ペトリネットを用いたルールベースの整合性検出に関する考察

3W－7

大本 周広　皆川 朋輝　藤田 一樹　稲積 宏誠
青山学院大学理工学部経営工学科

1．はじめに

知識ベースシステムにおいて知識獲得によるルールベースの生成及びその管理は重要な問題である。特に大規模ルールベースにおけるルールの追加、削除、変更に伴う効率的な検証方法が求められている[1]。近年ペトリネットを用いたルールベース検証方法が検討されている[2]。これはルール集合をペトリネット表現し、ルールの連鎖構造からエラーの検証を行うことで、一貫性と完全性を検証するものである。そこで、本稿では、ペトリネットによるエラー検証法に検討を加え、その検証能力の拡張について論じる。なお、対象とするルールベースは命題論理で表現されたルール集合であるものとする。

2．ペトリネットによるＲＢＳの表記 [1]

2.1 ペトリネットの基本表記

ペトリネット C(P,T,I,O)はルールベースの全体構造を示す。ルールはトランジションT、そのルールの前件部および後件部はプレースP、さらにマーキングは事象の生起状態を示すものとして下記のように定義する。

(1)プレース集合：P={p1,p2,…,pn}　n>0
(2)トランジション集合：T={ t1,t2,…,tn}
(3)トランジション ti の入力プレース：I(ti)
(4)　　出力プレース：O(ti)
(5)マーキング：μ={μ1,μ2,…,μn}
(6)プレース pi のマーキング数：μ(pi)
(7)マーキングμ^0の次段マーキング：$\mu'=\delta(\mu^0, ti)$
また、一般にマーキングμ^0に対する発火トランジション列 Ti={t1,t2,…,tn}の次段マーキングは次のように求めることができる。

$\mu'=\delta(\cdots(\delta(\delta(\mu^0, t1), t2)\cdots, tn)$

2.2 ルールの一般表記

ルール ri∈Ｒは、
前件部の集合：AC(ri)={AC1(ri),…,ACm(ri)}と
後件部の集合：CC(ri) ={CC1(ri),…,CCn(ri)}から成っている。

Applicability of Petri Nets for Rules- Based Verification
Narihiro Omoto, Tomoteru Minagawa, Kazuki Fujita,
Hiroshige Inazumi
Department of Industrial and Systems Engineering,
Faculty of Science and Engineering, Aoyama Gakuin University

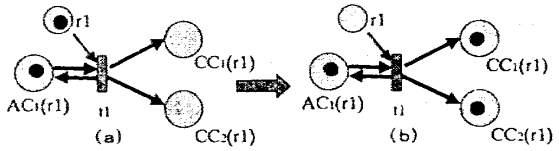

図1 ルールのペトリネット表現

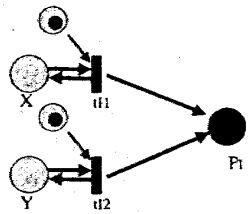

図2 矛盾検出構造

従ってルールの一般形は
AC1(ri)&…&ACm(ri)→CC1(ri)&…&CCn(ri)と表すことができ、次の関係を満たす。
I(ti)={AC1(ri),…,ACm(ri),ri}
O(ti)={AC1(ri),…,ACm(ri),CC1(ri),…,CCn(ri)}
例えば、r1：AC1(r1)→CC1(r1)&CC2(r1)は図１のような構造として表現される。特に r1 というプレースを付加することで発火を制御することが可能となり、ルールの前件部を含めて発火系列がマーキングとして保存されることになる。(ａ)が発火前(ｂ)が発火後を示している。

ここで新たに検出用付加プレース、トランジション（P_I、T_I）を導入する。例えば、背反な事実 X、Y の同時生起を検出するための構造を図２のように付加プレースと付加トランジションを用いて構成すると、P_I のマーキングμ_Iによりエラー検出が可能となる。そこでエラー検出のためのペトリネットC(P,T,I,O)のルールベース表記を以下に示す。

P={Pc,P_R,P_I}（Pc：節集合
　　P_R：ルールプレース集合
　　P_I：検出用付加プレース集合）
T={T_R,T_I}（T_R：ルールトランジション集合
　　T_I：検出用付加トランジション集合）
I：{T→P^∞ | AC(ri)∈I(ti);ri∈I(ti)}
O：{T→P^∞ | AC(ri)∈O(ti);CC(ri)∈O(ti)}
Pc={P_{CE},P_{CI},P_{CG}}
（P_{CE}：外部節，P_{CI}：推論節，P_{CG}：ゴール節）
μ={μ_C | μ_R | μ_I}
$\mu^0{}_C\in\{0,1\}$，$\mu^0{}_R=[1]$，$\mu^0{}_I=[0]$

3．エラー構造の検出法

エラー構造とエラーの原因となるルールの種類は表１のように分類されている。これに対してペトリネットモデルによる定義とエラー検出方法が示されている[2]。特に本稿で用いるもののみ下記に示す。

(1)*Redundancy*：

Tj∩Tk＝φ、μ'＝δ(μ⁰, Tj)、μ"＝δ(μ', Tk)のときヨi;μ⁰ci=0、μ'ci＝1、μ"ci＞1となる。

(2)*Conflict*：

Tn を付加トランジション列

μ'＝δ(μ⁰, Tj+Tn)のときヨi;μ' Ii＞1となる。

表１　エラー構造との対応づけ

エラー構造	問題となるルール
Redundancy	Redundant Rules Unnecessary IF Conditions Subsumed Rules
Conflict	Conflicting Rules
Circularity	Circular Rules
Dead-Ends	Dead-End IF Conditions Dead-End Goals
Unreachable Goals	Unreachable Concludions

4．エラー検出の拡張

従来においてエラー構造との関連ルールの中で十分に検討されていない Conflicting Rules と Unnecessary IF Conditions、特に事実の否定がルールの中に含まれる場合に生じるエラー検出方法に注目する。

4.1 Conflicting Rules

Conflict とは複数のルールが同じ条件節のもとで衝突する結果を導くことを指す。A→B の形式のルール集合に、A→not(B)のような否定形ルールの存在が引き起こすエラー検出に注目する。例えば A→B、A→not(B)は矛盾した結論を導いており、このようなルールが存在すると正しい推論が行われない。従って、ある事実の肯定と否定が存在する場合はその両プレースに図３のように付加プレースの導入によりConflict検出と同様のエラー検出が可能となる。

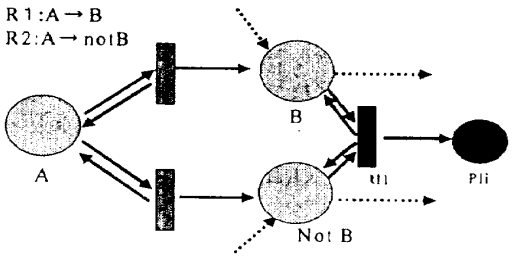

図３ 矛盾ルールの検出構造

4.2 Unnecessary IF Conditions

エラーの分類としては、Redundancy に含まれるが、２つのルールが同じ結論を導きそれらの条件節の中に互いに背反な条件が存在した場合に、その条件は本来簡略化されるべきである。これも否定形ルールが引き起こすエラーであり、以下のように定義する。

Unnecessary IF Conditionsが存在するなら

I(t1)∈{I(t1)∩I(t2), I(t1)∩I(tIi)}

I(t2)∈{I(t1)∩I(t2), I(t2)∩I(tIi)}

I(tIi)∈{I(t1)+I(t2)}

ただし、I(t1)+I(t2)=I(t1)∪I(t2)−I(t1)∩I(t2)

①μ'＝δ(μ⁰, t1), μ"＝δ(μ', t2) のとき

ヨi;μ⁰ci＝0, μ'ci＝1, μ"ci＞1

かつ

②P_{Ii}∈{O(tIi)}

ヨi;$\mu^0 P_{Ii}$＝0, $\mu' P_{Ii}$＝0, $\mu'' P_{Ii}$＝1 となる。

上記の①、②が同時成立することによりUnnecessary IF Conditionsの定義とする。図４にエラー検出構造を示す。

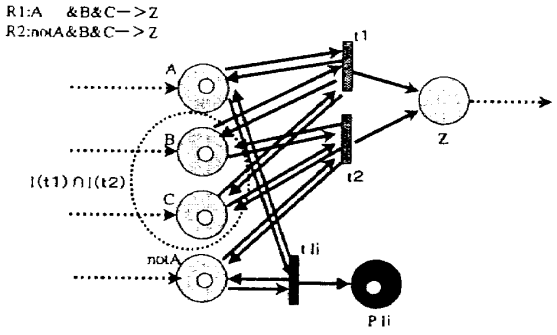

図4 Unnecessary IF Conditionsの検出構造

5．おわりに

ルールの追加削除が頻繁に行われる知識ベースではその洗練化が非常に重要となる。そこで、今後はエラーの検出にとどまらず、Redundancy を中心にルール簡約の視点からのアプローチを行っていく。今後はさらに詳細な検討を行う予定である。

参考文献

[1] Tin A.Nguyen, Walton A.Perkins, Thomas J.Laffey, and Deanne Pecora: Knowledge Base Verification, AI Magazine, Vol.8, No.2, pp.69-75, 1987

[2] Derek L. Nazareth :Investigating the Applicability of Petri Nets for Rules-Based System Verification, IEEE Transactions on Knowledge and Data Engineering, Vol.4, No.3, pp.402-414, 1993