Fairness and Conspiracy Concepts in Concurrent Systems

Anup Kumar Bandyopadhyay
Department of Electronics and Telecommunication Engineering
Jadavpur University , Kolkata – 700032, India
Email . akbandyopadhyay@etce.jdvu.ac.in

Abstract

Many different fairness notions are available in literature. One should choose the proper definition that match with the system under consideration. In this paper we consider two known fairness definitions, viz., weak fairness and strong fairness. It is argued that these concepts are suitable for determining the degree of fairness of a given system. For only starvation freedom we require a minimum degree of fairness which we call ***least fairness.*** This idea is illustrated using two practical examples. Conspiracy is another very important issue in concurrent system. We have defined conspiracy in connection with all the fairness notions. Conspiracy resistant implementation is illustrated using a starvation free solution to dining philosophers problem. Dijkstra's weakest precondition calculus is used as the analytical tool.

Keywords: Fairness, Starvation, Conspiracy, Concurrent Systems.

Introduction

Many different fairness notions are available in literature [1,2]. Among them weak fairness and strong fairness notions are most recognized [3]. An even stronger class of fairness concept viz. hyper fairness has been proposed [4,5]. All these concepts are useful for determining the degree of fairness of a given system. For proving starvation freedom we need some other view viz. minimum fairness requirements. In this paper we propose such fairness notions called ***least fairness***.

Consider a deadlock free loop connected message passing architecture as shown in Fig 1 [6].

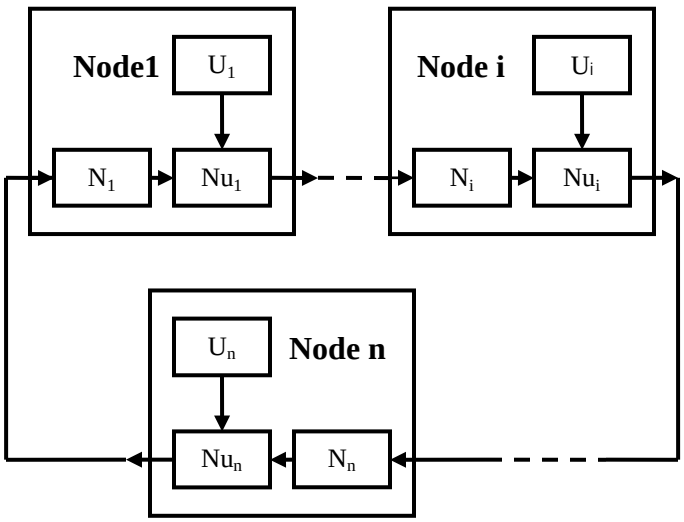

Figure 1. Loop connected message passing architecture

An arbitrary node i contains three processes viz., U_i (user), Ni (receiver) and Nu_i (sender). Among these processes N_i and Nu_i are running on a single processor. The user process U_i on the other hand may run on a different processor it may even be separated geographically from the other two processes. The user process runs application program and when necessary would send a message to its corresponding sender process Nu_i. Nu_i should check if there is any message available at N_i for reception. If there is none then and then only Nu_i should receive the message from U_i. Otherwise it should receive the message from N_i. Upon receiving a message from $Nu_{(i-1) \bmod n}$ the receiver process N_i should check its destination address. If the message is destined to itself then it will consume the same else it should send the message to Nu_i.

We define the process U_i to be **ready** when it intends to send a message to Nu_i. . A system is weakly fair referred to U_i when it is eventually always **ready** and the process succeeds infinitely many times in sending the messages. However, there will also be infinitely many instances when U_i will not be able to send messages in spite of its ready state. On the other hand for the same success rate if U_i requires being infinitely many times **ready**, leaving infinitely many instances when it does not have a message to be sent then the system is strongly fair with reference to the process U_i.

Let us also consider the famous dining philosophers problem [7,8]. The dining table configuration related to this problem is shown in Fig 2.

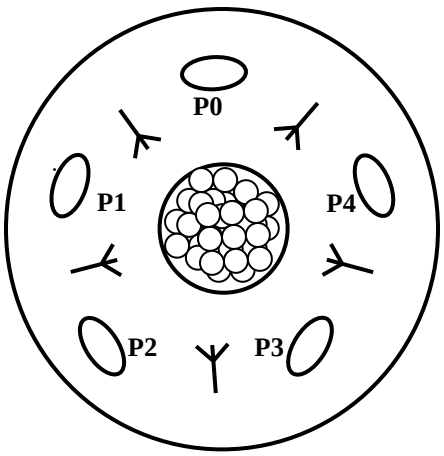

Figure 2. The dining table configuration illustrating dining philosophers problem

http://doi.acm.org/10.1145/1507195.1507203 DOI: 10.1145/1507195.1507203

Life cycle of each philosopher is defined by

```
begin
    think;
    feel hungry;
    acquire necessary forks;
    eat;
end;
```

and is elaborated in Fig 3 where the philosopher first acquire his left fork and then gets the right one. Interactions with the neighbors while acquiring the forks are also depicted in this representation. If all the philosophers follow this scheme then there is a possibility of deadlock, however, if we assume that there exists one philosopher who acquires his right fork first then this possibility may be avoided [8].

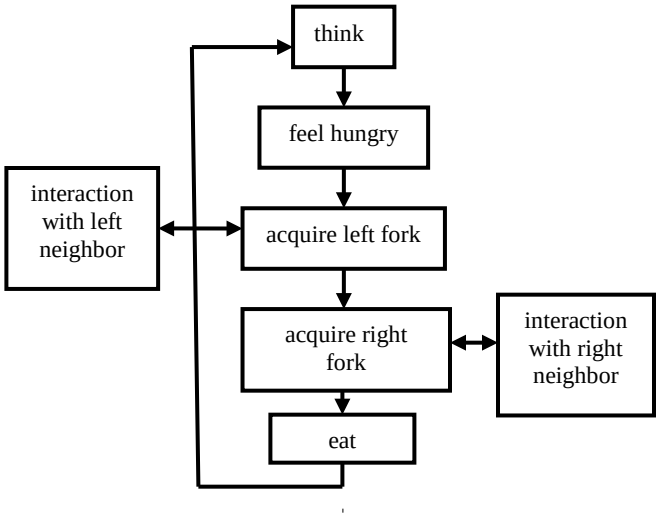

Figure 3. Process interactions in dining philosophers problem

The concerned system is fair referred to a philosopher, if it allows him to eat when he becomes hungry. The previous concepts of fairness may also be applied here to study how easily he will be allowed to do so. However in both the above examples there are possibilities when some process would starve. In a message passing network, for example, a group of processes may act in such a manner that some user process may not be allowed at all to transmit any message to its corresponding sender process Nu_i. In the second problem a hungry philosopher may starve to death due to the conspiracy of his fellow neighbor. In the following section we discuss conspiracy in connection with the above two examples.

Though both the above systems are deadlock free, they may not be free from starvation. In a deadlock free loop connected message passing network, as described above, the maximum number of messages that are allowed in the system is $2n-1$. Let there exists a sub set of nodes SN such that they could manage to push all the $2n-1$ messages in the system. Let us also assume that the destinations for all these messages are the nodes that belong to SN. We also assume that these nodes are always ready. When $2n-1$ messages are present, no node will be able to push any more messages into the system. When ever a message destined for node $k, k \in SN$ reaches its destination the process N_k will consume the same and would allow Nu_k to receive a new message from U_k. If this new message is again destined to a node belonging to SN then nodes not belonging to this set will not be able to push any message into the system, in other words those nodes will starve due to the **conspiracy** [9,10] of the nodes belonging to the set SN. If the cardinality of the set SN is n1 then we have $1 \le n1 \le (n-1)$.

Similarly conspiracy may occur in different solutions to the dining philosophers problem where a philosopher starves for ever when both his right and left neighbors continuously become critical (eat)[10].

Any concurrent system may be described by using Dijkstra's weakest precondition calculus [11,6]. In such description a transition rule P.r having post condition Q is specified by its weakest self precondition wsp(P.r,Q) and weakest cooperation requirement wcr(P.r,Q) . Weakest self precondition depends on the states of the process to which the transition rule under consideration belongs and therefore it depicts the sequential characteristics of the process under consideration. The weakest cooperation requirement for a transition rule on the other hand is a function of the states of other cooperating processes. We would use this technique as a tool to develop our concepts.

Fairness Definitions

In this section we would give formal definitions for all the fairness concepts discussed above. From the definitions we would derive other equivalent expressions using lemmas in temporal logic [12] proved in [13]. For completeness we have included all these results along with some new one in appendix.

Definition 1.

Weak fairness on r, asserts that if wsp(r,Q) eventually becomes true forever, then infinitely many execution of r must occur.

Mathematically

$$\Diamond\Box wsp(r,Q) \Rightarrow \Box\Diamond Q \qquad (1)$$

From (1) we get the following alternative expressions and explanations.

a) $\neg\Box\Diamond Q \Rightarrow \neg\Diamond\Box wsp(r,Q)$

Application of Lemma 11 and Lemma 12 on (a) we get

b) $\Diamond\varnothing Q \Rightarrow \varnothing\Box\ wsp(r,Q)$

That is if eventually Q never becomes true it implies that never wsp(r,Q) becomes true for ever.

Application of the definition for $\varnothing$ and lemma 11 and 15 on (b) we get

c) $\Diamond\Box\neg Q \Rightarrow \Box\Diamond\neg wsp(r,Q)$

That is if eventually Q remains false forever then it implies that wsp(r,Q) remains infinitely often false –even though it may also become infinitely often true.

Definition 2.

Strong fairness on r, asserts that if wsp(r,Q) infinitely often becomes true –even though it may also become infinitely often false–then infinitely many execution of r must occur.

http://doi.acm.org/10.1145/1507195.1507203 DOI: 10.1145/1507195.1507203

Mathematically

$\Box\Diamond wsp(r,Q) \Rightarrow \Box\Diamond Q$ (2)

From (2) we get the following alternative expressions and explanations.

a) $\neg\Box\Diamond Q \Rightarrow \neg\Box\Diamond wsp(r,Q)$

Application of Lemma 12 on (a) we get

b) $\Diamond\varnothing Q \Rightarrow \Diamond\varnothing wsp(r,Q)$

That is if eventually Q never becomes true it implies that eventually wsp(r,Q) never becomes true.

Application of definition for $\varnothing$ on (b) we get

c) $\Diamond\Box\neg Q \Rightarrow \Diamond\Box\neg wsp(r,Q)$

That is if eventually Q remains false forever then it implies that wsp(r,Q) eventually remains false forever

Definition 3.

Least fairness on r asserts that if wsp(r,Q) becomes true at an interval i then the truth of Q will eventually be established at some interval j≥i.

Mathematically

$i.\ wsp(r,Q) \Rightarrow \exists j: i \le j: j.Q = \Diamond^i Q$ (3)

From (3) we get the following alternative expressions

a) $\neg\Diamond^i Q \Rightarrow \neg\ i.\ wsp(r,Q)$

Considering the above relation is true for all i we get

$\forall i : \Box^i \neg Q \Rightarrow \forall i\ \neg i.\ wsp(r,Q)$

Applying rule (ii) on $\Box$ we get

b) $\Box\neg Q \Rightarrow \Box\neg wsp(r,Q)$

That is if Q is never true then it implies that wsp(r,Q) is also never true.

Application of the definition for $\varnothing$ on (b) we get

c) $\varnothing Q \Rightarrow \varnothing wsp(r,Q)$

That is if Q remains false forever then it implies that wsp(r,Q) remains false forever.

Three different fairness concepts have been described above. Among them the first two are associated with long term behavior of a system where the last one includes only distinct instances. Weak fairness, for example, asserts that if wsp(r,Q) eventually becomes true forever, then infinitely many executions of r must occur. This, however, includes the possibility of some happenings where wsp(r,Q) become true but the rule is not executed. Similar conditions are also possible for strong fairness scenario. Least fairness on the other hand asserts execution of the rule r for every distinct instances of the weakest self precondition wsp(r,Q) becoming true however it does not put any constraint on the response time. We can prove that strong fairness does imply weak fairness; however, no such conclusion may be drawn for least fairness.

Unfairness Definitions

Unfairness referred to ***weak fairness*** on r, asserts

$\neg(\Diamond\Box wsp(r,Q) \Rightarrow \Box\Diamond Q)$ (4)

Relation (4) may be expressed as

$\neg\ (\neg\Diamond\Box wsp(r,Q) \vee\ \Box\Diamond Q)$

$= \Diamond\Box wsp(r,Q) \wedge \neg\ \Box\Diamond Q$ (5)

That is, if wsp(r,Q) eventually becomes true forever even then infinitely many executions of r are not obtained then the system is unfair referred to ***weak fairness*** on r.

From (5) we may also get the following alternative expression by the application of lemma 21 and 12.

$\neg\varnothing\Box wsp(r,Q) \wedge \Diamond\varnothing Q$ (6)

Unfairness referred to ***strong fairness*** on r, asserts

$\neg(\Box\Diamond wsp(r,Q) \Rightarrow \Box\Diamond Q)$ (7)

Relation (7) may be expressed as

$\neg(\neg\Box\Diamond wsp(r,Q \vee \Box\Diamond Q)$

$= \Box\Diamond wsp(r,Q) \wedge \neg\ \Box\Diamond Q$ (8)

That is, if wsp(r,Q) infinitely often becomes true –even though it may also become infinitely often false even then infinitely many executions of r are not obtained then the system is unfair referred to ***strong fairness*** on r.

From (8) we may also get the following alternative expression by the application of lemma 20 and 12.

$\varnothing\Box\neg wsp(r,Q) \wedge \Diamond\varnothing Q$ (9)

Unfairness referred to ***least fairness*** on r, asserts

$\neg\ (\neg\varnothing Q \vee\varnothing\ wsp(r,Q))$ (10)

Relation (10) may be expressed as

$\varnothing Q \wedge \neg\varnothing wsp(r,Q)$ (11)

Application of lemma 7 on (11) we get

$\varnothing Q \wedge \Diamond wsp(r,Q)$ (12)

That is, if wsp(r,Q) becomes eventually true even then execution of r is never obtained then the system is unfair referred to ***least fairness*** on r.

Non execution of a rule throughout the life time of a system is defined as ***starvation***. This situation would arise due to complete non co operation of a set of rules in the system. Though the word ***starvation*** in the context of fairness asserts complete non-execution of the transition rule P.r under consideration, we would extend this definition to mean unfairness on P.r referred to a given fairness concept. This non cooperation from the fellow processes is considered as conspiracy. Before we give a formal description for conspiracy we introduce a new graph representation to study both intra-process and inter-process interactions.

http://doi.acm.org/10.1145/1507195.1507203 DOI: 10.1145/1507195.1507203

Has_Contribution Graph

Let a process P_i be defined by a set of transition rules $P_i.r_j, j = 1,2,...,n_i$, and the condition $wsp(P_i.r_k, Q_k)$ be obtained from the outputs of the rules $P_i.r_{\ell_q}, q = 1,2,...,n_k$. Similarly let $wcr(P_i.r_k, Q_k)$ be obtained from the outputs of the rules $P_{\ell_q}.r_{j_m}, q = 1,2,...,n_{ik}, m = 1,2,...,n_{qk}$.

We define the ***has_contribution*** graph HC_i for the process P_i by a set of nodes N and a set of directed edges E. The nodes in N represent all the transition rules $P_i.r_j, j = 1,2,...,n_i$ and $\forall k: P_{\ell_q}.r_{j_m}, q = 1,2,...,n_{ik}$, $m = 1,2,...,n_{qk}$ An edge in E on the other hand may be of two types.

Type I edges are the one that represent the contribution to the weakest self preconditions from different transition rules belonging to the process P_i. Symbolically these edges are described by the ordered pairs $\forall k: (P_i.r_{\ell_q}, P_i.r_k), q = 1,2,...,n_k$. We also define that the edges $(P_i.r_{\ell_q}, P_i.r_k), q = 1,2,...,n_k$ are incident to the node $P_i.r_k$.

Type II edges on the other hand are described by the pairs $\forall k: (P_{\ell_q}.r_{j_m}, P_i.r_k), q = 1,2,...,n_{ik}, m = 1,2,...,n_{qk}$. The edges are marked by $\leftrightarrow$ to show that the constituent rules have effects on each other. Type II edges $(P_{\ell_q}.r_{j_m}, P_i.r_k), q = 1,2,...,n_{ik}$, $m = 1,2,...,n_{qk}$ are incident to the nodes $P_i.r_k$ and $P_{\ell_q}.r_{j_m}, q = 1,2,...,n_{ik}$, $m = 1,2,...,n_{qk}$

The set of rules $R_{ik} = \{ P_{\ell_q}.r_{j_m}, q = 1,2,...,n_{ik}, m = 1,2,...,n_{qk} \}$ is defined as the ***set of cooperating rules*** for $P_i.r_k$.

has_contribution graph for the total system **S** is defined as $\forall i: HC_i$.

Type I degree of a node is defined by the number of Type I edges incident to the node. Similarly

Type II degree of a node is the number of Type II edges incident to the node.

Type II complexity or simply ***complexity*** of a system **S** is the maximum Type II degree of a node contained in the has_contribution graph for the total system **S**

Illustrative Example

To illustrate the above concept we consider formal representation of the deadlock free solution to the dining philosophers problem using "Event Ordering" as described in [8]. The solution requires that, upon getting hungry, four of the five philosophers (Type Left) should peak up their left fork first where the fifth philosopher (Type Right) should peak up his right fork first. States of the i^{th} philosopher is depicted in Table 1. The state transition rules for the Type Left philosophers are described briefly in Table 2 and the corresponding has_contribution graph is shown in Fig 4. Similar representation of type right philosopher may be obtained by swapping rule 2 and rule 3. From the has_contribution graph shown in Fig 4 it may also be noted that the complexity of the system is 1.

State	Informal Semantics
philosopher(i).thinking	philosopher(i) is thinking
philosopher(i).hungry	philosopher(i) is hungry
philosopher(i).lifted (i-1)	philosopher(i) has lifted the $(i-1)^{th}$ fork
philosopher(i).lifted (i)	philosopher(i) has lifted the i^{th} fork
philosopher(i).eat	philosopher(i) finished eating
philosopher(i).putdown	philosopher(i) has putdown both the forks

Table – 1. States for the i^{th} Philosopher

Rule	State Transition Profile
$P_i.r_1$	philosopher(i).thinking → philosopher(i).hungry
$P_i.r_2$	philosopher(i).hungry→ philosopher(i).lifted (i-1)
$P_i.r_3$	philosopher(i).lifted (i-1)→ philosopher(i).lifted (i)
$P_i.r_4$	philosopher(i).lifted (i)→ philosopher(i).eat
$P_i.r_5$	philosopher(i).eat→ philosopher(i).putdown
$P_i.r_6$	philosopher(i).putdown→ philosopher(i).thinking

Table – 2. State Transition Rules for Type Left Philosopher

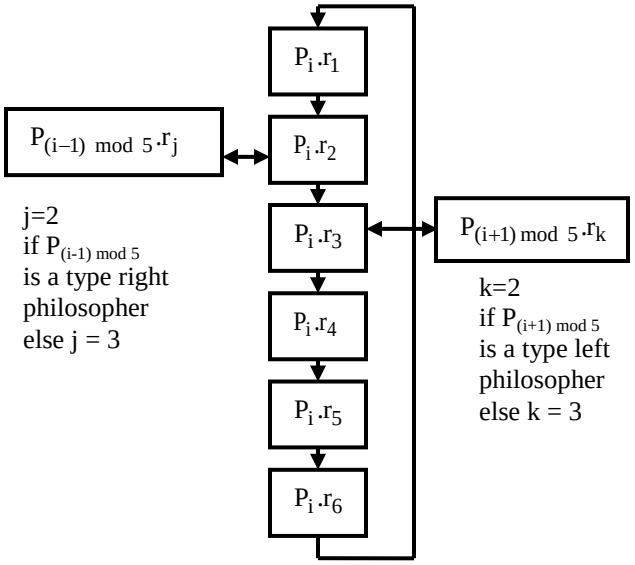

Fig 4 . Has_Contribution Graph for the process P_i

In the has_contribution graph shown in Fig 4 there are two type II edges viz., $(P_{(i-1) \bmod 5}.r_j, P_i.r_2)$ and $(P_{(i+1) \bmod 5}.r_k, P_i.r_3)$. Each of these two edges represents an abstract description of the fork acquiring procedure. Transition rule $P_i.r_2$, for example, would check if the fork left to him has not yet been lifted by the $(i-1)^{th}$

philosopher, if so then and only then the rule $P_i.r_2$ will be executed. The transition rule $P_{(i-1) \bmod 5}.r_j$ would also perform similar action for the (i-1)th philosopher. Similar explanation may be given for the other type II edge viz, ($P_{(i+1) \bmod 5}.r_k$, $P_i.r_3$).

Since no protocol is defined yet to resolve the conflict between the rules $P_i.r_2$ and $P_{(i-1) \bmod 5}.r_j$ as well as $P_{(i+1) \bmod 5}.r_k$ and $P_i.r_3$, any one or both the philosophers $P_{(i-1) \bmod 5}$ and $P_{(i+1) \bmod 5}$ may repeatedly acquire the corresponding forks resulting the ith philosopher to starve. That is, the rules $P_i.r_2$ and/or $P_i.r_3$ will be blocked by the above actions. This is known as conspiracy. We give a formal definition as follows.

Conspiracy Definitions

Consider a transition rule $P_i.r_j$. Let there exists a system state, reachable from the initial state, in which both wsp($P_i.r_j$,Q) and wcr($P_i.r_j$,Q) are true that is, there exists a sequence of state transitions starting from the initial state such that $P_i.r_j$ may be executed. In spite of this there might exist one non-terminating state transition sequence Σ such that the following predicate remains invariant.

$$\varnothing Q \wedge \Diamond \text{wsp}(P_i.r_j, Q)$$

$$\wedge\ \exists \ell, k : \text{wsp}(P_\ell.r_k, Q_{\ell k}) \wedge \text{wcr}(P_\ell.r_k.Q_{\ell k}) \qquad (13)$$

The first conjunct describes complete starvation referred to the transition rule $P_i.r_j$ where the second one describes condition for deadlock freedom. Let the non-terminating sequence Σ be participated by a subset Π of processes belonging to the system under consideration. Some of the processes belonging to Π are repeatedly executed by making the weakest cooperation requirement wcr($P_i.r_j$,Q) false at each cycle restricting execution of the transition rule $P_i.r_j$. Such behavior of the participating processes is defined as ***conspiracy*** and the processes themselves are defined as ***villains.*** Let the set of villains referred to the transition rule $P_i.r_j$ be denoted by $V_{i,j}$ clearly $V_{i,j} \subseteq \Pi$ also $\bigcup_{\forall i,j} V_{i,j}$ gives the set of all interactive processes in the system **S.**

From (13) we may also note that the first conjunct describes unfairness referred to ***least fairness*** on $P_i.r_j$. Considering unfairness conditions referred to ***strong fairness*** and ***weak fairness*** we may define two more predicates as follows.

$$(\Box\Diamond \text{wsp}(r,Q) \wedge \neg\, \Box\Diamond Q) \wedge \exists \ell, k : \text{wsp}(P_\ell.r_k, Q_{\ell k}) \wedge \text{wcr}(P_\ell.r_k.Q_{\ell k}) \qquad (14)$$

$$(\Diamond\Box \text{wsp}(r,Q) \wedge \neg\, \Box\Diamond Q) \wedge \exists \ell, k : \text{wsp}(P_\ell.r_k, Q_{\ell k}) \wedge \text{wcr}(P_\ell.r_k.Q_{\ell k}) \qquad (15)$$

Invariance of the predicate presented in (14) describes ***conspiracy*** related to ***strong fairness*** where, the invariance of the predicate presented in (15) describes conspiracy related to ***weak fairness.***

Interactive Rules

Since all the fairness concepts are related to process interactions we need to study the same in detail. There could be two different type of interactions between a pair of transition rules viz., competitive and non competitive. In the first type of interaction the two transition rules will compete for a single goal where in the second type such challenge is absent. Since fairness issues are relevant to the first type only we would restrict our discussion around that. Let $P_i.r_k$ and $P_j.r_\ell$ be the two transition rules that may perform competitive interaction. The predicate in(P.r) denotes that the rule P.r has been executed. For a non terminating iterative process we associate an integer index m such that in(P.r^m) denotes that the rule P.r has been executed in the mth iteration cycle of the process P. Also the predicate in(P_i^j) denotes the fact that the process P_i is executing its jth cycle Competitive interaction then asserts

$$\text{in}(P_i.r_k^q) \Rightarrow \neg \text{in}(P_j.r_\ell^m) \wedge \text{in}(P_j^m)$$

where, q and m are the current iteration cycle for the processes P_i and P_j respectively. in(P_j^m) in the conjunct is necessary to ensure that the rule P_j is executing its mth cycle.

The rule $P_j.r_\ell^m$ may be blocked by the predicate in($P_i.r_k^q$) during the qth cycle of the process P_i. However, after complete execution of this cycle the corresponding predicate becomes in($P_i.r_k^{q+1}$) which is false before the interaction at the (q+1)th cycle. Hence for the next interaction $P_j.r_\ell^m$ will again get a chance for execution.

In order to express different fairness conditions for the interaction one may consider in(P.r) as the post condition for the rule P.r For example, the above interaction is weakly fair iff

$$\Diamond\Box\ \text{wsp}(P_i.r_k, \text{in}(P_i.r_k)) \Rightarrow \Box\Diamond\, \text{in}(P_i.r_k)$$

and

$$\Diamond\Box\ \text{wsp}(P_j.r_\ell, \text{in}(P_j.r_\ell) \Rightarrow \Box\Diamond\, \text{in}(P_j.r_\ell)$$

System with Interconnected Competitive Interactions

The sub graph of a has_contribution graph for a system S formed by deleting all the type I edges is defined as the ***interaction sub graph,*** and the set of edges included in this sub graph, i.e., the set of all Type II edges is defined as the ***interaction set*** for a system **S.** An interaction sub graph is a ***spanning interaction sub graph*** when it includes all nodes of the original has_contribution graph. We associate an attribute with each Type II edge to express the relevant fairness concept related to the corresponding interaction. This fairness, however, is the one that is obtained when the participating rules are operating in isolation.

Consider the spanning interaction sub graph for a system **S** with complexity 1 as shown in Fig 5. Let there be n number of sequential processes having competitive interaction between each pair of rules $P_{i_j}.r_{k_{j+1}}$ and $P_{i_{(j+1) \bmod n}}.r_{k_j}$. Collection of edges

http://doi.acm.org/10.1145/1507195.1507203 DOI: 10.1145/1507195.1507203

$E_{II} = \{\forall j : (P_{i_j}.r_{k_j+1},\ P_{i_{(j+1) \bmod n}}.r_{k_j})\}$ is the interaction set for the system **S.** Let us also assume that the system is deadlock free. Deadlock freedom for such cyclic structures is possible by proper event ordering [14]. However, starvation may not be overruled, because, it may be possible that every time when a particular interaction is invoked one specific rule will be denied from execution by the other rule. For example, the rule $P_{i_j}.r_{k_j+1}$ may be smart enough so that at each turn it is executed disallowing execution of the rule $P_{i_{(j+1) \bmod n}}.r_{k_j}$. In the following theorem we would find a method for avoiding starvation.

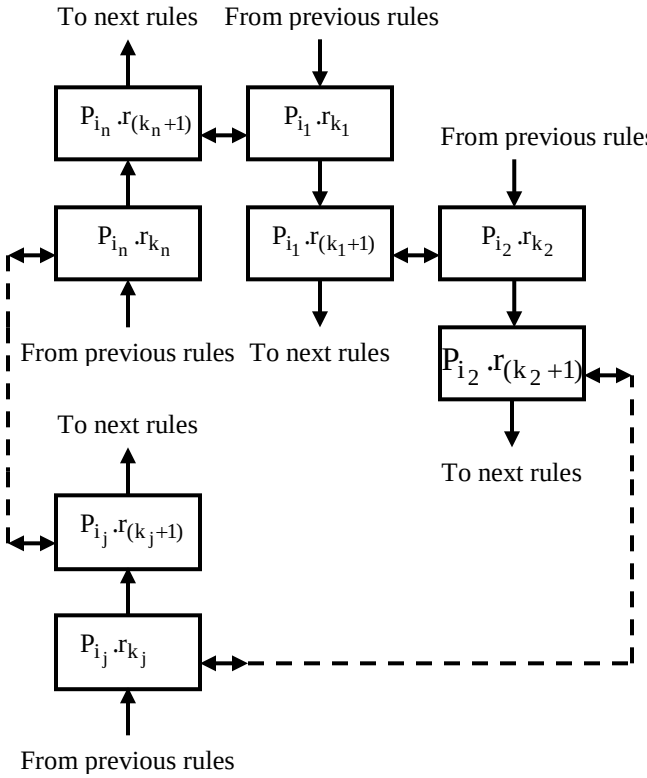

Fig 5 : Portion of the ***has_contribution*** graph for a system **S**

Theorem 0

Starvation of any rule referred to least fairness in the above system **S** may be avoided if fairness associated with each of the edges included in the interaction set is least fair.

Proof :

We would prove this theorem by contradiction. Without loss of generality let us assume that in the above system starvation of a rule $P_{i_j}.r_{(k_j+1)}$ can not be avoided in its $m_j{}^{th}$ cycle. Since the system is deadlock free and all the individual interactions are least fair, this can only happen if the rule is permanently blocked. Let us consider that it is blocked by the predicate $in(P_{i_{(j+1) \bmod n}}.r_{k_{(j+1) \bmod n}}^{m_{(j+1) \bmod n}})$ i.e.,

$\varnothing in(P_{i_j}.r_{(k_j+1)}^{m_j})$

$\Rightarrow \square\, in(P_{i_{(j+1) \bmod n}}.r_{k_{(j+1) \bmod n}}^{m_{(j+1) \bmod n}}) \wedge \varnothing in(P_{i_{(j+1) \bmod n}}^{m_{(j+1) \bmod n}+1})$

That is for permanent blocking, execution of the rule $P_{i_{(j+1) \bmod n}}.r_{(k_{(j+1) \bmod n}+1)}$ in its $m_{(j+1) \bmod n}{}^{th}$ cycle should also be blocked by the predicate $in(P_{i_{(j+2) \bmod n}}.r_{k_{(j+2) \bmod n}}^{m_{(j+2) \bmod n}})$, because otherwise the process $P_{i_{(j+1) \bmod n}}$ will complete its execution in this cycle and if the predicate $in(P_{i_{(j+1) \bmod n}}.r_{k_{(j+1) \bmod n}}^{m_{(j+1) \bmod n}+1})$ is false then this happening will provide an opportunity to the rule $P_{i_j}.r_{(k_j+1)}$ for execution. Repeated occurrence of the above set of events would eventually cause execution of the rule $P_{i_j}.r_{(k_j+1)}$ contradicting the initial assumption, i.e.

$\varnothing in(P_{i_j}^{m_j}..r_{(k_j+1)})$

$\Rightarrow \square\, in(P_{i_{(j+1) \bmod n}}.r_{k_{(j+1) \bmod n}}^{m_{(j+1) \bmod n}}) \wedge \varnothing in(P_{i_{(j+1) \bmod n}}^{m_{(j+1) \bmod n}+1})$

$\Rightarrow \varnothing\, in(P_{i_{(j+1) \bmod n+1}}.r_{k_{(j+1) \bmod n+1}}^{m_{(j+1) \bmod n}})$

$\Rightarrow \square\, in(P_{i_{(j+2) \bmod n}}.r_{k_{(j+2) \bmod n}}^{m_{(j+2) \bmod n}}) \wedge \varnothing in(P_{i_{(j+2) \bmod n}}^{m_{(j+2) \bmod n}+1})$

Proceeding in this manner we would require that the rule $P_{i_{(j-1) \bmod n}}.r_{k_{(j-1) \bmod n+1}}$ be blocked by the predicate $in(P_{i_j}.r_{k_j})$ in its $m_{(j-1) \bmod n}{}^{th}$ cycle i.e.

$\varnothing\, in(P_{i_{(j-1) \bmod n}}.r_{k_{(j-1) \bmod n+1}}^{m_{(j-1) \bmod n}}) \Rightarrow \square\, in(P_{i_j}.r_{k_j}) \wedge \varnothing in(P_{i_j}^{m_j+1})$

In the above scenario we would get $\forall j : \varnothing in(P_{i_j}^{m_j+1})$, that is the system would run into deadlock contradicting the assumption for deadlock freedom. Hence the initial assumption that that, in a deadlock free system the rule $P_{i_j}.r_{(k_j+1)}$ would starve is false.

Consider some interval when the instance of the rule $P_{i_j}.r_{k_j+1}^{m_j}$ is blocked. Since the system is deadlock free, there should exist a process $P_{i_r}.r_{k_r}$ that is executing. After a finite amount of time it should finish its execution and release the rule $P_{i_{(r-1) \bmod n}}.r_{k_{(r-1) \bmod n+1}}^{m_{(r-1) \bmod n}}$. This process will continue till $P_{i_{(j+1) \bmod n}}.r_{k_{(j+1) \bmod n}}^{m_{(j+1) \bmod n}}$ finishes its execution and releases $P_{i_j}.r_{k_j+1}^{m_j}$. This proves the theorem.

http://doi.acm.org/10.1145/1507195.1507203 DOI: 10.1145/1507195.1507203

Starvation Free Solution to Dining Philosopher Problem

We have shown above that a philosopher in the deadlock free solution to dining philosophers problem may suffer from starvation. This may occur due to the conspiracy of his neighbors i.e., the neighboring philosophers may repeatedly acquire the forks denying the concerned philosopher in getting the same. Since the interaction sub graph of this system is a spanning interaction sub graph, theorem 0 is applicable in this case. That is, if the fork allocation procedures are made fair then such continuous denial can be avoided and we would get a starvation free solution. Fairness in fork allocation may be achieved by using any starvation free [13] mutual exclusion protocol [15,16].

Conclusion

Different fairness definitions and their various representation schemes have been proposed. Unfairness and conspiracies related to these concepts have also been described. Concept of starvation has been generalized. A theorem to derive a starvation free solution from a given deadlock free algorithm has been proved. Application of this theorem has been illustrated using Dining Philosophers Problem. In the deadlock free loop connected message passing architecture shown in Fig 1 the node N_i has priority over U_i. Therefore, it is not possible to apply technique like theorem 0 to get a starvation free solution. Separate technique need to be developed for this problem that will be discussed in a next paper.

References

[1] Apt, K. R., Francez, N. and Katz, S. (1988): Appraising fairness in languages for distributed programming. Distributed Computing, vol. 2, pp 226–241.

[2] Joung, Y. J. (2001): On fairness notions in distributed systems: A characterization of implementability. Information and Computation, vol. 166, pp 1–34.

[3] Owicki, S. S., Lamport, L. (1982): Proving liveness properties of concurrent programs. ACM-TOPLAS, vol. 4, pp 455-495.

[4] Attie, P.C., Francez, N. and Grumberg, O. (1993): Fairness and hyperfairness in multi-party interactions. Distributed Computing, vol. 6, pp 245–254.

[5] Lamport, Leslie (2000): Fairness and hyperfairness., Distributed Computing, vol. 13, pp 239–245.

[6] Bandyopadhyay, A. K. and Bandyopadhyay, J. (2000): On the derivation of a correct deadlock free communication kernel for loop connected message passing architecture from its user's specification. ELSVIER Journal of System Architecture, vol. 46, pp. 1257-1261.

[7] Dijkstra, E.W. (1971): Hierarchical ordering of sequential processes. Acta Informatica , vol. 1, pp 115– 138.

[8] Banerjee, Jayasri, Bandyopadhyay, Anup Kumar and Mandal, Ajit Kumar (2007): Application of Dijkstra's Weakest Precondition Calculus to Dining Philosophers Problem. ACM SIGSOFT Software Engineering Notes, Vol. 32, July 2007.

[9] Best, E. (1984): Fairness and conspiracies, Information Processing Letters, vol.18, pp 215–220.

[10] Völzer, Hagen (2005): On Conspiracies and Hyperfairness in Distributed Computing, P. Fraigniaud (Ed.): DISC 2005, LNCS 3724, pp. 33–47.

[11] Dijkstra, E. W. (1976): A discipline of programming, Englewood Cliffs, NJ: Prentice– Hall, 1976.

[12] Schneider, F. B. (1997): On Concurrent Programming. Springer Verlag, N. Y., Inc.

[13] Bandyopadhyay, Anup Kumar (2007): Modeling Fairness and Starvation in Concurrent Systems. ACM SIGSOFT Software Engineering Notes, Volume 32, November 2007.

[14] Banerjee, Jayasri, Bandyopadhyay, Anup Kumar and Mandal, Ajit Kumar (2007): Some Investigations on Deadlock Freedom Issues of a Cyclically Connected System Using Dijkstra's Weakest Precondition Calculus" , ACM SIGPLAN Notices, Vol. 42, pp 10-15.

[15] Peterson, Gary L. (1981): Myths about the mutual exclusion problem. Information Processing Letters, vol. 12, No. 3, pp 115-116.

[16] Banerjee, Jayasri, Bandyopadhyay, Anup Kumar and Mandal, Ajit Kumar (2007): On the Correctness Issues in Two - Process Mutual Exclusion Algorithms. ACM SIGSOFT Software Engineering Notes, Nov. 2007.

Appendix

Following results are available from [13]

Lemma	Statement
1	$\neg \Diamond^i P = \Box^i \neg P$
2	$\neg \Diamond_i P = \Box_i \neg P$
3	$\neg \Diamond_{m,n} P = \Box_{m,n} \neg P$
4	$\neg \Diamond P = \Box \neg P$
5	$\varnothing P \vee \varnothing Q = \varnothing(P \wedge Q)$
6	$\varnothing P \wedge \varnothing Q = \varnothing(P \vee Q)$
7	$\neg \varnothing P = \Diamond P$
8	$\varnothing \neg P = \Box P$

Table 3. Lemmas proved in [13]

Lemma 9 :

$\Diamond \neg P = \neg \Box P$

Proof :

$\Diamond \neg P = \neg \Box \neg \neg P$

$= \neg \Box P$

Hence the Lemma is proved.

Lemma 10

$\varnothing P = \Box \varnothing P$

Proof :

$\Box \varnothing P = \Box \Box \neg P$

$= \Box \neg P$

$= \varnothing P$

Hence the Lemma is proved.

Lemma 11
$\varnothing\Box P = \neg\Diamond\Box P$
Proof :
$\varnothing\Box P = \Box\neg\Box P$
$= \neg\Diamond\Box P$
Hence the Lemma is proved

Lemma 12
$\Diamond\varnothing P = \neg\Box\Diamond P$
Proof :
$\Diamond\varnothing P = \Diamond\neg\Diamond P$
$= \neg\Box\Diamond P$ (by lemma 9)
Hence the Lemma is proved

Lemma 13 :
$\varnothing\Diamond P = \varnothing P$
Proof :
$\varnothing\Diamond P = \Box\neg\Diamond P$
$= \Box\Box\neg P$
$= \varnothing P$
Hence the Lemma is proved

Lemma 14
$\Diamond \Box\neg P=\neg\Box\Diamond P$
Proof :
$\Diamond \Box\neg P = \neg\Box\neg\Box\neg P$
$=\neg\Box\Diamond P$
Hence the Lemma is proved

Lemma 15
$\neg\Diamond\Box P = \Box\Diamond\neg P$
Proof :
$\neg\Diamond\Box P = \Box\neg\Box P$
$=\Box\Diamond\neg P$
Hence the Lemma is proved

Lemma 16
$\neg\Box\varnothing P = \Diamond P$
Proof :
$\neg\Box\varnothing P = \Diamond\neg\varnothing P$
$= \Diamond\Diamond P$
$= \Diamond P$
Hence the Lemma is proved

Lemma 17
$\neg\varnothing\Box P = \Diamond\Box P$
Proof :
$\neg\varnothing\Box P = \neg\neg\Diamond\Box P$
$= \Diamond\Box P$
Hence the Lemma is proved.

Lemma 18
$\neg\varnothing\Diamond P = \Diamond P$
Proof :
$\neg\varnothing\Diamond P = \Diamond\Diamond P$
$= \Diamond P$
Hence the Lemma is proved.

Lemma 19
$\neg\Diamond\varnothing P = \Box\Diamond P$
Proof :
$\neg\Diamond\varnothing P = \neg\neg\Box\neg\varnothing P$
$= \Box\Diamond P$
Hence the Lemma is proved.

Lemma 20
$\Box\Diamond P = \varnothing\Box\neg P$
Proof:
$\Box\Diamond P = \Box\neg\Box\neg P$
$= \varnothing\Box\neg P$
Hence the Lemma is proved.

Lemma 21
$\Diamond\Box P = \neg\varnothing\Box P$
Proof :
$\Diamond\Box P = \neg\Box\neg\Box P$
$= \neg\varnothing\Box P$
Hence the Lemma is proved.

Lemma 22 :
$\varnothing\varnothing P = \Box\Diamond P$
Proof :
$\varnothing\varnothing P = \neg\Diamond\varnothing P$
$= \neg\Diamond \neg\Diamond P$
$= \Box \neg\neg\Diamond P$
$= \Box\Diamond P$
Hence the Lemma is proved.

http://doi.acm.org/10.1145/1507195.1507203 DOI: 10.1145/1507195.1507203