知識工学と人工知能　41－10
（１９８５．７．４）

データフロー解析を用いた知識の誤り検出

古屋　博行　　森原　一郎　　石田　亨　　和佐野　哲男
（日本電信電話株式会社　横須賀電気通信研究所）

１．はじめに

プロダクションシステム（ＰＳ）を利用したエキスパートシステムの開発にあたっては、専門家の知識をルール形式で記述し、そのルールを頻繁に追加、更新しながらシステムのリファインが行われる。　ＰＳは、ルールが互いに独立であることから、ルールの更新、追加が行い易いという利点を持つ反面、ルール間の関係がとらえにくいため、更新、追加時にルールの誤り（不足、冗長、矛盾）が生じ易い。　このような誤りを人手で発見することは、非常な労力がかかるため、ルールの誤りを効率的に検出する手法が必要となっている。　現在まで報告されているルールの誤り検出方法には、大別して以下の２つのアプローチがある。

(1) ルール間の関係を静的に解析することにより、ルールの誤り検出を行う。　この方法は、実行効率が良い反面、誤り検出能力が低い。
(2) ルールに誤りがないことを論理的に証明する。　この方法は、誤りの検出能力が高い反面、実行効率が悪い。

(1) の手法としては、Suwa［８］、Davis ［３、４］、渡辺［９］がある。　Suwaは、ルールのＬＨＳ（注１）の各条件のパラメータが取り得る値の組み合せの表を作成することにより、
①ＬＨＳが同一であるのに、ＲＨＳが異なるアクションを持つ場合（矛盾）
②パラメータのとりうる組合わせに対応するルールが存在しない場合　（不足）、及び、
③ＬＨＳ、ＲＨＳの両方が同一なルール（冗長）を検出している。
また、Davis 、渡辺は、ルールのＬＨＳ、ＲＨＳで記述しているパラメータをもとに、ルール間の連鎖関係を求め、グラフで表現することにより、新たに追加されるルールと既存のルール間で、ＬＨＳでのパラメータの不足、冗長を検出している。
(2) の手法としては、古川他［５］の知識同化の研究がある。　これは、Prologの節で記述されたルールや、事実を対象として、①制約条件と矛盾する事実やルール、②冗長な事実やルールをdemo述語を用いて検出するものである。

我々は、実行効率、検出能力の双方とも(1) と(2) の中間にあると考えられるデータフロー解析を用いてルールの誤りを検出する手法を考案した。　この手法により、(1) の手法では検出できないWM（ワーキングメモリ）の状態に依存した次の誤りが効率良く検出可能となる。
(i) 与えられたWMの初期状態に対して、目標とする結論状態に達するルール連鎖がない。（ルールの不足）
(ii)与えられたWMの初期状態に対して、導かれる結論が異なる。（ルールの矛盾）
(iii) WMの取り得る全ての状態に対して、起動されることのないルールが存在する。（冗長なルール）

注１：ルールの条件部をLHS、実行部をＲＨＳと呼ぶ。

以下、本稿では、ＰＳに対するフローグラフの作成方法、及び上記誤りを検出するためのフロー解析アルゴリズムを提案する。

２．プロダクションシステムのモデル

本稿で対象とするプロダクションシステム（ＰＳ）のモデルについて説明する。ＰＳは、ワーキングメモリ（WM）と呼ばれるデータベースと、WMの内容を書き換えるルールの集合であるプロダクションメモリ（ＰＭ）からなる。

２．１　プロダクションシステムの構成

(1) WMの構成

WMは、WMの要素（ＷＭＥ：ワーキングメモリエレメント）の集合である。

ここで、

ＷＭＥ　::=　＜パラメータ　値＞（注１）

(2) ルールの構成

ルール　::=　ＩＦ　条件の列　ＴＨＥＮ　アクションの列

条件の列　::=　条件｜条件　ＡＮＤ　条件の列

条件　::=　＜パラメータ　値の集合＞

アクションの列　::=　アクション｜アクション　ＡＮＤ　アクションの列

アクション　::=　＜パラメータ　値＞

２．２　実行モデル

WMがとる状態には、初期状態、中間状態及び結論状態がある。

①初期状態：ルールが起動される直前の状態。

②中間状態：ルールが実行中の状態。

③結論状態：結論となるＷＭＥからなるWMの状態。

ＰＳは、WMの初期状態をもとに以下の処理を繰り返し行う。

(1) 条件照合（MATCH ）処理

ルールのＬＨＳの条件式とWMの要素とを照合し、ＬＨＳの全ての条件式を満足するようなルールの集合（競合集合）を作成する。　ここで照合したWMの要素が条件を満足するとは、ＷＭＥの値∈条件の値の集合の時を言う。

(2) 競合解決処理（CONFLICT RESOLUTION ）

競合集合の中からある戦略により起動すべきルールをひとつ選択する。このモデルでは、競合解決手法は、特に限定していない。　競合集合がφであれば、ＰＳの実行を中止する。

(3) 実行処理（ACT ）

選択されたルールのＲＨＳのアクションを実行し、WMの要素の書き換えを行う。即ち、書き換えられるパラメータについては、ＲＨＳ実行以前の値は全て失われている。　（注２）

注１：EMYCINでは、＜条件関数　コンテクスト　属性　値＞を条件式として採用しているが、我々のモデルでは、コンテクストと属性をひとつのパラメータとして表現している。

注２：ＲＨＳの実行モデルには、大別して　①ＷＭＥの追加、削除型（OPS5）、②ＷＭＥの書き換え型(LOOPS)がある。　本稿では、②の型に焦点を絞って議論を進める。

3．プロダクションフローグラフ

プログラムにおけるフローグラフは、実行文をノード、実行文間の制御移行をノード間のエッジで表現している。 これに対してルールでのフローグラフ（以後プロダクションフローグラフ：ＰＦＧと呼ぶ）は、各ルールをノード、ルール間の実行可能関係をエッジで表現することにより作成する（図１(1) 参照）。 ノードはルール実行開始前のWMの状態を入力し、ルール実行後のWMの状態を出力するものと考える。

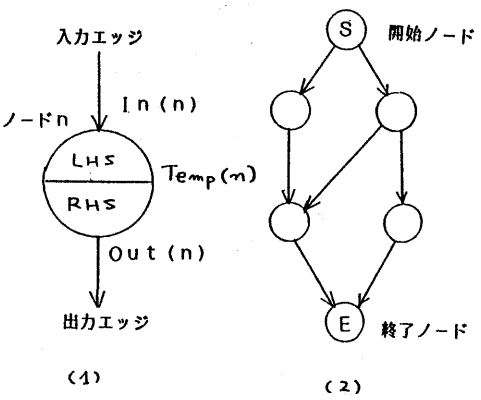


図1．プロダクションフローグラフ

3．1 WMの状態の集合

ＰＦＧを用いてWMの状態を解析するために、WMの状態の集合という概念を導入する。これは、『WMのとりうる状態』を表現するための道具である。 例えば、WMの状態ＷＭ１、ＷＭ２に対して、WMの状態の集合 ＊WMは次のように定義される。

Ｄｅｆ：ＷＭ１のＷＭＥ、＜パラメータ 値１＞と、ＷＭ２のＷＭＥ、＜パラメータ 値２＞について、それぞれの値の和集合を要素としたＷＭＥを考えることにより、＜パラメータ ｛値１、値２｝＞が得られる。 各パラメータについて、このように和集合をとったものをＷＭ１、ＷＭ２の集合と呼び、＊WMで表わす。□

さらに、WMの状態の集合＊ＷＭ１、＊ＷＭ２に対する集合演算∪と∩を次のように定義する。

Ｄｅｆ：＊ＷＭ１∪＊ＷＭ２（＊ＷＭ１∩＊ＷＭ２）は＊ＷＭ１の要素、＜パラメータ 値の集合１＞と＊ＷＭ２の要素、＜パラメータ 値の集合２＞について値の集合の和集合（積集合）をとることである。□

Ｄｅｆ：ノードｎに関する＊WMには、以下のようなものがある。

Ｉｎ（ｎ）：ノードｎの入口での＊WM、即ちノードｎに対応するルールの実行以前の＊WM。

Ｏｕｔ（ｎ）：ノードｎの出口での＊WM、即ちノードｎに対応するルール実行後の＊WM。

In_i（ｎ）：ノードｉからノードｎへのエッジ上での＊WM。

Out_j（ｎ）：ノードｎからノードｊへのエッジ上での＊WM。

Ｔｅｍｐ（ｎ）：ノードｎ実行中（ＬＨＳ処理後、ＲＨＳ処理前）の＊WM。計算過程を説明するのに用いる。□

3．2 ノード上の関数

Ｉｎ（ｎ）（Ｏｕｔ（ｎ））から、Ｏｕｔ（ｎ）（Ｉｎ（ｎ））を決定する関数ｆｎ（ｇｎ）を以下のように導入する。

Ｄｅｆ：関数f_n：Ｉｎ（ｎ）－＞Ｏｕｔ（ｎ）を１入力、１出力のノードについて以下のように定義する。

(1) Ｉｎ（ｎ）がノードｎに対応するルールのＬＨＳを満足する場合には、ルール実行

後の＊WMをＯｕｔ（ｎ）とする。
ここで、“満足する“とは、ＬＨＳの条件で記述された値の集合とＩｎ（ｎ）の値の集合との間に共通部分がある。 即ち条件が成立する可能性がある場合を言う。
Ｏｕｔ（ｎ）は、以下のように計算する。
①各パラメータについて条件に記述された値の集合とＩｎ（ｎ）の値の集合の共通部分を値としたものをＴｅｍｐ（ｎ）とする。
②アクションに設定されたパラメータについては、Ｔｅｍｐ（ｎ）の値の集合を無効とし、アクション中に指定された値を有効とする。
(2) Ｉｎ（ｎ）がノードｎに対応するルールのＬＨＳを満足しない場合には、Ｏｕｔ（ｎ）＝Φとする。□

Ｄｅｆ：関数ｇｎ：Ｏｕｔ（ｎ）－＞Ｉｎ（ｎ）を１入力、１出力ノードについて、以下のように定義する。
(1) Ｏｕｔ（ｎ）がノードｎに対応するルールの実行後に取り得る＊WMならば、そのルール実行前の＊WMをＩｎ（ｎ）とする。
Ｏｕｔ（ｎ）が、”実行後に取り得る”とは、以下の２つの条件を満足する場合である。
①各パラメータについて、アクションに指定された値とＯｕｔ（ｎ）の値の集合との間に共通部分がある。
②各パラメータについて後述するＴｅｍｐ（ｎ）の値の集合と条件に記述された値の集合との間に共通部分がある。
Ｉｎ（ｎ）は以下のように計算する。
①アクションに記述されたパラメータについては、Ｏｕｔ（ｎ）での値の集合を無効とし、´＿´で置き換え、Ｔｅｍｐ（ｎ）とする。 ´＿´は、その値が任意であることを示す。
②各パラメータについて条件に記述された値の集合とＴｅｍｐ（ｎ）の値の集合の共通部分を値としたものをＩｎ（ｎ）とする。
(2) Ｏｕｔ（ｎ）がノードｎに対するルールの実行後に取り得る＊WMでないなら、Ｉｎ（ｎ）＝Φとする。□

３．３ プロダクションフローグラフの作成方法

プロダクションルールは、プログラムの実行文とは異なり、一般的にはいつも実行される可能性がある。 しかし、ルールＡ（ノードａ）とルールＢ（ノードｂ）が以下の条件を満す場合には、明らかに連続して実行される可能性がないことがわかる。

条件：Ｉｎ（ａ）を任意とした時に得られるＯｕｔ（ａ）の各パラメータについて、ルールＢのＬＨＳの条件に記述された値の集合との間に共通部分がない場合。□

そこでＰＦＧの作成にあたっては、上記の条件を満すノード間には、あらかじめエッジを張らないことにする。 図１(2) にＰＦＧの例を示す。 図に示すように、特別なノードとしてルールの実行開始を示す開始ノード（Ｓ）とルールの実行終了を示す終了ノード（Ｅ）を付加しておく。 このようにすると、ルール起動前の＊WMはＯｕｔ（Ｓ）で、ルール実行後の結論状態はＩｎ（Ｅ）で表わすことができる。 Ｏｕｔ（Ｓ）、Ｉｎ（Ｅ）

は、ＰＦＧ作成時に初期設定されているものとする。ＰＦＧを作成する過程で、どのルールからも制御の渡らないルール（冗長なルール）及び、どのルールに対しても制御を渡すことがないルール（ルールの不足）が検出可能である。

４．データフロー解析による誤り検出方法

本章ではＰＦＧ上の各地点での＊WMをデータフロー解析を用いて求めることにより、ルールの不足、冗長、矛盾を検出するアルゴリズムを述べる。 データフロー解析を用いる利点は、ノードの入（出）口の＊WMを、全てのルールの実行系列を求めることなしに、得ることができることにある。 具体的には、各ノードの入（出）口の状態からノードの出（入）口の状態を求める処理を、全ノードについて繰り返し収束するまで計算する。
以下では、ＰＦＧ上での ①前向き（各ノードの入口の＊WMから出口の＊WMを求める）②後向き（各ノードの出口の＊WMから、入口の＊WMを求める）の２方向のデータフロー解析を行うアルゴリズムを述べる。 そのためＰＦＧの各ノードは、開始／終了ノードを含めて、r-post orderの順で番号づけされているものとする。 記法として以下のものを導入しておく。

Ｐｒｅｄ（ｎ）：ノードｎの直前の先行ノード（immediate predecessor ）の集合。
Ｓｕｃｃ（ｎ）：ノードｎの直後の後続ノード（immediate successor ）の集合。

４．１ 前向きのフロー解析

ノードｎが複数の入力エッジを持つ場合には、各Ｏｕｔ（ｎ）は、以下の式で求められる。

$$Out(n) = \bigcup_{i \in Pred(n)} f_n(In_i(n)) \quad (式1)$$

ここでＯｕｔ（ｎ）＝$Out_j(n) = In_n(j)$（ｊ∈Ｓｕｃｃ（ｎ））である。

（式１）に対しKildall の反復アルゴリズム（Hecht ［６］）を適用することにより、次のような手順で前向きのフロー解析を行う。
(1) 各ノードについてＯｕｔ（ｎ）の初期設定を以下のように行う。
①Ｏｕｔ（Ｓ）、Ｉｎ（Ｅ）については、ＰＦＧ作成時に用いたものをそのまま用いる。
②それ以外のノードについては、Φを設定する。
(2) 全てのノードについてr-post orderの順に（式１）を計算し、Ｏｕｔ（ｎ）を更新する。
(3) Ｏｕｔ（ｎ）が一定になるまで、(2) を繰り返す。
このようにして得られたＯｕｔ（ｎ）から、次の誤りが検出可能である（図２参照）。
①Ｏｕｔ（ｎ）がΦであるノードが存在するならば、そのルールは与えられた初期状態に対して起動されることのないルールである。（冗長なルール）
②（In_j（Ｅ），但し ｊ∈Ｓｕｃｃ（Ｅ））に結論状態と共通部分のあるものが存在しないなら、与えられた初期状態に対してルールが不足していると言える。（ルールの不足）
さらに、前向きのデータフロー解析で得られたＯｕｔ（ｎ）を用いて、３．３で述べた

2ルール間の実行可能性をチェックすることにより、実際には制御が渡らないＰＦＧ上のエッジを切ることが出来る。（図２の場合は、ノード２から３へのエッジが切られる。）
後述する後向きのデータフロー解析は、このようにして更新されたグラフを基に行う。

ルール例

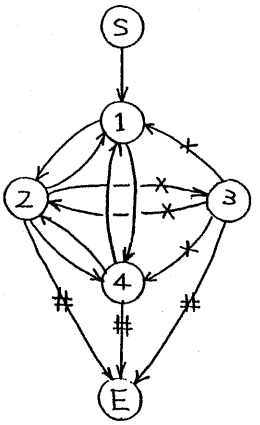
1. IF <X [0、∞)> THEN <Y 3>
2. IF <Y [0、∞)> AND <Z [0、∞)>
   THEN <W 1>
3. IF <Y {2}> AND <Z {3}>
   THEN <W 2>
4. IF <W [2、∞)> THEN <Y 4>

初期状態Out(S)

(<W φ><X [0、3]><Y φ><Z [0、3]>)

結論状態In(E)

(<Y {2}>)



ノード出口での*WM

ノード	初期設定	フロー解析による収束時
1	Φ	(<W 1><X [0、3]><Y {3}><Z [0、3]>)
2	Φ	(<W 1><X [0、3]><Y {3、4}><Z [0、3]>)
3	Φ	Φ
4	Φ	(<W 1><X [0、3]><Y {4}><Z [0、3]>)

注１：X印は、実行可能関係がないことを示す。

注２：＃印は、結論状態と共通部分を持たないことを示す。

この例では、以下の誤りが検出される。

①ノード３の出口でのWMの状態の集合は、Φであることから、ルール３が冗長である。

②ノードEに至る全てのエッジで、結論状態に至るものが存在しないことから、ルールが不足している。

図２．前向きのデータフロー解析の例

４．２　後向きのフロー解析

ノードｎが複数の出力エッジを持つ場合には、Ｉｎ（ｎ）は以下の式で求められる。

$$In(n) = \bigcup_{j \in Succ(n)} g_n(Out_j(n)) \quad (式2)$$

ここでＩｎ（ｎ）＝Ｉｎ$_j$（ｎ）＝Ｏｕｔ$_n$（ｉ）（ｉ∈Ｐｒｅｄ（ｎ））である。

（式２）に対しKildall の反復アルゴリズムを応用することにより、次の手順で後向きのフロー解析を行う。

(1) 各ノードについてＩｎ（ｎ）の初期設定を以下のように行う。

①Ｏｕｔ（Ｓ）、Ｉｎ（Ｅ）については、ＰＦＧ作成時に用いたものをそのまま用いる。

②それ以外のノードについては、Φを設定する。

(2) 全てのノードについてr-post orderの逆順に、（式２）を計算し、Ｉｎ（ｎ）を更新する。

(3) Ｉｎ（ｎ）が一定になるまで(2) を繰り返す。

このようにして得られたＩｎ（ｎ）から、次の誤りが検出可能である（図３参照）。

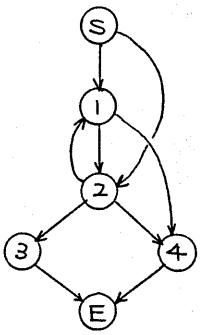
①Ｉｎ（ｎ）がΦの場合には、ノードｎに対応するルールが、実行されたとしても結論を導かない。　従ってこの場合、そのルール自身が冗長であるか、ルールが不足している可能性がある。（ルールの冗長、不足）

②開始ノードから始まるエッジ上の＊WMの和集合（$\bigcup_{i}$Ｉｎ（ｉ）、但しｉ∈Ｓｕｃｃ（Ｓ））とＯｕｔ（Ｓ）との差集合がφでない場合には、この差集合に対応する初期状態からルールを起動すると結論に達しない。　（ルールの不足）

③結論状態が同一のパラメータについて複数のWMＥを持つ場合、それぞれのWMＥを基に後向きのデータフロー解析を行い、Ｉｎ（ｊ）（但し、ｊ∈Ｓｕｃｃ（Ｓ））を求める。　求めた複数のＩｎ（ｊ）が共通部分を持つなら、その共通の値を持つ初期状態に対しては、異なる結論が得られることがある。（ルールの矛盾）

ルール例

1. IF <X [4、∞)> AND <X (−∞、8]>
   THEN <Y 8>
2. IF <X [3、∞)> AND <X (−∞、6]>
   THEN <Y 7>
3. IF <W {3}> AND <Y {7}>
   THEN (Z true)
4. IF <W {3}> AND <Y {8}>
   THEN <Z false>



初期状態Ｏｕｔ（Ｓ）

（<W [3、6]><X [3、8]><Y φ><Z φ>）

結論状態Ｉｎ（Ｅ）

（<Z {true、false}>）

上記の例では、以下の誤りが検出される。

①互いに異なる結論に対し、各ノードでの入口のWMの状態集合を求める。　これにより開始ノードの出口でのWMの状態集合Ｏｕｔ（Ｓ）の要素としてそれぞれ<X [3、6]>、<X [4、8]>が得られる。　これらの値の共通部分[4、6]の要素を初期状態に持つと矛盾した結論が得られることがわかる。

②フロー解析の結果求められた　ｉｎ（ｉ）（但し、ｉ　Ｓｕｃｃ（Ｓ））は、パラメータWとして値3を持ち、初期状態[3、6]を包含していない。　従って、初期状態でパラメータWの値として[4、6]のものをとると結論が得られない。

結論を（<Z {true}>）とした時の各ノードの入口でのWMの状態集合

ノード	初期設定	フロー解析による収束時
1	Φ	Φ
2	Φ	(<W 3><X [3、6]><Y _><Z _>)
3	Φ	(<W 3><X _><Y 7><Z _>)
4	Φ	Φ

結論を（<Z {false}>）とした時の各ノードの入口でのWMの状態集合

ノード	初期設定	フロー解析による収束時
1	Φ	(<W 3><X [4、8]><Y _><Z _>)
2	Φ	Φ
3	Φ	Φ
4	Φ	(<W 3><X _><Y 8><Z _>)

図３．後向きのデータフロー解析の例

５．考察・まとめ

今回提案したフロー解析を用いたルールの誤り検出アルゴリズムについては、次のことが考えられる。

(1) モデルの記述能力

本稿で述べたルールのモデルは、EMYCINのルールで用いられているルールの条件＜条件関数、コンテクスト、パラメータ、値＞を基本的に表現可能である。

しかし、LOOPS 、OPS5等のルールを表現するには、以下の制約がある。

①条件、アクション共に値は定数でなければならない。

即ち条件で変数へのバインディングが行われたり、アクションでＸ＝Ｘ＋１のように変数を用いた演算を記述することはできない。

②OPS5のような追加、削除型のＰＳには、今のところ対応できない。

①については、フロー解析を用いる所からくる制約であるが、②については、今後モデルの拡張を行う予定である。

(2) 処理速度

データフロー解析の処理時間のオーダは、おおむねノード数の２乗である。

(3) 検証の効果

本アルゴリズムで検出できる誤りは、これまで提案された静的なルールの誤り検出に比べ、

①各ルールがＰＳ全体に及ぼす影響を検出できる

②ＷＭＥの参照、更新のパターンだけでなく、ＷＭＥの値の追跡を行っている

という点で秀れている。 現在、筆者らは、知的教育システムMASTERS ［７］を開発中であり、本アルゴリズムは、MASTERS の教育戦略ルールの誤り検出を目的として考案したものである。 今後、インプリメント手法について検討し、評価を行う予定である。

６．謝辞

本研究の機会を与えて戴いたＮＴＴ横須賀電気通信研究所知識ベース研究室寺島室長に深謝します。

７．参考文献

［１］Barr.A et al:The Handbook of Artificial Intelligence,vol 1～3,Pitman(1981)

［２］Bobrow,D.G and Stefik.M : The Loops Manual(1983)

［３］Davis.R : Knowledge Acquisition in Rule -Based Systems : Knowledge about Representations as a Basis for System Construction and Maintenance、Pattern-Directed Inference Systems、New York:Academic Press 、99/134(1977)

［４］Davis.R : Interactive Transfer of Expertise:Acquisition of New Inference Rules、Artificial Intelligence 12,127/157(1979)

［５］古川他：知識獲得機構の実現法、情報処理学会知識工学と人工知能研究会資料、30-2(1983)

［６］Hecht,M.S : Flow Analysis of Computer Programs、North Holland(1977)

［７］森原他：知識ベースを利用したＣＡＩシステム：MASTERS 、情報処理学会知識工学と人工知能研究会資料、39-4(1985)

［８］Suwa.M et al: An Approach to Verifying Completeness and Consistency in a Rule-Based Expert System,The AI Magazine,fall,16/21 (1982)

［９］渡辺：知識獲得支援システム、計測と制御、Vol 22、No.9、23/29 (1983)

[10] Zadeck,F.K: Incremental Data Flow Analysis in a Structured Program Editor Proc.ACM SIGPLANS Notices Vol 19、No.6、(1984)