Process Flow Model

1N-5

中村 明
(株)東芝

(0)はじめに

Process Flow Modelは、UNIXのプロセス管理の複雑な機構を解析する目的で筆者によって考案されたペトリネットの一変種である[1]。

ペトリネットの構成[2][3]にしたがって、Process Flow Graph, Process Flow, Marked Process Flow,更にサービス関数、要求関数が定義されて、Process Flow Model の枠組が組み立てられる。

ここで対象とするUNIXシステムはスワッパというプロセス番号0を持ったシステム特殊プロセスと、PMAX個まで生成消滅するプロセス群より成立っているものとする。

(1) Process Flow

[定義1] プロセス番号iの Process Flow Graph:PFGi は、3組 (P_i , T_i , δ_i) から成立する。ここで、P_iはプレースの集合、T_iはトランジションの集合、 δ_i: $P_i x P_i \rightarrow T_i$は、結合部分関数である。

プレース集合 P_iは互いに素の活動プレース集合PA_iと受動プレース集合PP_iとNIL_iから成立っている。NIL_iはシステム中でプロセスが生成されていない事を意味する。スワッパ(プロセス番号 0)のプレース集合は$P_0 = PA_0 \cup PP_0 \cup \{a_0\}$と表わせ、$a_0$はプロセススケジュールの為の特殊プレースである。

トランジション集合$_i T_i$は同じく互いに素の、要求トランジション集合 TR_iと放棄トランジション集合 TG_iと獲得トランジション集合 TC_iとサービストランジション集合 TS_iと実行トランジション集合 TE_iとから成立する。

上記3種のプレース集合と5種のトランジション集合には次の関係が成立する。

$\delta_i(\{NIL_i\} x PP_i \cup PP_i x PP_i \cup PP_i x \{NIL_i\}) = TR_i$

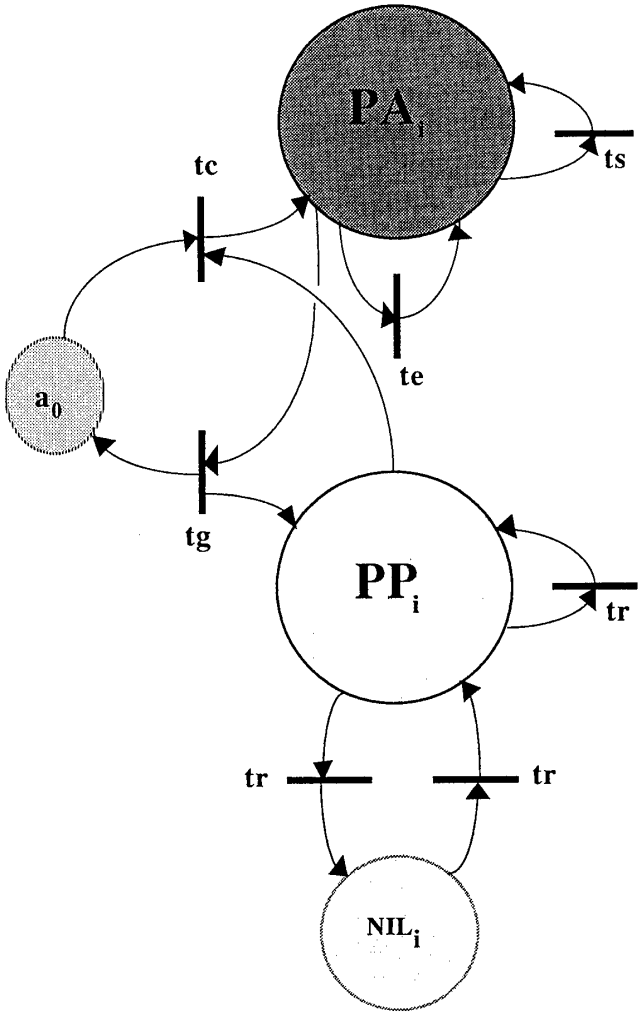
$\delta_i(PP_i x PA_i) = TC_i$

$\delta_i(PA_i x PP_i) = TG_i$

$\delta_i(PA_i x PA_i) = TS_i \cup TE_i$

[定義2] Process Flow:PFは4組で

PF=(P, T, A, R) よりなりたつ。ここで、$P=\cup_i P_i$, $T=\cup_i T_i$, Aは有向アークの集合、$R=\cup_i R_i$は要求集合である。r∈Rは、プロセスの要求リストである。プロセスiは、アーク集合A_iをもつ。最も単純なProcess Flowの構成を図1に示す。



(図1)単純なProcess Flowの構成

上図で、tr,tg,tc,ts,teは、各々トランジション集合 TR_i, TG_i, TC_i, TS_i, TE_iに属する。

Process Flowでは、各Process Flow Graphは次の結合アーク集合C_iによって全体が結合されている。

$C_i = \{<tg, a_0> \mid tg \in TG_i\} \cup \{<a_0, tc> \mid tc \in TC_i\}$

[定義3]Marked Process Flowは 2組 : MPF=(PF, μ)から成立する。ここで

Process Flow Model
NAKAMURA AKIRA
Toshiba Coporation

$\mu=\{\mu_0, \mu_1, - - -, \mu_{PMAX-1}, \mu_0'\}$は、マーキング空間Mにおける、マーキングされたプレースの状態を表す。マーキング空間Mは、次の式で表される。

$M=\{P_0-a_0\} \times P_1 \times - - \times P_{PMAX-1} \times \{a_0, \varnothing\}$

ここで∅は、トークンがない状態を表す。

(2)トランジションの発火

トランジションの発火は、P/Tペトリネットと、同様に定義される。5種のトランジション発火の特徴について以下に述べる。

(a) $tg \in TG_i$の発火の時

プロセス番号iのプロセスは、要求リストに要求r_iを書込み、現行権を放棄し、他のプロセスからのサービスを待つ状態に入る。トークンは受動プレース集合に入る。

(b)$tc \in TC_i$の発火の時

走行可能プロセスの中から、次に走行するプロセスとして選択され、現行権を獲得し走行を開始する。トークンは活動プレース集合に入りその中を動き回る。

(c)$ts \in TS_i$の発火の時

要求リストrを探索し、要求に基づきサービス関数SV: $TS \times R \rightarrow \Pi(TR_i \cup \{\varnothing\})$の発火列を実行する。SV(ts、r)の全てのトランジションの発火を終了するまで、新たな発火は行なわれない原子的操作である。即ち、要求に対するサービスを待っているプロセスの要求トランジションを全て発火させる。これは、UNIXのカーネル関数wakeup,signal等に対応する。

(d)$tr \in TR_i$の発火の時

他のプロセスjのトランジション$ts \in TS_j$の発火に伴ってのみ発火する受動的なトランジションである。発火すれば新たな要求を要求関数ρ_i: $\{TR_i \cup TG_i\} \times R_i \rightarrow R_i$に基づき要求リストrに書込む。

サービストランジションの発火に連動した要求トランジションの発火の様子を通常のペトリネットに近い形で思い描くとすれば図2のようになるであろう。

(e)$tr \in TR_i$の発火の時

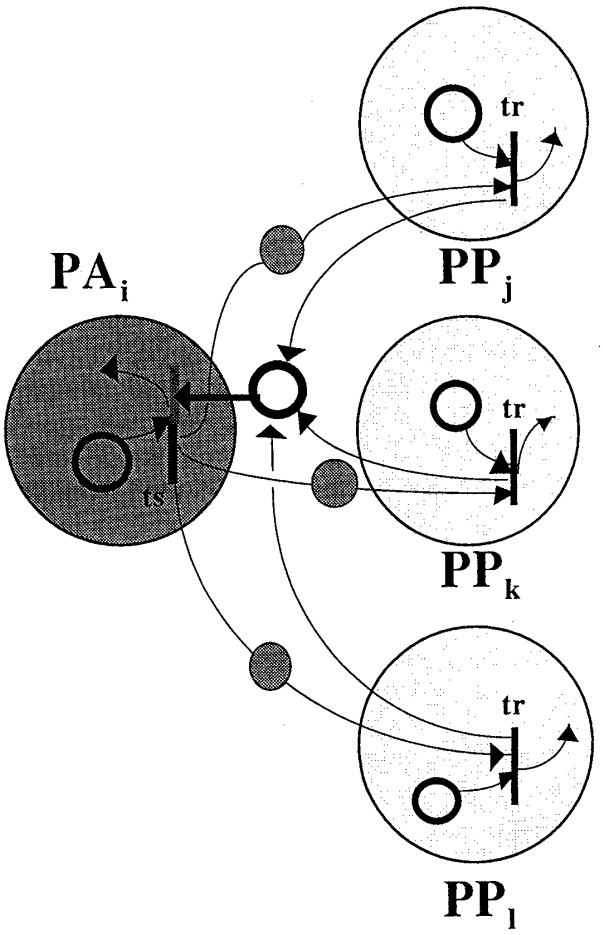
プロセスi自身の操作を実行する。

(3)システム状態空間

マーキングと要求リストの組(μ, r)は、一定時刻におけるシステム状態をあらわしている。$SS=M \times R$をシステム状態空間として定義する。発火によるSS中でのシステム状態の変化を調べる目的で次システム状態関数$\sigma: SS \times T \rightarrow SS$が定義されて、システム解析の有力な武器になる。

(4)結論

UNIXのプロセス管理を解析するために考案されたペトリネットの変種であるProcess Flow Modelの概要について述べた。当モデルは進化の激しいUNIX野プロセス管理解析に有効な基礎を与えるのを目指したものである。



(図2)サービス・要求トランジションの連動発火

[参考文献]

[1]中村: オペーレティングシステム構築法,丸善, 1986.

[2]Peterson, J: Petri Net Theory and The Modeling of System, Prentice-Hall, 1981.

[3]Reisig, W: Petri Net:An Introduction, Springer-verlag,1985.