6F-3 ＰＩＥにおけるアクティビティ制御 ～メタ述語の実現方式

後藤 厚宏 , 田中 英彦 , 元岡 達
(東京大学 工学部)

１．はじめに

ＰＩＥでは、多数台のアクティビティコントローラ（ＡＣ）によって関係木を保持し、実行時に生成される多数のゴールフレームを関係付ける。ＡＣは相互にノード制御コマンドを送受し合い、メタ述語等を実現する［１］。これまでに本機構の性能評価を目的として、シミュレータ上に幾つかのメタ述語を有する論理型言語のプロトタイプを実装し、性能評価を行なってきた。本報告では、シミュレータにおけるメタ述語の実現方式について述べる。評価結果については別稿［２，３］を参照されたい。

２．プロトタイプの仕様

シミュレータ上に実装した論理型言語の仕様はおおよそ以下の通りである。本言語は基本的に純Ｐrolog であり、リテラルの選択順序は原則として規定されず、また定義節も特に指定がない限り並列に適用される。

メタ述語としては以下の３つが用意されている。
①ｎｏｔ（未定義変数を持たないゴール），
②ｇｕａｒｄ，
③ｓｅｑ（定義節の逐次適用）

これらのメタ述語は主として、ＰＩＥにおけるメタ述語の実現手法の確認と、逐次Ｐrolog 上のプログラムをシミュレータの例題として採用することを目的として実装した。また、算術論理演算，print ，探索ストラテジ制御用述語が評価型述語として用意されている。

関係木を構成するノードの属性は、型（約15種）と数個のフラグ（約10種）によって与えられる。以下、この属性を［type,flags］として表す。一方、関係木を操作するために送受されるコマンドは

＜宛先ノード，コマンド名，引数＞

の型式を持ち、約15種用意されている。

これらのうち、上記のメタ述語の実現に関する型、フラグ、コマンドを図１に示す。

```
                -- types --
[AND] : all  success  ->[SUCC],
        one failure ->[FAIL].
[OR]  : one  success  ->[SUCC],
        all failure ->[FAIL].
[AOR] : all failure ->[FAIL],
        one success ->continue.
[NOT] : not-node.
[GOAL]: goal-node.
[GSS] : guard success and suspend
        goal-node.
[SUCC]: success-node.
[FAIL]: failure-node.
[SQ]  : sequentially applied defini-
        tions.
[GU]  : guarded definitions.

                -- flags --

[Sq]  : sequential.
[G]   : guard select node, or goal-
        node including more than one
        guard-bar.
[Su]  : suspended goal-node.
[O]   : OR flag.

              -- commands --

<success, a> <failure, a>:     node(a)
     succeeds/fails and is pruned.
<guard_success, a> :
     guard_part succeeds at node(a) or
     at descendants of node(a).
<son, a, n> :
     node(a) is the n-th son-node.
<parent_change, a, b> :
     parent-node  is  changed  from
     node(a) to node(b).
<son_change, a, b> :
     son-node is changed from  node(a)
     to node(b).
<eliminate> : can be eliminated.
<stop> : receiver node or
     its descendants should be pruned.
<activate> <suspend>:
     receiver node or its  descendants
     should be activated/suspended.
```

Fig.1 Node Attributes and Node Control Commands (partial)

３．メタ述語の実現例

図２に従ってメタ述語の実行の様子を示す。

（１）ゴールノードmにおいてguard を有する定義節が適用されると、中継ノードｍ［ＡＯＲ，Ｇ］となる。フラグＧは guarded clauses の導入点を示す。一方、新ゴールノードにおいてもＧフラグをセットし、対応するゴールフレームが内部に guard-bar を含むことを示す。

（２）メタ述語ｎｏｔを有するノードａはand 分割され、２つのゴールノードｃ，ｄと、それらを関係付けるＡＮＤノードに置き代る。この時、ゴールノードｃ，ｄは guard-barにおいて分割されたサブゴールフレームに相当する。このため、フラグＧとノードｃ，ｄ間の逐次実行を指示するフラグＳｑが中継ノードａにおいてセットされる。一方ノードｄにおいてはフラグＳｕがセットされ、親ノードａによって活性化されるまて実行が停止される。

（３）メタ述語ｎｏｔにより、ゴールノードｃは定義節の適用が全て失敗したとき成功ノードとなり、＜success ＞コマンドを親ノードａに送る。ノードａは自身の属性中のフラグＧからguard 部が成功したことを知り、それを＜guard-success ＞コマンドによって親ノードｍに知らせる。

（４）ノードｍは guarded clauses の導入点であるため、最初に＜guard-success ＞コマンドを送ってきた子ノードを選択し、 guard-body を活性化するために＜activate＞コマンドを送る返す。また他の子ノードｂに対しては＜stop＞コマンドを送り、実行を強制終了させる。ノードｍから送られてきた＜activate＞コマンドは停止中のゴールノードｄまで伝搬し、guard-bodyの実行がスタートする。

４．おわりに

今後は、応用プログラムを広く検討し言語において要求されるメタ述語の機能を整理するとともに、ＡＣおよびコマンド通信網の設計を進める予定である。

＜参考　文献＞

［１］　後藤　他“高並列推論エンジンＰＩＥについて”　Logic　Prog.　Conf.´８３　ＩＣＯＴ.

［２］　丸山　他“ＰＩＥにおけるアクティビティ制御～コマンドトラヒックのシミュレーション評価”本大会　６Ｆ－４.

［３］　濱中　他：“ＰＩＥの並列度評価”本大会.６Ｆ－10

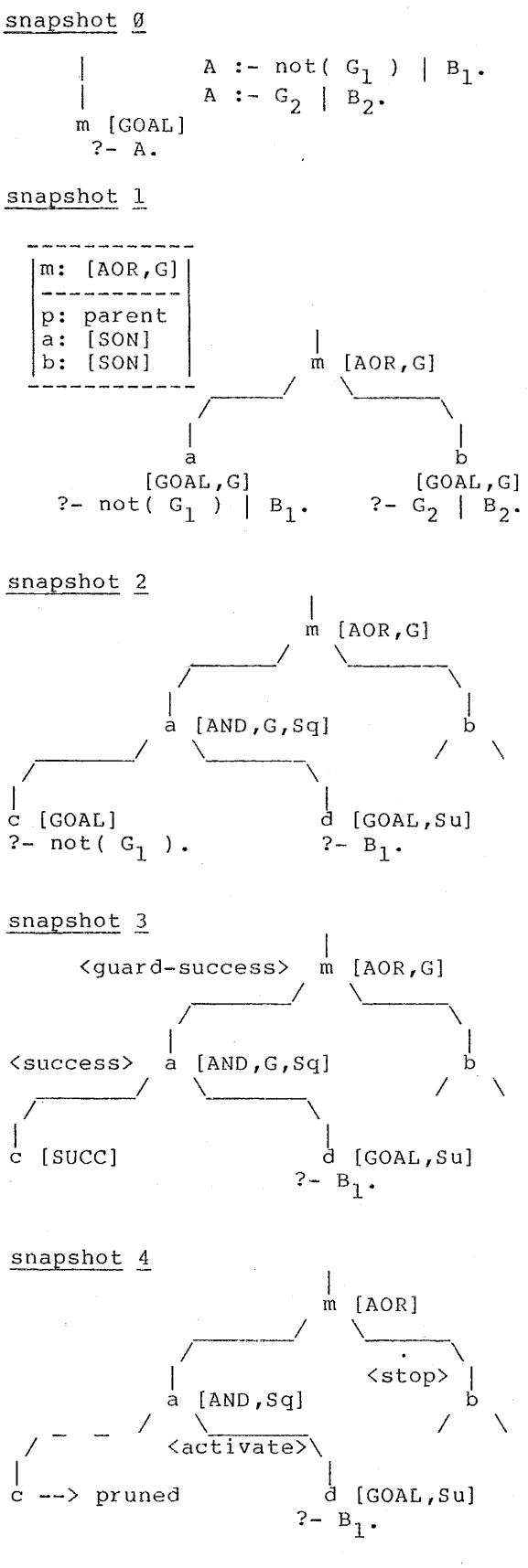

Fig.2　Snapshots of an Execution Example