〔公　開〕

ＴＲ－Ｃ－０１５５

サービス競合検出効率化手法

河原崎　裕朗
Yasuro KAWARASAKI

１９９６　３．１５

ＡＴＲ通信システム研究所

サービス競合検出効率化手法

平成８年３月１５日

河原崎 裕朗

目次

1　はじめに

本稿は，サービス競合検出手法の効率化手法についてまとめたものである．

ATRでは，通信サービス仕様をＦＳＭとして表現し，それに対して到達可能性解析を行い，その結果を用いてサービス競合を検出する手法（非決定性・異常な状態への遷移）を提案している．しかし，到達可能性解析には状態爆発（解析を行うために必要となる時間や記憶容量が指数関数で増加する）という問題がある．本稿は，ペトリネットの静的解析の結果得られるT-インバリアント(ある状態からその状態に遷移するまでの各遷移の遷移回数のベクトル)を用いて遷移の上限を設定することにより，状態爆発を回避する手法について述べる．

サービス競合の分類としては提案されていても，具体的な検出手法の提案が行われていない，異常な遷移・遷移の消失・状態の消失に関しても，T-インバリアントを用いた効率的な検出手法を提案する．

2　非決定性検出手法の効率化

よく知られているように，通信サービスはＦＳＭ（Finite State Machine)により表現できることから，プロダクションルールによりサービス仕様を記述し，複数のサービスを合成したときのＦＳＭを調べることにより，サービス競合を検出・解消する手法が提案されている．このうち状態遷移の非決定性によりサービス競合を検出する手法に対して，ペトリネットによる静的な分析の結果を用いてルールの適用回数の上限を設定することにより効率化を行った．

2．1　プロダクションルールによる通信サービス仕様記述

簡単に通信サービス仕様記述のためのプロダクションルール形式の仕様記述について説明する．

通信サービス仕様をプロダクションルールで記述する場合，一般的にはプロダクションの条件部は現在のサービス状態とプロダクションルール適用の契機となるイベントからなり，帰結部はプロダクションルール適用後のサービス状態となる．プロダクションルールでは着目するサービスの状態しか記述しないので，実際の状態(実状態)とルールの条件部の状態とは必ずしも一致する必要はない．実際の状態にルールの条件部が含まれるならばこのルールが適用される．また，サービスが加算的(incrementally)に追加できるためには，新しい状態に遷移するプロダクションルールが作られた場合に，この新しいルールが以前のルールよりも優先されて適用される必要がある．このため，プロダクションルールの適用規則に優先適用の規則が必要となる．優先適用規則は”２つのルールが適用可能な場合には条件部の状態の記述が詳細な方のルールが適用される”と表現される．以上をまとめると，

- プロダクションルールの形式表現;

　　現状態　イベント:　次状態

- プロダクションルールの適用規則:

　　ルールのイベント＝実際に起きたイベント，かつ

　　現状態　$\subseteq$　実状態のとき　適用される．

- ルールの優先適用規則:

　　２つのルールが適用可能な場合には，

　　(ルール１の条件部の状態)　$\subseteq$　(ルール２の条件部の状態)　のとき

　　ルール２が適用される．

ここでサービスの状態は，端末個々の状態と端末間の状態の組み合わせとして表現される．また，個々の端末に関する遷移をプロダクションルールとして表現することも可能であるが，端末を表わす変数を導入することにより記述すべきルールの数を軽減することができる．

2．2　非決定性検出手法の概要

状態遷移の非決定性とは，ある任意の状態から複数の遷移が可能となりどちらの遷移を行うかを決定できないことである．既提案の非決定性検出手法の概要は以下の通りである．

1）合成サービス仕様において遷移可能な状態の集合を求める
2）非決定性を起こす可能性のあるルールの組み合わせの集合を求める
3）ルールの組み合わせの集合の各要素の現状態を共に満たす状態が遷移可能な状態の集合の中に存在するかを判定する．存在する場合には，そのルールの組み合わせによる非決定性が検出されたことになる．

この手法においては1）の遷移可能な状態を求めるための計算量が指数関数で増加することが問題となる．

2．3　ペトリネットを用いた分析

ペトリネットでは静的な解析により可到達性，可逆性，有界性，活性などの性質を検証することが可能である．プロダクションルールによる通信サービス仕様記述をペトリネットを用いて分析するためには，プロダクションルールからペトリネットへの変換が必要である．

ペトリネットにおけるプレース，トランジションは，ＦＳＭにおける状態，イベントにそれぞれ対応づけられる．トランジションの入力アークはルールの現状態に出力アークはルールの次状態にそれぞれ対応する．(図7 参照)ペトリネットにおけるトークンは，通信サービスにおける端末に対応付けられる．特に，検証においては端末を識別する必要があるので，トークンとしてカラードトークンが，アークとしてカラードアークを使用するペトリネットを用いる．このようなペトリネットをカラードペトリネットと呼ぶ．カラードペトリネットでは，各アークの色に合ったカラードトークンが存在するときにのみトランジションが発火可能となる．トランジションの発火は，FSMにおける遷移に対応づけられる．

ペトリネットにおける可到達性，可逆性，有界性，活性は，ＦＳＭにおける状態の可到達性，可逆性，状態数の有限性，デッドロックを起こさないと言う性質にそれぞれ相当する．したがって，プロダクションルールからペトリネットへの変換ができればＦＳＭにおける上記性質をペトリネットの解析により検証することができる．

ここでは，上記性質のうち可逆性について説明する．ＦＳＭにおける可逆性を通信サービス仕様に当てはめると，任意の状態からその状態へ必ず到達できるということである．この遷移におけるイベントの生起系列はペトリネットにおけるトランジションの発火回数として行列式により表現でき，Ｔインバリアントと呼ばれる．従って，ＦＳＭにおける可逆性の検証による状態生成はＴインバリアントで表わされているルールの適用回数の範囲内で状態を生成すればよい．これにより不要なプロダクションルールの適用による状態生成を回避することができる．

プロダクションルールからカラードペトリネットへの変換は，機械的に行うことが可能であるが，カラードペトリネットのTインバリアントを求めることは困難なため，一度色なしペトリネットに変換し色なしペトリネットのＴインバリアントを求め，それを用いてＴインバリアントを求める手法が提案されている．Ｔインバリアントは，行列計算による静的な分析により求めることが可能である．

Ｔインバリアントの分析においては，極小Ｔインバリアントを求めることにより計算量を削減することができる．極小Ｔインバリアントとは，他のＴインバリアントの和であらわすことのできないＴインバリアントである．言い換えれば，

すべてのTインバリアントは極小Tインバリアントの和として表現することが可能である．

極小Tインバリアントからは，任意の状態からその状態への遷移が存在するということしか分からないため，通信サービスの初期状態から初期状態への遷移と対応しているとは限らない．つまり，通信サービスの取りうる任意の状態からその状態へのループもTインバリアントとして検出される．任意の状態が初期状態から到達可能であれば，初期状態から任意の状態までの遷移が別のTインバリアント（複数の極小インバリアントの和の場合もある）に含まれることはTインバリアントから明らかである．したがって，初期状態から任意の状態までの遷移に必要なTインバリアントを補う方法を検討する必要がある．

２．４　極小Tインバリアントの総和を上限とする状態生成手法

この手法は，初期状態から任意の状態までの遷移に必要なTインバリアント補う方法として，総ての極小Tインバリアントの和を用いる方法である．この手法では，求められた極小Tインバリアントの各ルールに対する適用回数の上限をそれぞれ足しあわせ，それを上限として状態を生成する．

この手法ではルールの適用回数の上限を設定することにより，ルールの適用の回数と生成される状態数を削減することができる．付図１に評価用の簡易STRインタプリタを用いた生成時間を，付図２に評価用の簡易STRインタプリタを用いた生成された状態数を，付図３にプロトタイプシステムを用いた生成時間を示す．簡易STRインタプリタを用いた結果において，生成された状態数が一定で生成時間が増加する場合がある．これは，簡易STRインタプリタでは，記憶容量を削減するために，ルール適用後に生成済み状態の判定を行っているため，ルールの適用回数の削減が十分ではないためである．ルール毎に既に適用した状態を管理し，ルール適用前にルール適用するか否かを判断することによりルールの適用回数を削減することは可能である．しかし，端末数の増加に伴って判定すべき状態の数は増加するため，生成時間が一定になることはない．

２．５　極小Tインバリアントを用いた非決定性検出手法

２．５．１　考え方

Tインバリアントを用いた状態生成法により状態生成を効率的に行うことが可能となったが，非決定性検出に限ればすべての状態を生成するのではなく非決定性を起こす可能性のある状態への可到達判定を行えば十分である．そこで，競合を起こす可能性のあるルールを含む極小Tインバリアントだけに限定した状態生成を行うことにより検出の効率化を行った．競合を起こす可能性のあるルールを含む極小Tインバリアントだけでは遷移できない可能性があるため，初期状態から極小Tインバリアントが遷移可能になる任意の状態までの遷移に必要なTインバリアントを補う方法の検討が必要である．

(1) Tインバリアントを補う方法

極小Tインバリアントが初期状態から遷移できない場合として以下が考えられる．

1） ルールの現状態に初期状態から極小Tインバリアントに含まれるルールだけでは遷移できない状態が含まれる

2）初期状態からは遷移できない極小Tインバリアント

2）に関しては，状態生成の時に除外することにより対処可能である．1）に関しては，極小Tインバリアントに含まれるルールを分析することにより補足を行う方法を検討した．

(2) ルールの分類

ルールの現状態に含まれるプリミティブの数と次状態に含まれる個々のプリミティブの数により以下の3種類に分類できる．

プリミティブの数が増加するルール

プリミティブの数が減少するルール

プリミティブの数が変化しないルール

例えば，

idle(A) off-hook(A): dial-tone(A).

の場合，プリミティブの総和に関する分類では変化しないルールになるが，idleに関しては減少するルール・dial-toneに関しては増加するルールに分類される．

ルールに含まれる総てのプリミティブについても上記と同様に分類できる．

単調増加するルール

含まれるプリミティブすべてに対してプリミティブの数が減少するルールを含まないもの

単調減少するルール

含まれるプリミティブすべてに対してプリミティブの数が増加するルールを含まないもの

変化しないプリミティブを含むルール

プリミティブの数が変化しないプリミティブを含み，単調増加するルール・単調減少するルール以外のルール

その他

上記以外のルール

ルールの現状態に初期状態から極小Tインバリアントに含まれるルールだけでは遷移できない状態が含まれる場合には，極小Tインバリアントに含まれるルールの中に変化しないプリミティブを含むルールが存在する．したがって，そのプリミティブが単調増加するルールを含む極小Tインバリアントと組み合わせることにより，初期状態から遷移可能なTインバリアントが構成できる．

2．5．2　アルゴリズム

上記の考え方に基づいた非決定性検出のアルゴリズムは以下の通りとなる．

(1) 非決定性を起こす可能性のあるルールの組み合わせの抽出

サービス合成後のルール集合の中で，非決定性を起こす可能性のあるルールの組み合わせ（イベントが同じで優先適用の関係にない）をすべて求める．

(2) 極小Tインバリアントを求める

サービス合成後のルール集合の極小Tインバリアントを求める．

(3) 状態生成による到達可能性の解析

(3-1) (1)で求めた全てのルールの組み合わせ(r_A,r_B)に対して以下を行う
(3-2) 極小Tインバリアントの合成
r_A,r_Bを含む極小Tインバリアントt_A,t_Bを合成してTインバリアントtを得る.
(3-3) 変化しないプリミティブを含むルールが含まれる場合
変化しないプリミティブそれぞれについて，そのプリミティブの数が増加するルールを含む極小Tインバリアントをtに追加する.
(3-4) 状態生成
Tインバリアントtを用いてルール適用回数の上限を設定し，状態生成を行い，状態生成の途中で非決定性が検出された場合には，中断する．すべての状態生成を終了するまでに非決定性が検出されなかった場合には，このルールの組み合わせでは非決定性が起こらない.

２．５．３　評価

この手法は，極小Tインバリアントに対する競合を起こす可能性のあるルールを含む極小Tインバリアントの割合により有効性が変化するため，一般的な定量的な評価は困難である.

付図４に評価用の簡易STRインタプリタを用いた生成時間を，付図５に評価用の簡易STRインタプリタを用いた生成された状態数を示す.

付録１に，CW+CFVに対してこのアルゴリズムを適用した結果を示す.

3　異常な遷移の検出

3．1　異常な遷移の定義

異常な遷移とは，サービス合成により既存の状態間にサービス合成前には存在しなかった新たな遷移が生じることである．サービス合成により新たな状態とそれに関る遷移が生じる場合には，既提案の未定義な状態への遷移により検出可能であるため異常な遷移の発生では対象としない．

プロダクションルール形式の仕様記述の場合の遷移の消失を形式的な定義は以下の通りである．

サービスAとサービスBを合成において，サービスAの状態S_{Ai}に対してサービスAのルールr_AとサービスBのルールr_Bが適用可能で，それぞれのルールを適用した結果それぞれS_{Aj}とS_{Ak}に遷移することを想定する．この時サービスAのルールr_AとサービスBのルールr_Bのイベントはそれぞれe_A，e_Bとする．r_Bと同じルールがサービスAに含まれる場合には，新たな遷移が生じない．r_Bと同じルールがサービスAに含まれない場合には，各ルールの遷移先の状態との組み合わせにより，以下の３種類に分類できる．

1)r_Bと同じイベントを持つルールr_Aが存在し遷移先の状態S_{Aj}とS_{Ak}が同じ場合，遷移の際に使用されたルールは異なるが遷移前の状態と遷移後の状態が合成前と変化しないため，新たな遷移が生じない．

2)r_Bと同じイベントを持つルールr_Aが存在し遷移先の状態S_{Aj}とS_{Ak}が異なる場合，遷移前の状態と遷移後の状態が合成前と変化するため，新たな遷移が生じる．

3)r_Bと同じイベントを持つルールr_Aが存在しない場合，遷移の契機となるイベントが異なるため，新たな遷移が生じる．

図１に，上記の4種類の状態遷移を示す．

ここでは，２つのサービスを合成する場合に関して説明したが，任意の数のサービスを合成する場合も同じ議論が成り立つ．

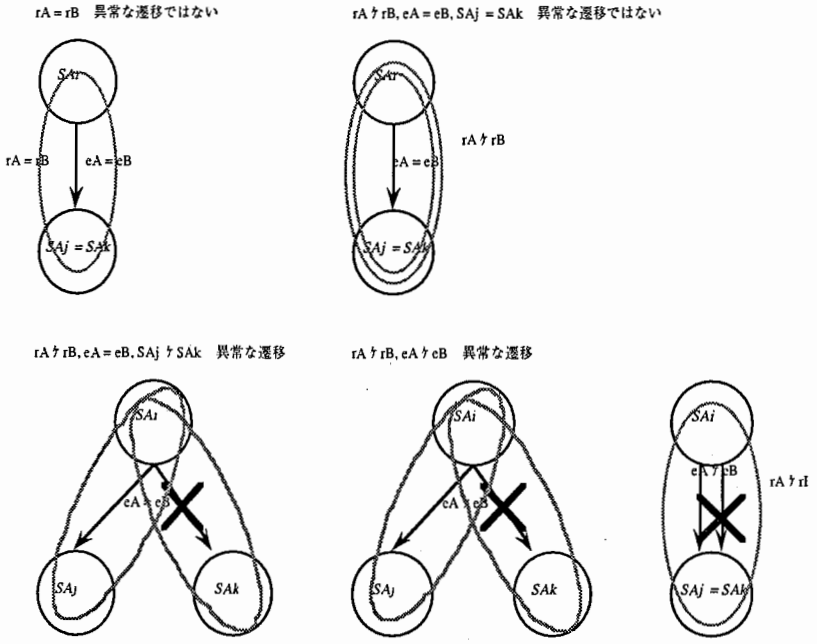


図１　異常な状態遷移

３．２　異常な遷移検出アルゴリズム

異常な遷移の発生は，サービス合成前のＴインバリアントと合成サービスのＴインバリアントを比較することにより，検出することが可能である．サービス合成前のＴインバリアントをt_A,t_B，合成サービスのＴインバリアントをt_{AB}とすると，t_{AB}にt_A, t_Bに含まれないＴインバリアントが含まれ，そのＴインバリアントが初期状態から遷移可能な場合には，それが異常な遷移となる．

異常な遷移の具体例が見つかっていないため，評価は完了していない．

4　遷移の消失の検出

4．1　遷移の消失の検出の定義

遷移の消失とは，サービス合成により既存のサービスの状態遷移が消失することである．

プロダクションルール形式の仕様記述の場合の遷移の消失を形式的に定義すると以下の通りである．

サービスAのルールr_Aに対してサービスBのルールr_Bが優先して適用される場合，ルールr_Aの遷移が消失する

図2に，緊急電話サービスと三者通話サービスの競合例を示す．

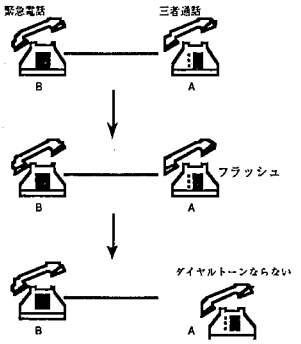


図2　遷移の消失の例

ここで，遷移の消失と状態の消失について考察する．状態の消失とは，サービス合成により既存のサービスで定義された状態が消失することである．しかし，状態が消失する時にはその状態への遷移も同時に消失するため，遷移の消失の検出を行えば必ず状態の消失も検出することが可能である．

4．2　遷移の消失の検出検出アルゴリズム

遷移の消失の発生は，サービス合成前のTインバリアントと合成サービスのTインバリアントを比較することにより，検出することが可能である．サービス合成前のTインバリアントをt_A,t_B，合成サービスのTインバリアントをt_{AB}とすると，t_A, t_Bにt_{AB}に含まれないTインバリアントが含まれ，そのTインバリアントが初期状態から遷移可能な場合には，それが遷移の消失となる．

緊急電話サービスと三者通話サービスの組み合わせにおいて，正しく検出されることを確認した．

5　まとめ

ＦＳＭとしてプロダクションルールにより記述されたサービス仕様の競合について，既提案されている非決定性検出手法の効率化手法を述べ，非決定性検出によるサービス競合検出手法の有効性を増すことができた．極小Ｔインバリアントの総和を上限とする状態生成手法は，異常な状態への遷移検出の効率化に適用することが可能である．また，検出手法の提案されていなかった異常な遷移の生成，状態の消失，遷移の消失についても効率的な検出手法を述べた．

非決定性検出効率化手法で用いたＴインバリアント補足法では，冗長なＴインバリアントを補足してしまう可能性があるため，よりよい補足法の検討の余地がある．

参考文献

[1] T. F. Browen, et. al., "The Feature Interaction Problem in Telecommunications Systems," Proc. of 7th IEE Int. Conf. on Soft. Eng. for Telecom. Systems, 1989

[2] 原田 他,「通信サービス競合検証システムの試作と評価」, SSE92-8, 1992

[3] 井上 他,「通信サービスにおける異常な状態への遷移の解消支援法」, 1994春季全国大会 B-620

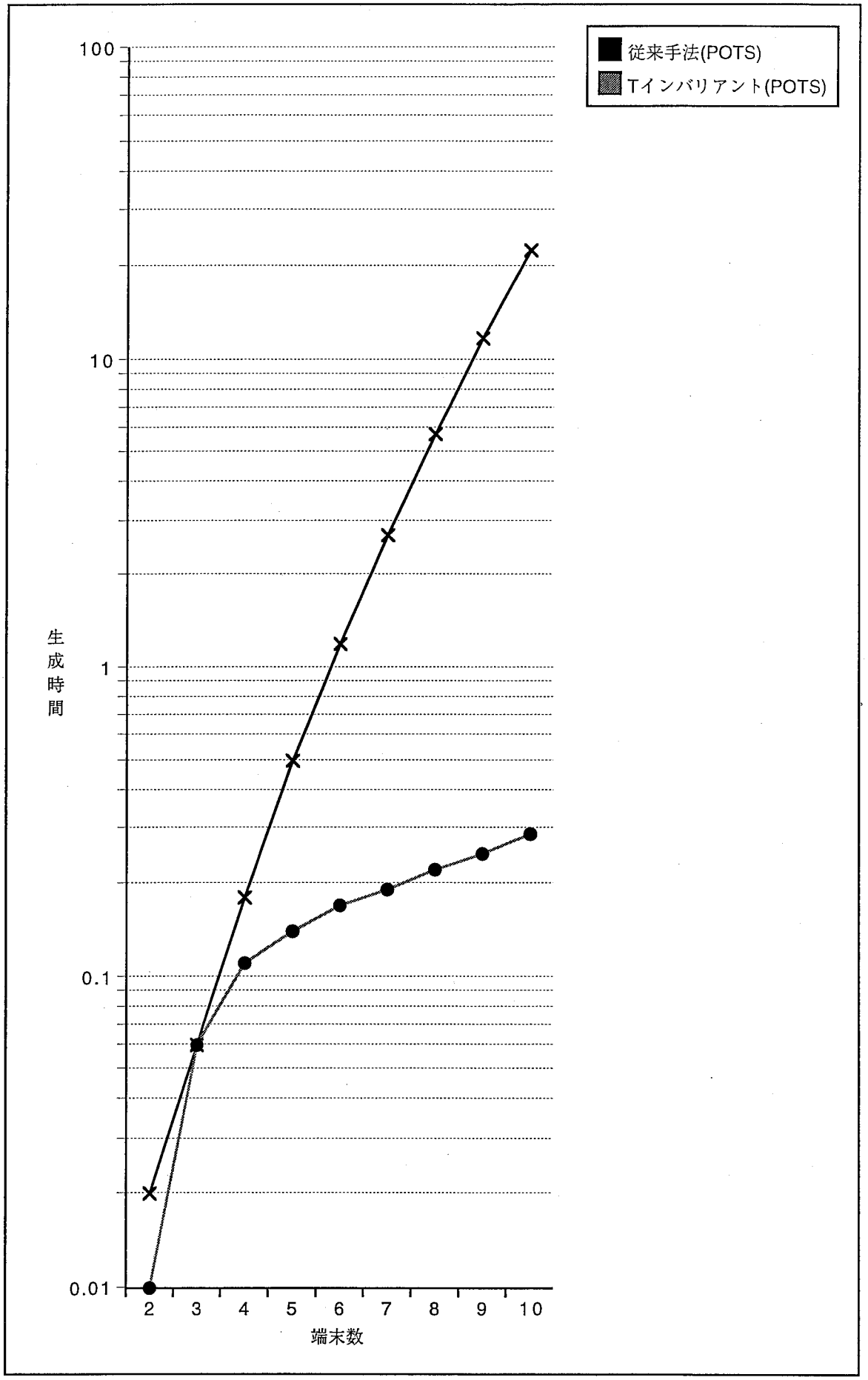
[4] Y. Hirakawa, et. al.,, "Telecommunication Service Description Using State Transition Rules," Sixth Int. Workshop on Software Specification and Design, Comp., Italy, Oct. 1991.

[5] 上田 他,「通信システムサービス仕様におけるデッドロック検出方式」, CST94-25,1994

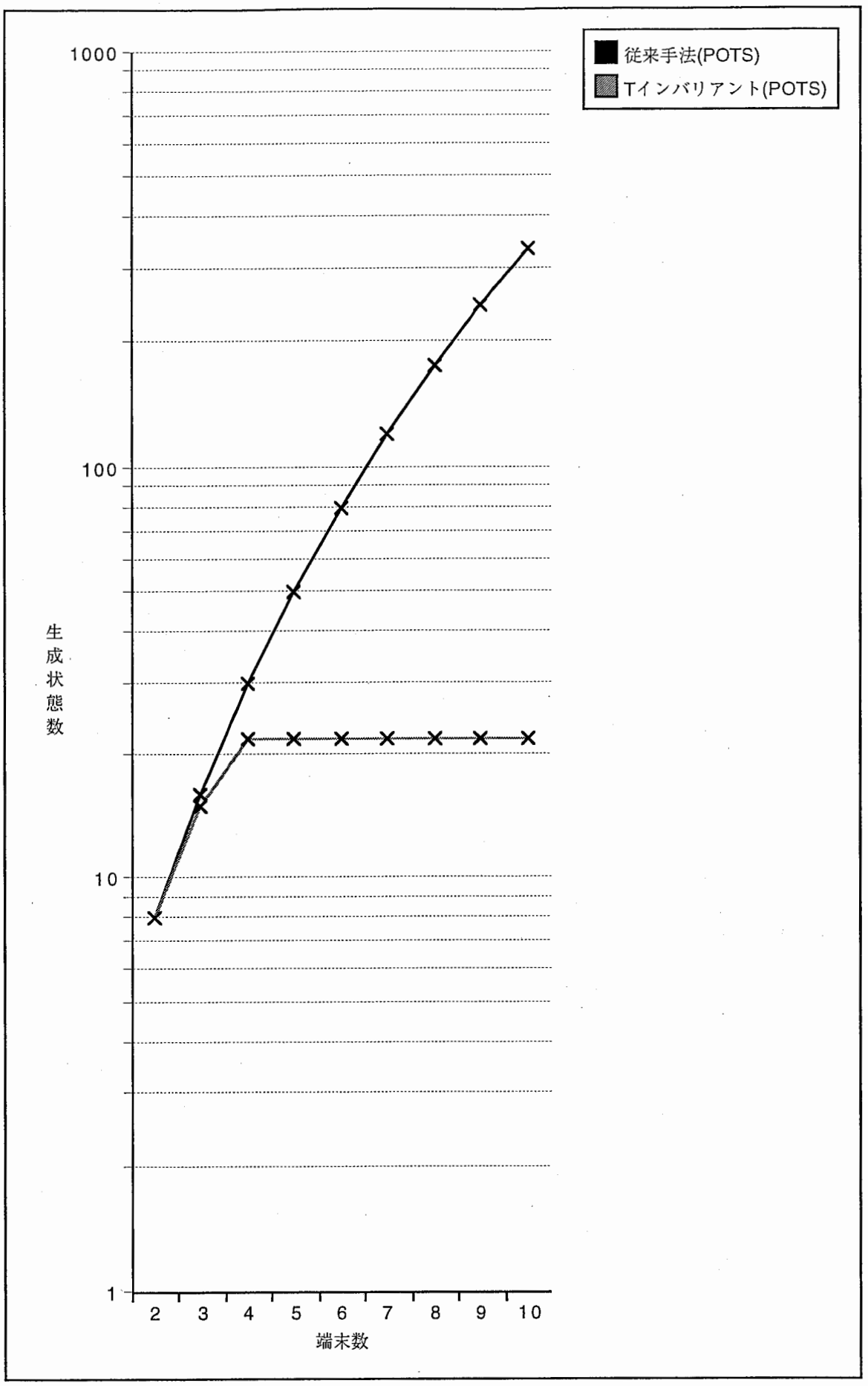
[6] 上田 他,「カラードペトリネットによる通信システムサービス仕様の検証方式」, SICE第14回離散事象システム研究会,pp.411-416,1994

[7] 河原崎 他,「サービス競合検出法の一考察」, 1994秋季全国大会 B-543

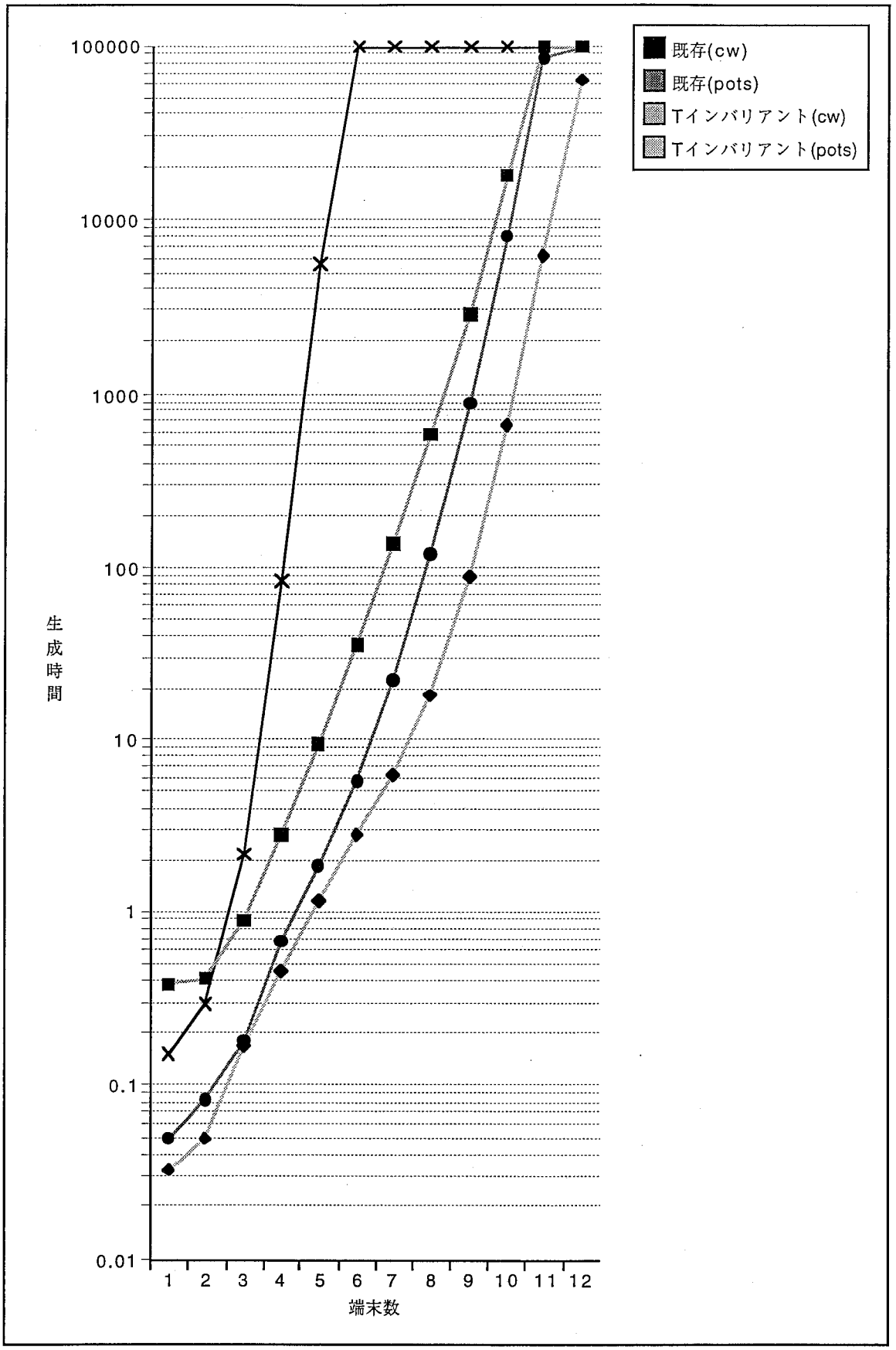
[8] 河原崎 他,「サービス競合検出法の一考察」, SSE95-48,1995



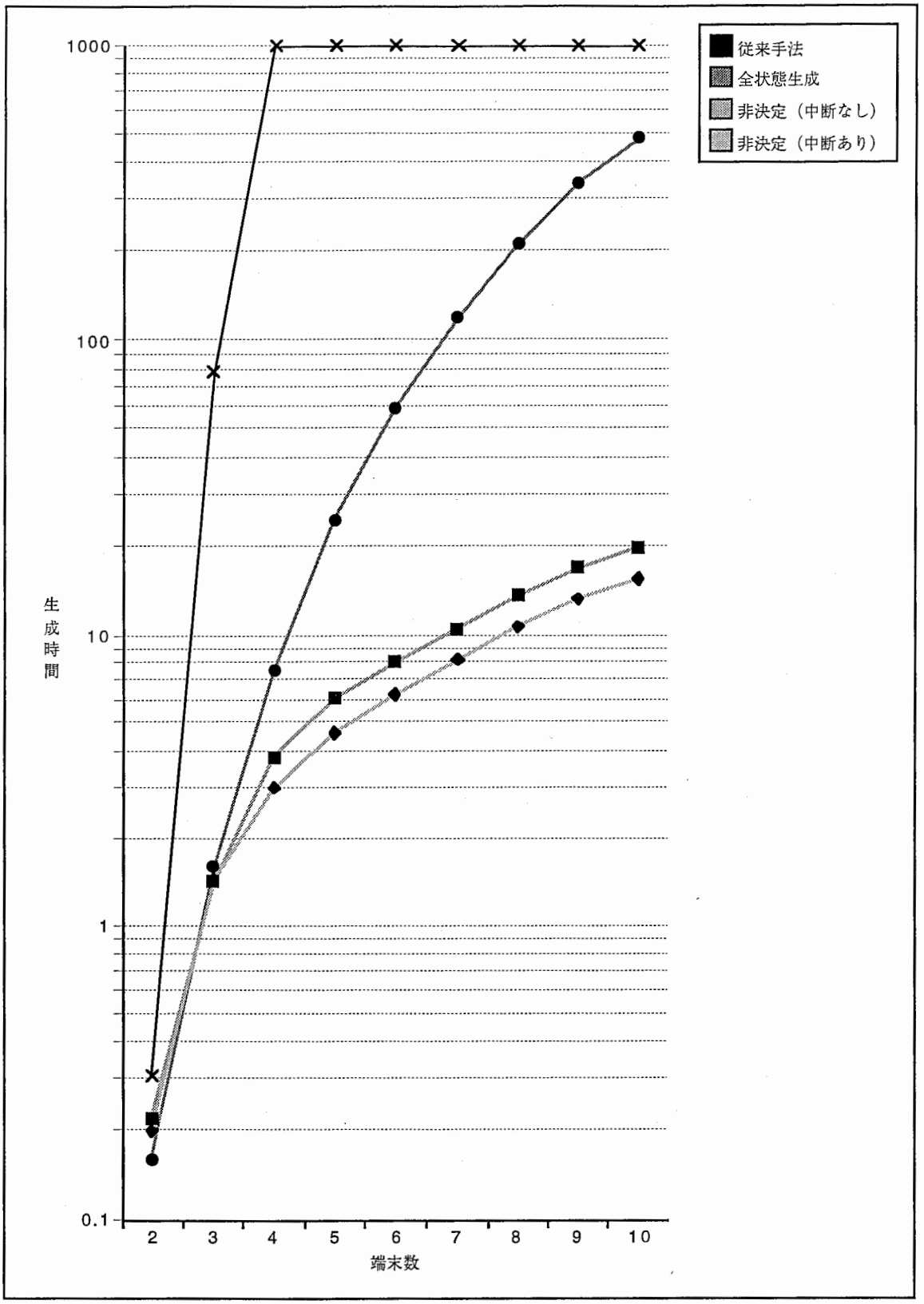
付図1　極小Tインバリアントの総和を上限とする状態生成手法（簡易・時間）



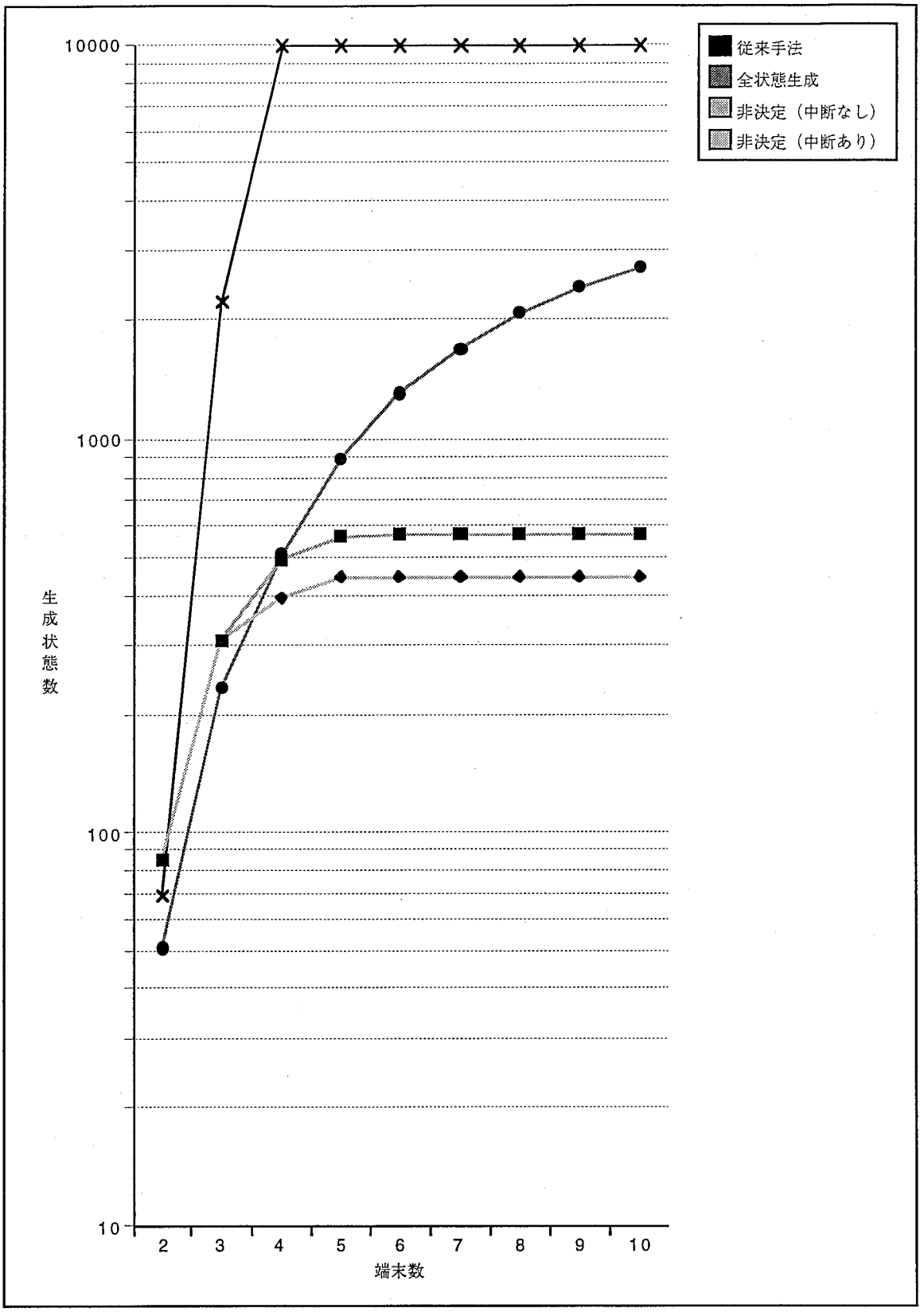
付図２　極小Ｔインバリアントの総和を上限とする状態生成手法（簡易・状態数）



付図3　極小Tインバリアントの総和を上限とする状態生成手法（プロトタイプ・時間）



付図４　極小Ｔインバリアントを用いた非決定性検出手法（簡易・時間）



付図５　極小Ｔインバリアントを用いた非決定性検出手法（簡易・状態数）

付録1 CW+CFVへのT-インバリアントを用いた非決定性検出効率化手法の適用

STRルール

共通ルール

```
idle(T0) offhook(T0): dial-tone(T0).                                    r-1
dial-tone(T0) onhook(T0): idle(T0).                                     r-2
dial-tone(T0),idle(T1) dial(T0,T1): Calling(T0,T1).                     r-3
dial-tone(T0),not[idle(T1)] dial(T0,T1): busy(T0).                      r-4
Calling(T0,T1) offhook(T1): Talk(T0,T1).                                r-5
Talk(T0,T1) onhook(T0): idle(T0),busy(T1).                              r-6
busy(T0) onhook(T0): idle(T0).                                          r-7
```

固有ルール(CW)

```
path(T0,T1),dial-tone(T2),cond:m-cw(T0) dial(T2,T0):
    CW-calling(T2,T0),path(T0,T1).                                      r-8
CW-calling(T2,T0),Talk(T0,T1) flash(T0):
    cw(T0),Talk(T0,T2),Hold(T0,T1).                                     r-9
Talk(T0,T2),Hold(T0,T1),cond:cw(T0) flash(T0):
    Talk(T0,T1),Hold(T0,T2).                                           r-10
cond:cw(T0),Talk(T0,T2),cond:path-passive(T0,T1),
    cond:path-passive(T1,T0) onhook(T0): ringing(T0,T1),busy(T2).      r-11
Talk(T0,T2),path-passive(T0,T1),path-passive(T1,T0),cw(T0)
    onhook(T2): Talk(T0,T1),idle(T2).                                  r-12
cw-ringing(T0,T2),Talk(T0,T1) onhook(T0): ringing(T0,T2),idle(T1).
                                                                       r-13
cw-ringing(T0,T1),ringback(T1,T0),Talk(T0,T2) onhook(T2):
    Talk(T0,T1),idle(T2).                                              r-14
ringing(T0,T1),Hold(T0,T1),cw(T0) offhook(T0): Talk(T0,T1).            r-15
cond:idle(T0) cw(T0): m-cw(T0).                                        r-16
cond:idle(T0),m-cw(T0) cw(T0): empty.                                  r-17
```

固有ルール(CFV)

```
dial-tone(T0),not[idle(T1)],idle(T2),cond:m-regcfv(T1,T2)
    dial(T0,T1): Calling(T0,T2).                                       r-18
dial-tone(T0),not[idle(T1)],not[idle(T2)],cond:m-regcfv(T1,T2)
    dial(T0,T1): busy(T0).                                             r-19
idle(T0),idle(T1),m-regcfv(T0,T1) cfv(T0,T1): idle(T0),idle(T1).       r-20
idle(T0),idle(T1) cfv(T0,T1): idle(T0),idle(T1),m-regcfv(T0,T1).       r-21
idle(T0),idle(T1),m-regcfv(T0,T2) cfv(T0,T1):
    idle(T0),idle(T1),m-regcfv(T0,T1).                                 r-22
```

(1) 非決定性を起こす可能性のあるルールの組み合わせの抽出

r-8とr-18, r-8とr-19の組み合わせが考えられる.

(2) 極小Tインバリアントを求める

極小Tインバリアントは以下の１３個である.

```
t1    (1,0,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
t2    (1,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
t3    (1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
t4    (1,0,0,0,0,0,1,1,1,0,1,0,0,0,1,0,0,0,0,0,0)
t5    (1,0,0,0,1,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0)
t6    (1,0,0,0,0,0,0,1,1,0,0,1,0,0,0,0,0,0,0,0,0)
t7    (1,1,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0)
t8    (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0)
t9    (0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0)
t10   (1,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0)
t11   (1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0)
t12   (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0)
t13   (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1)
```

(3) 状態生成による到達可能性の解析

(3-1) r-8とr-18の組み合わせ

(3-2) 極小Tインバリアントの合成

r-8を含む極小Tインバリアントはt4とt6，r-18を含む極小Tインバリアントはt10である．したがって，t1 =t4+t10は

```
t1    (2,0,0,0,1,1,2,1,1,0,0,0,0,0,1,0,0,1,0,0,0,0)
```

に，t2=t6+t10は

```
t2    (2,0,0,0,1,1,1,1,1,0,0,1,0,0,0,0,0,1,0,0,0,0)
```

となる．

(3-3) Tインバリアントtに含まれるルールの分類

T1,T2共に，m-cw・path・m-regcfvに関しては変化しないプリミティブを含むルールが含まれる．

m-cwが増加するルールを含む最少の極小Tインバリアントはt8，pathが増加するルールを含む最少の極小Tインバリアントはt1，m-regcfvが増加するルールを含む最少の極小Tインバリアントはt12である．したがって，これらを追加したTインバリアントは

(4-5-2) 極小Tインバリアントの集合Tの全ての要素について以下を行う

t1 =t1 +t1+t8+t12は

```
t1    (3,0,1,0,2,2,3,1,1,0,0,0,0,0,1,1,1,1,0,1,1,0)
```

```
t2=t2+t1+t8+t12
T2    (3,0,1,0,2,2,2,1,1,0,0,1,0,0,0,1,1,1,0,1,1,0)
```

となる．

(3-4) 状態生成

t1 を用いてルール適用回数の上限を設定し状態生成を行うと，以下の２つのルールの共に満たす現状態が存在することが確認され，非決定性が検出される．

```
path(T0,T1),dial-tone(T2),cond:m-cw(T0) dial(T2,T0):
      CW-calling(T2,T0),path(T0,T1).       r-8
dial-tone(T0),not[idle(T1)],idle(T2),cond:m-regcfv(T1,T2)
      dial(T0,T1): Calling(T0,T2).  r-18
```

t1 と**t2**は同じルールの組み合わせ(r-8,r-18)に対するＴインバリアントであるため，**t2**を用いてルール適用回数の上限を設定した状態生成を行う必要はない．