〔公　開〕

ＴＲ－Ｃ－００７９

通信サービスにおける要求の理解

柴　田　健　次
Kenji　SHIBATA

１９９２．１２．１８

ＡＴＲ通信システム研究所

通信サービスにおける要求の理解

柴田 健次

もくじ

通信サービスにおける要求の理解

柴田 健次

ATR 通信システム研究所

京都府 相楽郡 精華町 光台 2 丁目 2 番地

Tel: +81-7749-5-1241

E-mail: shibata@atr-sw.atr.co.jp

要 旨

通信サービスに対する設計者の要求を理解しサービス仕様を獲得するシステムについてまとめたものである.

要求理解はソフトウェア開発工程の上流に位置し, この段階で曖昧さあるいは間違いを残すと後の開発工程に多大な影響を及ぼし生産性の低下を招く. 従って, 要求を正しく理解することが生産性の向上にとって重要である. 要求を正しく理解するには要求に含まれる曖昧さを扱えることが必要となる. また, 通信サービス仕様を形式的に記述することによりサービス仕様作成段階で検証を可能とし生産性向上の一助となる. 当研究室では通信サービス仕様を形式的に記述する手法として, 状態遷移規則に基づいた動作記述手法 (STR) を開発し実際に使用している. 本稿では要求理解を「曖昧, 断片的な要求から状態遷移規則に基づいた動作記述手法を用いたサービス仕様を生成すること」 と定義する. 要求記述として STR と同形式の記述を採用することを前提としたとき, 要求理解が状態の到達可能性解析問題に帰着できる. ここで,

1. STR で与えられた動作から指定した状態の到達可能性が判定可能か否か
2. 曖昧さを含む要求の取り扱いが可能か否か

ということが問題となる. 本稿では 1 に対して決定可能であることを示し, 2 に対して仮説推論を導入した到達可能性解析で対処することを示す. さらに要求理解システムの部分システムとなる到達可能性解析システムについてその手法, 特徴, 試作システムの概要について述べる.

1 はじめに

近年の通信サービスに対する要求の複雑・多様化に伴い, 新たなサービスの開発をより容易に確実に行うことが必要となっている. 通信サービスの開発においては, サービス仕様の誤りや曖昧さを要因とした開発工程の後戻りが生産性に多大な影響を及ぼすため, 開発の上流工程であるサービス仕様を決定する段階までに誤りのないことを確認することが重要である [1][2][3][4].

通信サービスは複数の個別に設計された個別サービスを合成して得られる. それらは個別サービスに共通に使用される部分と各個別サービスに特有の部分とに分けられる. 開発に際しては個別サービスを全く新たに作ることはせず, 既存のサービスで利用できる部分を利用することで生産性を向上させている. ま

た，通信サービスの記述形式として近年では形式的記述が検討され，検証の容易性などの見地から有効であるとみなされてきている．本稿でも通信サービスの記述として形式的記述を対象とする．

正しいサービスを迅速に得るためにはサービス開発依頼者 (以後ユーザと呼ぶ) がどのような通信サービスを開発したいのかを正しく認識することが重要である．サービス仕様を状態遷移に基づくルール形式で記述し要求記述もサービス記述と同形式で記述するとき，既存のサービスの部分については，要求記述がサービス記述のどこに対応するかを求めることで既存のサービス仕様を利用できる．また，一般にユーザ要求には曖昧さ·不正確さが含まれているが，このような不完全さを有するユーザ要求を正しく認識するための手段が必要となる．完全な状態指定による到達可能性解析は従来でも行われている [8] が，不完全な状態指定については行われていない．そこで，不完全な状態指定を扱えるように拡張した仮説推論を導入した状態の到達可能性解析手法を提案した [5].

本稿では，通信サービスのような状態遷移システムに対して，不完全な状態指定に対処できる状態の到達可能性解析手法について述べる．本稿で扱う到達可能性解析は，既存サービスに対する仕様が与えられている場合を対象とし，既存サービス仕様の範囲内で行うことと限定している．サービス仕様の範囲内で指定した状態の到達可能性が判定可能か否かが問題となるが，それが可解であることを示す．また，システム試作の結果，時間変化を対象とした仮説推論が不完全な要求を入力として正しいサービス仕様を導出する方法として有効であることを確認した.

本稿は以下のように構成される．第2章で通信サービスにおける要求理解の定義をする．第3章で，当研究室で開発され使用されている通信サービス仕様記述言語 STR (State Transition Rule) について紹介し，STR を用いた要求および要求理解について説明する．また，STR の性質について述べ要求理解が状態の到達可能性解析問題に帰着できることについて説明する．第4章で具体的に到達可能性解析手法について説明する．通信サービスに対する曖昧さを含む要求を認識する手段となる，時間変化を対象とした仮説推論について述べ，処理方式を説明し具体的なシステムについて説明する．第5章で試作したシステムによる実施例を示し，第6章で評価を行う．評価は通信サービスのような状態遷移を扱うシステムに対する仮説推論が適切に行われているか否かという観点を中心に行う.

2 要求理解

通信サービスに対する要求理解とは，ユーザが何をしたいかを認識し，ユーザの目的とするサービス仕様を要求理解システムが獲得することである．一般に通信サービスに関わるすべての端末の状態をユーザが把握することは困難であり，通常は通信サービスに直接関与する実現して欲しい端末の状態を指定すると考えるのが妥当である．一方，対象としているサービス仕様記述によって実現できる端末の状態は，ユーザが指定した端末だけの状態では十分ではない場合がある．従って，要求には不完全さが含まれると考えられる．ここでは不完全さとして，曖昧·断片的な記述を考える．断片的とはサービス仕様の全てを入力するのではなく，サービス仕様を部分的に入力することである．また曖昧とは断片的に記述された仕様において，入力情報の一部が欠落していることである．このとき，要求理解を 「曖昧·断片的な入力から完全かつ正確なサービス仕様に変換すること」と定義する.

3 通信サービス記述手法

3.1 STR 記述手法の定義

本節で通信サービスを簡易に記述する手法として動作記述をルール形式 (STR:State Transition Rule) で記述する STR 記述 [6] について説明する. STR ルールは“現状態 イベント : 次状態. ”の形式で記述される. 図 1は STR ルールの例である. 図 1の一番目の規則 pots-1 は「端末 A が idle の時 端末 A が offhook すると, 端末 A は dial-tone を受ける状態に遷移する」 ことを規定している.

pots-1) idle(A) offhook(A): dial-tone(A).
pots-2) dial-tone(A), idle(B) dial(A, B): ringback(A,B), ringing(B,A).
pots-4) dial-tone(A), ~(idle(B)) dial(A, B): busy(A).
pots-5) ringback(A, B), ringing(B, A) offhook(B): path(A, B), path(B, A).

図 1: STR ルールの例

以下に本稿で使用される用語の定義をする.

[定義 1] STR ルール

動作の振舞いを 「現状態」, 「イベント」, 「次状態」で表したルール. 「現状態」および「次状態」は「状態記述要素」の集合で表される. 「状態記述要素」は端末の状態を表すプリミティブである. 「状態記述要素」が記号 ~ で括られているとき, その「状態記述要素」が存在しないことを表す. 「現状態」には ~ の記述が許されるが, 「次状態」では許されない.

[定義 2] STR ルールの適用

STR ルールの「現状態」が端末状態の部分集合となっているときその STR ルールはその端末に適用可能である.

[定義 3] STR ルールの包含関係

STR ルール R_1 の「現状態」が STR ルール R_2 の「現状態」の部分集合となっているとき, R_1 は R_2 に包含されると言う. 包含する STR ルールと包含される STR ルールが共に適用可能な端末状態では包含する STR ルールを適用する.

[定義 4] サービス

ルール形式で記述される STR ルールの集合であり適用される順番が規定されている.

[定義 5] 不完全なサービス

サービスの条件を満たしていない STR ルールの集合.

上述の形式で表される通信サービスは端末状態の遷移経過を表現することができ, 次のような特徴を有する.

[性質1] STRルールは同じ「状態記述要素」を含む端末状態から動作の違いによって異なる状態への遷移を規定する．端末状態を分析し遷移後の状態を特定することを意味する．

[性質2] STRルール間には時間に関しての明確な定義はないが，STRルールを適用して遷移する各時点間の端末状態には時間的順序関係がある．

[性質3] 端末状態の順序集合によって一つの通信サービスが定義され，その時に適用されたSTRルールの順序集合によってサービス仕様が決定される．

3.2 通信サービスに対する要求

通信サービスに対する要求として部分的サービス仕様である不完全なSTRルールの部分集合を考える．この要求に対し，既存STRルール集合により完全なサービス記述を求めることで，要求理解を行う．部分的サービス仕様である不完全なSTRルールの部分集合として，本稿では「状態記述要素」の集合の組を対象とする．例えば ({idle(a)}, {path-passive(a, b)}) である．この「状態記述要素」の集合の組を入力としたとき，その状態間の遷移を実現する完全なSTRルールを求めることとする．

前章で説明した要求の定義をSTR記述に対応させると以下のようになる．曖昧な要求記述とはある時点の状態から次の時点の状態への遷移記述で「状態記述要素」が欠落している場合である．また，断片的な要求記述とは，状態遷移が記述されてはいるが初期状態から状態遷移をして初期状態へ戻る完全な遷移が指定されていない場合である．図2にSTR記述による曖昧な要求記述例および断片的な要求記述例を示す．図の(1)に曖昧な要求記述を示す．要求記述のうち状態の記述に欠落がある場合を示している．一つのSTRルールで記述される場合と複数のSTRルールで記述される場合が存在する．(2)に断片的な要求記述を示す．3つの状態遷移を表す記述はそれぞれ正しいが，STRルールの R_{i+1} に対応する状態遷移記述がないため初期状態から初期状態へ戻る遷移が完全には指定されていない場合を示している．

また，要求理解の定義を，STR記述に対応させると，以下のようになる．

(1) 「現状態」，「次状態」，あるいは，「イベント」の一部情報が欠落して入力された不完全なSTRルールから，完全なSTRルールに変換する．

(2) 入力されたSTRルールの部分集合から，通信サービスを記述するのに必要かつ十分なSTRルールの集合を生成する．

3.3 要求理解が状態の到達可能性解析問題に帰着可能な理由

図2は要求記述に状態記述の欠落がある場合を示している．曖昧な要求として状態 s_1, s_2, s_4 と s_1, s_5 が与えられたとき，この状態間の遷移を規定するSTRルールがデータベースに存在すれば一つの完全なSTRルールで記述でき，曖昧な要求から完全な仕様を求めたことになるので要求を理解したことになる．また，一つのルールでは要求を満たすことができない時，複数のルールによって完全なルールを求められる場合がある．すなわち一つの要求状態があるSTRルールの現状態と一致し，他の状態が別のあるSTRルールの次状態と一致すれば，その間に適用されるルールを求めることで完全な仕様を求めることができる．一

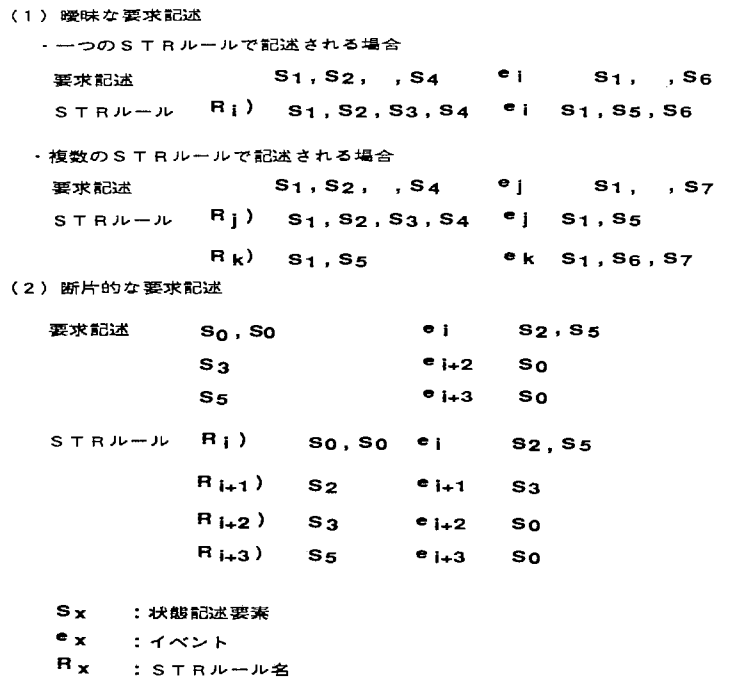


図 2: 曖昧，断片的な要求記述例

方，到達可能性解析問題は「指定したある状態から別の指定したある状態に到達可能か否かを判定すること」であり，状態遷移が規則で規定されている場合判定の結果として到達経路や適用規則等の情報を得ることが可能である．従って，ここで定義する要求理解は適用できるルールを求めることであるので，ある状態からある状態への到達可能性解析を行うことに他ならない．

4 到達可能性解析手法

到達可能性解析とは，ある状態が別のある状態から有限回の状態遷移を経て到達できるか否かを判定することである．本稿での到達可能性解析の目的は，要求理解システムを実現するために必要な要求の実現可能性の判定を行うことである．本章でははじめにSTR記述を対象として状態の到達可能性について議論し，その後状態遷移システムに対する不完全な状態指定に対処できる状態の到達可能性解析手法について説明する．

4.1 STR記述を対象とした状態の到達可能性解析

STR記述は不特定多数の端末を対象とした記述であり，指定された状態への到達可能性を判定するのに他の端末と相互に関連する場合が考えられる．従って，いくつの端末で構成されるシステムに対して指定した状態の到達可能性を判定すればよいかを決定することは困難である．この問題に対し，システムを構成する端末の数で定義されるシステムサイズと到達可能な状態との関係からあるシステムサイズで到達可能な状態が不変になるという性質を導き，そのようなシステムサイズのシステムで到達可能性の判定を行えばよいことを示した[14].

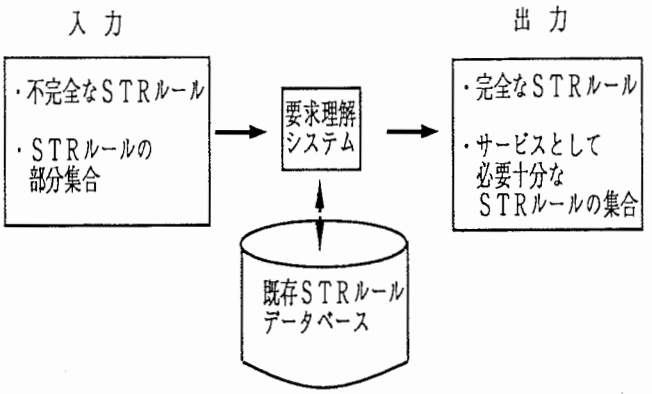


図 3: 要求理解システムイメージ

図4のような状態遷移規則が与えられたとき，状態 $(s_0(P_1), s_2(P_2))$ が状態 $(s_0(P_1), s_1(P_2))$ から到達可能か否かを判定するときにいくつの端末で構成されるシステムで到達可能性を判定すれば良いかを決定するために，端末数と到達可能状態との間に成立する性質を調べ以下のような性質を導いた．なお，以下で使用されている m は到達可能性の判定対象となる端末数と，与えられる状態遷移規則において定義される遷移する端末数の最大値のうちの大きい方の値である．なお，詳細については文献 [14] を参照されたい．

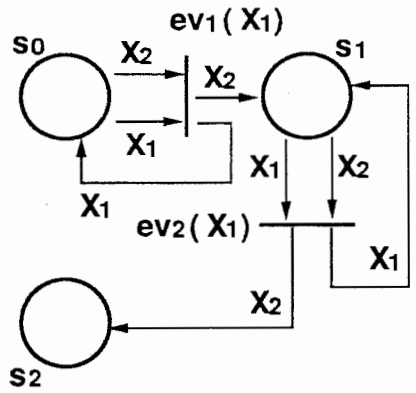
[性質] $R_{(t_1,\cdots,t_m)}(T_{m\times 2^i}) = R_{(t_1,\cdots,t_m)}(T_{m\times 2^{i+1}})$ かつ $R_m(T_{m\times 2^i}) = R_m(T_{m\times 2^{i+1}})$ を満たす値 i が存在するとき，$j > i$ において $R_{(t_1,\cdots,t_m)}(T_{m\times 2^i}) = R_{(t_1,\cdots,t_m)}(T_{m\times 2^j})$ が成立する．これは，$m \times 2^i$ 個のプロセスで構成されるシステムで $(t_1,\cdots,t_m)$ から到達可能な状態と $m \times 2^{i+1}$ 個の端末で構成されるシステムで $(t_1,\cdots,t_m)$ から到達可能な状態とが等しくなった時到達可能な状態が全て現れることを示している．ここで，$R_{(t_1,\cdots,t_m)}(T_{m\times 2^i})$ は，$m \times 2^i$ 個の端末で構成されるシステムにおいて $(t_1,\cdots,t_m)$ から到達可能な状態の集合を表す．$(t_1,\cdots,t_m)$ は一番目の端末の状態が t_1，m 番目の端末の状態が t_m である m 個の端末の状態を示している．

図5の (a) に，図4に対応するペトリネットモデルを示す．図5の (b) における P_i はカラートークンを示し端末を意味している．本稿で扱うシステムでは，端末の初期状態を表す s_0 のプレースに任意個存在することが許される．これはカラーの数に制限のないことを意味している．有限なカラーに対する到達可能性問題は従来扱われている [9][10][11] が制限のないカラーに対する到達可能性は検討されていない問題である．

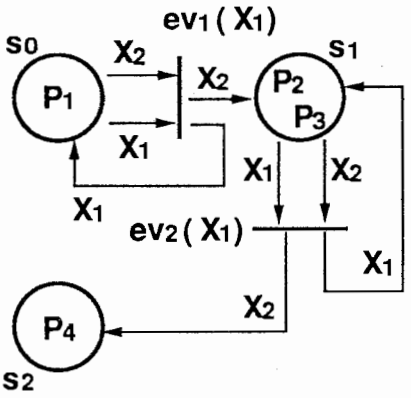
$\mathbf{rule_1)}$ $\quad \mathbf{s_0(X_1), s_0(X_2)} \quad \mathbf{ev_1(X_1):} \quad \mathbf{s_0(X_1), s_1(X_2)}$

$\mathbf{rule_2)}$ $\quad \mathbf{s_1(X_1), s_1(X_2)} \quad \mathbf{ev_2(X_1):} \quad \mathbf{s_1(X_1), s_2(X_2)}$

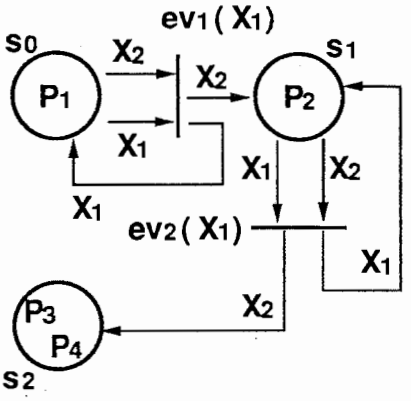
図 4: STR ルールの例



(a) 図2に対応するペトリネットモデル



(b) あるシステム状態を示した
ペトリネットモデル



(c) (b)の次の状態を示すペトリネットモデル

図 5: ペトリネットモデルによる表現

4.2 到達可能性解析への要求事項

本節で，到達可能性解析に対する必要な機能について説明する．

前章で説明したように，要求として不完全な記述を許容しているので，要求理解に到達可能性解析を適用する場合，不完全な記述を扱える到達可能性解析を行う必要がある．そのために，仮説推論を用いた到達可能性解析を行う．

また，STR ルールデータベースには，包含関係を有する STR ルールが存在する可能性がある．これは，STR ルールデータベース内には，全体として一つのサービスが格納されているわけではなく，個別のサービスに対応する STR ルールが格納される場合があるためである．つまり，個別のサービスとしては完全なサービス記述が STR ルールを用いて定義されていても，別の個別サービスを記述する STR ルールとの間で，包含関係を有する STR ルールが存在する可能性がある．包含する STR ルールと包含される STR ルールが共に適用可能な状態で，包含される STR ルールを適用すると正しい推論が行われない．従って，

推論過程で，包含されるSTRルールを適用したのか，包含するSTRルールを適用したのかの区別をする必要がある．

4.3 仮説推論を用いた到達可能性解析手法

曖昧さが含まれる要求を処理するために，通常の到達可能性解析に，仮説を扱う機能を追加した．何を仮説とするかについて説明する．ここでは，適用するルールに存在する「状態記述要素」を仮説としている．はじめから，ルールに存在しない「状態記述要素」を仮説として選択することも可能であるが，その場合，何が適切な選択かの基準がなく，考えられる状態の数が爆発する．また，たとえ，選択したとしても，もし，同じ「状態記述要素」が仮説としてたてられるとき，本来はそちらを採用すべきなのだが，矛盾となってしまい，正しい推論が行われなくなる．

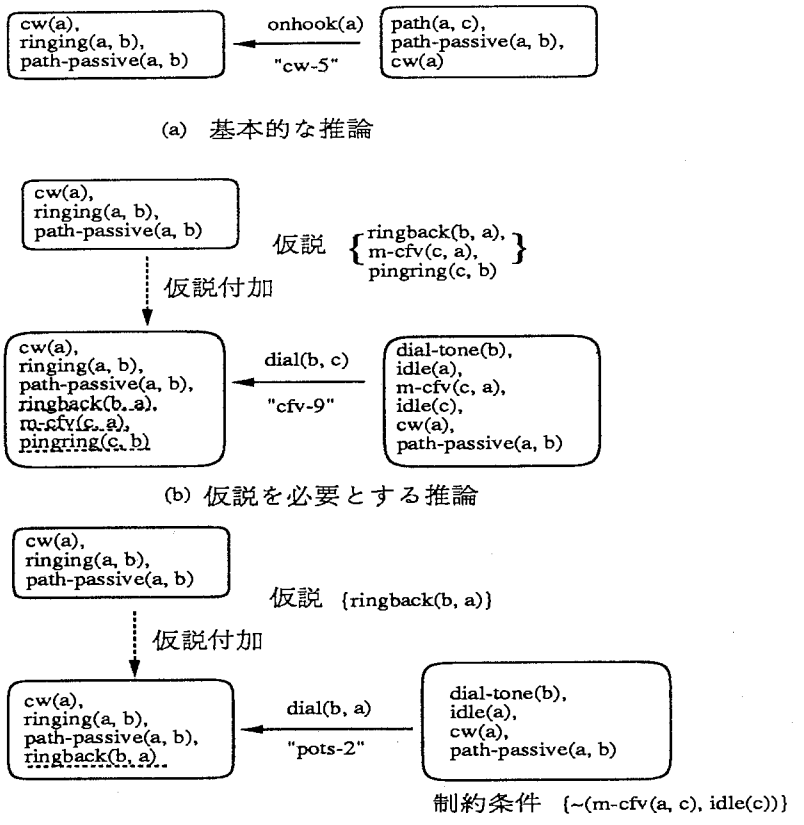
STRルールデータベースと，二つの指定状態S_1，S_2が与えられた時，STRルールの範囲内で，S_1がS_2から到達可能か否かを解析するアウトラインについて示す．第3章の性質1および性質2に示したように，動作を表すルールの「現状態」にマッチする状態が多く存在する．従って，後向きに推論をする方が効率が良くなると考えられるので，ここでは推論方式として，後向き推論を採用する．S_1を実現する一つ前の状態をSTRルールを基に後向き推論をする．そして，S_2に到達するまで，この推論を繰り返す．最終的に指定した端末の状態がS_2で指定した状態になれば，S_1がS_2から到達可能であると判断する．

基本的には，指定された状態S_1とSTRルールの「次状態」がマッチするSTRルールをサーチしていく(図6 (a))．

次に，仮説を必要とする推論(図6(b))について示す．仮説推論は，完全な知識，すなわち，真偽がはっきりした知識のみで与えられたゴールの説明ができない時，ゴールを説明できる仮説を見つける推論である．仮説推論においては，仮説とすることができるものの中からどれを仮説とするかという仮説選択問題がある．本方式では，仮説としては，すべての「状態記述要素」を対象としている．仮説の選択については，対象とするSTRルールを基にする．現在のシステム状態に，「状態記述要素」を仮定することで，あるSTRルールの適用が可能となり，一つ前の状態を導くことができる時，この仮定される「状態記述要素」が仮説である．指定された状態S_1がSTRルールの「次状態」とマッチはしないが，その状態を「次状態」に含むSTRルールがある場合，そのSTRルールに不足している「状態記述要素」を仮説推論を進める条件として付加して，一つ前の状態を導く．

包含されるSTRルールを適用する推論(図6(c))の場合は，包含されるSTRルールが適用される条件として，包含するSTRルールの「現状態」にあって，包含されるSTRルールの「現状態」にない「状態記述要素」が存在しないということを仮説推論を進める条件として付加する．本稿では，この条件を規則包含に起因する制約条件と呼ぶ．

図6は，指定された状態S_1が“cw(a), ringing(a, b), path-passive(a, b)”の場合の推論の具体例である．



STRルール

```
cw-5)  path(A, C), path-passive(A, B), cw(A)
                onhook(A):
          cw(A), ringing(A, B), path-passive(A, B).
cfv-9)  dial-tone(A), idle(C), m-cfv(B, C), idle(B)
                dial(A, B):
          ringback(A, C), ringing(C, A), m-cfv(B, C), pingring(B, A).
pots-2) dial-tone(A), idle(B)
                dial(A, B):
           ringback(A, B), ringing(B, A).
```

図 6: 推論の例

4.3.1 状態の矛盾判定について

推論過程で導かれる状態の矛盾判定について述べる．前述したように，不完全さを補完するため，「状態記述要素」を仮説として加えて推論する．このとき，選択された仮説をたてることができるか否かを判定する仮説検証が必要である．

一つの状態として共存できるか否かの検証に，論理的な矛盾，および意味的な矛盾を用いる．論理的な矛盾とは，ある「状態記述要素」の存在と非存在が同時に主張されている場合である．意味的な矛盾とは，通信サービスにおいて意味的に一つの状態として共存できない「状態記述要素」が存在する場合である．また，一つの状態に，同じ「状態記述要素」が複数存在する，「状態記述要素」の重複の場合も意味的な矛盾とする．例えば，一つの状態に，“ringback(a, b), ringback(a, b)” のような組合せが存在する場合は矛盾である．

推論過程で一度導出された「状態記述要素」は，その重複が許されないため，後の推論過程で仮説とす

$$\text{path(A, B)} \overset{\times}{\leftrightarrow} \text{ringing(B, A)}$$
$$\text{path(A, B)} \overset{\times}{\leftrightarrow} \text{busy-dial(B, A)}$$
$$\text{path(A, B)} \overset{\times}{\leftrightarrow} \text{dial-tone(B)}$$
$$\text{idle(A)} \overset{\times}{\leftrightarrow} \text{busy-dial(B, A)}$$
$$\text{idle(A)} \overset{\times}{\leftrightarrow} \text{path(B, A)}$$
$$\text{ringing(A, B)} \overset{\times}{\leftrightarrow} \text{busy(B)}$$
$$\text{ringing(A, B)} \overset{\times}{\leftrightarrow} \text{hangup(B)}$$
$$\text{ringback(A, B)} \overset{\times}{\leftrightarrow} \text{hangup(B)}$$
$$\text{cw-ringing(A, B)} \overset{\times}{\leftrightarrow} \text{busy(B)}$$
$$\text{path(A, B)} \overset{\times}{\leftrightarrow} \text{path-passive(A, B)}$$
$$\text{path(A, B)} \overset{\times}{\leftrightarrow} \text{ringing(A, C)}$$
$$\text{cw-ringing(A, B)} \overset{\times}{\leftrightarrow} \text{dial-tone(A)}$$
$$\text{cw-ringing(A, B)} \overset{\times}{\leftrightarrow} \text{cw-ringing(A, C)}$$

図 7: 矛盾する状態の組合せ例

ることはできない．また，矛盾する状態の組合せとなる「状態記述要素」も同様である．さらに，規則包含に起因する制約条件についても，それと矛盾する「状態記述要素」が後の推論過程で仮定されると，制約条件が発生した時の世界で矛盾が生じることになるため，仮説とすることはできない．

4.3.2 通信に関する知識

一般に，すべての解を列挙して，その中から目的に合った解を取り出すという手順は，列挙すべき解が組合せ的に爆発するので，現実的には実行できない．従って，無駄な計算を省いた，効率良い解の探索手法が必要である．ここでは，通信の知識を使用し，探索の効率化を図る．

本システムでは，通信に関する知識を使用することにより冗長，無駄な探索の除去を行っている．通信の知識としては，グローバル状態の組合せ矛盾知識を使用する．これは，二つのプリミティブ状態に対して，次のような基準で矛盾状態と規定した状態の組合せである．

1. 端末 a,b 間がパスの状態の時は，他の状態と共存しない．

2. アイドル状態とそうでない状態は共存しない．

図 7に例を示す．

4.4 時間変化を推論の対象とする仮説推論

仮説推論は従来いろいろな分野で適用されているが，状態遷移システムという，時間変化を推論の対象とした場合についての利用は行われていない．また，本稿で扱っている状態遷移記述は，複数の端末の状

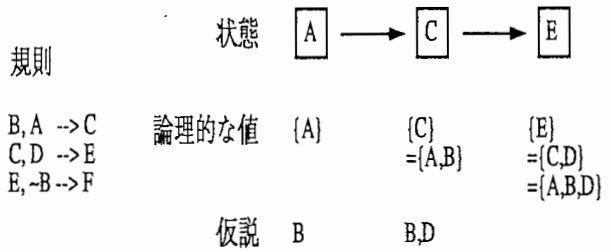


図 8: 一階述語論理推論

態の変化を同時に定義するものであり，一階述語論理の範疇で記述できる世界ではない．一階述語論理，あるいは，さらに制限したホーン節の範囲では，仮説推論における効率化を対象とした研究が行われている．しかし，時間変化を扱う世界での仮説推論については対象とされていない．従って，時間変化を扱うために，論理を対象とした推論とは異なる，状態の矛盾の取り扱い方法を考案する必要がある．

以下に，論理の世界と状態遷移の世界との違いについて説明する．状態遷移の世界では，各状態は時間的に異なっている．従って，状態に対し，仮説をたてた場合，元の状態にその仮説を加えた世界において矛盾が生じるか否かをチェックする必要がある．一方，論理の世界では，規則が適用されて，状況が変わったとしても，一つの閉じた世界での変化であるので，その時点の状態について，矛盾のチェックをすればよいことになる．

仮説を導入した一階述語論理推論の例を図 8に示す．A，B，C，D，E，F をある「状態記述要素」とする．A に対して B を仮定することで一番目の規則が適用でき，C が導出される．この時点で，A，B，C のすべてが論理的に真であるとみなされる．次に，D を仮定することにより，二番目の規則が適用でき，E が導出される．この時点で，A，B，C，D，E のすべてが論理的に真であるとみなされる．ここで，三番目の規則を適用するためには，~B を仮定しなければならないが，論理的に矛盾を起こすので，仮定することはできない．このように，一階述語論理推論の場合は，導出された状態は，過去のすべての事実を反映している．

これに対し，通信サービスのような状態遷移システムでは，導出される状態はそれぞれ独立である．従って，仮説を用いて状態を導出した時は，元の状態に仮説を加えた世界を考えることになる．そして，一つの仮説の導入により，各時点での広げられた世界における無矛盾性を検証する必要がある．例えば，図 9において，1 から 3 のように 仮説 “m-cfv(c, b), ~(idle(c))” をたてることにより，2 の状態が導かれるが，この仮説の導入によって，3 の状態に矛盾が生じないか否かをチェックすることが必要である．

4.5 解析例

図 10に “idle(a),idle(b),idle(c)” の状態から “path(b, c), path(c, b), busy(a)” の状態へ到達可能か否かを判定する例を示す．解析結果は，同図に示すように，“path(b, c), path(c, b), busy(a)” の状態から，2→3→4→7→8 と推論し，“idle(a), idle(b), idle(c)” の状態になるので，到達可能と判定される．

[矛盾チェックの説明]

1 の状態に対し，STR ルール pots-4 が適用できる．pots-4 は，cw-1 に包含されるので，規則包含に起

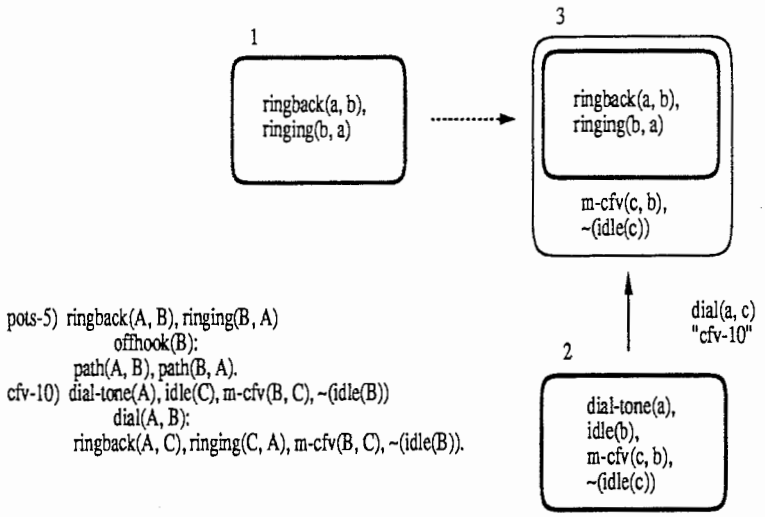


図 9: 仮説世界の説明図

因する制約条件として，~(m-cw(b)) が設定される．従って，以降の推論では，仮説 m-cw(b) をたてることはできない．2 の状態 path(b, c) に対し，STR ルール cw-1 を適用しようとする時，仮説 "m-cw(b), ringback(d, b), cw-ringing(b, d)" が必要であるが，~(m-cw(b)) と矛盾するため cw-1 は適用できない．3 から 6 を導出する時に，"cw-ringing(b, c)" の存在を仮定したことと，"cw-ringing(A, B)" と "cw-ringing(A, C)" が矛盾する状態の組合せとして登録されていることから，"cw-ringing(b, e)" も以降の推論で仮説とすることはできない．6 の状態 "path(b, d), m-cw(b)" に対して，STR ルール cw-1 を用いる時は，仮説 "cw-ringing(b, e), ringback(e, b)" が必要となるが，~(cw-ringing(b,e)) と矛盾するため，cw-1 は適用できない．5 の状態で，m-cw(b) が導出される．2 で設定された ~(m-cw(b)) と一階述語論理の世界では矛盾するが，状態遷移システムにおいては異なる時間での状態であり，~(m-cw(b)) がこれと矛盾する仮説をたててはいけない条件として扱うことで，矛盾は起こさない．

4.6 到達可能性判定システム

4.6.1 概要

状態の到達可能性解析を利用した通信サービスに対する要求理解システムの概要について説明する．

本到達可能性解析システムは，STR 記述仕様に則って記述された STR 規則を STR ルールデータベースとしたとき，二つの状態を指定し，その状態間の遷移を実現することが可能か否かを判定するシステムである．到達可能性解析は前章までに説明したように，仮説推論を用いた後向き推論によって行い，状態およびイベントをノードとする到達木を作成する．作成される到達木により，到達可能と判定された場合，どのような状態遷移か，どの規則が使用されたか，どのような仮説を必要としたか，を求めることができ，与えられた曖昧さを含んだ要求を実現するサービス仕様を生成することが可能となる．

本システムの入出力を以下に示す．

- 入力

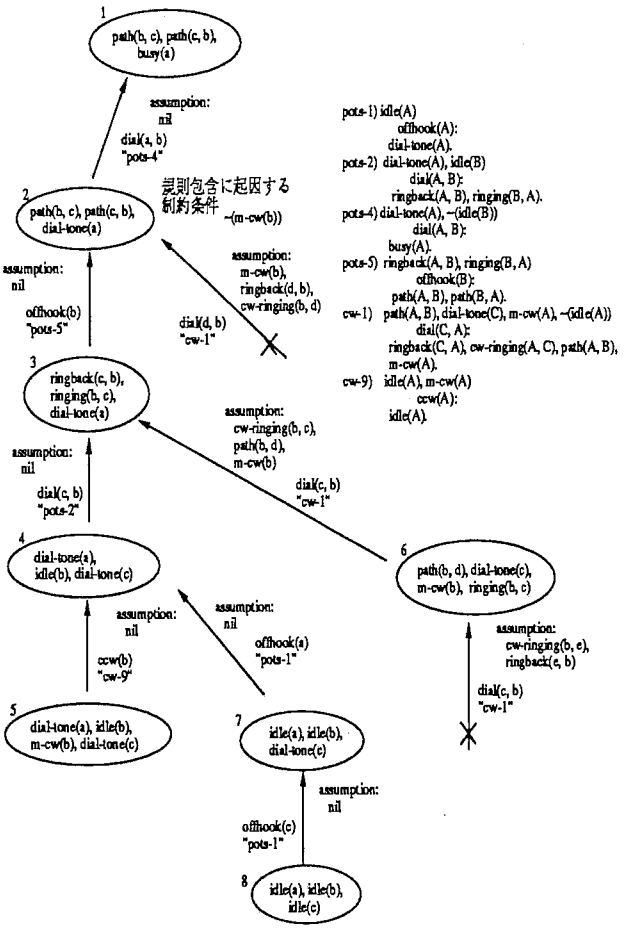


図 10: 解析例

- 到達可能か否かを求めようとする二つの端末の状態.
- 実現されては困る状態.

- 出力
  - 要求を満足するルール
  - 適用順を決定したサービス仕様
  - 補完された要求

4.6.2 システム構成

システムの内部構造について説明する．グローバルな状態を表した状態およびイベントをノードとする木構造で表現する．状態ノードは，次の要素から構成される (図 11)．これらは，時間変化を対象とした推論を実現する上で必要となる形態である．特徴的なのは Restriction と， History の属性である．

- State

  現在の端末のグローバル状態を示す．適用される STR ルールの「現状態」のインスタンスと，ルー

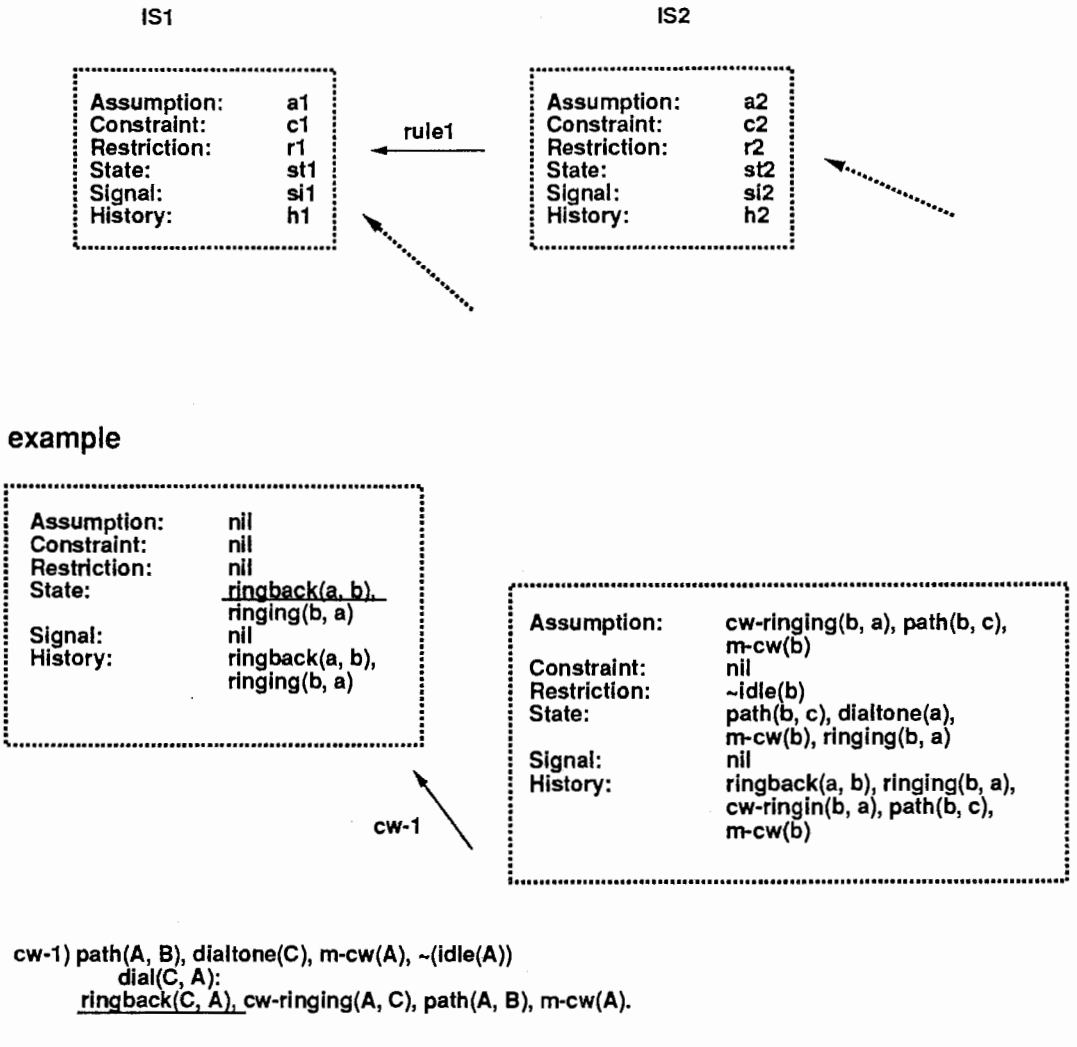


図 11: 状態ノードデータ構造

ル適用以外のプリミティブ状態より構成される.

- Assumption

  一つ前の状態導出に必要な仮説プリミティブ状態を保持する. ここに示されたプリミティブ状態を仮定することにより, この時点のグローバル状態が導出される.

- Constraint

  包含関係に起因する制約条件. あらかじめ STR 規則が与えられた時, 規則の包含関係を保持し, それを基に生成する. 詳しくは, 到達可能性判定システム仕様書を参照されたい. 仮説としてたてることのできないプリミティブ状態となる.

- Restriction

  仮説としてたてることのできないプリミティブ状態を保持する. Constraint データと, State データから, 計算する. また, 一つ前の状態ノードの Restriction データと, 現在の Assumption データから計算する. 詳しくは, 矛盾の取り扱いで述べる.

- Signal
  内部信号が発生していることを示す.

- History
  現在までに存在しているとみなされるプリミティブ状態. 以後の推論で, 仮定されてはいけないプリミティブ状態を表す. たてようとする仮説によって矛盾が生じるか否かを判定するのに使用される.

4.6.3 矛盾の取り扱い

状態ノードの Attribute 値による矛盾のチェックについて説明する.

- a2 と r1 との間でチェック
  仮説 a2 をたてることができるか否かを, r1 との間でチェックする.

- a2 と h1 との間でチェック
  仮説 a2 をたてることができるか否かを, h1 との間でチェックする. 例えば, cw-ringing(b, d) を仮定しようとした場合, History データとの矛盾チェックで, cw-ringing(b, d) と cw-ringing(b, a) とが, 共存できないので, この仮説はたてることはできない. 仮説をたてることができるか否かのチェックは, 共存可能なデータか否かを判定することによって行う. 共存可能なデータか否かは, 一つの端末が取り得る状態のデータベースを参照して求める.

- a2 と r1 との間で演算を行いチェック
  r1 として ~(A, B, C) が設定されていたとき a2 として A をたてる場合を考える. r1 として ~(A, B, C) が設定されていることの意味するものは, A, B, C 共に存在することを否定するものであり, すべてが同時に仮定されなければ問題はおこらない. 従って, A を仮説としてたてることは可能であり, その時点で, r2 には, ~(B, C) が設定される.

- st2 と c2 との間で演算を行いチェック
  Constraint データとして ~(A, B) の形式のデータが設定される時

  - State データとの演算をして, 結果を Restriction に設定する.
    ~(A, B)·A → (~A + ~B)·A → ~B·A
    ~B を Restriction データとする.
    ~(A, B)·(A, C) → (~A + ~B)·(A, C) → ~A·(A, C) + ~B·(A, C) → ~B·(A, C)
    ~B を Restriction データとする.
  - ~(A, B) が Restriction データに設定された後, A or B が仮定される場合
    (~A + ~B)·A → ~B·A
    ~B を Restriction データとする. このように, A または B が仮定されるのは可能. A(B) が仮定された後, B(A) を仮定することはできない.

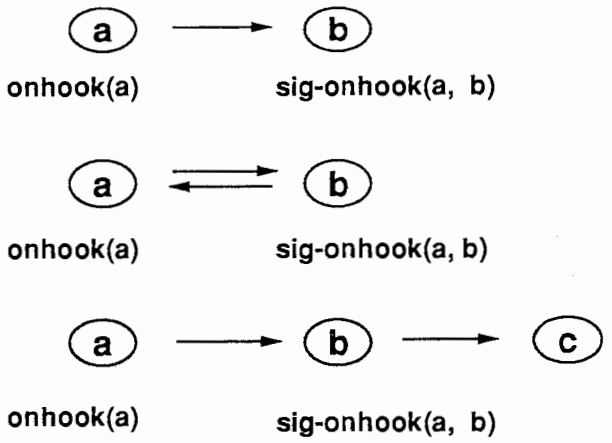


図 12: onhook と sig-onhook の関係

4.6.4 onhook 規則

本節で，onhook 規則について説明する．onhook 規則の詳細については，テクニカルレポート [7] を参照されたい．

onhook 規則について，簡単に以下に説明する．図 12に示すように，端末 a と関係のある端末に対して，内部信号が発生される．

例えば，次の STR 規則が存在するとき，

pots-6) path(A, B) onhook(A): idle(A).

pots-7) path(A, B) sig-onhook(B, A): busy(A).

“path(a, b), path(b, a)” という状態において，端末 a が onhook するとき STR 規則 pots-6 が適用され，端末 a は idle 状態になり，端末 b に内部通信信号 sig-onhook(a, b) の受信が生じ，端末 b は busy 状態となる．

onhook 規則については，内部信号を扱うため，局所的な推論を行うことになる．つまり，一つの onhook 規則，あるいは，sig-onhook 規則が適用された段階では，グローバルな端末状態は不安定である．

onhook 規則の処理を以下に示す．

1. onhook 規則の右辺がユニファイ可能なルールを選択する．
   onhook 規則の処理中であることを示す inner-signal が設定されていない状態を対象とする．

2. 関係する端末に内部信号を設定．ただし，Delivery-Range で宣言されている状態に対しては必要ない．
   onhook した端末を A，関係する端末を X としたとき，inner-signal(A, X) を設定．

3. sig-onhook 規則の右辺がユニファイ可能なルールを選択
   inner-signal(A, B) と，sig-onhook(A, B) とをユニファイさせる．inner-signal を消去

4. 全ての inner-signal がなくなるまで 3 を繰り返す．

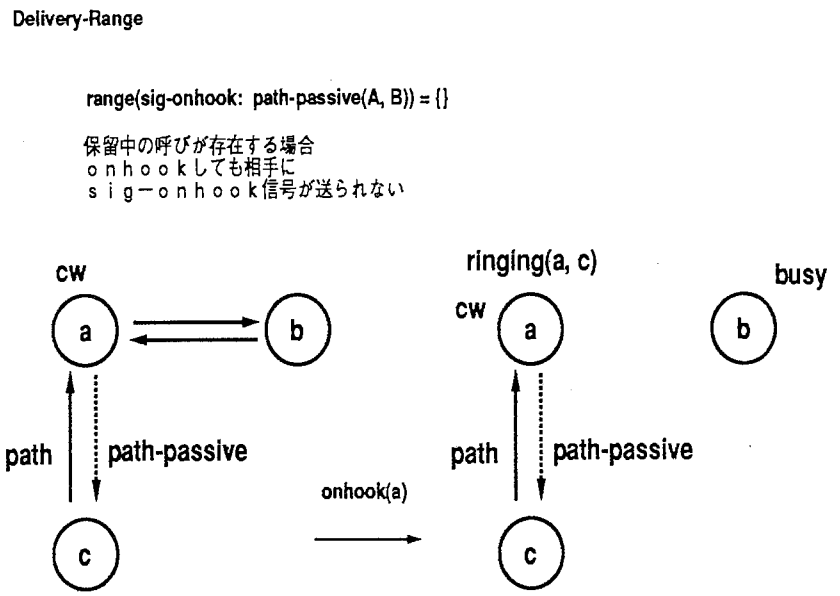


図 13: Delivery-Range の説明

5. 局所推論終了.

ここで, onhook 規則を先に適用する理由を列挙しておく.

- sig-onhook 規則には, empty が存在し, すべての端末について, 成立する. 従って, すべての端末に適用しなければならない. 状態爆発の要因となる.
- sig-onhook は, いくつ適用されるかわからない. empty のものについて全て適用できる. 無駄な探索を行っている可能性がある.
- onhook 規則を先に後向き推論で適用することにより, 状態遷移のうち不安定な状態 (inner-signal が設定されている状態) では, 表現する状態が前向き推論を適用した場合と, 異なる. しかし, onhook 規則により変化する端末と, 関係する sig-onhook 規則により変化する端末は, 互いに独立になっている為, 適用順によらず正しい推論を行うことができる.
- onhook 規則の適用により, onhook する端末を特定できる.

4.6.5 禁止状態集合の取り扱い

禁止状態集合の宣言は, 一つのプロセスの状態記述に指定された集合が含まれることを禁止するものである.

Inhibited-Primitive-Sets:
{cw-ringing(A, B), cw-ringing(A, C)}(busy)

cw-1) path(A, B), dial-tone(C), m-cw(A), not[idle(A)]

dial(C, A):

ringback(C, A), cw-ringing(A, C), path(A, B), m-cw(A).

端末 A が話中着信を受けている状態で，他の着信を受けることが可能であり，複数の呼びの着信を許してしまう．これを禁止するために上記の禁止状態集合が宣言される．

後向き推論をする時の取り扱い方として，STR ルールから，あらかじめ，cw-ringing(A, B), cw-ringing(A, C) の状態になる動作を起こした端末がその動作で busy 状態になるような規則を追加しておく．すなわち，この場合は，以下の規則を追加する．

path(A, B), dial-tone(D), m-cw(A), not[idle(A)], cw-ringing(A, C)

dial(D, A):

busy(D), path(A, B), m-cw(A), cw-ringing(A, C).

禁止状態になったとき，その動作を起こした端末を busy 状態にする場合の追加する規則の生成について以下に記す．

(a) 禁止状態集合の一つのプリミティブ状態を次状態に含む STR ルールを抽出．

(b) 抽出したルールの現状態に，禁止状態集合の他の一つのプリミティブ状態をあわせたものを新しいルールの現状態とする．

(c) 抽出したルールを適用することにより，取り除かれるプリミティブ状態をのぞいた現状態を次状態におく．

(d) 動作を起こす端末に対して，busy 状態とし，次状態に追加する．

4.6.6 timeover イベントの処理

timeover イベントはイベント記述が他のルールと異なるが，通常の処理と同様に行う．

pots-t-1) dial-tone(A)

timeover(dial-tone(A)):

busy(A).

5 実施例

付録に断片的な要求に対し，到達可能性解析を行った実施例を示す．付録 1 は，対象とした STR ルール (POTS と CW) である．この STR ルールデータベースにおいて，要求を ({idle(a), idle(b), idle(c)}, {path(b, c), path(c, b), busy-dial(a)}) としたときの解析結果が付録 2 である．内部データ構造の詳細については，到達可能性判定システム仕様書を参照されたい．

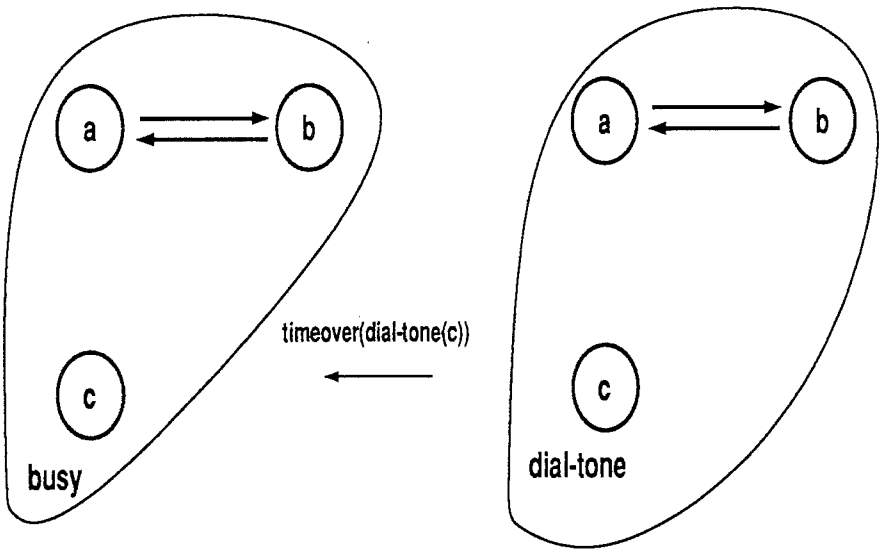


図 14: timeover イベントの処理

解析結果の到達木のノード種別の意味は以下の通りである.

- Restriction
  Restriction データとの間で矛盾が生じた状態
- Double
  同一のプリミティブ状態が複数存在したことによる矛盾状態
- Coexistence-h&a
  仮説をたてるとき, History データとの間で矛盾が生じた状態
- Coexistence-els
  State データとして実現できない状態が存在した矛盾状態
- Global
  グローバル状態の組合せ矛盾知識により矛盾となった状態
- Same
  同一の状態ノードが既に存在することにより推論を止めた状態
- End
  要求を満たす目標状態に到達した状態
- Nothing
  内部通信信号が存在している状態とユニフィケーションしたが, イベントとユニフィケーションできない状態
- Next
  矛盾を起こしていない状態で推論を続けることのできる状態

表 1: 推論の深さに対する状態ノード数の変化

深さ	矛盾状態数	目標状態数	総状態数	到達率
2	82	0	123	0
4	516	0	736	0
6	1325	21	1810	1.56
8	2058	52	2736	2.46
10	2672	60	3560	2.20
12	3588	60	4842	1.65
14	5309	60	7326	1.12

- Total
  到達木に存在する全状態ノード数

6 評価および課題

6.1 要求理解の立場からの評価

通信サービスに対する要求として，二つの指定状態を与えた時，それを実現する STR ルールの集合を求める場合を考える．本稿で示した仮説を使用する推論手法により，要求を実現する STR ルール集合を求めることが可能となる．評価として，仮説を使用した場合と使用しない場合とを比較し，仮説を使用することが必要であることを示す．

次の要求を考える．

[要求] ({idle(a), idle(b), idle(c)}, {path(b, c), path(c, b), busy(a)})

STR ルールデータベースを Call waiting と POTS としたときの，推論の深さに対する状態ノード数の変化および到達率の変化を表 1 に示す．

表の矛盾状態数は，推論過程で状態が矛盾であると判定された状態ノード数である．これには，既に作成された同じ状態，仮説をたてることができないと判定された状態，通信の知識を用いて矛盾状態であると判定された状態が含まれている．このように状態を矛盾であると判定することにより，無駄な探索，冗長な探索の除去ができ，探索の効率化が図られている．

仮説を使用することによって，要求解析に貢献することを，表 2 に示す．

表 2: 仮説の使用による目標状態への到達状況

仮説	深さ	目標状態数	総状態数
使用	6	21	1810
	8	52	2736
未使用	6	0	34
	8	0	34

表 3: 通信知識の有無による状態数の比較

知識	深さ	状態数	目標状態数
有り	2	110	1
	3	297	2
無し	2	212	1
	3	1194	4

6.2 通信知識の有効性

表3に STR ルールデータベースとして Call waiting と POTS を対象としたとき，要求 ({dial-tone(a), idle(b), m-cw(c)},{path(b, c),path(c, b), busy(a)}) において推論時に生成される状態数の，通信の知識を用いた場合 と 用いない場合との比較を示す.

このような基本的な知識を使用するだけでも，表に示すように，かなりの割合で探索空間を減らすことが可能である.

6.3 課題

- 本稿で示した状態の到達可能性手法については，組合せ爆発が起こってしまう．これを起こさないようにする方法を検討することが必要である．通信サービスという領域の性質を用いてそれを実現することを考える．つまり，通信サービスを初期状態からサービス状態の進行に沿ったフェーズ分けを考える．例えば，基本電話サービスにおいて，開始処理・相手特定処理・相手状況確認処理・パス接続処理・パス切断処理の 5 段階の処理フェーズが考えられる．その各フェーズに対応するように動作を表すルールを分割し，全てのルールを一度に対象としないようにすることで組合せ爆発の発生を押える.
- ユーザは予め起こって欲しいサービス状態，あるいは，起こって欲しくないサービス状態を全て把握しているわけではない．複数の可能性がある場合，通常はより詳細に記述されている状態を優先するなど，優先順位をつけて探索をするが，通信サービスの場合は，機械的に決定す

ることは困難である．そこで，ユーザにどれを優先するかを決定してもらうことで，探索の効率化を図る．また，通信サービスに対する要求として，端末数も含める．これにより，決められた端末数の範囲で探索すれば良いことになる．また，要求として，実現して欲しい状態だけではなく，特にユーザの望まないサービス状態も含める．これにより，探索範囲を減少させることができる．

7 まとめ

本稿では，通信サービスのような状態遷移システムに対して，不完全な状態指定に対処可能な状態の到達可能性解析手法について述べた．

不完全な状態指定を扱うために，仮説推論を導入した．本稿で扱ったような通信サービスがルール形式で与えられている場合の仮説の選択，仮説の検証について明確にした．また，状態遷移システムと標準論理との違いを述べ，標準論理とは異なる状態の矛盾の検出法が必要なことを明らかにした．

上記の検討に基づいて試作したシステムについて，内部構造を説明し，処理上注意すべきことを述べ，試作システムを通し，本手法の評価を行った．仮説を使用することにより，不完全な状態指定を扱えることを確認したが，探索空間の拡大という問題が発生する．これに対し，通信固有の知識を適用することで探索空間を減少させることは可能であるが，実用システムとするためには，さらに検討が必要である．

参考文献

[1] 島袋 潤, 永松 裕嗣, 新保 勲, 大津 和之「交換サービス仕様の作成・検証支援システム」情報処理学会第４４回全国大会 4k-8, March 1992

[2] 長谷川 晴朗,「ペトリネットを用いた通信サービス仕様設計支援」電子情報通信学会論文誌 B-I Vol. J74-B-I No.6 pp.445-455, June 1991

[3] Howard B. Reubenstein and Richard C. Waters, "The Requirements Apprentice : Automated Assistance for Requirements Acquisition," IEEE Transactions on Software Engineering, Vol. 17, No. 3, pp. 226-240, March 1991

[4] Barry W. Boehm, "Guidelines for Verifying and Validating Software Requirements and Design Specifications," EURO IFIP 79, P. A. Samet (editor) pp. 711-719, North-Holland Publishing Company, IFIP, 1979

[5] 柴田 健次, 平川 豊, 竹中 豊文,「仮説推論を用いた状態の到達可能性解析手法とその通信サービスインタラクション検証への適用」電子情報通信学会交換システム研究会資料 SSE 91-178, March 1992

[6] Hirakawa, Y. and Takenaka, T., "Telecommunication service description using state transition rules," Proceeding of Sixth International Workshop on Software Specification and Design, pp. 140-147, Oct. 1991

[7] 平川 豊, 「STR(State Transition Rule) 記述仕様書」, ATR Technical Report, TR-C-0073 Jan. 1992

[8] 楢崎 修二, 堀田 英一, 「通信サービス記述のための知識と動作に基づく様相論理 SSL」 情報処理学会第４４回全国大会, 1992, 4K-9

[9] Jensen, K., "Colored Petri Nets and the Invariant-method," Theoretical Computer Science 14 pp.317-336, 1981

[10] Jensen, K., "Colored Petri Nets," Lecture Notes in Computer Science, Vol. 254, Springer-Verlag pp.248-299, 1987

[11] Peterson, J. L.,"A note on colored petri nets," Information Processing letters, Vol. 11, No. 1, Aug. 1980

[12] 国藤, 「仮説推論」 人工知能学会誌, vol.2 No.1 (1987. 3)

[13] 三宅 芳雄, 「理解と知識」 人工知能学会誌, Vol. 6, No. 6, Nov. 1991

[14] Shibata, K., Hirakawa, Y., and Takenaka, T., "Reachability analysis for a behavior description independent of the number of processes," IJWCC'90(IPSJ SIG-DPS Technical Report), pp. 85-93, July 1990

[付録1]STRルール(CW)

```
####################################
###  CW Service                  ###
###     New Description 19910611 ###
####################################
Primitives:
idle(A),dial-tone(A),busy(A),busy-dial(A,B),
path(A,B),hangup(A),r-path(A,B),ringing(A,B),
ringback(A,B),path-passive(A,B),
cw(A),m-cw(A),cw-ringing(A,B),
Events:
onhook(A),sig-onhook(A,B),offhook(A),dial(A,B),timeover(A),
flash(A),cw(A)
Limited-Time-Primitives:
dial-tone(A) 30sec
busy(A) 30sec
busy-dial(A,B) 30sec
cw-ringing(A,B),busy(A) 30sec
Internal-Signal-Deliveration:
onhook(A) --> sig-onhook
Macro-Primitives:
Calling(A,B)    = {ringback(A,B),ringing(B,A)}
Talk(A,B)       = {path(A,B),path(B,A)}
Busy(A,B)       = {(busy(A)|busy-dial(A,B))}
CW-calling(A,B) = {ringback(A,B),cw-ringing(B,A)}
Inhibited-Primitive-Sets:
{cw-ringing(A,B),cw(A)} (busy)
{cw-ringing(A,B),cw-ringing(A,C)} (busy)
Delivery-Range:
range(sig-onhook: busy-dial(A,B))    = {}
range(sig-onhook: path-passive(A,B)) = {}
range(sig-onhook: cw-ringing(A,B))   = {}
Rules:
### onhook
#pots-6)path(A,B) onhook(A): idle(A).
```

```
#pots-8)Busy(A,B) onhook(A): idle(A).
#pots-9)dial-tone(A) onhook(A): idle(A).
#pots-11)ringback(A,B) onhook(A): idle(A).
#pots-12)hangup(A) onhook(A): idle(A).
#cw-2)cw-ringing(C,A),path(A,B) onhook(A): ringing(C,A).
cw-2-mod)cw-ringing(A,C),path(A,B) onhook(A): ringing(A,C).
cw-5)cond:cw(A),path(A,C),path-passive(A,B)
onhook(A): ringing(A,B),path-passive(A,B).
cw-13)busy(A),cw-ringing(A,B) onhook(A): ringing(A,B).
#cw-15)cw-ringing(A,B),hangup(B) onhook(A): ringing(A,B).
#cw-15-mod)cw-ringing(A,B),hangup(A) onhook(A): ringing(A,B).
cw-15)cw-ringing(A,B),hangup(A) onhook(A): ringing(A,B).

### sig-onhook
pots-7)path(A,B) sig-onhook(B,A): busy(A).
pots-10)ringing(A,B) sig-onhook(B,A): idle(A).
cw-10)cw-ringing(A,B) sig-onhook(B,A): empty.
cw-6)path-passive(A,B),cw(A),cond:path(A,C)
sig-onhook(B,A): empty.
cw-7)path(A,C),path-passive(A,B),cw(A)
sig-onhook(C,A): path(A,B).
cw-12)ringing(A,B),path-passive(A,B),cw(A)
sig-onhook(B,A): idle(A).

### offhook
pots-1)idle(A) offhook(A): dial-tone(A).
pots-5)Calling(A,B) offhook(B): Talk(A,B).
cw-11)ringing(A,B),path-passive(A,B),cw(A)
offhook(A): path(A,B).

### dial
pots-2)dial-tone(A),idle(B) dial(A,B): Calling(A,B).
pots-3)dial-tone(A) dial(A,A): busy(A).
pots-4)dial-tone(A),not[idle(B)] dial(A,B): busy-dial(A,B).
cw-1)cond:path(A,B),dial-tone(C),cond:m-cw(A),not[idle(A)]
dial(C,A): CW-calling(C,A).
```

```
### flash
cw-3)CW-calling(C,A),path(A,B)
flash(A): cw(A),Talk(A,C),path-passive(A,B).
cw-4)path(A,C),path-passive(A,B),cond:cw(A)
flash(A): path(A,B),path-passive(A,C).
cw-14)busy(A),cw-ringing(A,B) flash(A): path(A,B).

### cw
cw-8)idle(A) scw(A): idle(A),m-cw(A).
cw-9)idle(A),m-cw(A) ccw(A): idle(A).

### timeover
#cw-t-1)busy(A),CW-calling(B,A)
# timeover(busy(A),cw-ringing(A,B)): Talk(A,B).
#pots-t-1)dial-tone(A) timeover(dial-tone(A)): busy(A).
#pots-t-2)busy(A) timeover(busy(A)): hangup(A).
#pots-t-3)busy-dial(A,B) timeover(busy-dial(A,B)): hangup(A).
```

[付録 2] 断片的要求と解析結果

[要求] ({idle(a), idle(b), idle(c)}, {path(b, c), path(c, b), busy-dial(a)})

[結果] 端末数 3, 推論の深さ 6 のとき,

```
Restriction       = 45
Double            = 325
Coexistence-h&a   = 330
Coexistence-els   = 365
Global            = 257
Same              = 159
End               = 21
Nothing           = 0
Next              = 536
Total             = 2038
```

[結果] 要求を満足する一つの到達木の内部データ

```
#:|state-7612|
#<Structure STATE F1C78E> is a structure of type STATE.
It has 9 slots, with the following values:
 BACKWARD:                    ROOT
 FORWARD:                     (#:|path-7613| #:|path-7616|
#:|path-7619| #:|path-7622|
#:|path-7625| #:|path-7628| #:|path-7631|)
 SUBSTANCE:                   (A B C)
 ASSUME:                      NIL
 CONSTRAINT:                  NIL
 RESTRICTION:                 NIL
 STATE:                       ((BUSY-DIAL A B) (PATH B C) (PATH C B))
 SIGNAL:                      NIL
 HISTORY:                     ((BUSY-DIAL A B) (PATH B C) (PATH C B))
#:|path-7613|
(#:|state-7614| #:|state-7615|)
#<Structure RULE E918C6> is a structure of type RULE.
It has 6 slots, with the following values:
 NAME:                        POTS-5
 VARIABLES:                   (?A ?B)
```

```
 EVENT:                          (OFFHOOK ?B)
 PRE-STATE:                      ((RINGBACK ?A ?B) (RINGING ?B ?A))
 POST-STATE:                     ((PATH ?A ?B) (PATH ?B ?A))
 INCLUDED-RELATIONS:             NIL
#:|state-7614|
#<Structure STATE F2C5AE> is a structure of type STATE.
It has 9 slots, with the following values:
 BACKWARD:                       #:|path-7613|
 FORWARD:                        (#:|path-7677| #:|path-7688|
#:|path-7699| #:|path-7708|
#:|path-7715| #:|path-7717| #:|path-7719|)
 SUBSTANCE:                      (A B C)
 ASSUME:                         NIL
 CONSTRAINT:                     NIL
 RESTRICTION:                    NIL
 STATE:                          ((BUSY-DIAL A B) (RINGBACK C B) (RINGING B C))
 SIGNAL:                         NIL
 HISTORY:                        ((BUSY-DIAL A B) (PATH B C) (PATH C B)
(RINGBACK C B) (RINGING B C))
#:|path-7719|
(#:|state-7720|)
#<Structure RULE E91906> is a structure of type RULE.
It has 6 slots, with the following values:
 NAME:                           POTS-4
 VARIABLES:                      (?A ?B)
 EVENT:                          (DIAL ?A ?B)
 PRE-STATE:                      ((DIAL-TONE ?A) (~IDLE ?B))
 POST-STATE:                     ((BUSY-DIAL ?A ?B))
 INCLUDED-RELATIONS:             ((< POTS-4 CW-1))
#:|state-7720|
#<Structure STATE EDBC6E> is a structure of type STATE.
It has 9 slots, with the following values:
 BACKWARD:                       #:|path-7719|
 FORWARD:                        (#:|path-8153| #:|path-8164|
#:|path-8175| #:|path-8184|
#:|path-8191| #:|path-8193| #:|path-8195|)
```

```
 SUBSTANCE:                    (A B C)
 ASSUME:                       NIL
 CONSTRAINT:                   (NOT (M-CW B) (PATH B C))
 RESTRICTION:                  ((NOT (IDLE B)) (NOT (M-CW B) (PATH B C)))
 STATE:                        ((DIAL-TONE A) (RINGBACK C B) (RINGING B C))
 SIGNAL:                       NIL
 HISTORY:                      ((BUSY-DIAL A B) (DIAL-TONE A) (PATH B
C) (PATH C B) (RINGBACK C B) (RINGING B C))
#:|path-8191|
(#:|state-8192|)
#<Structure RULE E91866> is a structure of type RULE.
It has 6 slots, with the following values:
 NAME:                         POTS-2
 VARIABLES:                    (?A ?B)
 EVENT:                        (DIAL ?A ?B)
 PRE-STATE:                    ((DIAL-TONE ?A) (IDLE ?B))
 POST-STATE:                   ((RINGBACK ?A ?B) (RINGING ?B ?A))
 INCLUDED-RELATIONS:           NIL
#:|state-8192|
#<Structure STATE F3274E> is a structure of type STATE.
It has 9 slots, with the following values:
 BACKWARD:                     #:|path-8191|
 FORWARD:                      (#:|path-8197| #:|path-8199| #:|path-8201|)
 SUBSTANCE:                    (A B C)
 ASSUME:                       NIL
 CONSTRAINT:                   NIL
 RESTRICTION:                  ((NOT (IDLE B)) (NOT (M-CW B) (PATH B C)))
 STATE:                        ((DIAL-TONE A) (DIAL-TONE C) (IDLE B))
 SIGNAL:                       NIL
 HISTORY:                      ((BUSY-DIAL A B) (DIAL-TONE A)(DIAL-TONE
C) (IDLE B) (PATH B C) (PATH C B)
(RINGBACK C B) (RINGING B C))
#:|path-8197|
(#:|state-8198|)
#<Structure RULE E918A6> is a structure of type RULE.
It has 6 slots, with the following values:
```

```
 NAME:                          CW-8
 VARIABLES:                     (?A)
 EVENT:                         (SCW ?A)
 PRE-STATE:                     ((IDLE ?A))
 POST-STATE:                    ((IDLE ?A) (M-CW ?A))
 INCLUDED-RELATIONS:            NIL
#:|state-8198|
#<Structure STATE F003A6> is a structure of type STATE.
It has 9 slots, with the following values:
 BACKWARD:                      #:|path-8197|
 FORWARD:                       (#:|path-8204| #:|path-8206| #:|path-8208|)
 SUBSTANCE:                     (A B C)
 ASSUME:                        ((M-CW B))
 CONSTRAINT:                    NIL
 RESTRICTION:                   ((NOT (IDLE B)) (NOT (PATH B C)))
 STATE:                         ((DIAL-TONE A) (DIAL-TONE C) (IDLE B))
 SIGNAL:                        NIL
 HISTORY:                       ((BUSY-DIAL A B) (DIAL-TONE A)
(DIAL-TONE C) (IDLE B) (M-CW B) (PATH
B C) (PATH C B) (RINGBACK C B)(RINGING B C))
#:|path-8208|
(#:|state-8209| #:|state-8210|)
#<Structure RULE E91926> is a structure of type RULE.
It has 6 slots, with the following values:
 NAME:                          POTS-1
 VARIABLES:                     (?A)
 EVENT:                         (OFFHOOK ?A)
 PRE-STATE:                     ((IDLE ?A))
 POST-STATE:                    ((DIAL-TONE ?A))
 INCLUDED-RELATIONS:            NIL
#:|state-8210|
#<Structure STATE F07C9E> is a structure of type STATE.
It has 9 slots, with the following values:
 BACKWARD:                      #:|path-8208|
 FORWARD:                       (#:|path-8221| #:|path-8224| #:|path-8227|)
 SUBSTANCE:                     (A B C)
```

```
 ASSUME:                    NIL
 CONSTRAINT:                NIL
 RESTRICTION:               ((NOT (IDLE B)) (NOT (PATH B C)))
 STATE:                     ((DIAL-TONE C) (IDLE A) (IDLE B))
 SIGNAL:                    NIL
 HISTORY:                   ((BUSY-DIAL A B) (DIAL-TONE A)
(DIAL-TONE C) (IDLE A) (IDLE B) (M-CW
B) (PATH B C) (PATH C B) (RINGBACK C
B) (RINGING B C))
#:|path-8227|
(#:|state-8228|)
#<Structure RULE E91926> is a structure of type RULE.
It has 6 slots, with the following values:
 NAME:                      POTS-1
 VARIABLES:                 (?A)
 EVENT:                     (OFFHOOK ?A)
 PRE-STATE:                 ((IDLE ?A))
 POST-STATE:                ((DIAL-TONE ?A))
 INCLUDED-RELATIONS:        NIL
#:|state-8228|
#<Structure STATE EDB756> is a structure of type STATE.
It has 9 slots, with the following values:
 BACKWARD:                  #:|path-8227|
 FORWARD:                   END
 SUBSTANCE:                 (A B C)
 ASSUME:                    NIL
 CONSTRAINT:                NIL
 RESTRICTION:               ((NOT (IDLE B)) (NOT (PATH B C)))
 STATE:                     ((IDLE A) (IDLE B) (IDLE C))
 SIGNAL:                    NIL
 HISTORY:                   ((BUSY-DIAL A B) (DIAL-TONE A)
(DIAL-TONE C) (IDLE A) (IDLE B) (IDLE
C) (M-CW B) (PATH B C) (PATH C B)
(RINGBACK C B) ...)
```