属性値を直接設定。

(b) $ CLEAR

属性値を削除。対象の属性が初期値を持てばその値を設定し、持たない場合は未設定状態とする。

② 属性間の関係に関する操作種別

(a) $ IF-ADDED（属性名）

パラメータで指定した属性に値が書き込まれると起動する。

例：発アドレスが決まれば、自動的に着アドレスが決まる。（ホットライン）

着アドレス属性に「$ IF-ADDED (Originating address)＋ $ VALUE (Data)」を記述する

(b) $ IF-REMOVED（属性名）

パラメータで指定した属性の値が削除されると起動する。

属性間の関係に関する操作は、属性個別の操作と組み合わせて規定する。

③ 適用条件に関する規定

(a) $IF-VALUE （条件式）

条件式::＝｛属性名，属性値｜呼要素名｜not ｜｛or｜and｝｝

呼要素名::＝　端末変数名｛，端末変数名｝

条件式が成立すると、$IF-VALUE の後に記述された属性操作を適用する。

(6) イベントと属性操作の対応

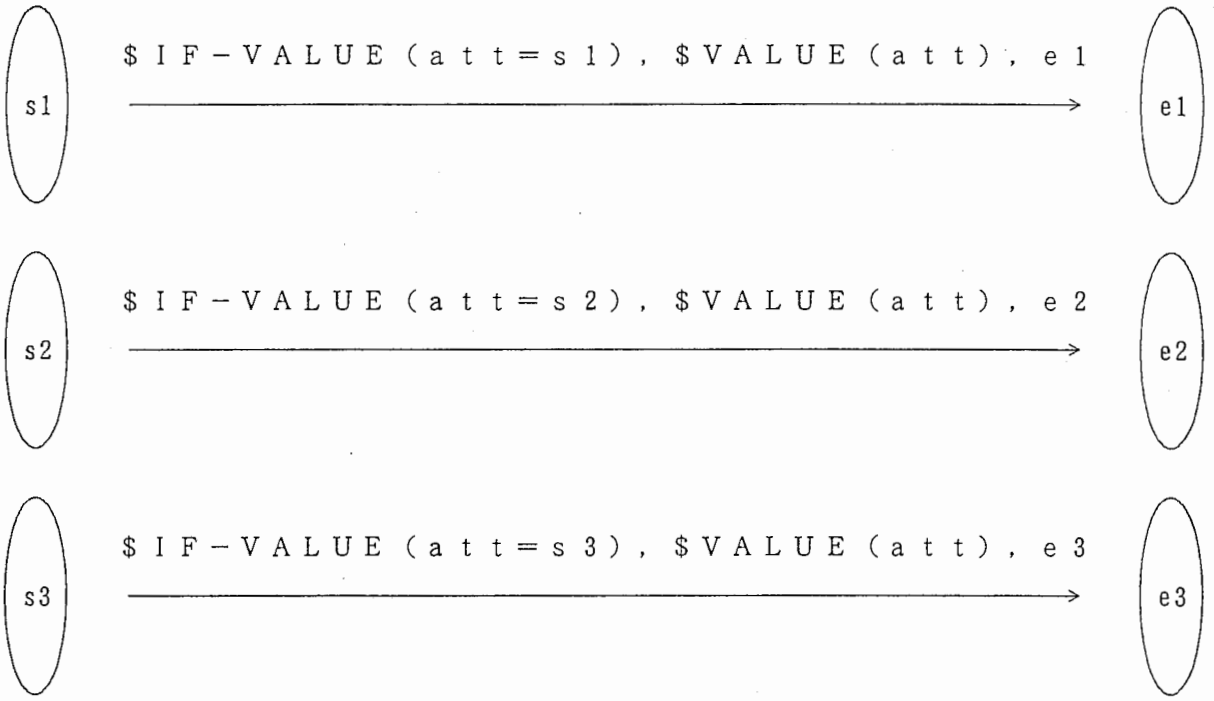
イベントは　主属性操作　＋　従属性操作に対応する。

また、$IF-VALUE を用いて、属性値対応に、操作内容を規定できる。

1イベント

⇩

```
$IF-VALUE(att=s1), $VALUE(att), e1
$IF-VALUE(att=s2), $VALUE(att), e2
$IF-VALUE(att=s3), $VALUE(att), e3
```



(7) 辞書定義

ＳＴＲ状態とドメインモデルの抽象状態を対応させるには、呼要素対応の定義が必要なため、辞書は、ＳＴＲ状態に対応して、３タイプの呼要素対応に記述する。

ＳＴＲ状態	モデル状態
ＳＴＲ状態	┌ 呼要素タイプ１のモデル状態 │ 呼要素タイプ２のモデル状態 └ 呼要素タイプ３のモデル状態

３．４　ドメインモデルの構築方法

ドメインモデルとは、実世界の情報（状態，状態遷移）をドメインモデル上の抽象化された情報として表現する事により、実世界をモデル化する事である。

ドメインモデルの構築は、３．２節に示したドメインモデルの各構成要素を決める事となる。

(1)　属性の抽出

対象ドメインをモデル化するための固有の概念を抽出する必要がある。例えば、通信サービスをドメインとする場合は、”呼び”の状態変化に着目してモデル化した。即ち、発呼により生成され、切断によって消滅する呼のライフサイクルモデルである。通信サービスをどのような回線をどの端末間に設定するかを決定していく過程として考え、”どのような”という部分を属性によって表現する。各属性は(1)に示す形式で規定する。値はその属性がとりうる値の範囲を、初期値は呼が生成された状態で、その属性が特定の値を持つ場合にその値を規定する。回線交換サービスの場合の呼の属性を構成する要素例を表２に示す。

表３－２　呼の属性規定の例

属性名称	取りうる値の範囲	初期値
接続状態	[通信前／通信中／通信後]	通信前
端末状態	[聴取前／聴取中]	聴取前
端末アドレス	端末アドレス	―――
通信速度	[64kb/s／192kb/s／・・・・]	64kb/s
透過性	[透過／非透過]	非透過
遅延特性	ｍｓ	―――
セル廃棄率	％	―――
課金手段	[課金ﾒｰﾀ／ｸﾚｼﾞｯﾄｶｰﾄﾞ／ｷｬｯｼｭ]	課金メータ
課金アドレス	[発ｱﾄﾞﾚｽ／着ｱﾄﾞﾚｽ／第３者]	発アドレス

取りうる値の範囲を［・・・・］で規定した属性は、その中のどれかが選択される。

他の属性は、規定情報を単位とする値が設定される。

属性の中には初期値をもたないものがある。これらの属性は呼の発生後、通信中状態に至るまでにユーザ動作によって値が決定していく。属性種別は不変ではなく、ネットワークの進歩や要求の多様化に応じて追加、変更もありうる。表２の例では、アナログネットワークでは接続状態、端末状態、端末アドレスの３種類だった属性が、Ｎ－ＩＳＤＮ(Narrow-band ISDN) で通信速度や透過性が追加になり、さらにＢ－ＩＳＤＮ(Broad-band ISDN) では遅延特性やセル廃棄率が加わる。また、着側課金やクレジットカード課金のような要求の多様化に対応した属性規定もある。

(2) 属性操作

主属性操作により、特定の呼要素の特定の属性を書換え、従属性操作により、他の呼要素の属性を書き換える様に規定できる。

例題. 一連の属性操作により、呼要素A－B，B－Aのパス接続状態を接続中にし、呼要素A－C，C－Aのパス接続状態を保留中にする例を示す。

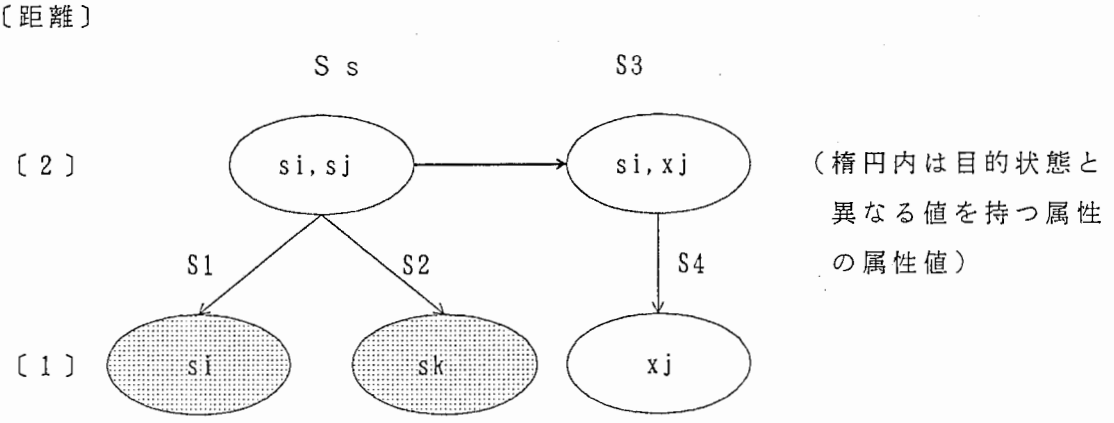
```
$IF-VALUE(call-ellem=A-B)
    $VALUE(path-state),connect,
$IF-VALUE(call-ellem=B-A)
    $VALUE(path-state),connect,
$IF-VALUE(call-ellem=A-C)
    $VALUE(path-state),hold
$IF-VALUE(call-ellem=C-A)
    $VALUE(path-state),hold
```

(3) 仕様抽出機構

開始状態と目標状態の属性値の差を距離として、基本的にはこの距離が短くなる属性操作を適用する事により、抽象状態遷移を抽出する。

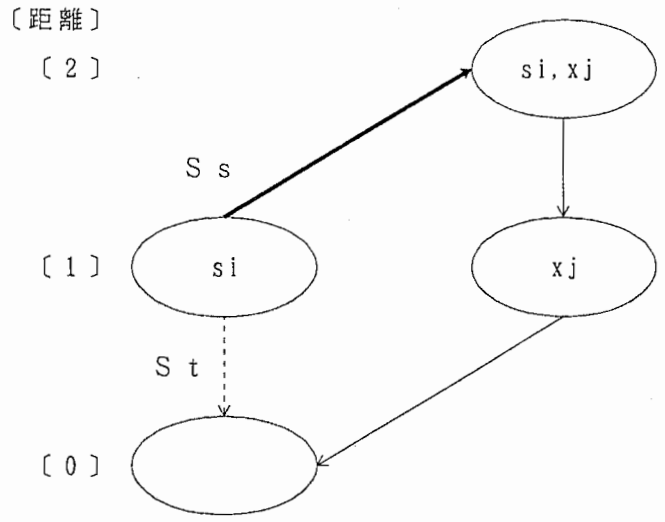
しかし、常に、距離が短くなるとは限らず、また、3コ以上の端末の抽象状態が目標状態に到達するには、ある端末に対応する抽象状態が一時的に距離が離れる事もあり得る。そこで、仕様抽出機構としては、状態遷移ルートを距離が一時的に縮まらない状態遷移（横滑り迂回）、一時的に離れる状態遷移（遠回り迂回処理）を必要とする。

また、距離そのものの考えかたを各呼要素毎ではなく、全呼要素として算出し、目標状態への到達完了を評価する事により、後者についても対応可能となる。



S1とS2が制約違反状態のケースで属性jを書き換えて迂回している。

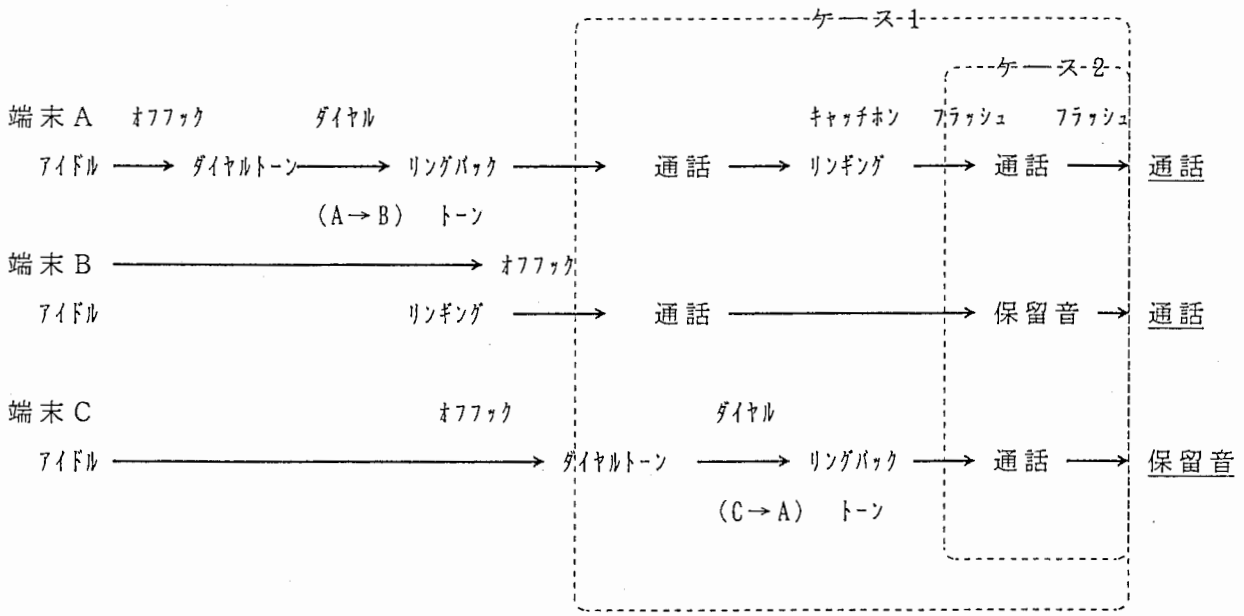
図3－6　横滑り迂回の例



（楕円内は目的状態と異なる値を持つ属性の属性値）

Ｓｓ⇨Ｓｔが制約違反遷移のケースで属性ｊを書き換えて迂回している。

図３－７　遠回り迂回の例



…… が囲んだ部分が欠落情報だとすると以下のルート抽出機構が必要である。

ケース１：全端末に対応する呼要素が目標状態に到達するまでルート抽出を続行する。
（２端末に対応する呼要素が目標状態に到達してもルート抽出を中止しない。）
ケース２：ある２端末によるイベントにより、他端末の状態を変化させる。

図３－８　３端末が関係する通信サービス（ｷｬｯﾁﾎﾝ）の例

(4) 辞書

辞書は、ＳＴＲ状態対応にドメインモデルの対応情報を呼要素のタイプ毎に記述する。ＳＴＲ状態の実際に可能な和集合を作る時、呼要素毎の状態定義に矛盾が生じる様な事は生じない。

下表の例では、Hold(A,B),Calling(A,C)　の場合、

A-B,B-A のpathstate=connected

A-C,C-A のpathstate=initial　　となり、呼要素毎に矛盾なく定義できている。但し、ｎｉｌの項については、他のモデル状態の定義でｎｉｌ以外の値設定がある場合はｎｉｌ以外の値を有効とする。

表３－３　３コ以上の端末を持つ通信サービスの辞書の定義例

ＳＴＲ表記	モデル状態
path-passive (A,B)	--　A1:tone ,A2:A ,A3:nil
	--　A1:nil ,A2:nil,A3:nil,A4:nil
	A-B　A1:connect,A2:A ,A3:B ,A4:nil
	others　A1:nil,A2:nil,A3:nil,A4:nil
ringing(A,C)	C　A1:tone ,A2:A ,A3:nil
	A　A1:nil ,A2:nil,A3:nil
	---　A1:nil ,A2:nil,A3:nil,A4:nil
	A-C　A1:initial,A2:A ,A3:B ,A4:nil
	others　A1:nil,A2:nil,A3:nil,A4:nil
path-passive (A,B) ringing(A,C)	A　A1:tone ,A2:A ,A3:nil
	B　A1:nil ,A2:A ,A3:nil
	C　A1:tone ,A2:A ,A3:nil
	A-A　A1:nil ,A2:nil,A3:nil,A4:nil
	B-B　A1:nil ,A2:nil,A3:nil,A4:nil
	C-C　A1:nil ,A2:nil,A3:nil,A4:nil
	A-B　A1:connect,A2:A ,A3:B ,A4:nil
	A-C　A1:initial,A2:A ,A3:B ,A4:nil
	B-A　A1:nil,A2:nil,A3:nil,A4:nil
	B-C　A1:nil,A2:nil,A3:nil,A4:nil
	C-A　A1:nil,A2:nil,A3:nil,A4:nil
	C-B　A1:nil,A2:nil,A3:nil,A4:nil

３．５　期待効果

３．３節に示す拡張を加える事により、一度、１組の２端末が目標状態に到達しただけで、ルート抽出を止めてしまう事はなく、また、ある２端末間におけるイベントにより、他端末の状態を変化させる事ができる事となり、この結果、３コ以上の端末を対象とした通信サービスの欠落情報の補完が可能となる。

4．手法の検証

4．1　例題

付録－1に例題を示す。

4．2　有効性

4．1においてケーススタディとして示したにキャッチホン，3者通話の3コ以上の端末が出現する規則を含む欠落情報の補完が正常に実施できる事が判った。

従って、本手法に示した拡張を加える事により、新規サービスが3コ以上の端末を含む通信サービスであっても、要求理解が可能となる。

5．評価

5．1　可用性

本手法では入力となる要求仕様が状態遷移により記述できる事という条件の下で、通信サービス以外の領域を対象とするものでも、ドメイン知識が抽出できれば、辞書は容易に与え得るものであり、また、仕様抽出機構も基本的には利用可能であると考える。

例えば、通信サービスと密接に関係する保守システムの保守サービス等においても、容易に本手法の適用は属性の抽出ができれば、属性操作は特に、新規の操作方法を加える必要もなく、仕様抽出機構もそのまま利用可能と考える。

5．2　効果

非専門家がどの様な新規サービスを入力しても、欠落情報の補完を行える事がドメインモデルによる要求理解の目的であるが、従来の手法では、3コ以上の端末が1つのイベントにより、状態遷移する様な規則を欠落情報とする場合の補完はできなかった。

この原因は、属性操作の規定方法が複雑な規定に適合しなかったためであるが、3．3節に示した拡張を取り込む事により、3コ以上の端末が1つのイベントにより、状態遷移する様な規則を欠落情報とする場合のモデル化を実現する事が可能となった。

但し、新規サービスの補完のためにドメインモデル自体の拡張（ドメイン知識、辞書の追加または変更）を必要とする事が考えられるが、この場合は自動的に補完を行う事はできない。

5．3　残課題

本手法ではドメイン知識の獲得手法を示すものではないが、「非専門家がどの様な新規サービスを入力しても、欠落情報の補完を行える事」を目的とするため、新規サービスの補完のためにドメインモデル事態の拡張を必要とする場合には、非専門家からのドメイン知識の獲得が必要となる。これは、非専門家にモデル化された世界を意識させずに、一部のドメイン知識を獲得する事が必要となる。そのためには、非専門家が認識できる要求仕様を定義する世界で、メタ知識に基づくドメインモデルとの会話による情報を設定により、この知識獲得が可能となると考えている。

6．まとめ

ドメインモデルによる要求理解の目的は、「非専門家がどの様な新規サービスを入力しても、欠落情報の補完を行える事」であるが、従来の手法では、3コ以上の端末が1つのイベントにより、状態遷移する様な規則を欠落情報とする場合の補完はできなかった。そこで、3者通話、キャッチホン、ＵＰＴ、8者会議等の通信サービスをケーススタディの対象として、ドメインモデルの拡張方法を研究し、その結果に基づき本レポートに示す拡張を行った。

また、ＰＢＸの持つ多彩なサービスの一部を対象としてドメインモデルの適応性について研究を実施したが、属性値を追加するだけで新規サービスの補完は可能であり、その他の枠組みの拡張等は不要であるという見通しを得た。

計算時間は、従来手法に基づき試作したプロトタイプでは、サブセコンドで終了しているため、本手法による拡張を取り入れた結果、計算量は増加するが、サブセコンドの範囲で可能であると考える。

今後、現状では実体の想像が難しい通信サービス（例えばマルチメデイア等）を対象とした補完を考慮すると、一部の属性や属性値の追加が必要となる事が考えられ、属性の分類や階層化等のドメイン知識の再編集を容易化するドメインモデルの枠組みの変更伴う非専門家からの知識獲得の手法の研究が重要な課題となる事が考えられる。

［参考文献］

(1) H.B.Reubenstein,et al.“The Requirement Apprentice:Automated Assistance for Requirements Acquisition”IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, Vol.1 No.3,MARCH 1991

(2)吉村　晋，他「事例ベース推論を適用した通信ソフトウェア開発支援システムにおける要求仕様獲得　とヒューマンインタフェースアーキテクチャについて」電子情報通信学会 AI92-95

(3) N.A.M.Maiden & A.G.Sutcliffe“Requirements Critiquing Using Domain Abstractions” PROCEEDINGS FIRST INTERNATIONAL CONFERENCE ON REQUIREMENTS ENGINEERING

(4)中村　光宏，田倉　昭，太田　理「ドメインモデルを用いたサービス仕様生成の構想」電子情報通信学会　交換システム研究会SSE94-17,1994 年4 月

付録 — 1

3コ以上の端末がある場合の2例題を以下に示す。

(1) 属性と属性値

呼要素のタイプと属性の対応関係を以下の様に定義する。

タイプ1 ＝ [termstate , terminal-address , mregst]

タイプ2 ＝ [pathstate , terminal-address , mregst]

タイプ3 ＝ [pathstate , terminal-address , mregst]

	属性名	属性値	意　味
1	pathstate （パス状態）	initial connected disconnected hold	初期状態 パスが接続された状態 パスが切断された状態 パスが保留された状態
2	terminal-address （端末アドレス）	（端末変数名）	端末アドレス
3	termstate （端末状態）	initial tone	初期状態 音源聴取状態
4	mregst （メモリ登録状態）	regst clear	登録あり 登録解除

(2) キャッチホンにおける例題

Calling(A,B) 欠落情報 CW-calling(A,C),Talk(A,B)
開始状態 目標状態

初期状態としてidle(X) が定義されている場合、入力状態を付加すると

Calling(A,B),idle(C) 欠落情報 CW-calling(A,C),Talk(A,B)

距離 5

呼要素	開始状態	目標状態
A	[tone , nil , nil]	[tone , nil , nil]
B	[tone , nil , nil]	[tone , nil , nil]
C	[initial , nil , nil]	[tone , nil , nil]
A－A	[nil , nil , nil]	[nil , nil , nil]
B－B	[nil , nil , nil]	[nil , nil , nil]
C－C	[nil , nil , nil]	[nil , nil , nil]
A－B	[initial , nil , nil]	[connect , nil , nil]
A－C	[nil , nil , nil]	[initial , nil , nil]
B－A	[initial , nil , nil]	[connect , nil , nil]
B－C	[nil , nil , nil]	[nil , nil , nil]
C－A	[nil , nil , nil]	[initial , nil , nil]
C－B	[nil , nil , nil]	[nil , nil , nil]

主属性操作 $VALUE(pathstate),connectをpathstate が,initialを選び適用。
距離 3

呼要素	開始状態	次状態1
A	[tone , nil , nil]	[tone , nil , nil]
B	[tone , nil , nil]	[tone , nil , nil]
C	[initial , nil , nil]	[initial , nil , nil]
A－A	[nil , nil , nil]	[nil , nil , nil]
B－B	[nil , nil , nil]	[nil , nil , nil]
C－C	[nil , nil , nil]	[nil , nil , nil]
A－B	[initial , nil , nil]	[connect , nil , nil]
A－C	[nil , nil , nil]	[nil , nil , nil]
B－A	[initial , nil , nil]	[connect , nil , nil]
B－C	[nil , nil , nil]	[nil , nil , nil]
C－A	[nil , nil , nil]	[nil , nil , nil]
C－B	[nil , nil , nil]	[nil , nil , nil]

主属性操作 $VALUE(termstate),tone をtermstate が,initialを選び適用。
距離 2

呼要素	状態1	状態2
A	[tone , nil , nil]	[tone , nil , nil]
B	[tone , nil , nil]	[tone , nil , nil]
C	[initial , nil , nil]	[tone , nil , nil]
A－A	[nil , nil , nil]	[nil , nil , nil]
B－B	[nil , nil , nil]	[nil , nil , nil]
C－C	[nil , nil , nil]	[nil , nil , nil]
A－B	[connect , nil , nil]	[connect , nil , nil]
A－C	[nil , nil , nil]	[nil , nil , nil]
B－A	[connect , nil , nil]	[connect , nil , nil]
B－C	[nil , nil , nil]	[nil , nil , nil]
C－A	[nil , nil , nil]	[nil , nil , nil]
C－B	[nil , nil , nil]	[nil , nil , nil]

主属性操作 $VALUE(pathstate),initialを呼要素がC-A,A-C を選び適用。
距離　0

呼要素	状態2	目標状態
A	[tone , nil , nil]	[tone , nil , nil]
B	[tone , nil , nil]	[tone , nil , nil]
C	[tone , nil , nil]	[tone , nil , nil]
A－A	[nil , nil , nil]	[nil , nil , nil]
B－B	[nil , nil , nil]	[nil , nil , nil]
C－C	[nil , nil , nil]	[nil , nil , nil]
A－B	[connect , nil , nil]	[connect , nil , nil]
A－C	[nil , nil , nil]	[initial , nil , nil]
B－A	[connect , nil , nil]	[connect , nil , nil]
B－C	[nil , nil , nil]	[nil , nil , nil]
C－A	[nil , nil , nil]	[initial , nil , nil]
C－B	[nil , nil , nil]	[nil , nil , nil]

この結果は、

```
Calling(A,B),cond: idle(C)          offhook(B) : Talk(A,B).
Talk(A,B),idle(C)                   offhook(C) : Talk(A,B),dial-tone(C).
dial-tone(C),cond:Talk(A,B)         dial(C,A)  : CW-calling(C,A).
```

(3)　3者通話による例題

Hold(A,B),Calling(A,C)　　欠落情報　　Talk(A,B),Talk(B,C),Talk(C,A)
開始状態　　目標状態

上記の場合、初期状態としてidle(X) が定義されていても、入力状態を付加する対象はない。

距離　6

呼要素	開始状態	目標状態
A	[nil , nil , nil]	[nil , nil , nil]
B	[nil , nil , nil]	[nil , nil , nil]
C	[nil , nil , nil]	[nil , nil , nil]
A－A	[nil , nil , nil]	[nil , nil , nil]
B－B	[nil , nil , nil]	[nil , nil , nil]
C－C	[nil , nil , nil]	[nil , nil , nil]
A－B	[disconnect, nil , nil]	[connect , nil , nil]
A－C	[initial , nil , nil]	[connect , nil , nil]
B－A	[disconnect, nil , nil]	[connect , nil , nil]
B－C	[nil , nil , nil]	[connect , nil , nil]
C－A	[initial , nil , nil]	[connect , nil , nil]
C－B	[nil , nil , nil]	[connect , nil , nil]

属性操作 $VALUE(pathstate),connectedをpathstate がinitial である呼要素に適用。

距離　4

呼要素	開始状態	状態1
A	[nil , nil , nil]	[nil , nil , nil]
B	[nil , nil , nil]	[initial , nil , nil]
C	[nil , nil , nil]	[initial , nil , nil]
A－A	[nil , nil , nil]	[nil , nil , nil]
B－B	[nil , nil , nil]	[nil , nil , nil]
C－C	[nil , nil , nil]	[nil , nil , nil]
A－B	[disconnect, nil , nil]	[disconnect, nil , nil]
A－C	[initial , nil , nil]	[connect , nil , nil]
B－A	[disconnect, nil , nil]	[disconnect, nil , nil]
B－C	[nil , nil , nil]	[nil , nil , nil]
C－A	[initial , nil , nil]	[connect , nil , nil]
C－B	[nil , nil , nil]	[nil , nil , nil]

主属性操作 $VALUE(pathstate),connectedをpathstate がdisconnectedである呼要素に適用。
距離　2

呼要素	状態1	状態2
A	[nil , nil , nil]	[nil , nil , nil]
B	[initial , nil , nil]	[initial , nil , nil]
C	[initial , nil , nil]	[initial , nil , nil]
A－A	[nil , nil , nil]	[nil , nil , nil]
B－B	[nil , nil , nil]	[nil , nil , nil]
C－C	[nil , nil , nil]	[nil , nil , nil]
A－B	[disconnect, nil , nil]	[connect , nil , nil]
A－C	[connect , nil , nil]	[connect , nil , nil]
B－A	[disconnect, nil , nil]	[connect , nil , nil]
B－C	[nil , nil , nil]	[nil , nil , nil]
C－A	[connect , nil , nil]	[connect , nil , nil]
C－B	[nil , nil , nil]	[nil , nil , nil]

従属性操作 $VALUE(pathstate),connectedをpathstate がdisconnectedである呼要素に適用。
距離 0

呼要素	状態2	目標状態
A	[nil , nil , nil]	[nil , nil , nil]
B	[initial , nil , nil]	[initial , nil , nil]
C	[initial , nil , nil]	[initial , nil , nil]
A－A	[nil , nil , nil]	[nil , nil , nil]
B－B	[nil , nil , nil]	[nil , nil , nil]
C－C	[nil , nil , nil]	[nil , nil , nil]
A－B	[connect , nil , nil]	[connect , nil , nil]
A－C	[connect , nil , nil]	[connect , nil , nil]
B－A	[connect , nil , nil]	[connect , nil , nil]
B－C	[nil , nil , nil]	[connect , nil , nil]
C－A	[connect , nil , nil]	[connect , nil , nil]
C－B	[nil , nil , nil]	[connect , nil , nil]

この結果は、

```
Calling(A,C),cond:Hold(A,B)    offHook(C) : Talk(A,C).
Talk(A,C),Hold(A,B)            flash(A)   : Talk(A,B),Talk(B,C),Talk(C,A).
```

(4) 辞書

ＳＴＲ表記	モデル内部
Hold(A,B)	A1:tone ,A2:A ,A3:nil A1:nil ,A2:nil,A3:nil,A4:nil A1:connect,A2:A ,A3:B ,A4:nil
recall- dialtone(A)	A1:tone ,A2:A ,A3:nil A1:nil ,A2:nil ,A3:nil,A4:nil A1:disconnect,A2:A,A3:nil,A4:nil
Calling(A,C)	A1:tone ,A2:A ,A3:nil A1:nil ,A2:nil,A3:nil,A4:nil A1:initial,A2:A ,A3:B ,A4:nil
Talk(A,B)	A1:tone ,A2:A ,A3:nil A1:nil ,A2:nil,A3:nil,A4:nil A1:connect,A2:A ,A3:B ,A4:nil
dial-tone(C)	A1:tone ,A2:C ,A3:nil A1:nil ,A2:nil,A3:nil,A4:nil A1:initial,A2:A ,A3: * ,A4:nil
offhook(A)	P1:originating-address, $VALUE(originating-address),A $IF-VALUE(termstate)=initial, $VALUE(termstate),tone $IF-VALUE(termstate)=nil, $VALUE(termstate),tone
offhook(B)	P1:terminating-address, $IF-VALUE(pathstate)=initial, $VALUE(pathstate),connect $IF-VALUE(termstate)=initial, $VALUE(termstate),tone $IF-VALUE(termstate)=nil, $VALUE(termstate),tone
dial(A,B)	P1:originating-address,P2:terminating-address, $IF-VALUE(call-ellem=(P1,P2)) $VALUE(pathstate),initial $IF-VALUE(call-ellem=(P2,P1)) $VALUE(pathstate),initial
flash(A)	P1:originating-address, $IF-VALUE(path-state)=disconnect $VALUE(path-state),connect $IF-VALUE(path-state)=nil $VALUE(path-state),connect $VALUE(originating-address),B $VALUE(terminating-address),C
flash(A)	P1:originating-address, $IF-VALUE(path-state)=connect $VALUE(path-state),disconnect