Acta Informatica 19, 195–220 (1983)

Acta Informatica

Fairness and Related Properties in Transition Systems – A Temporal Logic to Deal with Fairness

J.P. Queille and J. Sifakis

IMAG, BP53X, F-38041 Grenoble Cedex, France

Summary. In this paper we propose a notion of fairness for transition systems and a logic for proving properties under the fairness assumption corresponding to this notion.

We start from an informal characterization of the unfairness as the situation where some event becomes possible infinitely often but has only a finite number of occurrences, which induces various definitions of fairness by considering different classes of events. It results from the comparison of these definitions that the concept of fairness which is useful is "fair reachability" of a given set of states P in a system, i.e. reachability of states of P when considering only the computations such that if, during their execution, reaching states of P is possible infinitely often, then states of P are visited infinitely often.

This definition of fairness suggests the introduction of a branching time logic FCL, the temporal operators of which express, for a given set of states P, the modalities "it is possible that P" and "it is inevitable that P" by considering fair reachability of P. The main result is that, given a transition system S and a formula f of FCL expressing some property of S under the assumption of fairness, there exists a formula f' belonging to a branching time logic CL such that: f is valid for S in FCL iff f' is valid for S in CL. This result shows that proving a property under the assumption of fairness is equivalent to proving some other property without this assumption and that the study of FCL can be made via the "unfair" logic CL, easier to study and for which several results already exist.

Finally, the proposed notion of fairness is compared to other related notions such as the absence of livelock, the absence of starvation and the finite delay property.

I. Introduction

Fairness is a property whose study becomes important when non-deterministic models are used to represent systems, i.e. models such that from a state it is

possible to execute different transitions (actions) and this choice is not specified. In such models there may exist diverging (infinite) computation sequences such that during their execution some *event becomes possible infinitely often* but it has not an infinite number of occurrences.

The preceding statement is an informal characterization of the unfair sequences of a system. In fact, the existence of such sequences may lead to unfair situations where, although an event is realizable infinitely often, it never occurs because conflicts are resolved in a non equitable manner.

The given informal definition is sufficient to apprehend intuitively the concept of fairness. However, many difficulties arise when a formal definition is to be given. In this case, the expression "*some event becomes possible infinitely often*" must be assigned a precise meaning: the class of the events which are of interest has to be characterized; also, "*becomes possible*" and "*infinitely often*" must be defined in a given model. The big variety of choices to be made in order to obtain a formal definition of fairness, explains perhaps the fact that many authors use this concept without attempting to introduce it formally.

Part II of this paper deals with the search for a "satisfactory" definition of fairness which agrees with the informal characterization given above. It is shown that this characterization induces different notions of fairness corresponding to the choice of different classes of events and of different notions of the "possibility" for an event to occur. However, the concept of fairness which regroups the different definitions obtained in this manner, is shown to be too strong and not really useful in practice; it implies that there exist systems such that all their infinite computation sequences are unfair. For this reason, we have "relativized" this concept by defining the notion of unfair computation sequence with respect to some given set of states.

Roughly speaking, an infinite computation sequence is *unfair* with respect to a given set of states P, if the sequence s of the states visited during its execution is such that s contains,

- an infinite number of occurrences of states from which it is possible to reach states of P,
- only a finite number of occurrences of states of P.

It is important to notice that according to this definition a sequence may be fair with respect to a set of states and unfair with respect to some other set. Fairness relative to a set of states is, we believe, the notion which is important in practice since most of the "interesting" system properties express reachability relations of some set of states [20, 21].

The problem of fairness becomes crucial when a property is to be proved in formal systems based on non-deterministic models. In the descriptions with such models, are introduced computation sequences which are not feasible under the assumption of any "reasonable" scheduling policy. These sequences are difficult to distinguish from the feasible ones because, for infinite sequences, fairness cannot be decided by examining any prefix of them. As a result, proving properties valid under the assumption of fairness seems to be a non-trivial problem.

Two different approaches are possible to the problem of proving a property PR for some system S when considering only fair computation sequences.

1) Either transform S into S' such that S' is a system whose computation sequences are all the fair computation sequences of S, and prove the validity of PR for S'. S' can be obtained from S by adjoining to it a "fair scheduler" [2, 16].
2) Or, give a method for associating to PR some other property PR' such that: PR' is valid for S iff PR is valid for S under the assumption of fairness.

In part III, we consider the latter approach by defining a conditional branching time logic for fairness, FCL. In this logic temporal operators express, for a given set of states P, the modalities "it is possible that P" and "it is inevitable that P" by considering only the computation sequences which are fair with respect to P.

The introduction of FCL sets the problem of its comparison with the existing branching time logics [1, 3, 4, 13, 18]. We have shown that given a formula f of FCL it is possible to find a formula f' of a conditional time logic CL such that for any model S: "f is valid for S in FCL" is equivalent to "f' is valid for S in CL".

This result is interesting from several points of view. In particular it shows that the study of FCL can be made via "unfair" logics, easier to study in our opinion, and for which many results already exist (decision procedure, fixed point characterization of temporal operators [4, 3, 18]). In particular, the results on the fixed point characterization of the temporal operators lead to a mechanizable proof method used by the system CESAR [18] to verify properties under the assumptions of fairness.

In part IV, the proposed notion of fairness is compared to other related properties such as starvation, presence of livelock and the finite delay property.

II. Searching for an Adequate Definition of Fairness

II.1. Informal Definition

The following informal characterization of fairness is given which will "guide" our search for formal definitions, in the sense that each definition must describe situations which agree with this characterization.

A computation sequence is *fair* if, during its execution, every event which becomes possible infinitely often is realized infinitely often.

Obviously, all the finite computation sequences are fair. We consider that the fair functioning of a system is characterized by the set of all the fair computation sequences. A sequence which is not fair is said to be unfair.

Example 1. If a divergent computation in a non-deterministic program is such that from each one of the states visited during its execution it is possible to reach termination then this computation sequence is unfair.

Example 2. For a system of processes sharing in mutual exclusion a common resource, the fair functioning is characterized by the property: if a process is present infinitely often before the critical section and the resource is free then it accesses the resource infinitely often.

Example 3. For a transmission line which may lose messages but is never cut, the emission of a message always gives rise to one of the two events: reception of a message or loss of a message. A functioning where an infinity of messages are emitted while only a finite number of messages are received, is an unfair functioning. Symmetrically, a functioning where only a finite number of messages are lost is an unfair functioning.

II.2. Transition Systems

Transition systems is the model used throughout this article.

A *transition system* is a triple $S=(Q, T, \{\xrightarrow{t}\}_{t\in T})$ where,

- Q is a countable set of *states*
- $T=\{t_1, \ldots, t_m\}$ is a set of *transitions*
- $\{\xrightarrow{t}\}_{t\in T}$ is a set of binary relations on $Q(\xrightarrow{t}\subseteq Q\times Q)$ in bijection with the transitions.

Transitions systems have been widely used to model discrete systems (see for example [12, 14, 20]).

Transitions are action names, the effect of which is described by the corresponding relations; $q\xrightarrow{t}q'$ means that it is possible to execute t from the state q and the resulting state after its execution is q'. The state q' is said to be a *direct successor* of q. We denote by *enable* (t) the set of the states $\{q \mid \exists q'(q\xrightarrow{t}q')\}$.

Remark that in a transition system the relation $\rightarrow=\bigcup_{t\in T}\xrightarrow{t}$ is not total; there may exist *sink states* i.e. states q such that $\nexists q'\in Q\ (q\rightarrow q')$. We denote by $SINK$ the set of the sink states $(SINK=Q-\bigcup_{t\in T} enable\ (t))$.

* For a sequence $\sigma\in T^*$, $\sigma=t_{i_1}t_{i_2}\ldots t_{i_s}$, the notation $q\xrightarrow{\sigma}q'$ is used to represent the fact

$$\exists q_0 q_1 \ldots q_s,\ q=q_0,\ q'=q_s \quad \text{and} \quad q_{j-1}\xrightarrow{t_{ij}}q_j \quad \text{for } j=1, \ldots s.$$

* We use T^∞ to represent the set of the infinite sequences on T. A sequence $\sigma\in T^*\cup T^\infty$ is said to be *applicable* from a state q_0 (this fact is denoted by $q_0\xrightarrow{\sigma}$) if there exists a sequence of states, $q_0q_1\ldots q_i\ldots$ such that $q_0\xrightarrow{\sigma_i}q_i$ where σ_i is the prefix of σ of length i.

* A *computation sequence* from a state q_0 is a sequence σ of $T^*\cup T^\infty$ such that σ is applicable from q_0 and if σ is finite then $\exists q'(q_0\xrightarrow{\sigma}q'$ *and* $q'\in SINK)$.

* *An execution sequence* from a state q_0 is a sequence of states visited when a computation sequence is executed from q_0.

Obviously, every discrete system can be represented, at some level of abstraction, by a transition system. However, there exists a class of non-deterministic programs with guarded commands for which this representation is trivial. These programs are of the following form,

$S :: x := K;$

do $c_1\rightarrow a_1 \,[]\, \ldots \,[]\, c_m\rightarrow a_m$ **od**

where, $x=(x_1, \ldots, x_s)$ is the vector of state variables, K is a constant, the c_i's are conditions on x and the a_i's are tuples of assignments of the form

– $x_j := f(x)$, with f a (deterministic) function on x,
– $x_j :=$ **random** $P(x)$, where $P(x)$ is a set of possible values for x_j; the meaning of this assignment is that x_j takes an arbitrary value from this set.

A program of this form can be obviously considered as a transition system $(Q, T, \{\xrightarrow{t}\}_{t \in T})$ whose state space Q is contained in the cartesian product of the domains of the components of x and such that each transition t_i corresponds to the guarded command $c_i \rightarrow a_i$.

II.3. Fairness with Respect to Transitions

According to the informal characterization of fairness in II.1, if for an infinite computation sequence σ in a transition system, a transition t is enabled infinitely often then this transition must occur infinitely often in σ. Hence we have the definition.

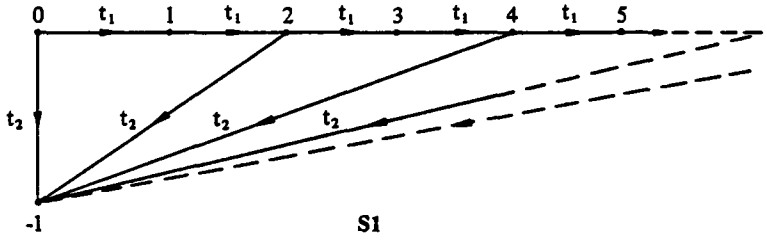
Definition 1 (fairness with respect to transitions). An infinite computation sequence σ, applicable from a state q_0, is unfair if there exists a transition t such that

– t has only a finite number of occurrences in σ, i.e. there exists σ', suffix of σ, $\sigma' \in (T-\{t\})^\infty$,
– during the execution of σ, t is enabled infinitely often, i.e. there exists an infinity of sequences σ'' such that σ'' is a prefix of σ and $q_0 \xrightarrow{\sigma'' t}$.

Consider the program PROG 1 and the transition system $S1$ representing it,

PROG 1 :: $x := 0$;

do
$t_1: x \geqq 0 \rightarrow x := x+1$ []
$t_2: x \equiv 0 \pmod 2 \rightarrow x := -1$
od



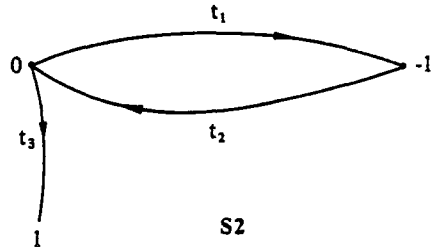
Under the introduced assumption of fairness, PROG 1 terminates because the only possible infinite computation sequence t_1^∞ is unfair.

The property of fairness derived from Definition 1 is *strong fairness* [13, 15, 17]. All the known types of fairness such as the *finite delay property* [11], *weak fairness* [7, 13, 15, 17] or *justice* [14] are particular cases of it.

One could expect that this property, as all the important correctness properties, be preserved by elementary syntactical transformations which do not alter program behaviour. Consider the program PROG2 represented by the transition system S2.

PROG2 :: $x := 0$;

do
$t_1: x = 0 \rightarrow x := x - 1$ ▯
$t_2: x < 0 \rightarrow x := x + 1$ ▯
$t_3: x = 0 \rightarrow x := x + 1$
od

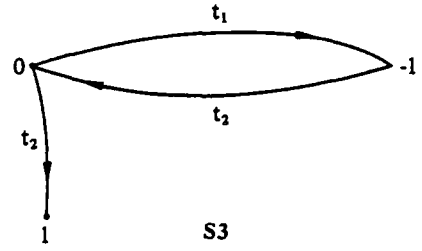


S2

This program terminates under the fairness assumption derived from Definition 1 (the unique infinite computation sequence $(t_1 t_2)^\infty$ is unfair with respect to t_3). Consider the program PROG3, represented by S3, which is obtained by merging the transitions t_2 and t_3 of PROG2 into one transition.

PROG3 :: $x := 0$;

do
$t_1: x = 0 \rightarrow x := x - 1$ ▯
$t_2: x \leqq 0 \rightarrow x := x + 1$
od



S3

Notice that under the given fairness assumption PROG3 does not terminate because the sequence $(t_1 t_2)^\infty$ is fair. From this remark one can conclude that either PROG2 and PROG3 are not equivalent or that the proposed definition is not sufficiently general to capture the concept of fairness.

Clearly, an equivalence relation allowing to distinguish PROG2 from PROG3 would be too strong. Futhermore the sequence $(t_1 t_2)^\infty$ in PROG3 is unfair by virtue of the informal definition given in II.1; during its execution the state 0 is visited infinitely often and only one of the two possible alternatives from this state is chosen. Thus, we introduce another type of fairness presented hereafter.

II.4 Fair Choice from States

Definition 2 (fair choice from states). An execution sequence of a transition system, $q_0 q_1 \ldots q_i \ldots$, is *unfair* if some state q_i occurs infinitely often and there exists a state q_i' such that $q_i \rightarrow q_i'$ and the subsequence $q_i q_i'$ has only a finite number of occurrences in this sequence.

That is, if an execution sequence is unfair then there exists a state q_i occuring infinitely often and a suffix of this sequence which does not contain any occurrence of the subsequence $q_i q_i'$, where q_i' is a direct successor of q_i.

According to the fairness assumption derived from this definition, PROG 3 terminates since the execution sequence $(0(-1))^{\infty}$ is unfair.

Notice that the two definitions correspond to two different notions of fairness because PROG 1 does not terminate under the assumption of fair choice from states as no state is visited infinitely often during the execution of t_1^{∞} while PROG 3 does not terminate under the assumption of fairness with respect to transitions.

Remark. One can wonder whether Definition 2 is still adequate in the case where a state has an infinite number of direct successors i.e. one of the relations $\{\xrightarrow{t}\}_{t\in T}$ is not image-finite.

A positive answer can be given by noticing that if a state q has an infinite number of successor $q_0, q_1, \ldots, q_i, \ldots$, then there exists a (fair) scheduling policy allowing to consider each one of the successors q_i infinitely often. For instance, such a policy is obtained by choosing successively,

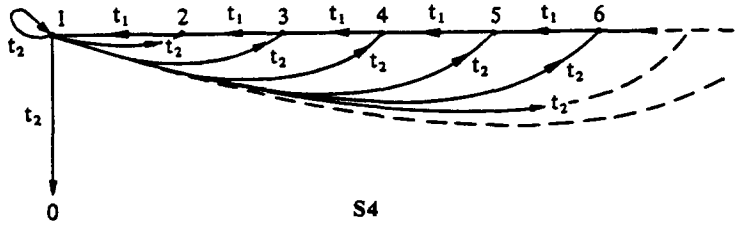
$$q_0, q_0, q_1, q_0, q_1, q_2, q_0, q_1, q_2, q_3, q_0, q_1, q_2, q_3, q_4, q_0, \cdots$$

each time the state q is visited.

Thus according to Definition 2 the program PROG 4, represented by the transition system S4, terminates within a finite but unbounded number steps.

PROG 4 :: $x := 1$;

do
$t_1 : x > 1 \rightarrow x := x - 1$ []
$t_2 : x = 1 \rightarrow x :=$ **random** $[0, +\infty[$
od

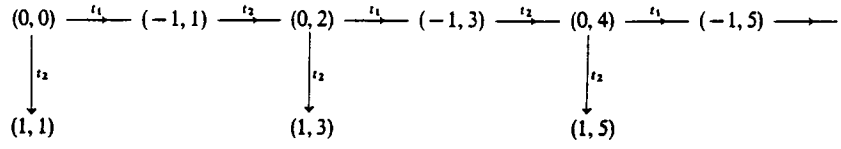


II.5. Fair Reachability of Predicates

Clearly, an adequate definition of fairness should cover both cases: a sequence should be considered as unfair if it is unfair according to Definition 1 or according to Definition 2. However, even after such an extension, the definition obtained is still not satisfactory. For example, transform PROG 3 by adjoining a counter variable y whose value represents the number of executions of its transitions. The program PROG 5 so obtained, is represented by the transition system S5.

PROG 5 :: $x := 0;\ y := 0;$

do
$t_1 : x = 0 \rightarrow x := x - 1;\ y := y + 1$ ▯
$t_2 : x \leqq 0 \rightarrow x := x + 1;\ y := y + 1$
od



S5

This program does not terminate under the fairness assumptions derived from Definitions 1 and 2. The only diverging computation sequence, $(t_1\, t_2)^\infty$ is fair according to Definition 1 and the corresponding execution sequence is fair according to Definition 2. However, y is independent of x and PROG 5 behaves exactly as PROG 3 if only the variable x is observed. Futhermore, the sequence $(t_1\, t_2)^\infty$ is unfair according to the informal definition because reaching a termination state is possible infinitely often during its execution and this event is never realized. For this reason we introduce the following type of fairness.

Definition 3 (fair reachability of predicates). An execution sequence s applicable from a state q_0 is unfair if there exists a predicate P on Q such that

– there exists an infinity of occurrences of states q_i in s such that for every q_i there exists a state q_i' and a sequence σ_i such that $q_i \xrightarrow{\sigma_i} q_i'$ and $q_i' \in P$.
– there exists only a finite number of occurrences of states of P in S.

Obviously, in program PROG 5 the sequence $(t_1\, t_2)^\infty$ is unfair according to this definition, given that there exists a set of states,

$$SINK = \{(1, 1), (1, 3), (1, 5), (1, 7), \ldots\},$$

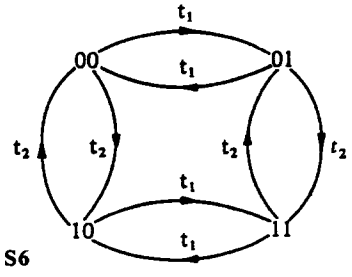
the set of termination states, which is reachable from every state visited during the execution of this sequence.

Definition 3 does not cover Definitions 1 and 2 as it is shown by the following examples:

a) Consider the program PROG 6 represented by S6.

PROG 6 :: $x := 0;\ y := 0;$

do
t_1 : **true** $\rightarrow x := (x + 1) \bmod 2$ ▯
t_2 : **true** $\rightarrow y := (y + 1) \bmod 2$
od



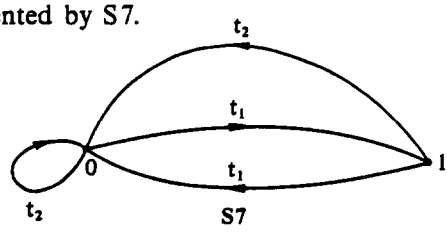
S6

The execution sequence corresponding to $(t_1\, t_2\, t_1\, t_2)^\infty$ is unfair according to definition 2 because the state (0, 0) is visited infinitely often and the subsequence (0, 0) (1, 0) does not occur. On the contrary, it is fair with respect to Definition 3 since all the states are visited infinitely often.

b) Consider the program PROG 7 represented by S7.

PROG 7 :: $x := 0$;

do
t_1 : **true** $\rightarrow x := (x+1) \bmod 2$ ▯
t_2 : **true** $\rightarrow x := 0$
od



The execution sequence corresponding to t_1^∞ is fair according to Definition 3 but t_1^∞ is not fair according to Definition 1.

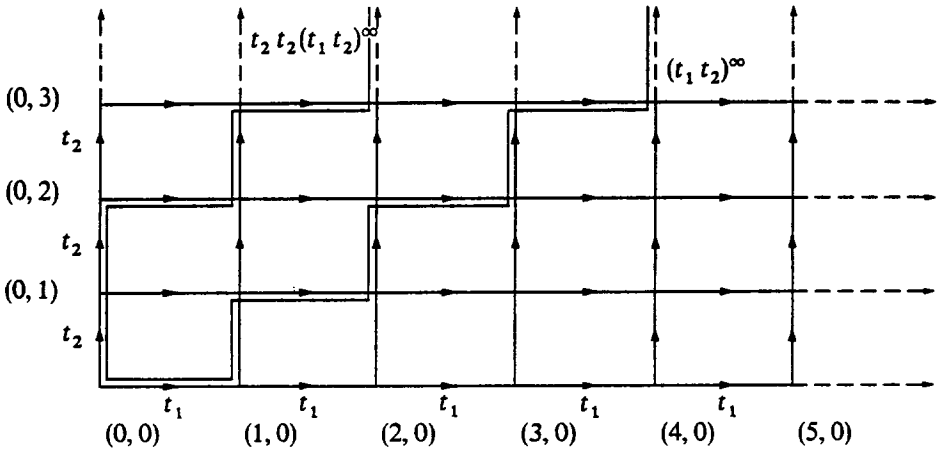
II.6. Relative Fairness

The presentation of the different types of fairness helps to realize that this phenomenon has many facets. It is now important to find a concept useful in practice.

The idea of considering that a sequence is fair if and only if it is fair according to all three presented definitions is not satisfactory for many reasons. One reason is that we are not sure to cover all the cases of fairness which are of practical interest. In fact, it will be shown later that, as a consequence of our informal definition, many other definitions of fairness can be proposed by considering different classes of events with respect to which unfair situations can arise. Another reason is that such a definition is too strong and it yields a concept of fairness not interesting in practice. Consider, for instance, the program PROG 8 represented by the transition system S8.

PROG 8 :: $x := 0$; $y := 0$;

do
t_1 : $x - y \leqq 1000 \rightarrow x := x+1$ ▯
t_2 : $y - x \leqq 1000 \rightarrow y := y+1$
od



S8

For any infinite computation sequence of PROG 8 it is possible to find a set of states P reachable from every state of the corresponding execution sequence and such that no state of P occurs in this sequence. For example, $(t_1 t_2)^\infty$ is unfair according to Definition 3 because from every state visited by this sequence, states of the execution sequence corresponding to $t_2 t_2 (t_1 t_2)^\infty$ are reachable.

This remark is rather disconcerting if we suppose that PROG 8 represents a controller of two processes, x and y being respectively the number of times each process has access to a service (the difference between access requests is always supposed to be less than 1000). It shows that for a given program some type of fairness may not be of practical interest; depending on what it is intended to do, some unfair situations are not considered in the properties expressing its specifications. For PROG 8, all its computation sequences are fair with respect to transitions and *this* is the important fairness property if this program represents a controller.

This example shows that instead of searching for some general definition of fairness it is important to "relativize" this concept. Depending on specifications only some type of fairness may be of importance. Futhermore, it is interesting to restrict Definitions 1 and 3 to a given transition t or to a given predicate respectively in order to obtain the notions of fairness w.r.t. t or w.r.t. P. For this, replace "if there exists a transition t (resp. predicate P on Q) such that" in these definitions, by "with respect to a given transition t (resp. predicate P on Q) if".

Relative fairness allows to focus on the possibility for a special class of events to occur. In particular, *fair termination* [9] can be defined as fairness with respect to HALT (HALT is the set of termination states) i.e. a program terminates fairly if all its infinite execution sequences are unfair with respect to HALT.

In this section we show that the different notions of fairness in a transition system S can be expressed as cases of fairness with respect to predicates in a transformed transition system S_T.

Given $S=(Q, T, \{\xrightarrow{t}\}_{t\in T})$ a transition system, we define the transition system $S_T=(Q_T, T, \{\overset{t}{\rightsquigarrow}\}_{t\in T})$ such that,

- $Q_T=\{(q_1, t, q_2)\in Q\times T\times Q \mid q_1 \xrightarrow{t} q_2\}$
- $(q_1, t, q_2) \overset{t'}{\rightsquigarrow} (q_1', t', q_2')$ iff $q_1'=q_2$ and $t'=t''$

Notice that S_T is "equivalent" to S in the sense that every computation sequence from (q_1, t, q_2) of S_T is a computation sequence from q_2 and conversely. Thus, every state (q_1, t, q_2) of Q_T is equivalent to the state q_2 with the difference that all its direct predecessors are states of the form $(-, -, q_1)$ i.e. states equivalent to q_1, and all the transitions leading to this state are t-transitions. Consequently, if during the execution of the same computation sequence in S and S_T starting from equivalent states, the state (q_1, t, q_2) is reached in S_T then the state q_2 is reached in S after executing t from state q_1. Hence, we have the propositions,

Proposition 1. *If a computation sequence σ is unfair according to Definition 1 in S then there exists a transition t such that every execution sequence s corresponding to σ in S_T is unfair with respect to the predicate,*

$$after(t) = \{(q_1, t', q_2) \in Q_T \mid t' = t\}.$$

Proposition 2. *If an execution sequence s is unfair according to Definition 2 in S then there exists two states q and q' of Q such that every execution sequence s' in S_T corresponding to s is unfair with respect to the predicate,*

$$between(q, q') = \{(q_1, t, q_2) \in Q_T \mid q = q_1, q' = q_2\}.$$

As a consequence of Propositions 1 and 2 it is possible to study the three introduced types of fairness in a unified manner. For this, it is necessary to transform the transition system under study S into an equivalent system S_T whose states memorize the immediate past. From a practical point of view this transformation can be done by adjoining to a program auxiliary variables so as to express the predicate with respect to which fairness is studied. In particular, the study of fairness with respect to predicates on Q does not require any transformation while the study of fairness with respect to transitions can be done by adjoining a boolean variable per transition.

Studying fairness properties of S as fair reachability of predicates of S_T, allows to better apprehend the underlying concept. Remark that there exist predicates on Q_T whose fair reachability property cannot be expressed by using Definitions 1-3 on S. For instance, consider predicates of the form,

$$P(q, t) = \{(q_1, t', q_2) \mid t = t', q_2 = q\},$$

which characterize the situations where the state q is reached after executing transition t. The existence of an execution sequence s' which is unfair with respect to $P(q, t)$ in S_T implies the existence of an execution sequence s in S which is unfair by virtue of the informal definition of II.1 because during its execution the event "q is reached after executing transition t" becomes possible infinitely often and it is realized only a finite number of times. Such unfairness situations cannot be captured by application of Definitions 1-3 on S.

In the same manner, other fairness properties of S can be defined by considering fair reachability of predicates in $(S_T)_T$ i.e. by considering fairness with respect to events depending on the present state and the last two computation steps. In fact, $q_1 \xrightarrow{t_1} q_2 \xrightarrow{t_2} q_3 \xrightarrow{t_3} q_4$ in S is equivalent to

$$((q_1, t_1, q_2), t_2, (q_2, t_2, q_3)) \overset{t_3}{\rightarrow} ((q_2, t_2, q_3), t_3, (q_3, t_3, q_4)) \quad \text{in } (S_T)_T.$$

This approach can be generalized to obtain any fairness property of S by considering for $i \in \mathbb{N}$, the reachability of predicates in $S_T i$, $S_T i = (S_T i - 1)_T$ with $S_{T0} = S$, whose states memorize the last i computation steps. For example, in the controller represented by PROG 8, consider for a given integer i the functioning corresponding to all the sequences which are fair in $S_T i$ with respect to all the predicates P_σ where P_σ is a predicate on $Q_T i$ satisfied by the states representing an event of the form,

$$q_1 \xrightarrow{t_{j_1}} \dots q_i \xrightarrow{t_{j_i}} q_{i+1}, \quad \text{where } \sigma = t_{j_1} t_{j_2} \dots t_{j_i} \in (t_1 + t_2)^*.$$

The infinite computation sequences in this functioning contain infinitely often all the sequences of length i whose execution is possible infinitely often. For $i=3$ the sequence $(t_1 t_2)^\infty$ is not a possible sequence of this functioning while the sequence $(t_1 t_1 t_1 t_2 t_2 t_2 t_1 t_2)^\infty$ is one since it contains infinitely often all the sequences of lenght 3 on $\{t_1, t_2\}$.

Finally, notice that considering the sequences which are fair with respect to all P_σ for $\sigma \in (t_1 + t_2)^*$ leads to some notion of *absolute fairness* with respect to t_1 and t_2. These sequences contain infinitely often any finite sequence which is possible infinitely often during their executions.

As a conclusion to part II, we summarize the approach followed in the search for a satisfactory definition of fairness and the main conclusions:

– Following the informal definition of II.1, we have tried to find a definition of fairness which is sufficiently general and such that this property be preserved by some elementary transformations which do not alter program behaviour.
– The types of fairness considered until now, in fact fairness with respect to transitions, do not allow to characterize all the situations of practical interest. Furthermore, this type of fairness is too sensitive to elementary transformations giving equivalent programs.
– It is possible to define a big variety of fairness properties according to the class of events considered. Program specifications define which of them are significant correctness properties. This fact brought us along to relativize this concept.
– The different types of fairness for a given transition system S can be studied as properties of fair reachability of a predicate P in a transition system S' obtained by transforming adequately S. Thus, the important notion is fair reachability of predicates. Only this type of fairness is considered in the sequel.

III. A Temporal Logic to Deal with Fairness

In this section a logic for dealing with the properties of the fair functioning of transition systems is given. The expressibility of this logic is compared with the expressibility of existing branching time logics. The main result is that there exists a *conditional branching time logic CL* [1, 4] such that for a given transition system (model) S and for each formula f' of the introduced logic, there exists a formula f of CL for which: f' is valid for S iff f is valid for S in CL; i.e. f' under the assumption of fairness is equivalent to f. This result allows the application to the introduced logic of well-known results for CL (decision procedure, fixedpoint characterization of the temporal operators).

III.1. The Conditional Branching Time Logic CL

We introduce a conditional branching time logic CL obtained by augmenting the propositional calculus in the following manner.

The formulas of CL are built from a set of propositional variables V and the constants **true** and **false** by using the logical connectives $\vee$, $\wedge$, $\neg$, $\Rightarrow$ and two binary temporal operators POT and $INEV$.

Each one of the arguments of the temporal operators play very different roles; in order to better distinguish them, we prefer writing $POT[f_1](f_2)$ and $INEV[f_1](f_2)$ instead of $POT(f_1, f_2)$ and $INEV(f_1, f_2)$. Also, the abbreviations $ALL[f_1](f_2)$ and $SOME[f_1](f_2)$ are used for respectively $\neg POT[f_1](\neg f_2)$ and $\neg INEV[f_1](\neg f_2)$. The class of the models for CL is defined in terms of transition systems as follows:

* Given a transition system $S=(Q, T, \{\xrightarrow{t}\}_{t\in T})$, represent by $EX(q)$ the set of all the execution sequences starting from state q. In order to simplify the notations, for a sequence s of $EX(q)$ we represent by $s(k)$ the k-th element of it, if it is defined; if not, we take $s(k)=\omega$ where ω represents some fictitious non accessible state adjoined to Q. Thus, the relation $s(0)\xrightarrow{k}s(k)$ is satisfied iff $s(k)\neq\omega$.

* Given CL and a transition system $S=(Q, T, \{\xrightarrow{t}\}_{t\in T})$, an interpretation of CL is a function $|\ |$ associating to each formula of CL a subset of Q such that:

- $|\mathbf{true}|=Q$
- $\forall f\in CL, |\neg f|=Q-|f|$
- $\forall f_1, f_2\in CL, |f_1\wedge f_2|=|f_1|\cap|f_2|$
- $\forall f_1, f_2\in CL,$

$$q\in|POT[f_1](f_2)|\equiv\exists s\in EX(q)\,\exists k\in\mathbb{N}\left[q\xrightarrow{k}s(k)\ \textbf{and}\ \overset{k-1}{\underset{i=0}{\forall}}\, s(i)\in|f_1|\ \textbf{and}\ s(k)\in|f_2|\right]$$

$$q\in|INEV[f_1](f_2)|\equiv\forall s\in EX(q)\,\exists k\in\mathbb{N}\left[q\xrightarrow{k}s(k)\ \textbf{and}\ \overset{k-1}{\underset{i=0}{\forall}}\, s(i)\in|f_1|\ \textbf{and}\ s(k)\in|f_2|\right]$$

Obviously, $|POT[f_1](f_2)|$ represents the set of the states q of S such that there exists an execution sequence from q containing some state q' which belongs to $|f_2|$, and all the states between q and q' (excluding q') belong to $|f_1|$. We say that $|POT[f_1](f_2)|$ is the set of the states from which (some state of) $|f_2|$ is *potentially reachable under the condition* $|f_1|$. In the same way, $|INEV[f_1](f_2)|$ is the set of the states from which $|f_2|$ is *inevitably reachable under the condition* $|f_1|$ in the sense that every execution sequence, starting from a state q of this set, contains some state q' of $|f_2|$ and all the states between q and q' (excluding q') belong to $|f_1|$.

The interpretation of the dual operators ALL and $SOME$ is,

$\forall f_1, f_2\in CL,$

$$q\in|ALL[f_1](f_2)|\equiv\forall s\in EX(q)\,\forall k\in\mathbb{N}\left[q\xrightarrow{k}s(k)\ \textbf{and}\ \overset{k-1}{\underset{i=0}{\forall}}\, s(i)\in|f_1|\ \textbf{implies}\ s(k)\in|f_2|\right]$$

$$q\in|SOME[f_1](f_2)|\equiv\exists s\in EX(q)\,\forall k\in\mathbb{N}\left[q\xrightarrow{k}s(k)\ \textbf{and}\ \overset{k-1}{\underset{i=0}{\forall}}\, s(i)\in|f_1|\ \textbf{implies}\ s(k)\in|f_2|\right]$$

$|ALL[f_1](f_2)|$ represents a set of states such that if q belongs to $|f_1|$ then every possible successor of q belong to $|f_2|$ *as long as* f_1 remains true. In an analoguous manner, $|SOME[f_1](f_2)|$ represents a set of states such that if q belongs to $|f_1|$ then there exists an execution sequence applicable from q whose states belong to $|f_2|$ *as long as* f_1 remains true. The formulas $\neg f_1\vee ALL[f_1](f_2)$ and $\neg f_1\vee SOME[f_1](f_2)$ express for branching time logic a

similar notion as the **until** operator for linear time logic [10, 19]; both of them express the fact that "f_2 remains true until f_1 becomes false".

The logic CL constitutes a natural generalization of the branching time logic L [18] obtained by adjoining to the propositional calculus the unary temporal operators POT and $INEV$ such that $POT = \lambda f \cdot POT[\mathbf{true}](f)$, $INEV = \lambda f \cdot INEV[\mathbf{true}](f)$. The logic CL has been introduced in [1] and studied in [4] where a decision procedure is given. Moreover, a logic similar to L has been studied in [3].

II.2. Interpretation of the Temporal Operators as Invariants and Trajectories

The following results constitute a generalization of well-known results for L [21, 18] and show that the interpretation of the temporal operators of CL can be expressed as least or greatest fixed points of predicate transformers and correspond to the well-known notions of invariant and trajectory. Reasoning in terms of these notions makes easier the understanding of the properties of CL. A similar fixed point characterization of the operators of CL is given in [4].

Let $S = (Q, T, \{\xrightarrow{t}\}_{t \in T})$ be a transition system. Represent by $(2^Q, \cup, \cap, \bar{})$ the lattice of the subsets of Q and $[2^Q \to 2^Q]$ the set of the internal mappings of 2^Q. For $h, g \in [2^Q \to 2^Q]$, $h \cup g$, $h \cap g$, $\bar{g}, \tilde{g}$ and Id denote the functions $h \cup g = \lambda x \cdot h(x) \cup g(x)$; $h \cap g = \lambda x \cdot h(x) \cap g(x)$, $\bar{g} = \lambda x \cdot \overline{g(x)}$, $\tilde{g} = \lambda x \cdot \bar{g}(\bar{x})$, $Id = \lambda x \cdot x$. We also introduce the notations $gfpx \cdot g(x)$ and $lfpx \cdot g(x)$ to denote the greatest and the least fixed point of the monotonic function $g = \lambda x \cdot g(x)$.

Definition. Given $S = (Q, T, \{\xrightarrow{t}\}_{t \in T})$, $P \subseteq Q$ and $q \in Q$, *pre* is a function of $[2^Q \to 2^Q]$ such that,

$q \in pre(P) \equiv \exists\, q' (q \to q' \text{ and } q' \in P)$.

Definitions. Given a transition system $S = (Q, T, \{\xrightarrow{t}\}_{t \in T})$ and $C \subseteq Q$,

a) an *invariant* of S is a subset J of Q such that

$$\forall\, q, q' \in Q\, (q \in J \text{ and } q \to q' \text{ implies } q' \in J)$$

b) a *conditional invariant of S under the condition C* is a subset J of Q such that,

$$\forall\, q, q' \in Q\, (q \in J \cap C \text{ and } q \to q' \text{ implies } q' \in J)$$

c) a *trajectory* of S is a subset W of Q such that,

$$\forall\, q \in Q\, (q \in W - SINK \text{ implies } \exists\, q' (q \to q' \text{ and } q' \in W))$$

d) a *conditional trajectory of S under the condition C* is a subset W of Q such that,

$$\forall\, q \in Q\, (q \in W \cap C - SINK \text{ implies } \exists\, q' (q \to q' \text{ and } q' \in W))$$

Proposition 3. *Given a transition system S and a subset C of Q,*
a) J is an invariant iff $J = J \cap \tilde{pre}(J)$
b) J is a conditional invariant under C iff $J = J \cap (\bar{C} \cup \tilde{pre}(J))$.

Proposition 4. *Given a transition system S and a subset C of Q,*
a) W is a trajectory iff $W = W \cap (pre(W) \cup \tilde{pre}(W))$
b) W is a conditional trajectory under C iff

$$W = W \cap (\bar{C} \cup pre(W) \cup \tilde{pre}(W)).$$

Proposition 5. *Let* f_1, f_2 *be two formulas of CL, S a transition system and* $|\ |$ *an interpretation of CL in S.*

a) $|POT[f_1](f_2)| = lfp\, x \cdot |f_2| \cup |f_1| \cap pre(x)$ *and dually,*
$|ALL[f_1](f_2)| = gfp\, x \cdot |f_2| \cap (|\neg f_1| \cup \tilde{pre}(x))$

b) $|INEV[f_1](f_2)| = lfp\, x \cdot |f_2| \cup |f_1| \cap (pre(x) \cap \tilde{pre}(x))$ *and dually,*
$|SOME[f_1](f_2)| = gfp\, x \cdot |f_2| \cap (|\neg f_1| \cup pre(x) \cup pre(x))$.

According to Proposition 5, $|ALL[f_1](f_2)|$ and $|SOME[f_1](f_2)|$ are respectively the greatest conditional invariant and trajectory under $|f_1|$ contained in $|f_2|$. In particular, $|ALL[\mathbf{true}](f_2)|$ and $|SOME[\mathbf{true}](f_2)|$ are respectively the greatest invariant and trajectory contained in $|f_2|$. These results suggest a proof method in *CL* by iterative evaluation of the formulas, which is used in [18].

III.3. The Logic FCL

III.3.1. Preliminary Results. Given a transition system *S*, a state q_0 of *S*, and a countable set *A* of subsets of *Q*, we define $FEX(q_0, A)$ to be the set of the execution sequences starting from q_0 and which are fair with respect to any member *a* of *A* i.e. $FEX(q_0, A) = \bigcap_{a \in A} FEX(q_0, \{a\})$.

Proposition 6. *For every transition system S, any state* q_0 *of it, and any countable set A of subsets of Q, the set* $FEX(q_0, A)$ *is not empty.*

Proof. a) Suppose that *A* is finite, $A = \{a_1, \ldots, a_s\}$. Then a computation sequence σ, fair with respect to any member of *A*, can be iteratively computed as follows,

$q := q_0$; $\sigma := \lambda$; (λ is the empty word of T^*)
while true **do**
  for $j := 1$ **to** $|A|$ **do**
    if $\exists\, q_j$ and σ_j such that $q \xrightarrow{\sigma_j} q_j$ and $q_j \in a_j$
      then $q := q_j$; $\sigma := \sigma\sigma_j$
    fi;
    if $q \in SINK$ **then stop fi**
  od
od

Let $\sigma = \sigma_{11}\sigma_{12} \cdots \sigma_{1|A|}\sigma_{21}\sigma_{22} \cdots \sigma_{2|A|} \cdots \sigma_{k_1} \cdots \sigma_{k|A|} \cdots$

the sequence thus defined where σ_{w_r} represents the sequence concatenated to the variable σ for $j = r$ during the *w*-th iteration (we take $\sigma_{w_r} = \lambda$ if the condition "$\exists q_r$ and σ_r such that $q \xrightarrow{\sigma_r} q_r$ and $q_r \in a_r$" is not satisfied).

The sequence σ is fair with respect to any element of A: suppose that some set a_u is reachable infinitely often during the execution of σ but its states are not infinitely often visited. Then there exists some integer k such that $\forall m,\ m \geqq k$ *implies* $\sigma_{mu} = \lambda$; futhermore, the state q_{u-1} reached after the application of the sequence,

$$\sigma_{11} \cdots \sigma_{1|A|} \cdots \sigma_{k_1} \cdots \sigma_{ku-1}$$

is such that it is not possible to reach from it any state of a_u. This implies that from every possible successor of q_{u-1} it is not possible to reach some state of a_u. Thus σ is fair with respect to a_u (contradiction).

b) Suppose that A has an infinity of elements. In this case a fair computation sequence σ with respect to any member of A, can be defined in the following manner.

$q := q_0$; $\sigma := \lambda$; (λ is the empty word of T^*)

for $i := 1$ **to** infinity **do**
  for $j := 1$ **to** i **do**
    if $\exists q_j$ and σ_j such that $q \xrightarrow{\sigma_j} q_j$ and $q_j \in a_j$
      then $q := q_j$; $\sigma := \sigma\sigma_j$
    fi;
    if $q \in SINK$ **then stop fi**
  od
od

Let $\sigma_{11}\sigma_{21}\sigma_{22}\sigma_{31}\sigma_{32}\sigma_{33} \cdots \sigma_{k_1}\sigma_{k_2} \cdots \sigma_{kk} \cdots$

the sequence thus defined where σ_{wr} represents the sequence concatenated to σ for $i = w$ and $j = r$. For the same reasons as in the case where A is finite, σ is fair with respect to any member of A. □

Corollary. *Given a countable set A of subsets of Q, $q_0 \in Q$ and an execution sequence s, $s \notin FEX(q_0, A)$, it is possible for any integer k to find a sequence s', $s' \in FEX(q_0, A)$ such that the sequence $s(0) \ldots s(k)$ is a prefix of s'.*

III.3.2. Obtaining the logic FCL. Given CL and an interpretation of it, a formula f represents a property of both the fair and unfair sequences of a transition system. If one is interested in the property expressed by this formula f when restricting to fair functioning, the following approach can be adopted.

Let f be a formula of CL written in such a form that only the temporal operators POT and $INEV$ occur in it and let $\{f_1, \ldots f_s\}$ be the set of the subformulas of f which are second arguments of POT and $INEV$. We put $A(f) = \{|f_i|\}_{i=1}^{s}$.

Given that POT and $INEV$ express reachability of their second argument, we try to obtain a formula f', expressing the same property as f under the assumption of fairness, by restricting to the functioning corresponding to the set of the execution sequences which are fair with respect to any member of $A(f)$. To do this, we define for any countable set A of subsets of Q, and any pair of formulas f_1, f_2 of CL such that $|f_2| \in A$, the operators $F_A POT$ and $F_A INEV$ such that,

$$q \in |F_A POT[f_1](f_2)|$$
$$\equiv \exists s \in FEX(q,A)\, \exists k \in \mathbb{N} \left[q \xrightarrow{k} s(k) \quad \textbf{and} \quad \overset{k-1}{\underset{i=0}{\forall}} s(i) \in |f_1| \quad \textbf{and} \quad s(k) \in |f_2| \right]$$

$$q \in |F_A INEV[f_1](f_2)|$$
$$\equiv \forall s \in FEX(q,A)\, \exists k \in \mathbb{N} \left[q \xrightarrow{k} s(k) \quad \textbf{and} \quad \overset{k-1}{\underset{i=0}{\forall}} s(i) \in |f_1| \quad \textbf{and} \quad s(k) \in |f_2| \right]$$

i.e. these operators express respectively the fact that $|f_2|$ is potentially or inevitably reachable under $|f_1|$ by using only the sequences which are fair with respect to any member of A.

The formula f' expressing the same property as f under the assumption of fairness, is obtained by uniform substitution of the operators POT and $INEV$ in f by the operators $F_{A(f)}POT$ and $F_{A(f)}INEV$. This results from the observation that the temporal operators POT and $INEV$ express reachability of their second argument. Thus, the formula f' expressing the same property as f is obtained by restricting the interpretation of the operators to the set of the fair sequences (according to Definition 3) with respect to any predicate which is second argument of them.

Proposition 7. *For any pair of formulas f_1, f_2 of CL and any countable set A of subsets of Q such that $|f_2| \in A$,*

$$|F_A POT[f_1](f_2)| = |POT[f_1](f_2)|.$$

Proof. Given that $FEX(q,A) \subseteq EX(q)$, we have $F_A POT[f_1](f_2) \Rightarrow POT[f_1](f_2)$.

Suppose that $\exists q \in Q$ $q \in |POT[f_1](f_2)|$ and $q \notin |F_A POT[f_1](f_2)|$. This implies that there exists a sequence s, $s \notin FEX(q,A)$ and $s \in EX(q)$ such that,

$$\exists k \in \mathbb{N} \left[q \xrightarrow{k} s(k) \textbf{ and } \overset{k-1}{\underset{i=0}{\forall}} s(i) \in |f_1| \textbf{ and } s(k) \in |f_2| \right].$$

Then, it is possible to find, according to the corollary of Proposition 6, a sequence $s' = s(0) \ldots s(k)\, s''$ such that s' is fair with respect to each one of the members of A. This implies $q \in |F_A POT[f_1](f_2)|$ (contradiction). □

Proposition 8. *For any pair of formulas f_1 and f_2 of CL and any countable set A of subsets of Q such that $|f_2| \in A$,*

$$|F_A INEV[f_1](f_2)| = |F_{\{|f_2|\}} INEV[f_1](f_2)|$$

Proof. $FEX(q,A) \subseteq FEX(q, \{|f_2|\})$ because $|f_2| \in A$; consequently we have,

$$F_{\{|f_2|\}} INEV[f_1](f_2) \Rightarrow F_A INEV[f_1](f_2).$$

Suppose that for some state q, $q \in |F_A INEV[f_1](f_2)|$ and $q \notin |F_{\{|f_2|\}} INEV[f_1](f_2)|$. This implies that there exits a sequence s, $s \in FEX(q, \{|f_2|\})$ such that,

$$\forall k \in \mathbb{N} \left[q \xrightarrow{k} s(k) \textbf{ implies } \left(\overset{k-1}{\underset{i=0}{\exists}} s(i) \in |\neg f_1| \textbf{ or } s(k) \in |\neg f_2| \right) \right],$$

and there exists a member of $A-\{|f_2|\}$ with respect to which s is not fair. Thus s is an infinite sequence.

a) Suppose that $\forall i \in \mathbb{N}\ s(i) \in |f_1|$. This implies $\forall i \in \mathbb{N}\ s(i) \in |\neg f_2|$.

But since s is fair with respect to $|f_2|$ there exists some $j_0 \in \mathbb{N}$ such that $\forall j, j \geqq j_0$ implies $s(j) \in |\neg POT(f_2)|$. Then, according to the corollary of Proposition 6, it is possible to find a sequence s', fair with respect to $A-\{|f_2|\}$ starting from $s(j_0)$. The sequence $s(0) \ldots s(j_0)\, s'$ is fair with respect to any element of A and all its elements satisfy $\neg f_2$. Contradiction, since we supposed that $q \in |F_A INEV[f_1](f_2)|$.

b) Suppose that $\exists i \in \mathbb{N}\ s(i) \in |\neg f_1|$ and put $i_0 = \text{Min}\,\{i | s(i) \in |\neg f_1|\}$.

Then, for $0 \leqq i < i_0$ $s(i) \in |f_1|$ and for $0 \leqq i \leqq i_0$ $s(i) \in |\neg f_2|$.

Consider a sequence $s(0) \ldots s(i_0)\, s'$, where s' is such that it is fair with respect to any member of A. The fact that this sequence belongs to $FEX(q, A)$, $s(i_0) \in |\neg f_1|$ and $s(i) \in |\neg f_2|$ for $0 \leqq i \leqq i_0$ contradicts $q \in |F_A INEV[f_1](f_2)|$. □

The results of Propositions 7 and 8 considerably simplify the method for obtaining a formula f' expressing the same property as a formula f of CL under the assumption of fairness (f is supposed to be written in such a form that the operator $SOME$ does not occur in it). For this, one has to substitute every occurrence of $INEV[f_1](f_2)$ by $F_{(|f_2|)} INEV[f_1](f_2)$.

The simpler notation $FINEV[f_1](f_2)$ will be adopted in the sequel for $F_{(|f_2|)} INEV[f_1](f_2)$. We call FCL the logic obtained by adjoining to the propositional calculus the binary temporal operators POT and $FINEV$, their duals being respectively denoted by ALL and $FSOME$. Also, we call FL the sublogic of FCL constructed from the unary temporal operators $POT = \lambda f \cdot POT[\mathbf{true}](f)$ and $FINEV = \lambda f \cdot FINEV[\mathbf{true}](f)$.

Remark. Due to the result of Proposition 8 and by dualization one obtains,

$$q \in |FSOME[f_1](f_2)|$$

$$\equiv \exists s \in FEX(q, \{|\neg f_2|\})\ \forall k \in \mathbb{N} \left[q \xrightarrow{k} s(k) \quad \textbf{and} \quad \overset{k-1}{\underset{i=0}{\forall}} s(i) \in |f_1| \quad \textbf{implies} \quad s(k) \in |f_2| \right].$$

Notice that for the definition of the interpretation of $FSOME[f_1](f_2)$ only the fair execution sequences with respect to $|\neg f_2|$ are considered. This is not surprising because $FSOME[f_1](f_2)$ express the possibility of "remaining fairly in $|f_2|$" i.e. without using unfair sequences with respect to $|\neg f_2|$ in order to remain in $|f_2|$.

III.4. Comparison of the Logics CL and FCL

In this section FCL and CL are compared with respect to their capabilities to express properties of transition systems. This comparison is made in a progressive manner by considering successively FL and L, FL and CL, FCL and CL. To do this, we consider the logics $CL(V)$ and $FCL(V)$ defined on the same set of propositional variables V and such that the restrictions of their interpretation functions agree on V.

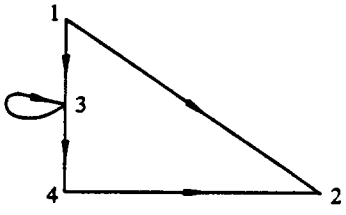
The problem studied is the search for a meaning preserving homomorphism μ of FCL into CL i.e. a function μ such that if $\mu(f_1)=f_2$ then the statements "f_1 is valid in FCL for S" and "f_2 is valid in CL for S" are equivalent (for any S). This means that the interpretations of f_1 and $\mu(f_1)$ are the same for every f_1 of FCL ($|f_1|=|\mu(f_1)|$). Obviously, if such a function μ exists, then proving the validity of a formula f_1 in FCL amounts to proving the validity of $\mu(f_1)$ in CL.

In order to simplify the notations, we omit the interpretation function and we use subsets of Q instead of formulas, whenever there is no risk of confusion. For example, we write $POT[f_1](f_2)$ to represent the set of the states $|POT[f_1](f_2)|$.

III.4.1. Comparison of $L(V)$ and $FL(V)$. Consider $L(V)$ and $FL(V)$, the sublogics obtained from $CL(V)$ and $FCL(V)$ by taking the conditions of the temporal operators to be equal to **true** (*POT*, *INEV*, *FINEV* stand respectively for POT[**true**], $INEV$[**true**], $FINEV$[**true**]).

Proposition 9. *The operator $FINEV$ of $FL(V)$ cannot be expressed in $L(V)$.*

Proof. Consider the following transition system. If $FINEV$ can be expressed by means of some formula of $L(V)$, then the set $FINEV\{4\}=\{3,4\}$ can be ob-



S9

tained from $\{4\}$ by carrying out a finite number of times the operations $\wedge$, $\neg$, POT and $INEV$. But this is not possible as shown hereafter.

1) From $\{4\}$ by carrying out logical operations, one obtains the sets,

$$\emptyset, \{4\}, \{1,2,3\}, \{1,2,3,4\}.$$

2) The applications of POT and $INEV$ to these sets respectively gives,

$$\begin{array}{l}\emptyset, \{1,3,4\}, \{1,2,3,4\}, \{1,2,3,4\}\\ \emptyset, \{4\}\quad\ , \{1,2,3,4\}, \{1,2,3,4\}\end{array}$$

3) The combination of the obtained sets via logical operators gives,

$$\{2\}, \{4\}, \{1,3\}, \{2,4\}, \{1,2,3\}, \{1,3,4\}, \{1,2,3,4\}.$$

No new set can be obtained by application of POT and $INEV$. □

The following proposition gives upper and lower approximations of $FINEV(f)$ by formulas of $L(V)$.

Proposition 10. *For any set of states f,* $INEV(f) \vee ALLPOT(f) \Rightarrow FINEV(f)$ *and* $FINEV(f) \Rightarrow INEV(f) \vee SOMEPOT(f)$.

Proof

a) $INEV(f) \vee ALLPOT(f) \Rightarrow FINEV(f)$.

Obviously, $INEV(f) \Rightarrow FINEV(f)$.
Suppose that $q \in ALLPOT(f)$ and $q \notin FINEV(f)$.
$q \notin FINEV(f)$ is equivalent to $q \in FSOME(\neg f)$ which means that there exists an execution sequence s, starting from q, whose states belong to $\neg f$ and which is fair with respect to f. Remark that, due to the fact that $q \in ALLPOT(f)$, s is infinite, since there is no sink state in $POT(f) \wedge \neg f$. Thus, s is not fair with respect to f as all its states belong to $POT(f)$ (contradiction).

b) $FINEV(f) \Rightarrow INEV(f) \vee SOMEPOT(f)$.

Suppose that for some state q, $q \in FINEV(f)$ and $q \notin INEV(f)$. This is equivalent to $q \notin FSOME(\neg f)$ and $q \in SOME(\neg f)$. Thus, there is no fair execution sequence with respect to f, starting from q and contained in $\neg f$ but there is an execution sequence s starting from q and contained in $\neg f$. Thus, s is unfair with respect to f. So, it is possible to reach f from every state of s and consequently all its states belong to $POT(f)$. $SOMEPOT(f)$ being the greatest trajectory in $POT(f)$ all the states of s belong to it. In particular, $q \in SOMEPOT(f)$. □

Remark. The implications are strict due to Proposition 9. For the transition system given in its proof we have for $f = \{4\}$

$$INEV(f) = \{4\},\ ALLPOT(f) = \emptyset,\ FINEV(f) = \{3,4\}\ SOMEPOT(f) = \{1,3\}.$$

Corollary. *For any set of states f,*
a) If $SOMEPOT(f) \Rightarrow INEV(f)$ *then* $FINEV(f) \equiv INEV(f)$
b) If $FINEV(f) \equiv INEV(f)$ *then* $ALLPOT(f) \Rightarrow FINEV(f)$.

Proposition 11. *For every transition system S and any set of states f of S,* $\neg SOME(\neg f \wedge POT(f))$ *is a valid formula iff every sequence is fair with respect to f.*

Proof. It is sufficient to show that,

$\exists q \in SOME(\neg f \wedge POT(f))$ iff there exists some sequence unfair with respect to f.

a) If there exists $q \in SOME(\neg f \wedge POT(f))$ then the trajectory $SOME(\neg f \wedge POT(f))$ is non-terminating because there is no sink state in $\neg f \wedge POT(f)$. Thus, there exists an infinite execution sequence whose states do not belong to f but f is reachable from every state of this sequence. So, it is unfair with respect to f.

b) If there exists an unfair execution sequence s with respect to f, it is possible to find some suffix s' of it such that the corresponding trajectory is in $\neg f$. The sequence s' is also unfair with respect to f and all its states belong to $POT(f)$. Thus, the greatest trajectory contained in $\neg f \wedge POT(f)$ is a non-empty set. □

III.4.2. Comparison of $CL(V)$ and $FL(V)$

Lemma 1. *For any set of states f, $INEV(f) \Rightarrow ALL[\neg f](POT(f))$.*

Proposition 12. For any set of states f, $FINEV(f) \equiv ALL[\neg f](POT(f))$.

Proof

a) $ALL[\neg f](POT(f)) \Rightarrow FINEV(f))$.

Suppose that $q \in ALL[\neg f]\, POT(f)$ and $q \notin FINEV(f)$.
$q \notin FINEV(f)$ is equivalent to $q \in FSOME(\neg f)$, which means that there exists an execution sequence s, fair with respect f, starting from q, the states of which are in $\neg f$. Remark that s is infinite because as long as f is not reached the system is at some state of $\neg f \wedge POT(f)$.

For s to be fair with respect to f, there must be a suffix s' of it, the states of which do not belong to $POT(f)$. This means that there exists an execution sequence in $\neg f$ starting from q and leading to some state of $\neg POT(f)$. But this fact contradicts $q \in ALL[\neg f]\, POT(f)$, equivalent to $q \notin POT[\neg f]\, \neg POT(f)$, which means that $\neg POT(f)$ is not reachable from q under the condition $\neg f$.

b) $FINEV(f) \Rightarrow ALL[\neg f]\, POT(f)$.

Suppose that for some q, $q \in FINEV(f)$, and $q \notin ALL[\neg f]\, POT(f)$. This implies $q \notin FSOME(\neg f)$ and $q \in SOME(\neg f)$ (by Lemma 1) which means that there are only unfair sequences with respect to f, starting from q and having all their states in $\neg f$.

This contradicts $q \notin ALL[\neg f]\, POT(f)$ which is equivalent to $q \in POT[\neg f]\, ALL(\neg f)$ *i.e.* there exists a finite sequence s, starting from q, leading to some state of $ALL[\neg f]$ and whose all states are in $\neg f$. Every sequence having s as prefix is fair with respect to f. □

This result shows that uniform substitution of $FINEV(f)$ and $FSOME(f)$ in a formula of $FL(V)$ by respectively $ALL[\neg f]\, POT(f)$ and $POT[f]\, ALL(f)$ gives an equivalent formula of $CL(V)$. As a consequence we have that $FL(V)$ is equivalent to the sub-logic of $CL(V)$ generated by adjoining to the predicate calculus on V the operators $\lambda f \cdot POT(f)$ and $\lambda f \cdot ALL[\neg f]\, POT(f)$.

III.4.3. Comparison of $CL(V)$ and $FCL(V)$

Lemma 2. *For any pair of sets of states f_1 and f_2, $INEV[f_1](f_2) \Rightarrow ALL[\neg f_2]\, POT[f_1](f_2)$.*

Proposition 13. *For any pair of sets of states f_1 and f_2, $FINEV[f_1](f_2) \equiv ALL[\neg f_2]\, POT[f_1](f_2)$.*

Proof

a) $ALL[\neg f_2]\, POT[f_1](f_2) \Rightarrow FINEV[f_1](f_2)$

Suppose that for some state q, $q \in ALL[\neg f_2]\, POT[f_1](f_2)$ and $q \notin FINEV[f_1](f_2)$.
$q \notin FINEV[f_1](f_2)$ is equivalent to $q \in FSOME[f_1](\neg f_2)$ which means that there exists an execution sequence s fair with respect to f_2, starting from q, the

states of which are in $\neg f_2$. Remark that s is infinite because $\neg f_2 \wedge POT[f_1](f_2)$ does not contain sink states. For s to be fair with respect to f_2, it must have a suffix all the states of which belong to $\neg POT(f_2)$. Thus,

$$q \in POT[\neg f_2] \neg POT(f_2)$$

which implies,

$$q \in POT[\neg f_2] \neg POT[f_1](f_2)$$

and this contradicts $q \in ALL[\neg f_2] POT[f_1](f_2)$.

b) $FINEV[f_1](f_2) \Rightarrow ALL[\neg f_2] POT[f_1](f_2)$

Suppose that, $q \in FINEV[f_1](f_2)$ and $q \notin ALL[\neg f_2] POT[f_1](f_2)$ for some state q. This implies,

$$q \notin FSOME[f_1](\neg f_2) \quad \text{and} \quad q \in SOME[f_1](\neg f_2) \quad \text{(by Lemma 2)}$$

which means that all the conditional trajectories under f_1 which are contained in $\neg f_2$ and start from q, are unfair with respect to f_2. These trajectories being unfair with respect to f_2 (when only execution paths in f_1 are considered) all their states belong to $POT[f_1](f_2)$.

But this contradicts $q \notin ALL[\neg f_2] POT[f_1](f_2)$ with means that there exists a possible successor q' of q such that $q' \in \neg POT[f_1](f_2)$. □

This result shows that FCL is the conditional time logic obtained by adjoining to the propositional calculus the operator POT. This logic is less expressive than CL as it is shown by the following counter-example:

Consider the transition system,

S10

If $INEV(f)$ is expressible by a formula of $FCL(V)$, then it is possible to compute $INEV\{2\} = \{2, 3\}$ by applying to $\{2\}$ logical and/or temporal operators of FCL.

It is easy to verify that this is not possible:

1) From $\{2\}$ by application of logical operators we obtain,

$$\emptyset, \{2\}, \{1,3\}, \{1,2,3\}.$$

2) Application of the conditional temporal operator POT gives (x successively takes the values, $\emptyset$, $\{2\}$, $\{1, 3\}$, $\{1, 2, 3\}$).

$$\begin{aligned} POT[\{1,2,3\}](x)&: \emptyset, \{1,2,3\}, \{1,2,3\}, \{1,2,3\} \\ POT[\{2\}](x)&: \quad \emptyset, \{2\}, \quad \{1,2,3\}, \{1,2,3\}, \\ POT[\{1,3\}](x)&: \emptyset, \{1,2,3\}, \{1,3\}, \quad \{1,2,3\}. \end{aligned}$$

Since there is no new set generated we conclude that for this transition system it is not possible to express $INEV\{2\}$ in FCL.

III.4.4. Example. The aim of this section is to show how, from a practical point of view, the presented results simplify the problem of proving a property under the assumption of fairness.

The following program, given in [8], is a "non-deterministic version" of a solution to the mutual exclusion problem discussed in [5] for two processes A and B ($p1$ and $p2$ are program counters for the processes A and B).

$PROG$:: $p1:=1$; $p2:=1$; $inA:=$false; $inB:=$false; $prty:=A$;

do

$p1=1 \rightarrow inA:=$true; $p1:=2$ []

$p1=2 \wedge inB \rightarrow p1:=3$ []

$p1=3 \wedge prty=B \rightarrow inA:=$false; $p1:=4$ []

$p1=4 \wedge prty=B \rightarrow$ skip []

$p1=4 \wedge prty \neq B \rightarrow inA:=$true; $p1:=2$ []

$p1=3 \wedge prty \neq B \rightarrow p1:=2$ []

$p1=2 \wedge \neg inB \rightarrow \langle$critical section$\rangle$; $p1:=5$ []

$p1=5 \rightarrow inA:=$false; $p1:=6$ []

$p1=6 \rightarrow prty:=B$; $p1:=1$ []

$p2=1 \rightarrow inB:=$true; $p2:=2$ []

$p2=2 \wedge inA \rightarrow p2:=3$ []

$p2=3 \wedge prty=A \rightarrow inB:=$false; $p2:=4$ []

$p2=4 \wedge prty=A \rightarrow$ skip []

$p2=4 \wedge prty \neq A \rightarrow inB:=$true; $p2:=2$ []

$p2=3 \wedge prty \neq A \rightarrow p2:=2$ []

$p2=2 \wedge \neg inA \rightarrow \langle$critical section$\rangle$; $p2:=5$ []

$p2=5 \rightarrow inB:=$false; $p2:=6$ []

$p2=6 \rightarrow prty:=A$; $p2:=1$

od

We want to verify that if a process is about to enter its critical section then it cannot be delayed until eternity. This is expressed by the formulas f_i, $i=1,2$,

$$(pi=1) \Rightarrow INEV(pi=5).$$

One can prove that these formulas are not satisfied while the formulas f_i' expressing the same property under the assumption of fairness,

$$(pi=1) \Rightarrow FINEV(pi=5)$$

are satisfied.

To make these proofs we have used the formula checker of CESAR [18] which allows the verification of formulas by iterative evaluation of the inter-

pretation of temporal operators as fixed points of predicate transformers (Proposition 5). We found $INEV(pi=5)=\emptyset$ and the validity of f_i' has been established by proving that,

$$(pi=1)\Rightarrow ALL[pi\neq 5]\ POT(pi=5) \quad \text{for } i=1,2.$$

IV. Fairness and Related Properties

The aim of this section is to compare the proposed notion of fairness to other related notions expressing the fact that there exist infinite sequences such that a transition is never executed.

Definitions. Let σ be an infinite computation sequence starting from a state q of a transition system S and t a transition of S. We say that,

- σ *avoids* t *if* $\sigma\in(T-\{t\})^\infty$.
- σ is a *livelock sequence* for t if it has only a finite number of prefixes containing t (or, equivalently, a suffix avoiding t).
- σ is a *starvation sequence* for t if it has only a finite number of prefixes σ' such that $q\xrightarrow{\sigma'}q'$ and $q'\in enable(t)$.
- σ is an *unfair sequence* for t if it is a livelock sequence for t and for any prefix σ' there exists $\sigma''\in T^*$ such that $q\xrightarrow{\sigma'\sigma''}t$ (this is another formulation for unfairness w.r.t. transition t).
- σ is a sequence not satisfying the *finite delay property* for t if it is a livelock sequence for t and there exists a prefix σ' of σ such that for every $\sigma''\in T^*$, $\sigma'\sigma''$ prefix of σ, we have $q\xrightarrow{\sigma'\sigma''}q'$ and $q'\in enable(t)$.

Remarks. 1) Any starvation sequence is a livelock sequence. Starvation sequences correspond to starvation situations as described in [5] while livelock sequences correspond to livelock situations studied in [12, 20].

2) If a sequence does not satisfy the finite delay property for some t then it is an unfair sequence w.r.t. t but obviously the converse is not true.

A sequence satisfies the finite delay property for any transition t iff it satisfies the finite delay property condition formulated in [11].

Notations. For a given transition system $S=(Q, T, \{\xrightarrow{t}\}_{t\in T})$,

- denote by $OP\langle S\rangle(P)$ the function obtained by evaluating the temporal operator OP of L for S and a given subset P of Q.
- $SONT\langle S\rangle(P)$ is an abbreviation of $SOME\langle S\rangle(P\cap\overline{SINK}(S))$ where $\overline{SINK}(S)$ represents the set of the states of S having one successor at least ($\overline{SINK}(S)=\bigcup_{t\in T}$ enable(t)). In fact, $SONT\langle S\rangle(P)$ represents the greatest nonterminating trajectory of S contained in P.
- denote by S_{-t} the transition system obtained from S by deleting the transition t i.e.

$$S_{-t}=(Q, T-\{t\}, \{\xrightarrow{t'}\}_{t'\in T-\{t\}}).$$

The following proposition gives a fixed point characterization of the situations where livelock sequences, starvation sequences, unfair sequences and sequences not satisfying the finite delay property exist.

Proposition 14. *Let* $S=(Q, T, \{\xrightarrow{t}\}_{t\in T})$ *be a transition system and* t *one of its transitions,*

a) there exists a livelock sequence for t *iff* $SONT\langle S_{-t}\rangle(Q)\neq\emptyset$

b) there exists a starvation sequence for t *iff*

$$SONT\langle S\rangle(\neg\mathbf{enable}(t))\neq\emptyset$$

c) there exists an unfair sequence for t *iff*

$$SONT\langle S_{-t}\rangle(POT\langle S\rangle\,\mathbf{enable}(t))\neq\emptyset$$

d) there exists a sequence which does not satisfy the finite delay property for t *iff*

$$SONT\langle S_{-t}\rangle(\mathbf{enable}(t))\neq\emptyset.$$

Proof. a) b) and d) have been proved in [20]. A similar proof can be made for c) by using *Proposition 11.* □

V. Conclusion

We have proposed a notion of fairness for transition systems and a logic for proving properties under the fairness assumption induced from this notion.

In part I we have tried to justify, by using several examples, the choice of our definition of fairness which is sufficiently general for covering the majority of the definitions proposed until now. However, a precise comparison is not always possible either because fairness is sometimes introduced by informal discussion or because it is defined on models of higher level than transition systems (for example, models where the notions of process and parallel composition are primitive [17]).

The approach proposed for proving a property under our fairness assumption has the advantage of avoiding the complexification of the system under study by adjoining a fair scheduler to it. In fact, a fairness assumption characterizes the set of all the possible (fair) scheduling policies. In our approach this assumption is incorporated with the formula to prove, without modifying the model. This result is especially interesting as the logic is relatively simple and sufficiently well-studied.

Acknowledgements. We wish to thank Krzysztof Apt, Pedro Guerreiro and Jacques Voiron for their helpful comments on the results of this paper and their presentation. Thanks are also due to the members of the working group "Sémantique du Parallélisme" for their critiques concerning the material of part II.

References

1. Abrahamson, K.: Modal logic of concurrent non deterministic programs. Semantics of concurrent computation, Lecture Notes in Computer Science **70**, pp. 21-33. Berlin, Heidelberg, New York: Springer, 1979
2. Apt, K.R., Pnueli, A., Stavi, J.: Fair termination revisited - With delay. In: Proc. 2nd Conference on *FST* and *TCS*, Bangalore, India, 1982 (to appear)
3. Ben-Ari, M., Manna, Z., Pnueli, A.: The temporal logic of branching time. 8th Annual ACM Symp. on Principles of Programming Languages, pp. 164–176, 1981
4. Clarke, E.M., Emerson, E.A.: Design and synthesis of synchronization skeletons using branching time temporal logic. In: Logics of Programs LNCS **131**, 52-71 (1981)
5. Dijkstra, E.W.: Solution of a problem in concurrent programming control. Comm. ACM **8**, 9, 569 (1965)
6. Dijkstra, E.W.: Hierarchical ordering of sequential processes. Acta Informat. **1**, 2 (1971)
7. Emerson, E.A., Clarke, E.M.: Characterizing correctness properties of parallel programs using fixpoints. Proc. ICALP 80, Lecture Notes in Comput. Sci. **85**, pp. 169–181. Berlin, Heidelberg, New York: Springer, 1980
8. Flon, L., Susuki, N.: Non determinism and the correctness of parallel programs. In: Neuhold, E.J. (ed.). Formal description of programming concepts, North-Holland 1978, pp. 589–608
9. Grumberg, O., Francez, N., Makowsky, J.A., de Roever, W.P.: A proof rule for fair termination of guarded commands. Technical Report RUU-CS-81-2 Univ. of Utrecht, Dept. of Comput. Sci., 1981
10. Gabbay, D., Pnueli, A., Shelah, S., Stavi, J.: On the temporal analysis of fairness. Proc. 7th Annual ACM Symp. on Principles of Programming Languages, pp. 163–173, 1980
11. Karp, R.M., Miller, R.E.: Parallel Program Schemata, J. Comput. Syst. Sci. **3**, 147–195 (1969)
12. Kwong, Y.S.: On the absence of livelocks in parallel programs. Semantics of Concurrent Computation, Lectures Notes in Comput. Sci. **70**, pp. 172–190. Berlin, Heidelberg, New York: Springer, 1979
13. Lamport, L.: "Sometime" is sometimes "not never" - on the temporal logic of programs. Proc. 7th Annual ACM Symp. on Principles of programming Languages, Las Vegas, pp. 174–185, 1980
14. Lehmann, D., Pnueli, A., Stavi, J.: Impartiality, justice and fairness: the ethics of concurrent termination. ICALP, LNCS **115**, 264–277 (1981)
15. Park, D.: On the semantics of fair parallelism Abstract Software Specifications. Lecture Notes in Comput. Sci. **86**, pp. 504–524. Berlin, Heidelberg, New York: Springer, 1980
16. Park, D.: A predicate transformer for weak fair iteration. Proc. of the Sixth IBM Symp. Math. Foundations of Comp. Sci., 1981
17. Plotkin, G.D.: A powerdomain for countable non-determinism. In: Proc. ICALP 82, LNCS **140**, 418–428 (1982)
18. Queille, J.P., Sifakis, J.: Specification and verification of concurrent systems in CESAR. Internat. Symp. Progr., Lecture Notes on Comput. Sci. **137**, pp. 337–351. Berlin, Heidelberg, New York: Springer, 1982
19. Schwartz, R.L., Melliar-Smith, P.: Temporal logic specification of distributed systems. Proc. 2nd Int. Conf. on Distributed Comput. Syst., pp. 446–454, 1981
20. Sifakis, J.: Deadlocks and livelocks in transition systems Mathematical foundations of Computer Science. Lecture Notes in Comput. Sci. **88**, pp. 587–600. Berlin, Heidelberg, New York: Springer, 1980
21. Sifakis, J.: A unified approach for studying the properties of transition systems. TCS **18**, 227–258 (1982)

Received June 21, 1982 / December 20, 1982