Generating non-conspiratorial executions

D. Ruiz [a], R. Corchuelo [a,*], J.L. Arjona [b]

[a] *Dep. de Lenguajes y Sistemas Informáticos, Universidad de Sevilla, ETSI Informática, Avda. Reina Mercedes, s/n, Sevilla 41012, Spain*
[b] *Dep. de Tecnologías de la Información, Universidad de Huelva, Pabellón Torreumbría, Ctra. Huelva-La Rábida, Huelva 21071, Spain*

Abstract

Avoiding conspiratorial executions is useful for debugging, model checking or refinement, and helps implement several well-known problems in faulty environments; furthermore, avoiding non-equivalence robust executions prevents conflicting observations in a distributed setting from occurring. Our results prove that scheduling pairs of states and transitions in a strongly fair manner suf-fices to prevent conspiratorial executions; we then establish a formal connection between conspiracies and equivalence robustness; finally, we present a transformation scheme to implement our results and show how to build them into a well-known distributed scheduler. Previous results were applicable to a subset of systems only, just attempted to characterise potential conspiracies, or were tightly bound up with a particular interaction model.

Keywords: Distributed systems; Concurrency; Conspiracies; Equivalence robustness; Fairness

1. Introduction

Conspiracies and equivalence robustness are well known topics in the concurrency literature [4,5,11]: an execution of a transition system is conspiratorial iff there is a state that is visited finitely many times despite it is reachable infinitely many times; it is equivalence robust if given a fairness assumption according to which it is fair, then no execution that is equivalent to it up to the order in which the transitions are executed may be unfair.

☆ The work reported in this article was supported by the Spanish Interministerial Commission on Science and Technology under grant TIC2003-02737-C02.

* Corresponding author.
E-mail addresses: druiz@us.es (D. Ruiz), corchu@us.es (R. Corchuelo), jose.arjona@diesia.uhu.es (J.L. Arjona).

Avoiding conspiratorial executions is obviously appealing for debugging or model checking purposes, but there are also intricate connections with refinement robustness and fault tolerance. Refining a transition system consists in decomposing some transitions so that the system is represented at a lower level of abstraction [3,7], and it is well known that conspiracies may prevent important liveness properties to be preserved after refinement [2,11,17]. For instance, strong fairness suffices to prove that no philosopher in the system in Fig. 1(a) may starve, but it is not enough for the refined system in Fig. 1(b) since an execution with trace $ini_1(ini_3end_1ini_1end_3)^\omega$ is strongly fair but the second and the fourth philosophers starve. Regarding fault tolerance, it has been recently proved that the dining philosophers, the consensus, or the committee co-

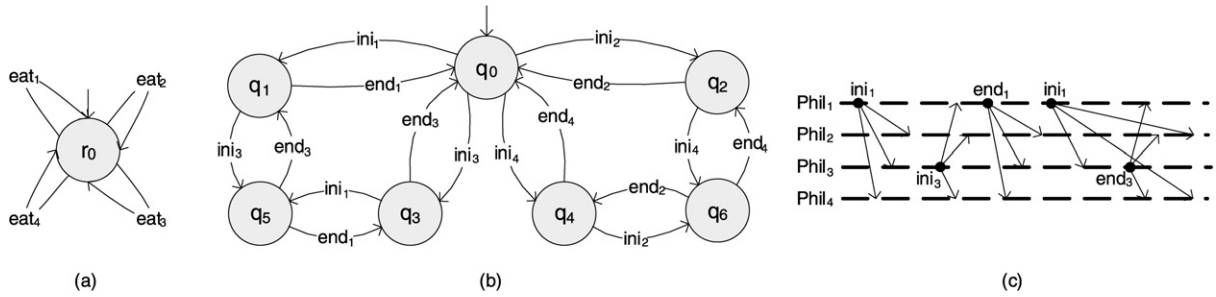

Fig. 1. (a) A four-philosopher system in which each *eat* transition co-ordinates a philosopher with its two neighbouring forks. (b) A refined system in which the *ini* and *end* transitions represent the start and finish of the corresponding *eat* actions. (c) Part of a concurrent execution with conflicting observations.

ordination problems may be implemented in a faulty environment if conspiracies are avoided [18].

If a transition system is used to model a distributed system, then its executions are called observations since they implicitly reflect the point of view of a process that perceives the actual concurrent executions through the messages it gets from the others. In this context, equivalence robustness is desirable to avoid conflicting observations of the same actual concurrent execution [2,5, 10,11]. For instance, assume that the system described in Fig. 1(b) is implemented so that each philosopher is an independent process that broadcasts the transitions it executes to the others; Fig. 1(c) depicts an excerpt of an execution in which the first and the third philosophers observe the trace $ini_1(ini_3 end_1 ini_1 end_3)^\omega$, which corresponds to a fair execution of the transition system; however, the second and the fourth philosophers observe the trace $ini_1(ini_3 end_1 end_3 ini_1)^\omega$, which is equivalent to the previous one but corresponds to an unfair execution of the transition system. Apart from its practical interest to avoid conflicting observations, equivalence robustness is also desirable for systems in which several fairness notions need to be combined [2,5] and a strong requirement for a fairness notion to be implementable by means of wait-free schedulers [9].

On account of the previous results, several authors have worked on schedulers to prevent conspiracies, viz.: Attie, Francez, and Grumberg proposed a scheduler that prevents conspiracies that are due to race conditions in a subset of systems written in IP that are known as conspiracy resistant [1,6], i.e., systems in which disabling a transition deliberately may not lead to deadlock; Völzer got inspiration from an earlier proposal by Joung [8] and designed a scheduler that does not require the system under consideration to be conspiracy resistant [18]; Ruiz et al. characterised potential conspiracies and designed a centralised scheduler that prevents conspiracies at the cost of reducing concurrency if the number of potentially conspiratorial situations detected exceeds a predefined threshold [15,16]. Contrarily, other authors have focused on devising fairness notions to characterise and rule out (some) conspiracies, viz.: Best's ∞-fairness [4], Queille and Sifakis's predicate reachability fairness [14], and Attie, Francez, and Grumberg's, Lamport's or Völzer's hyperfairness notions [1,12,18]. Attie et al. also proved that preventing conspiracies in conspiracy-resistant systems enforces equivalence robustness [1], and Joung worked on restricting and implementing a fairness notion so that it precludes the executions that are not equivalence robust [9,10]; recently, Kurki-Suonio pointed out that it is a better idea to introduce additional synchronisation so that a system can regulate itself and prevent the executions with unfair equivalents from occurring [11], but this idea was not developed further.

In this paper, we prove that scheduling pairs of states and transitions in a strongly fair manner suffices to prevent conspiracies in finite, functional transition systems; in turn, this suffices to preserve equivalence robustness if the system is connected. To the best of our knowledge, these results are novel since the previous attempts focused on trying to schedule transitions independently from states [1,18] or just trying to characterise potentially conspiratorial situations [15,16]. We also present a transformation scheme to implement our results and sketch how to build it into to a well-known scheduler [2] for Action Systems [2,11] and Interacting Processes [6]. The transformation improves on Attie et al.'s scheduler in that we do not require the original system to be conspiracy resistant or the conspiracies to be due to race conditions, and it is not bounded up with the IP language; it improves on Joung's or Völzer's results in that we do not commit to a particular communication model, which makes our solution more general; it improves on Ruiz et al.'s previous results in that our scheme does not rely on a heuristic to detect conspiracies, and it may be implemented in a distributed manner.

2. Preliminaries

Definition 1 *(Transition system).* A transition system S is a triple (Q, Q_0, T) in which Q is a set of states, $Q_0 \subseteq Q$ is a set of initial states, and $T \subseteq 2^{Q \times Q}$ is a set of transitions. For each state q and transition t, the set of t-successors of q is $t(q) = \{r \in Q \mid (q, r) \in t\}$. The domain of transition t is $\mathrm{dom}(t) = \{q \in Q \mid \exists r \in Q \cdot (q, r) \in t\}$; its image is $\mathrm{img}(t) = \{r \in Q \mid \exists q \in Q \cdot (q, r) \in t\}$. The set of outgoing transitions of state q is $\mathrm{out}(q) = \{t \in T \mid q \in \mathrm{dom}(t)\}$;[1] the set of incoming transitions is $\mathrm{in}(q) = \{t \in T \mid q \in \mathrm{img}(t)\}$. State q is reachable from state p iff there is a finite sequence of states $\langle r_0, r_1, \ldots, r_{n-1}\rangle \in Q^*$ and a finite sequence of transitions $\langle u_0, u_1, \ldots, u_n\rangle \in T^*$ such that $r_0 \in u_0(p), r_1 \in u_1(r_0), \ldots, q \in u_n(r_{n-1})$. S is connected iff for every $(p, q) \in Q \times Q$, q is reachable from p; S is finite iff Q is finite; S is functional iff $|t(q)| = 1$ for every $q \in Q$ and $t \in \mathrm{out}(q)$.

Definition 2 *(Execution).* Let $S = (Q, Q_0, T)$ be a transition system. λ is an execution of S iff it is an infinite sequence of steps $\langle (q_0, t_0), (q_1, t_1), (q_2, t_2), \ldots\rangle \in (Q \times T)^\omega$ such that $q_0 \in Q_0$ and for every $i \in \mathbb{N}$, $q_{i+1} \in t_i(q_i)$; $\Pi(S)$ denotes the set of all possible executions of S. Given a step of the form $(q, t) \in Q \times T$, we say that state q is visited, that transition t is executed, that every transition in $\mathrm{out}(q)$ is enabled, and that the transitions in $\mathrm{out}(q) \setminus \{t\}$ are discarded. The trace of execution λ is the sequence of transitions $\langle t_0, t_1, t_2, \ldots\rangle$.

Definition 3 *(Fairness).* Let $S = (Q, Q_0, T)$ be a transition system and λ an execution in $\Pi(S)$. λ is strongly fair with respect to T (or simply fair) iff every transition that is enabled infinitely many times in λ is executed infinitely many times. λ is strongly fair with respect to $Q \times T$ iff for all $q \in Q$ and $t \in \mathrm{out}(q)$, if q is visited infinitely many times in λ, then step (q, t) appears infinitely many times in λ. For instance, the execution $\langle (q_0, \mathit{ini}_1), ((q_1, \mathit{ini}_3), (q_5, \mathit{end}_1), (q_3, \mathit{ini}_1), (q_5, \mathit{end}_3))^\omega\rangle$ of Fig. 1(b) is fair with respect to T, but unfair with respect to $Q \times T$.

Definition 4 *(Conspiracy).* Let S be a transition system (Q, Q_0, T) and λ an execution in $\Pi(S)$. λ is conspiratorial iff there is a state $q \in Q$ such that there are infinitely many steps of the form (p, t), with $p \neq q$, such that q is reachable from p, but it is visited finitely many times only. S is conspiratorial iff there is a conspiratorial execution in $\Pi(S)$.

Definition 5 *(Equivalence robustness).* Let S be a transition system and $\lambda_1, \lambda_2 \in \Pi(S)$ two executions. λ_1 and λ_2 are equivalent iff their traces are permutations of each other. λ_1 is equivalence robust iff it is fair and for every $\lambda_3 \in \Pi(S)$ such that λ_3 is equivalent to λ_1, then λ_3 is fair. S is equivalence robust iff every execution in $\Pi(S)$ is equivalence robust.

3. Supporting results

In this section, we first establish a relationship between strong fairness with respect to $Q \times T$ and conspiracies, cf. Theorem 1; we then establish a connection between conspiracies and equivalence robustness in systems that are connected, cf. Theorem 2.

Theorem 1. *Let $S = (Q, Q_0, T)$ be a finite, functional transition system and $\lambda \in \Pi(S)$ an execution that is strongly fair with respect to $Q \times T$. λ is then strongly fair with respect to T and non-conspiratorial.*

Proof. Assume that λ is strongly fair with respect to $Q \times T$, but unfair with respect to T. This implies that there is a transition $t \in T$ that is enabled infinitely may times in λ, but it is executed finitely many times only. Since S is finite, there must be a state $q \in \mathrm{dom}(t)$ that is visited infinitely many times in λ, but this contradicts the hypothesis that λ is strongly fair with respect to $Q \times T$ since this implies that step (q, t) must appear infinitely many times in λ.

Assume now that λ is strongly fair with respect to $Q \times T$, but conspiratorial. This implies that there must be a state $p \in Q$ that is visited infinitely many times in λ and a state $q \in Q$ that is visited finitely many times despite it is reachable from p. Note that p must exist or, otherwise, S would not be finite. Without loss of generality, we may assume that q is the "closest" state to λ, i.e., there exists $t \in \mathrm{out}(p)$ such that $q \in \mathrm{img}(t)$. Since S is functional, $t(p) = \{q\}$, which contradicts the hypothesis that λ is strongly fair with respect to $Q \times T$ since this implies that the subsequence of steps $\langle (p, t), (q, v)\rangle$, for some $v \in \mathrm{out}(q)$, must appear infinitely many times in λ. □

Theorem 2. *Let $S = (Q, Q_0, T)$ be a finite, functional transition system and λ a fair execution in $\Pi(S)$. If S is connected and λ is not conspiratorial, then λ is equivalence robust.*

[1] Henceforth, we assume that $\mathrm{out}(q) \neq \emptyset$ for every $q \in Q$ since sink states may be removed by introducing a stuttering transition that maps them onto themselves [12].

$$
\begin{aligned}
(q_2,\delta_2)\in t'(q_1,\delta_1) \quad \Leftrightarrow \quad & \widehat{t'}(q_1)=\{q_2\} && (1)\\
\wedge \quad & \delta_1(q_1,\widehat{t'})=\min_{t\in T}\big\{\delta_1(q_1,t)\mid t\in \mathrm{out}(q_1)\big\} && (2)\\
\wedge \quad & \delta_2(q_1,\widehat{t'})\in\mathbb{N} && (3)\\
\wedge \quad & \big(\forall t\in T\setminus\{\widehat{t'}\}\cdot t\in\mathrm{out}(q_1)\Rightarrow \delta_2(q_1,t)=\delta_1(q_1,t)-1\big) && (4)\\
\wedge \quad & \big(\forall t\in T\setminus\{\widehat{t'}\}\cdot t\notin\mathrm{out}(q_1)\Rightarrow \delta_2(q_1,t)=\delta_1(q_1,t)\big) && (5)\\
\wedge \quad & \big(\forall q\in Q\setminus\{q_1\}, t\in T\cdot \delta_2(q,t)=\delta_1(q,t)\big) && (6)
\end{aligned}
$$

Fig. 2. Transitions of our transformation.

Proof. Assume that $\lambda_1 \in \Pi(S)$ is fair and non-conspiratorial, but there exists an equivalent execution $\lambda_2 \in \Pi(S)$ that is unfair. This means that there is at least a transition t that is enabled infinitely many times in λ_2 but executes finitely many times only. Let $q \in \mathrm{dom}(t)$ be one of the states that enable t infinitely many times, $u \in \mathrm{in}(q)$ one of the transitions by means of which q is visited infinitely many times, and $p \in \mathrm{dom}(u)$ one of the states in which u is executed infinitely many times, i.e., there is a transition $v \in \mathrm{out}(q)$ such that the subsequence of steps $\langle (p,u),(q,v)\rangle$ is repeated infinitely many times in λ_2. Note that p, q, and u must exist or otherwise S would not be finite. If λ_2 is equivalent to λ_1, it implies that u must be executed infinitely many times in λ_1, which in turn implies that there must exist $r, s \in Q$, $w \in \mathrm{out}(s)$ such that the subsequence of steps $\langle (r,u),(s,w)\rangle$ is repeated infinitely many times in λ_1. If $p = r$, then $q = s$ since S is functional, which implies that t is discarded infinitely many times in λ_1 and contradicts the hypothesis that this execution is fair. This is also the case when $p \neq r$ but $q = s$. If $p \neq r$ and $q \neq s$, then q must be reachable from s since we assume that S is connected; this in turn implies that q must be visited infinitely many times in λ_1 since this execution is not conspiratorial, but t is executed finitely many times, which contradicts the hypothesis that this execution is fair. □

4. Transformation scheme

In this section, we present a transformation scheme that uses Olderog and Apt's scheduling policy [13] to build a scheduler for $Q \times T$ into a given transition system, cf. Definitions 6 and 7, as well as Lemma 1; we then prove that it is correct, cf. Lemma 2 and Theorem 3.

Definition 6 *(Transformation scheme).* Let $S = (Q, Q_0, T)$ be a finite, functional transition system. We define its transformation as a new transition system $\Delta(S) = (Q', Q'_0, T')$, where:

1. $Q' = Q \times (Q \times T \to \mathbb{Z})$, i.e., every state is augmented with a map that associates a counter with every transition in every state. Each counter is interpreted as a priority so that the smaller it is, the greater the priority it represents becomes.
2. $Q'_0 = Q_0 \times (Q \times T \to \mathbb{N})$, i.e., the initial priorities are arbitrary natural numbers.
3. There is a total bijective mapping between T' and T that is denoted as $\widehat{\ }$ and verifies the formula in Fig. 2. That is, transition $t' \in T'$ may transit from (q_1,δ_1) to (q_2,δ_2) iff (1) $\widehat{t'}$ transits from q_1 to q_2, (2) $\widehat{t'}$ has the maximum priority amongst the transitions that are enabled in q_1, (3) the priority of $\widehat{t'}$ in q_1 is reset to an arbitrary natural number in q_2, (4) the priorities of the transitions that have been discarded in q_1 are increased in q_2, i.e., their counters are decreased, and (5), (6) the remaining transitions keep their priorities.

Definition 7 *(Projections).* Let S be a finite, functional transition system (Q, Q_0, T), and $\Delta(S) = (Q', Q'_0, T')$ its transformation. Given a state $q' \in Q'$ of the form (q,δ), we define its kernel as $\widehat{q'} = q$. If $\lambda' = \langle (q'_0,t'_0),(q'_1,t'_1),(q'_2,t'_2),\ldots\rangle$ is an execution of $\Delta(S)$, then we denote its projection on S as $\widehat{\lambda'} = \langle (\widehat{q'_0},\widehat{t'_0}),(\widehat{q'_1},\widehat{t'_1}),(\widehat{q'_2},\widehat{t'_2}),\ldots\rangle$.

Lemma 1. *Let $S = (Q, Q_0, T)$ be a finite, functional transition system. If $\lambda' \in \Pi(\Delta(S))$, then $\widehat{\lambda'} \in \Pi(S)$.*

Proof. It follows from (1) since no transition may be enabled in $\Delta(S)$ unless its counterpart is also enabled in S. □

Lemma 2. *Let $S = (Q, Q_0, T)$ be a finite, functional transition system, and $\Delta(S) = (Q', Q'_0, T')$ its transformation. For all $(q,\delta) \in Q'$, if (q,δ) is reachable from an initial state, then, for all $t \in T$, $r \in Q$, $\delta(r,t) \geqslant -|T| + 1$.*

Proof. [2] Let $\lambda' = \langle (q'_0, t'_0), (q'_1, t'_1), (q'_2, t'_2), \ldots \rangle$ be an execution of $\Delta(S)$ so that $q'_i = (q_i, \delta_i)$ for all $i \in \mathbb{N}$; let $U_i(q,k) = \{t \in T \mid \delta_i(q,t) \leqslant -k\}$ for all index $i \in \mathbb{N}$, state $q \in Q$, and natural number $k \in [1, |T|]$. Thus, proving that there is not a reachable state in which a counter is less than or equal to $-|T|$ amounts to proving that $|U_i(q,k)| \leqslant |T| - k$ for all $i \in \mathbb{N}$, $q \in Q$, and $k \in [1, |T|]$ since this implies that $|U_i(q, |T|)| = 0$ if $k = |T|$.

The proof follows by induction and reductio ad absurdum: initially, $\delta_0(q,t) \geqslant 0$ for all $q \in Q$ and $t \in T$; for the inductive step, we assume that there exists a state $q \in Q$ and a natural number $k \in [1, |T|]$ such that $|U_{i+1}(q,k)| \geqslant |T| - k + 1$ for some $i \in \mathbb{N}$. By (6), it is clear that such a state must be q_i since it is the only state in which a transition may change its priority. By (4)–(6), we conclude that for all $u \in U_{i+1}(q_i,k)$, $\delta_i(q_i,u) \leqslant -k+1$; thus, $|U_{i+1}(q_i,k)| \leqslant |T| - k + 1$ according to the induction hypothesis, which, in turn, implies that $|U_{i+1}(q_i,k)| = |T| - k + 1$ according to the reductio ad absurdum hypothesis. Furthermore, $\widehat{t'_i} \in U_{i+1}(q_i,k)$ according to (2), which implies that $\delta_{i+1}(q_i, \widehat{t'_i}) \geqslant 0$ according to (3). This is contradictory since $\delta_{i+1}(q_i,t) \leqslant -k$ for all $t \in U_{i+1}(q_i,k)$ according to the reductio ad absurdum hypothesis. □

Theorem 3. *Let $S = (Q, Q_0, T)$ be a finite, functional transition system and $\Delta(S) = (Q', Q'_0, T')$ its transformation. The executions in $\Pi(\Delta(S))$ are strongly fair with respect to $Q \times T$.*

Proof. Let λ' be an execution in $\Pi(\Delta(S))$ and assume that there is a state $q' \in Q'$ and a transition $t' \in \mathrm{out}(q')$ such that q' is visited infinitely many times, but the step (q', t') appears finitely many times in λ. According to (4), every time t' is discarded, the counter associated with $\widehat{t'}$ in state $\widehat{q'}$ decreases by one, thus increasing its priority. If this situation is repeated infinitely many times, then a state in which this counter might be an arbitrarily small integer might be visited, which would contradict Lemma 2. □

5. Conclusions

We have devised a transformation scheme for finite, functional transition systems that uses Olderog and Apt's policy to schedule pairs of states and transitions in a strongly fair manner. We have proved that this results in a system that is not conspiratorial; if the original system is connected, this in turn implies that the result is equivalence robust, too. Furthermore, the scheme is correct if the range of integers available includes the interval $[-|T| + 1, 0]$, i.e., it is implementable with finite counters.

Our result is useful to improve Back and Kurki-Suonio's shared-bus scheduler [2] for Actions Systems [11] or Interacting Processes [6]. Roughly speaking, it works as follows: when a process is ready to execute an action, it broadcasts a willingness message to inform the others; when a process concludes that it is safe to execute an action, it broadcasts a select message so that the other processes involved in that action can execute it; if more than one action may be executed, the process selects one of them at random. The authors described a protocol that ensures that only one process may broadcast a select message for a particular action occurrence, but the details are irrelevant here. It is not difficult to improve this scheduler so that it can avoid conspiracies: we require each process to be equipped with a copy of the transition system they implement as transformed by the scheme presented in Section 4, so that when a process is ready to send a select message it can know what the current state of the system is and restrict the choices available, i.e., it may safely delay the execution of an action if necessary. (Note that the proposal by Attie, Francez, and Grumberg also delays the execution of some actions, but it may lead to deadlock if the system under consideration is not conspiracy-resistant [1].) Obviously, for this idea to work it is necessary that all of the processes get the same observation of the actual concurrent execution, which is the case for shared-bus networks, and all of the actions must terminate in finite time, which was already a requirement for the original scheduler.

The importance of this improvement is twofold: on the one hand, Back and Kurki-Suonio proved that liveness properties are preserved by their scheduler as long as the system under consideration is not conspiratorial; their original scheduler cannot prevent conspiracies, whereas our simple modification avoids them completely. On the other hand, Kurki-Suonio argued that the problem with equivalence robustness lies in the classical notion of observation, according to which an observer must not interfere with the executions it observes [11]; he pointed out that an observer of a system is part of that system and, thus, it should be allowed to introduce additional synchronisation that does not result in new executions but restricts them so that no conflicting observations may occur; this is particularly interesting for the subset of observers that are responsible for the scheduling decisions, and our proposal to improve Back

[2] We use the proof method in Ref. [13, Theorem 4.1].

and Kurki-Suonio's scheduler may thus be seen as a practical realisation of this idea.

Acknowledgements

We are thankful to our reviewers for their insightful comments on an earlier version of this paper.

References

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

[2] R.-J. Back, R. Kurki-Suonio, Distributed cooperation with action systems, ACM Transactions on Programming Languages Systems 10 (4) (1988) 513–554.

[3] R.-J. Back, Q. Xu, Refinement of fair action systems, Acta Informatica 35 (2) (1998) 131–165.

[4] E. Best, Fairness and conspiracies, Information Processing Letters 18 (4) (1984) 215–220; Information Processing Letters 19 (3) (1984) 162.

[5] N. Francez, R.-J. Back, R. Kurki-Suonio, On equivalence-completions of fairness assumptions, Formal Aspects of Computing 4 (6) (1992) 582–591.

[6] N. Francez, I. Forman, Interacting Processes, Addison-Wesley, 1996.

[7] R.J. van Glabbeek, U. Goltz, Refinement of actions and equivalence notions for concurrent systems, Acta Informatica 37 (4–5) (2001) 229–327.

[8] Y.-J. Joung, Two decentralized algorithms for strong interaction fairness for systems with unbounded speed variability, Theoretical Computer Science 243 (1–2) (2000) 307–338.

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

[10] Y.-J. Joung, On fairness notions in distributed systems: Equivalence completions and their hierarchies, Information and Computation 166 (1) (2001) 35–60.

[11] R. Kurki-Suonio, Action systems in incremental and aspect-oriented modeling, Distributed Computing 16 (2–3) (2003) 201–217.

[12] L. Lamport, Fairness and hyperfairness, Distributed Computing 13 (4) (2000) 239–245.

[13] E.-R. Olderog, K.R. Apt, Fairness in parallel programs: The transformational approach, ACM Transactions on Programming Language Systems 10 (3) (1988) 420–455.

[14] J.-P. Queille, J. Sifakis, Fairness and related properties in transition systems, Acta Informatica 19 (1983) 195–220.

[15] D. Ruiz, R. Corchuelo, M. Toro, Fairness in systems based on multiparty interactions, Concurrency and Computation: Practice and Experience 15 (11–12) (2003) 1093–1116.

[16] D. Ruiz, R. Corchuelo, J.A. Pérez, M. Toro, An algorithm for ensuring fairness and liveness in non-deterministic systems based on multiparty interactions, in: Euro-Par 2002, pp. 563–572.

[17] H. Völzer, Refinement-robust fairness, in: CONCUR 2002, pp. 547–561.

[18] H. Völzer, On conspiracies and hyperfairness in distributed computing, in: DISC 2005, pp. 33–47.