数理解析研究所講究録
第 427 巻 1981 年 98-111

DATA FLOW ANALYSIS OF PROGRAM NETS

Hiroshima Univ., Faculty of Engineering

Kenji Onaga

1. Introduction

A flow chart is a widely used graphical model for expressing a structure of a program. An execution of the program is expressed by a trajectory of a single token through chart elements where the token is processed upon by operators (functions, procedures) in rectangular boxes or directed to one of several destinations by deciders in decision boxes. The token in the flow chart represents a mixture of control flows and data flows among program elements. At anytime moment, only a single token exists in the chart.

A program graph of Rodriguez [1] and Dennis [2] is a variation of Petri nets of Petri [3], specifically tuned to program representation. It can be seen as a simple case of GAN and GERTS of Elmaghraby [4] and Pritsker [5] in operations research areas. In the program graph a node represents either an operator, decider, merge, fork or switch. An edge represents a communication channel of tokens between nodes and serves also as a FIFO queue of holding token. A token represents abstract data whose structure varies from edges to edges. An execution of the program is expressed by a flow of token through the graph where tokens are transfered across a node from input edges to output edges by node firing. The output of a decider is binary-valued and is used to control a switch to open or close. See Fig 1 for a program net.

In this paper we deal with dataflow analysis of program nets (we prefer this terminology for emphasizing token flows) by means of firing sequences and presents some basic properties on maximal firing numbers, and terminating conditions.

2. Net Elements [2]

A program net N is token flow activity on a base digraph G=(V,E) with initial token distribution d° such that topology of G is depicted in Fig. 2 where

V: set of nodes of 5 types

E: set of directed edges e with initial d_e° tokens

s: source

t: sink

(s , s'): input edge with a single start token

(t', t): output edge with no token

for any node x there are directed paths from the source s and to the sink t.

Five types of nodes are as follows:

1) operator It possesses one or two inputs and one output. The output is regarded a funcion of the inputs.
2) decider It possesses one or two inputs and one output. The output is a binary-valued function of the inputs and used as the control signal to a switch.
3) Merge It possesses two inputs and one output. The output is a merge of the inputs.
4) Fork It possesses one input and two outputs. The outputs are copies of the input.
5) Switch It possesses one input and two outputs. A token on the input is directed to output T (or F) if the token value on the control input is T (or F).

Firing rules of respective node types are as follows:

1) AND-node (operator, decider, fork) is firable if each of its inputs has at least one token. If a firable node is fired, one token is removed from each input and one token is placed on each output.
2) OR-node (Merge) is firable if one of its inputs has at least one token. If a firable node is fired, one token is removed from one but not both of its inputs and one token is placed on the output.
3) Switch-node is firable if the data input and control input have at least one token each. If the value of the head token on the control input queue is T (or F), then the token on the data input is directed to the output terminal T (or F) and the control token is removed.

The firing rule of various node types is summarized in Table 1. We choose as the net element only those tabulated in Table 1 on a purpose of simplifying description. A node of multiple inputs and multiple outputs can be constructed in a manner shown in Fig. 3.

3. Terminating Firing-sequences of a Program Net without Switches

In this paper, we deal with a program net not containing switches. Since a switch changes data channeling from one node to the other depending on the binary value of the control token, firing behavior becomes unnecessarily complicated and hence the switch is better excluded from the net in the preliminary analysis.

Let $N=(G\ ,\ d^\circ)$ be a program net without switches, where $G=(V\ ,\ E)$ is the base graph and $d^\circ=(d_1^\circ, d_2^\circ, \ldots, d_m^\circ)$ is the initial token distribution, d_i° tokens on edge i. A <u>firing sequence</u> $F=(v_o, v_1, \ldots, V_{k-1})$ of N is a sequence of nodes such that

(i) node V_i is firable with respect to token distribution d^i, $i \geq 0$.

(ii) by firing of V_i, the token distribution changes from d^i to d^{i+1}.

F is called <u>terminating</u> if it is of a finite length $k < \infty$ and satisfies

(iii) any node x is dead (or not firable) with respect to the last token distribution d^k.

A terminating firing-sequence $\hat{F}$ is maximal in a sense that no firable node can be concatenated to it.

Let $\hat{f}_x$ be the number of times node x fires in $\hat{F}$, i.e. the number $\overset{\wedge}{\text{of}}$ times node x appearing in $\hat{F}$. In the final token distribution, we have

(i) for operator, decider or fork $\overset{z}{\smile}$, at least one input to z is token free.

(ii) for merge z, both inputs to z are token free

and hence a defining equation of $\hat{f}$,

$$\hat{f}_z = \begin{cases} \operatorname{Min}\left\{ \hat{f}_x + d^\circ_{xz},\ \hat{f}_y + d^\circ_{yz} \right\} & \text{if z is AND} \\ \\ (\hat{f}_x + \hat{f}_y) + (d^\circ_{xz} + d^\circ_{yz}) & \text{if z is OR} \end{cases}$$

where (x,z), (y,z) are two inputs to z.

4. Dead Nodes

A node is called <u>dead</u> in the net N if it can not be brought to be firable by any sequences of node firings. The purpose of this section is to characterize dead nodes in terms of a cycleberry.

A <u>cycleberry containing node v</u> CB_v is a connected subgraph $CB_v=(V_{CB}\ ,\ E_{CB})$ such that

(i) $v \in V_{CB}$

(ii) $(x,y) \in E_{CB}$ implies $x,y \in V_{CB}$

(iii) If $x \in V_{CB}$ is AND, then only one of its inputs is in E_{CB}. If $x \in V_{CB}$ is OR, then both of its inputs are in E_{CB}.

Some examples are shown in Fig. 4.

We list several lemmas with proof in the appendix.

[Lemma 1] If a cycleberry containing v is token free, then node v is dead in the net N.

[Lemma 2] If a cycleberry CB_v is token free in the present token distribution, then it remains so in all subsequent distributions caused by node firings.

[Lemma 3] The number of tokens on a cycleberry CB_v is non-decreasing with respect to node firings.

[Lemma 4] If program net $N=(G,d^\circ)$ has no token free cycleberry, then every node can fire at least once, i.e. not dead.

Through a sequence of above lemmas, we have actually proved the following basic property.

[Theorem 1] In a program net $N=(G,d^\circ)$, node v can be brought to be firable iff no cycleberry containing v is token free.

5. Maximal Firing Numbers

The <u>maximal firing number</u> $\hat{f}_x$ is a number of appearances of node x in a terminating firing-sequence $F=(v_o, v_1, \ldots, v_{k-1})$. In this section we present an algorithm for calculating $\hat{f}_x$'s.

《Calculation of $\hat{f}$》

1° Set $\hat{f}_x=0$ for all x in V.

2° Scan nodes in V in an arbitrarily fixed order and modify $\hat{f}_z$ as follows:

$$\hat{f}_z = \begin{cases} \min(\hat{f}_x+d^\circ_{xz},\ \hat{f}_y+d^\circ_{yz}) & \text{if } z \text{ is AND} \\ (\hat{f}_x+\hat{f}_y) + (d^\circ_{xz}+d^\circ_{yz}) & \text{if } z \text{ is OR} \end{cases}$$

3° Repeat 2° until no change occurs.

We observe 3 lemmas with proof in the appendix.

[lemma 5] If node v is immediately firable, then $\hat{f}_v \geqslant 1$.

[lemma 6] If there exists a token free cycleberry containing v, then $\hat{f}_v=0$ and conversely.

[lemma 7] Let v be immediately firable. Define f' by $\hat{f}'_z = \begin{cases} \hat{f}_z & \text{if } z \neq v \\ \hat{f}_v - 1 & \text{if } z = v \end{cases}$ and let d' be the token distribution resulted from d° by node firing of v. Then we have invariance.

$$\hat{f}'_z = \begin{cases} \min(\hat{f}'_x+d'_{xz},\ \hat{f}'_y+d'_{yz}) & \text{if } z \text{ is AND} \\ (\hat{f}'_x+\hat{f}'_y) + (d'_{xz}+d'_{yz}) & \text{if } z \text{ is OR} \end{cases}$$

[Theorem 2] Let the algorithm produce finite $\hat{f}$. Then there exist a terminating firing-sequence $\hat{F}$ whose firing number is $\hat{f}$.

[Proof]

Let $\hat{f} \neq 0$. We claim that there exists a firable node v with $\hat{f}_v > 0$. To show this let $\hat{f}_z > 0$. Assuming Z not firable, we trace token-free edges as follows:

(i) When Z is AND, there is at least one token-free input, say (x,z), with $\hat{f}_x \geqslant 1$.

(ii) When Z is OR, both inputs say (x,z) and (y,z), are token-free with $\hat{f}_x + \hat{f}_y \geqslant 1$.

Now ask if node x or y is firable. The argument is repeated until whether a firable node is reached or a token-free cycleberry is obtained. The second alternative should imply $\hat{f}_z = 0$ by Lemma 6, a contradiction.

Now we apply Lemma 7 repeatedly until $\hat{f}'$ becomes zero. Hence we have obtained a terminating firing-sequence $\hat{F}$ whose firing number is equal to $\hat{f}$.

QED.

[Example] Consider a program net of Fig.5.a, where AND-node is expressed by 0, OR-node by ▷ and token by •. We scan the nodes in an order of a, b, c, d. The result is tabulated as follows.

scanning	$\hat{f}_a$	$\hat{f}_b$	$\hat{f}_c$	$\hat{f}_d$	$\hat{f}_s = \hat{f}_t$
initial	0	0	0	0	0
1 st	1	1	1	1	0
2 nd	1	2	2	2	0
3 rd	1	3	2	2	0
4 th	1	3	2	2	0

The final token distribution is calculated by

$$\begin{cases} d_{xz} = d^\circ_{xz} + \hat{f}_x - \hat{f}_z & \text{if z is AND} \\ d_{xz} + d_{yz} = (d^\circ_{xz} + d^\circ_{yz}) + \hat{f}_x + \hat{f}_y - \hat{f}_z, & \text{if z is OR.} \end{cases}$$

and depicted in Fig.5 .b. We note that the source s and t are never fired. In general a node having no inputs or no outputs is assumed not firable. The above result can be checked by simulation of node firing.

6. Termination of Firing-sequences

In the previous section, we assumed the algorithm to terminate and produce finite $\hat{f}$. However it is easy to demonstrate that firing numbers grow unbounded on a directed circuit that has one OR-node to which there is an access from the source s and an AND-node from which the sink t is accessed, see Fig.6 .a. Since an unbounded firing number makes little sense in the program net, such circuit is suppressed by a directed path from the source consisted of AND-nodes only. See Fig. 6 .b.

Let $Q=(s, \ldots, x, v)$ be a directed path from s to v. By successive substitutions of defining equations of step 2° along Q, we have a chain of inequalities

$$\begin{aligned} f_v &\leqslant \sum_{(x,y)\in Q} d^\circ_{xy} + \hat{f}_s + \sum_{w_i=OR} [d^\circ_{z_i w_i} + \hat{f}_{z_i}] \\ &\leqslant \sum_{(x,y)\in Q} d^\circ_{xy} + \sum_{w_i=OR} \Big\{ d^\circ_{z_i w_i} + \sum_{(x,y)\in Q_i} [d^\circ_{xy} + \sum_{w_{ij}=OR} (d^\circ_{z_{ij} w_{ij}} + \hat{f}_{z_{ij}})]\Big\} \end{aligned}$$

where w_i is a OR-node on Q, z_i is an input node of w_i not on Q, Q_i is a directed path from s to z_i, and w_{ij} is a OR-node on Q_i and so forth, see Fig. 7 for topology. The above equation is refered to as an <u>expansion</u> of $\hat{f}_v$ along Q; $Q_i \ldots$; $Q_{ij} \ldots; \ldots$.

We begin with simple observations.

[Lemma 8] (i) If the net is acyclic, then $\hat{f}_v$ is not larger than a sum of token numbers of $s \rightarrow v$ directed paths and hence is finite.

(ii) If the net does not contain OR-nodes, then $\hat{f}_v$ is equal to the shortest distance from the dead nodes (token number is considered as the edge length) and hence finite.

[Proof]

(i) If net N is acyclic, no OR-node of Z_i, $Z_{ij} \ldots$ appears more than once in the expansion. This means the right hand side of the expansion is bounded above by a sum of tokens on $s \rightarrow v$ directed paths.

(ii) $\hat{f}$ is the maximal solution of a system of equations

$$\begin{cases} \hat{f}_y \leqslant \hat{f}_x + d^\circ_{xy} & \text{for all } (x,y) \in E \\ \hat{f}_z = 0 & \text{for all dead nodes } Z. \end{cases}$$

$\hat{f}$ is well known to be the shortest distance function.

QED.

To characterize the termination of firing-sequences, we define the OR-graph of the net $N=(G,d^\circ)$, $G_{OR}=(V_{OR},E_{OR})$ as follows:

(i) It is an ordinary graph consisting of s, t and 3 nodes α_x, β_x, γ_x for each OR-node x.

(ii) To each OR-node x of N, there corresponds a V-shaped digraph of nodes α_x, β_x, γ_x as shown in Fig. 8, where α_x, β_x are virtual inputs nodes and γ_x is the virtual output node of OR-node x.

(iii) If there is a directed path of AND-nodes in N from the output of OR-node x to α-input of OR-node $y \neq x$, draw edge (γ_x, α_y) in G_{OR}. However we do not allow self-loop (γ_x, α_x) or (γ_x, β_x) in G_{OR}.

Fig. 9 shows how to construct an OR-graph.

[Lemma 9] (i) Intermediate $\hat{f}_x$ of step 2° is non-decreasing with respect to increase in iteration of step 2°.

(ii) The algorithm either terminates and produces a unique finite $\hat{f}$ or does not terminate and $\hat{f}$ grows unbounded.

[proof] (i) Let $\hat{f}^i$ be intermediate $\hat{f}$ of i-th iteration. Since $\hat{f}^\circ=0$ and

$$\hat{f}^i_z = \begin{cases} \min(\hat{f}^{i-1}_x + d^\circ_{xz}, \hat{f}^{i-1}_y + d^\circ_{yz}) & \text{if } z \text{ is AND} \\ (\hat{f}^{i-1}_x + \hat{f}^{i-1}_y) + (d^\circ_{xz} + d^\circ_{yz}) & \text{if } z \text{ is OR,} \end{cases}$$

we can say $\hat{f}^1 \geqslant \hat{f}^0$ which implies further $\hat{f}^2 \geqslant \hat{f}^1$ which implies further $\hat{f}^3 \geqslant \hat{f}^2$ and so on.

(ii) It is well-known that a non-decreasing sequence of real numbers either converges to a unique value or diverges to infinity. QED

[Theorem 3] If the OR-graph of the program net N contains neither directed circuits nor source nodes excluding s, then the algorithm terminates and produces finite $\hat{f}$.

[Proof]

First we observe that unbounded firing number ever occurs at OR-nodes if any. This is because the firing number of a program net without OR-nodes is finite. Let v be an OR-node and expand $\hat{f}_v$ along some $s \rightarrow v$ path. By the nature of the OR-graph construction, it is easy to understand that H is a graphical expression how OR-nodes appear in an expansion of $\hat{f}_v$. If α_v is a source in the OR-graph H, it is meant that any expansion of $\hat{f}_v$ involves infinite appearances of $\hat{f}_v$ itself because we suppressed self-loop (γ_v, α_v) in the construction. Therefore under the stated condition of the theorem, no OR-node appears more than once in the expansion and hence a finite upper bound is produced on the right hand side.

QED.

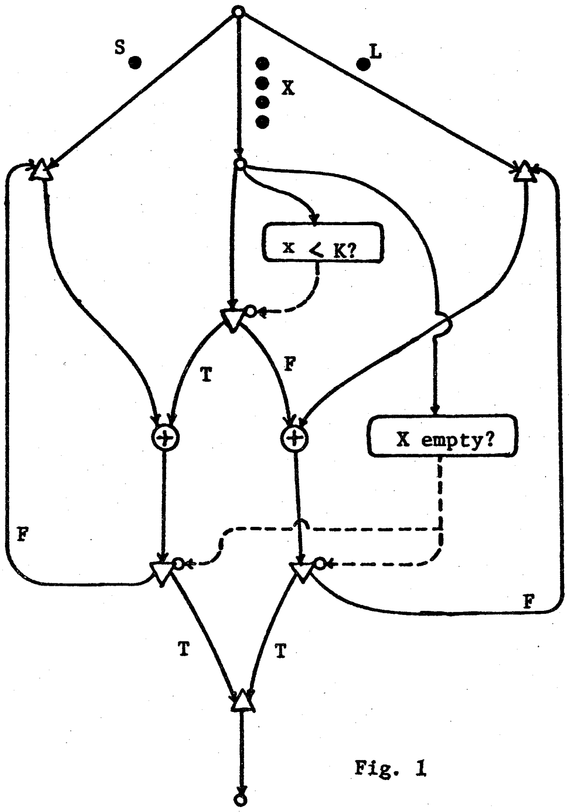
7. Conclusions

We have discussed on terminating firing-sequences $\hat{F}=(v_o, v_1, v_2, \ldots, v_{k-1})$ of a program net $N=(G,d^{\circ})$ and presented some basic properties and characterizations of dead nodes, maximal firing numbers, and termination. However these results should be considered as preliminary, for we have excluded switches from the net in order to simply description. In data flow analysis points of view, main function of a switch is controling of loops to open or close. Interesting topics to be pursued further are decomposition and functionality of nets containing loops.

Acknowledgement: This work was supported in part by the Grant in Aid for Scientific Research of the Ministry of Education, Science and Culture of Japan under Grant: Cooperative Research (A) 435013 (1979,1980), General Research (C) 555144 (1980).

References

1. J.E. Rodriguez:"A graph model for parallel conputation" Report MAC-TR-64, sept. 1969.
2. J.B. Dennis: "First version of data flow procedure language" Lecture Notes in Computer Science 19G., Springer-Verlag, New York, 1974.
3. C.A. Petri: "Kommunikation mit Automaten," Schriften des Rheinisch-Westfalischen Institutes fur Instrumentelle Mathematik an der Universitat Bonn, Heft 2, Bonn, 1962.
4. S.E. Elmaphraby: Activity Networks: Planning and Control by Network Models, John-Wiley, N.Y.1973
5. A.A.B. Pritsker and W.W. Happ: "GERT: graphical evaluation and review technique, part I, fundamentals" J. Ind. Eng. vol. 17 (5), pp 267-274 1966.
6. R.E. Karp and R.E. Miller: "Properties of a model for parallel computations: determinacy, termination, queueing" SIAM J. Appl. Math., vol. 14, No. 6, p 1390 (1966)
7. F. Commoner, A.W. Holt, S. Even, and A. Pnueli: "Marked directed graphs," J. of Computer and System Sciences. vol.5, No. 5, p.511 (1971)



```
Program

While X is non empty do
   begin x in X; X←X-{x};
      if x < K then S←S+x
      else L←L+x
   end
 Write S,L
```

A program and its net representation. The outputs of the switch are labeled with T and F.

⟶ data flow

- - - -→o control flow

Fig. 1

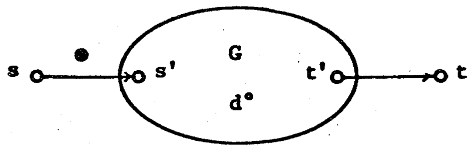


Fig. 2 Topology of G

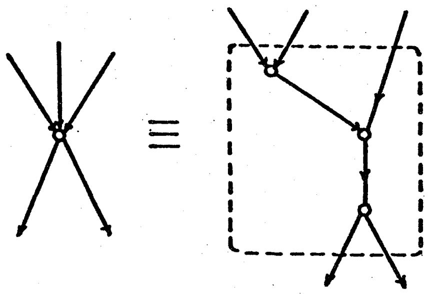


Fig. 3 Construction of Multiple Inputs & Outputs

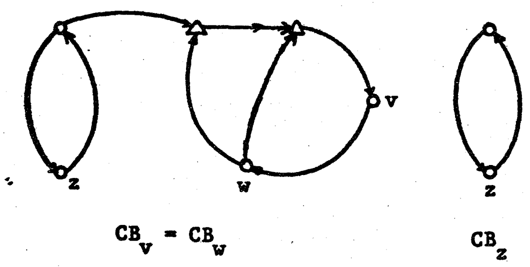


Fig. 4 Cycleberry (o : AND, ◁ : OR)

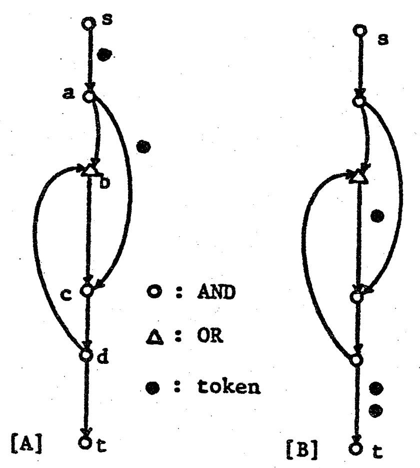
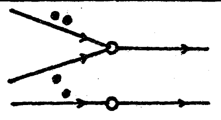
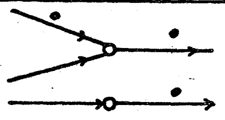
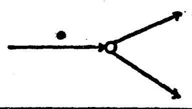
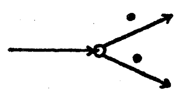
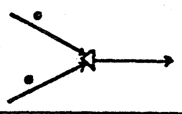
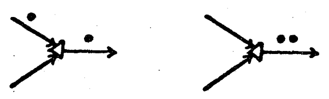
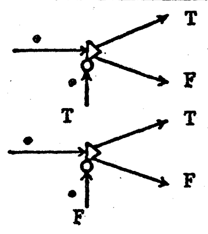
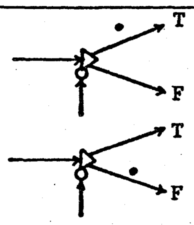


Fig. 5 An Example of Token Flows. Initial Token Pattern [A] and Final Pattern [B]

Table 1 Firing Rule

Node Type	Before Fire	After Fire
Operator Decider		
Fork		
Merge		
Switch		

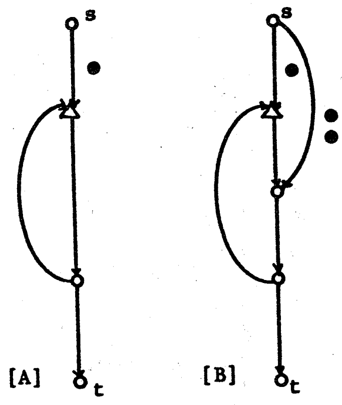


Fig. 6 Unbounded Firing Numbers and its Suppression

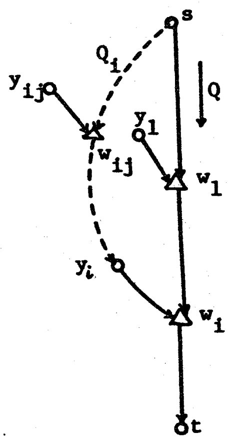


Fig. 7 Successive Substitution

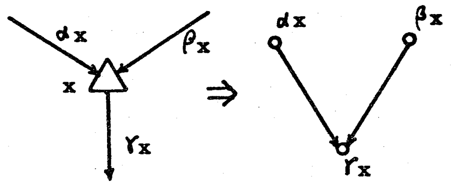


Fig. 8 V-shape Representation of the OR-node in the OR-graph

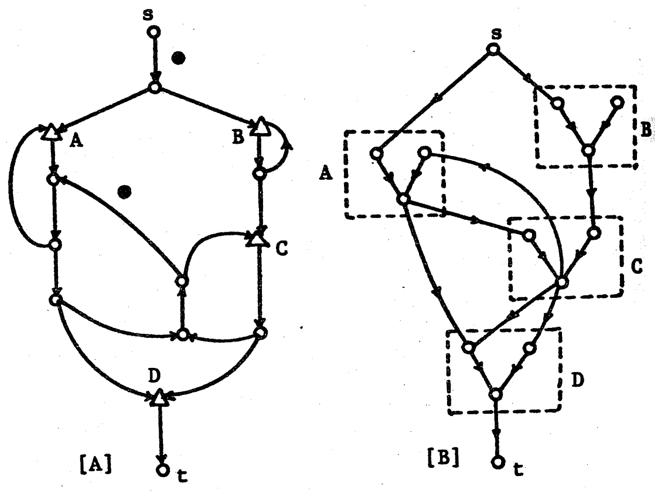


Fig. 9 Program Net and its OR-graph

Appendix

[Proof of Lemma 1]

We construct a firing precedence tree T_v such that v is the root and every ancestor nodes of x in T_v must fire in order for x to become firable.

1° v is the root and $\alpha \leftarrow v$.

2° Set α old.

3° Consider two cases:

(i) α is AND.

Only if there is at least one token-free input edge to α, choose one and draw a tree edge, say (x,α). If x is not old, then do $\alpha \leftarrow x$ and go to 2°.

(ii) α is OR.

Only if both input edges to are token-free, draw two tree edges, say (x,α) and (y,α). If x (or y) is not old, then do $\alpha \leftarrow x$ (or y) and go to 2°.

Note that a leaf of the tree T_v is either old or not old. If a leaf is not old, it is immediately firable by the rule of the construction. Observe that if every leaf x is old and if each old leaf x is coaleased onto internal node x, we get a token-free cycleberry containing node v. Moreover if every leaf is old, node v is led into a deadlock where there exists in T_v is never brought to be firable. Therefore when there is a token-free cycleberry containing v, there exists a firing precedence tree rooted on v, all of whose leaves are old, and hence v can never be brought to be firable. QED.

[Proof of Lemma 2]

By Lemma 1, no node in CB_v can be brought to be firable and hence no edge in CB_v obtains any token. QED.

[Proof of Lemma 3]

If no node of CB_v fires, the token count of CB_v remains the same. Let node x of CB_v be fired and consider two cases.

(i) x is AND, and has one input and two outputs in CB_v: firing of x increases the token count of CB_v.

(ii) x is AND, and has one input and one output in CB_v or x is OR: firing of x remains the token count of CB_v

[Proof of Lemma 4]

Let v be non-firable and suppose no cycleberry containing v be token-free. We show v can be brought to be firable after appropriate node firings. Let H^0 be a connected subgraph token-free with respect to the initial token distribution d^0 such that starting from v token free input edges are traced backward from non-firable nodes and added to H^0 until old nodes, or AND-nodes with no token-free input or OR-nodes with at least one positive token input are met.

Under the stated hypothesis H^0 has at least one source node a_0 (node with no input) that is immediately firable in d^0 . Otherwise H^0 contains a token-free cycleberry. Let this node a_0 be fired and denote the new token distribution by d^1. Let H^1 be the above mentioned subgraph with respect to d^1. H^1 neither contains a_0 nor other nodes not in H^0. Hence H^1 is a proper subgraph of H^0. Since by Lemma 2 token-freeness of the cycleberry is invariant with respect to node firing. H^1 inherits the same property of H^0, that is, there is at least one source node that is firable in d^1. The process is iterated until H is reduced to a single node v. This means v is firable.

QED.

[Proof of Lemma 5]

Obvious by the calculation of Step 2 . QED.

[Proof of Lemma 6]

Let $f_x^{(i)}$ be the firing number of node x after i-th scan. Let i=0, $f_z^{(i)}=0$ for any node z in CB_v and consider the calculation of step 2 .

(i) If Z is AND in CB_v, either $f_x^{(i)}+d_{xz}^0$ or $f_y^{(i)}+d_{yz}^0$ is zero and hence $f_z^{(i+1)}$ remains zero.

(ii) If Z is OR in CB_v, both $f_x^{(i)}+d_{xz}^0$, $f_y^{(i)}+d_{yz}^0$ are zero and hence $f_z^{(i+1)}$ remains zero.

Repetition of the above for i never increases f_z in CB_v beyond zero.

Conversely suppose $f_v=0$ be produced by the algorithm. There are two case to consider.

(i) If v is AND, then at least one input to v, say (x,v), is token free and $f_x=0$ as well.

(ii) If v is OR, then both two inputs, say (x,v) and (y,v) are token free, and $f_x=f_y=0$ as well.

Repeated application of the above tracing of token-free edges results in a cycleberry containing v.

QED.

[Proof of Lemma 7]

Observe that

$$\hat{f}'_z = \begin{cases} \hat{f}_z & \text{if } z \neq v \\ \hat{f}_v - 1 & \text{if } z = v \end{cases}$$

$$d'_{xz} = \begin{cases} d^\circ_{xz} & \text{if } x \neq v,\ z \neq v \\ d^\circ_{xz} - 1 & \text{if } z = v \text{ and } v \text{ is AND} \\ d^\circ_{xz} + 1 & \text{if } x = v \end{cases}$$

$$d'_{xz} + d'_{yz} = d^\circ_{xz} + d^\circ_{yz} - 1 \quad \text{if } z=v \text{ and } v \text{ is OR.}$$

We consider two cases.

(i) Z is AND.

$$\hat{f}'_x + \hat{d}'_{xz} = \begin{cases} \hat{f}_x + d^\circ_{xz} & \text{if } z \neq v,\ x \neq v \\ (\hat{f}_x - 1) + (d^\circ_{xz} + 1) = \hat{f}_x + d^\circ_{xz} & \text{if } x = v \\ \hat{f}_x + (d^\circ_{xz} - 1) = \hat{f}_x + d^\circ_{xz} - 1 & \text{if } z = v \end{cases}$$

Hence we have

$$\text{Min}\left\{\hat{f}'_x + d'_{xz},\ \hat{f}'_y + d'_{yz}\right\} = \begin{cases} \min\left\{\hat{f}_x + d^\circ_{xz},\ \hat{f}_y + d^\circ_{yz}\right\} = \hat{f}_z & \text{if } z \neq v \\ \min\left\{\hat{f}_x + d^\circ_{xz} - 1,\ \hat{f}_y + d^\circ_{yz} - 1\right\} = \hat{f}_z - 1 & \text{if } z = v \end{cases}$$

as was to be prooved.

(ii) Z is OR.

$$(\hat{f}'_x + \hat{f}'_y) + (D'_{xz} + d'_{yz}) = \begin{cases} (\hat{f}_x + \hat{f}_y) + (d^\circ_{xz} + d^\circ_{yz}) = \hat{f}_z & \text{if } z \neq v,\ x \neq v,\ y \neq v \\ (\hat{f}_x - 1 + \hat{f}_y) + (d^\circ_{xz} + 1 + d^\circ_{yz}) = \hat{f}_z & \text{if } x \text{ or } y = v \\ (\hat{f}_x + \hat{f}_y) + (d^\circ_{xz} + d^\circ_{yz}) - 1 = \hat{f}_z - 1 & \text{if } z = v \end{cases}$$

as was to be proved. QED.