REFINING AND EXTENDING THE PROCEDURAL NET

Mark E. Drunimond

Department of Artificial Intelligence
University of Edinburgh
Hope Park Square
Edinburgh, Scotland, U.K.

Abstract

This paper presents a new definition for Plans The objects defined are called Plan Nets, and are similar in spirit to Sacerdoti's Procedural Nets (1975) It is argued that Plan Nets are more descriptive tham Procedural Nets, because they can easily describe iterative behaviour The Plan Net definition is motivated by providing an operational semantics for the Procedural Net, and noticing that all Procedural Net state spaces are "loop free" This is seen to restrict the behaviours that can be described by the Procedural Net to those which do not. include iteration It is suggested that Plan Net. state spaces can contain loops, and thus can describe iterative behaviour

1. Paper Overview

In the next section we give Sacerdoti's definition of the Procedural Net, A simple method for deriving Procedural Net behaviours is also presented, and it is argued that the Procedural Net cannot describe iteration Section 3 defines and discusses the Plan Net Two sample Plan Nets are given An operational semantics is suggested for the Plan Net, and it is argued that Plan Nets can describe iteration The Procedural Net and the Plan Net are compared in Section 4 Section 5 concludes

2 The Procedural Net

A Procedural Net has been defined as "a network of actions at varying levels of detail, structured into a hierarchy of partially ordered time sequences " (Sacerdoti, 1975, p 10) The basic objects in a Procedural Net are actions, and some ordering relations on the actions Because of this, we refer to the Procedural Net as an "event space" representation A net can be drawn as an action-on node graph, with directed arcs between nodes An arc running from one node a to another node B means that the action denoted by a must occur "before" the action denoted by B

We can derive the possible behaviours of a given Procedural Net by analyzing the state space which it describes. A net's state space can be produced by playing a version of the "pebbling game" (Pippenger, 1980) While pebbling was not developed with this application in mind. it does capture our intuition of what "before" means in a Procedural Net In our version of this game, we place "pebbles" on the nodes of a Procedural Net as they are executed The net starts out pebble-free. and finishes up pebble laden each node must be pebbled; that, is, each action must be executed Pebble placement is carried out according to the rule: A node may be pebbled if all of its immediate predecessors arc pebbled.

The Procedural Net has been criticized recently (McDermott. 1983, Rosensehein. 1984) This paper addresses the Procedural Net's inability to describe "iterative" behaviour Such behaviour is difficult to model in a natural way using a Procedural Net Since the arcs of a net are taken to mean "before", one cannot simply direct an arc from an action "back into" the net

It is obvious that all Procedural Net state spaces will be loop free, since the number of pebbles on a net must increase monotonically There will never be an action which removes a pebble; thus never an action which can produce an earlier state This is due to the strict, "before" interpretation of the Procedural Net's arcs

While Sacerdoti did include a mechanism for dealing with iteration, it hides the notion of "process" inside a special replicate node His treatment of iteration poses problems Below, we suggest that by defining an alternative "event space" representation for plans, we can describe iterative behaviour. This new definition follows the belief that iteration must be expressed in terms of the structure of a plan, so that a planner can reason about the iteration

3. The Plan Net

In this section, we define Plan Nets, using some basic concepts from Net Theory (Brauer, 1979).

Definition 1. A Plan Net is a 6 tuple $\langle P,T,R_a,R_b,R_c,R_e\rangle$, where $P = \{p_1,p_2,\ldots,p_N\}$, a finite set of places; $T = \{t_1,t_2,\ldots,t_M\}$, a finite set of transitions; $N \geq 0$, $M \geq 0$; and $T \cap P = \phi$, the empty set. $R_a = \{(t_i,t_k) \mid \exists p_j \in P \ [(t_i,p_j)\in R_c \ \& \ (p_j,t_k)\in R_e]\}$, the Allow relation. $R_e \subset (P\times T)$, the Enable relation; $R_c \subseteq (T\times P)$, the Cause relation $R_b \subseteq (T\times T)$, the Before relation; R_b must be a strict partial order on T

Definition 2. Place p_i is an input place of transition t_j if and only if $(p_i,t_j) \in R_a$. Place p_i is an output place of transition t_j if and only if $(t_j,p_i) \in R_c$.

Definition 3. A marking of a Plan Net is a mapping μ: P → {0, 1}.

Definition 4. A transition t_j is enabled if for each input place, p_i, $\mu(p_i) = 1$. A transition t_j may fire when enabled. Firing t_j in a marking μ produces a new marking μ' such that

1) If p_i is an input place of t_j
   then $\mu'(p_i) = 0$;
2) If p_i is an output place of t_j
   then $\mu'(p_i) = 1$;
3) If p_i is not an input place
   and not an output place of t_j
   then $\mu'(p_i) = \mu(p_i)$.

A place is thought of as a "condition", a static thing which does or does not hold. If a place pi is marked (M(pi) = 1) it is considered to be believed [by the planning system], and if it is unmarked (u(pi) = 0), it is considered to be not believed [by the planning system].

Transitions are events, the Plan Net counterparts of a Procedural Net's actions. Events are dynamic entities; conditions are static Events are things which "happen"

The Allow relation (Ra) is an abstraction of two simpler relations: Cause (Re), and Enable (R_e) Events cause conditions, and conditions enable events The input places of a transition describe those conditions which must be believed to hold in the world for the event the transition denotes to be enabled The firing of a transition models the activation of its event. After firing, the output places of the transition describe the new conditions that are believed to hold

Sample plans in this formalism are given in Figures 1 and 2 The plan of Figure 1 is designed to solve a canonical Blocks World problem. An initial marking is included. The plan of Figure 2 is one for (endlessly) hammering a nail. The plans look large, but this is due to redundant information being included in the formalism When a plan is drawn as a graph it shrinks to more modest proportions (see Drummond, forthcoming),

C = <P,T,R_a,R_b,R_c,R_e>

P = { (on c a), (on c t3), (on a t1),
(on b t2), (on b c), (on a b),
(clear t3), (clear a), (clear t1),
(clear b), (clear t2), (clear c)}

T = { (move c a t3), (move a t1 b),
(move b t2 c) }

R_c = { ((move c a t3), (clear a)),
((move c a t3), (on c t3)),
((move c a t3), (clear c)),
((move b t2 c), (clear b)),
((move b t2 c), (on b c)),
((move b t2 c), (clear t2)),
((move a t1 b), (clear a)),
((move a t1 b), (clear t1)),
((move a t1 b), (on a b)) }

R_e = { ((on c a), (move c a t3)),
((clear t3), (move c a t3)),
((clear c), (move c a t3)),
((clear a), (move a t1 b)),
((clear b), (move a t1 b)),
((on a t1), (move a t1 b)),
((clear c), (move b t2 c)),
((clear b), (move b t2 c)),
((on b t2), (move b t2 c)) }

R_a = { ((move c a t3), (move b t2 c)),
((move b t2 c), (move a t1 b)),
((move c a t3), (move a t1 b)),
((move b t2 c), (move b t2 c)),
((move a t1 b), (move a t1 b)),
((move c a t3), (move c a t3)) }

R_b = { ((move c a t3), (move b t2 c)),
((move b t2 c), (move a t1 b)) }

$\mu(p) = 1$ if p ∈ {(on c a), (clear t3),
(clear c), (clear b),
(on b t2), (on a t1)}
$\mu(p) = 0$ otherwise.

Figure 1: A plan. (move X Y Z) means "Move block X from Y to Z".

4 Comparing the Nets

In this section we argue that the Plan Net representation is more powerful than the Procedural Net because of an explicit epistemological commitment to conditions and events The Procedural Net makes no clear distinction between them. Using the Allow and Before relations as defined above, we can produce state spaces which contain cycles; that is, ones which correspond to iterative behaviour.

NOAH (Sacerdoti, 1975), and its descendents, such as NONLIN (Tate, 1976), DEVISER (Vere. 1981). and SIPE (Wilkins, 1983) all use plans based on Sacerdoti's original Procedural Net The following comments are expressed principally in terms of NOAH, but apply equally to these newer planners

The Allow relation takes the form of a "before" link in a Procedural Net which has been introduced by pattern-directed operator invocation. Such a link might appear, for instance, between an action which must make block A clear, and an action which must stack A on B.