Proving Liveness Property under Strengthened Compassion Requirements*

Teng Long[1,2] and Wenhui Zhang[1]
{longteng, zwh}@ios.ac.cn

[1] State Key Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences, Beijing, China
[2] School of Information Science and Engineering
Graduate University of China Academy of Sciences, Beijing, China

Abstract. Deductive rules are useful for proving properties with fairness constraints and there have been many studies on such rules with justice and compassion constraints. This paper focuses on system specifications with strengthened compassion that impose constraints on transitions involving states and their successors. A deductive rule for proving liveness properties under strengthened compassion is presented with examples illustrating the application of the rule.

1 Introduction

Liveness properties are requirements that something good must eventually happen. A counterexample to such a property is typically a loop during which the good thing never occurs. For avoiding acceptance of unrealistic loops in which some process or action is infinitely ignored, fairness is needed for imposing restrictions on accepted runs on such models.

There are several different notions of fairness for dealing with different situations. *Justice* is a simple kind of fairness that may be represented by a set of state formulas $\{\varphi_1, ..., \varphi_n\}$ and this fairness condition requires that each φ_i must be true infinitely often along every path. In 1981, Lehmann, Pnueli and Stavi defined justice requirements in [1], to describe the situation that some states must be visited infinitely often. *Compassion* is a kind of generalizations of justice, suggested by Pnueli and Sa'ar in [2], and may be represented by a set of pairs of state formulas $\{\langle\psi_1, \varphi_1\rangle, ..., \langle\psi_n, \varphi_n\rangle\}$. This fairness condition requires that along every path, for each pair $\langle\psi, \varphi\rangle$, either the first part is true only finitely many times or the second one is true infinitely often.

Justice and *Compassion* are regarded as weak and strong fairness in [3], both of them constrain the fairness of actions. They can only deal with actions fairly in *one context* situation. On the other hand, the correctness of many population protocols rely on stronger fairness constraints that may constrain actions in *all contexts* situation, e.g., self-stabilizing leader election in ring networks [4] and

* Supported by the National Natural Science Foundation of China under Grant Nos. 60721061, 60833001, and the CAS Innovation Program.

token circulation in rings [5]. Self-stabilizing algorithms [6], such as the former one, can guarantee the robustness and fault-tolerance of distributed systems.

We study a kind of fairness based on compassion with additional constraints. An important character of this kind of fairness is the constraint of transition (involving states and their successors) which can deal with actions fairly in *all contexts* situations. The fairness is referred to as *strengthened compassion* represented by a set of pairs of state formulas: $\{\langle \psi_i, \{\varphi_{i1}, ..., \varphi_{im_i}\}\rangle \mid 1 \leq i \leq n\}$. This specification requires that along every path, either ψ_i is true only finitely many times or for all j, $j \in [1..m_i]$, φ_{ij} (with $1 \leq j \leq m_i$) is true infinitely often immediately after the state at which ψ_i holds.

For proving liveness properties, *deduction rules* have been presented in [7, 2] for systems with justice and compassion. In this paper, we also develop a deductive rule SC_RESPONSE for the strengthened compassion.

The rest of this paper is organized as follows. In Section 2, we present the computation model with strengthened compassion requirements and compare it with models under compassion. In Section 3, the deduction rule SC_RESPONSE is presented and its soundness and relative completeness are proved. In section 4, an illustrative example is used to shown the application of the approach. Finally, concluding remarks are presented in Section 5.

Related Works There has been a lot of research work on fairness. In [8], a process analysis toolkit for system analysis with different kinds of fairness was presented. In [9], a method for finding auxiliary constructs as the effective premises of deductive proof of liveness property was proposed. In [10], a method based on analysis of maximal strongly connected components was presented. It involves compassion requirements introduced by ranking abstraction. [11] extended the method to native[1] compassion. In [12], an automatic method to derive a deductive proof of liveness properties from symbolic model checking under compassion was presented. Strengthened compassion is closely related to extreme fairness [13] which restrict transitions by adding predicates. The difference between extreme fairness and strengthened compassion (to be formally defined later) is that strengthened compassion is defined with one-step pair of state formulas instead of the directions in extreme fairness.

2 Computational Model

We present a computational model with strengthened compassion. We first present a transition system without fairness constraints, and then add strengthened compassion constraints to the model.

2.1 Discrete Transition Systems

A transition system is triple $T = \langle V, \Theta, \rho\rangle$ where the components are as follows.

[1] The compassion requirements without the additional variable *dec* that comes from ranking abstraction.

- V : A finite set of typed *system variables* - containing data and control variables. A set of states (interpretation) over V is denoted by Σ. For a state s and a system variable $v \in V$, we denote by s[v] the value assigned to v by the state s.
- Θ : The *initial condition* - an assertion (state formula) characterizing the initial states.
- ρ : The *transition relation* - an assertion $\rho(V, V')$, relating the values V of the variables in state $s \in \Sigma$ to the values V' in a T-successor state $s' \in \Sigma$. If φ is a formula representing a set of states, then φ' is the formula with each v replaced by v'.

Computation A computation of T is an infinite sequence of state $\sigma : s_0, s_1, \ldots$, satisfying the following requirements: (1) $s_0 \models \Theta$. (2) For each j = 0, 1, . . . , the state s_{j+1} is in a T-successor of the state s_j. For each $v \in V$, we interpret v as $s_j[v]$ and v' as $s_{j+1}[v]$, such that $\langle s_j, s_{j+1} \rangle \models \rho(V, V')$.

2.2 Fair Discrete Systems

A strengthened compassion constraint is specified as a pair $\langle r, U \rangle$ where r is an assertion and U is a set of state assertions.

Strengthened Compassion A computation σ of a discrete transition system T is fair with respect to the strengthened compassion (abbreviated to *scp*) with respect to a set of fairness constraints (state assertions) $F = \{\langle r_i, U_i \rangle | i = 1, ..., n\}$ with $U_i = \{u_{i,1}, ..., u_{i,m_i}\}$, if for each $i \in \{1, ..., n\}$, σ contains only finitely many r_i-states, or σ contains infinitely many pairs of states, such that for all $j \in [1..m_i]$: (s_k, s_{k+1}), $s_k \models r_i$ and $s_{k+1} \models u_{i,j}$.

Discrete Systems with Strengthened Compassion Requirements A discrete system with strengthened compassion requirements (SCDS) $D = (T, F)$ is a pair of a discrete transition system and a set of strengthened compassion constraints such that a fair computation of D is a *scp* computation of T with respect to F.

2.3 A Discussion on Different Kinds of Fairness

We explain the differences between compassion and strengthened compassion by considering the transitions in Fig. 1.

- Compassion requirements constrains that infinitely enabled actions must eventually be taken.
  The compassion requirement: $\langle x = 1 \wedge y = 0, x = 0 \rangle$ constrains that if "$x = 1 \wedge y = 0$" is satisfied infinitely many times, then "x=0" must be satisfied infinitely many times.
  The infinite loop $(S_1 S_2 S_3 S_2)^\omega$ satisfy this constraint. In such a loop, the action "x=x-1" is enabled infinitely many times, and this action is taken infinitely times.
  The action "x=x-1" in the figure is enabled in both S_1 and S_2, however, the compassion requirement does not distinguish the two different transitions.

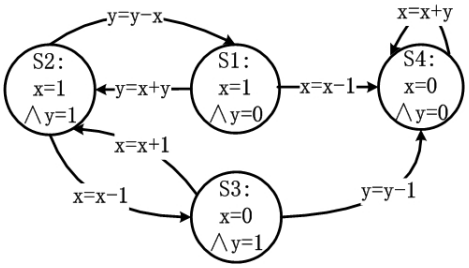

Fig. 1. Case 1

– Strengthened compassion requirements constrains that transitions with an infinitely enabled action must eventually be taken.
The strengthened compassion requirement $\langle x = 1 \wedge y = 0, \{x = 0\}\rangle$ constrains that if "$x = 1 \wedge y = 0$" is satisfied infinitely many times, then "x=0" must be satisfied infinitely many times *right after* "$x = 1 \wedge y = 0$".
The infinite loop $(S_1S_2S_3S_2)^\omega$ does not satisfy this requirement. In such a loop, the action "x=x-1" is enabled infinitely many times at S_1 and S_2, but the action "x=x-1" is never taken from S_1 (which satisfies $x = 1 \wedge y = 0$) in the loop (i.e., the transition $S_1 \xrightarrow{x=x-1} S_4$ is not taken).

3 Proving Properties

The liveness properties considered in this paper are response properties. Response properties are an important and widely studied type of liveness properties [2, 14]. A response property is of the form $p \Rightarrow \diamond q$ where p and q are state assertions. A SCDS D satisfies the response property $p \Rightarrow \diamond q$ whenever a reachable state of D satisfies p at which every fair computation starts, it will reach a q state at some point (including the starting point).

3.1 Proof Rule

In Fig. 2, we present proof rule $SC\_RESPONSE$ that establishes the response property $p \Rightarrow \diamond q$ for a SCDS D. The use of the rule requires a well-founded domain $\mathcal{A}$, and for each requirement $\langle r_i, U_i\rangle$, a helpful assertion φ_i and a ranking function $\Delta_i : \Sigma \mapsto W$ mapping states of D to elements of $\mathcal{A}$.

R1 requires that any p-state is either a goal state (i.e., a q-state), or a $(r_i \wedge \varphi_i)$-state for some $i \in \{1, \ldots, n\}$. R2 requires that any step from a $(r_i \wedge \varphi_i)$-state moves either directly to a q-state, or to another $(r_j \wedge \varphi_j)$-state, or stays at a state of the same level (i.e., a $(r_i \wedge \varphi_i)$-state). R3 requires that any step from a φ_i-state moves either directly to a q-state, or to another $(r_j \wedge \varphi_j)$-state with decreasing rank ($\Delta_i \succ \Delta_j$), or stays at a state with the same rank. R4 ensures that there exists at least one $u_{i,k}$ such that these $u_{i,k}$-states as the successors of $(\varphi_i \wedge r_i)$-states are not in φ_i. In other words, during the transitions among the states

Let p, q be assertions.
Let $\mathcal{A} = (W, \succ)$ be a well-founded domain.
Let $\{F_i = \langle r_i, \{u_{i,1}, .., u_{i,m_i}\}\rangle \mid i \in \{1, ..., n\}\}$ be a set of scp requirements.
Let $\{\varphi_i \mid i \in \{1, ..., n\}\}$ be a set of assertions.
Let $\{\Delta_i : \Sigma \to W \mid i \in \{1, ..., n\}\}$ be a set of ranking functions.

R1	p	$\Rightarrow q \vee \bigvee_{j=1}^{n}(r_j \wedge \varphi_j)$
$\forall i \leq n$:		
R2	$r_i \wedge \varphi_i \wedge \rho$	$\Rightarrow q' \vee \bigvee_{j=1}^{n}(r'_j \wedge \varphi'_j)$
R3	$\varphi_i \wedge \rho$	$\Rightarrow q' \vee (\varphi'_i \wedge \Delta_i = \Delta'_i) \vee \bigvee_{j=1}^{n}(r'_j \wedge \varphi'_j \wedge \Delta_i \succ \Delta'_j)$
R4	$\varphi_i \wedge r_i \wedge \rho \wedge \varphi'_i$	$\Rightarrow \neg u'_{i,k}$ for some $1 \leq k \leq m_i$
	p	$\Rightarrow \Diamond q$

Fig. 2. Proof Rule: SC_RESPONSE

inside of φ_i, there are no r_i to $u_{i,k}$ transitions. R_{2-4} together with the definition of strengthened compassion requirements guarantee that if an execution enters a loop (consisting of states of some φ_i) without leaving it, then it violates the strengthened compassion requirements, such that it must get out of all the unfair loops and finally reach q-state (the goal state).

3.2 Soundness of the Rule

The soundness is established as follows. Suppose that the premises of the rule are valid and the conclusion is not. We prove that this is a contradiction.

The conclusion is not valid means that there exists a computation $\sigma = s_0, s_1 \ldots$ and a position $j \geq 0$ such that $s_j \models p$ and no state s_k, for $k \geq j$ satisfies q. Without loss of generality, we take $j = 0$. According to premises of R_1 and R_2 and the assumptions that no states satisfy q, then any state s_w satisfies $r_i \wedge \varphi_i$ for some $i \in \{1, ..., n\}$. Since there are only finitely many different i's, there exists a cutoff index $h \geq 0$ such that for every i and $w \geq h$, $s_w \models r_i \wedge \varphi_i$ if and only if σ contains infinitely many $(r_i \wedge \varphi_i)$-positions.

Consider position $w_1 = h$. Choose i_1 to be the index such that $s_{w_1} \models r_{i1} \wedge \varphi_{i1}$. According to R_3 and the assumption that σ contains no q-positions, then either φ_{i1} holds at all positions $w \geq w_1$, or there exists a position $w_2 \geq w_1$ and index i_2 such that $s_{w_2} \models r_{i2} \wedge \varphi_{i2}$ and $\Delta_{i1}(s_{w_1}) \succ \Delta_{i2}(s_{w_2})$. We argue that the former case is not possible and then the latter leads to an infinite sequence of decreasing values of Δ.

- If φ_{i1} holds continuously beyond w_1, then due to premise of R_4, $r_{i1} \wedge \varphi_{i1}$ holding at $w_1 \geq h$ implies that $r_{i1} \wedge \varphi_{i1}$ (and therefore r_{i1}) holds at infinitely many positions without succeed infinitely many $u_{i1,k}$-states. This violates the requirement $\langle r_{i1}, \{U_{i1}\}\rangle$.
- If there exists a position $w_2 \geq w_1$ and index i_2 such that $s_{w_2} \models r_{i2} \wedge \varphi_{i2}$ and $\Delta_{i1}(s_{w_1}) \succ \Delta_{i2}(s_{w_2})$, we can continuously find i_3, i_4, ..., such that $\Delta_{i_1}(s_{w_1}) \succ \Delta_{i_2}(s_{w_2}) \succ \Delta_{i_3}(s_{w_3}) \succ \ldots$. According to the definition of

well-founded domain, it is impossible to find infinite positions to satisfy the decrease sequence.

Therefore there cannot exist a computation σ violating the response property $p \Rightarrow \diamond q$ if the premises of rule are all valid.

3.3 Relative Completeness of the Rule

Completeness means that, whenever a response property $p \Rightarrow \diamond q$ is valid over a SCDS $\mathcal{D}$, there exists a well-founded domain and auxiliary constructs such that the premises of the rule can be proved. The auxiliary constructs consist of a list of helpful assertions $\varphi_1, \ldots, \varphi_n$ and a list of ranking functions $\Delta_1, \ldots, \Delta_n$. We only consider relative completeness, in the sense that the followings are assumed: the premises of the rule can be proved when they are valid, and the language for expressing the assertions is sufficient to express information (including the expressions $\mathbf{E}(p\,\mathcal{S}\,q)$ and $\mathbf{E}(p\,\mathcal{U}\,q)$ defined below) that are necessary for proving the validity of the premises.

Operators

1. For assertions p and q , the formula $\mathbf{E}$(p $\mathcal{S}$ q) captures the set of states that are reachable from a q-state by a p-path all of whose states, except possibly the first, satisfy p. In this expression we use the *since* temporal operator $\mathcal{S}$.
2. The formula $\mathbf{E}$(p $\mathcal{U}$ q) captures the set of states that originate a path leading to any q-state, such that all the states in the path, except possibly the last, satisfy p. In this expression we use the *until* temporal operator $\mathcal{U}$.
3. The formula $\mathbf{EX}$(p) captures the set of states that are the immediate predecessors of p-states.

Considering a SCDS $\mathcal{D}$ and a response property $p \Rightarrow \diamond q$, we present an algorithm which extracts a deductive proof according to the rule of a response property $p \Rightarrow \diamond q$. It defines the values $\delta_1, \ldots, \delta_m$ of the respective ranking functions $\Delta_1, \ldots, \Delta_m$ on different sets of states, and identify an associated requirement $\langle r_i, U_i \rangle$, and a helpful assertion φ_i for each $i \in \{1, .., m\}$. The algorithm *Auxiliary_constructs* is presented as Algorithm 1.

The expression $accessible_{\mathcal{D}}$ captures the set of all accessible states with $\mathcal{D}$. The expression of *pend* describes all states which are reachable from any accessible p-state by a finite q-free path. *prefix* is a list that is supposed to be a prefix of some δ. The list operation $*$ denotes the concatenation of two lists. ψ is the set of Y-states without r-states which are the predecessors of u_j-states. φ is the set of ψ-states which can be reached by r, i.e. φ-states are those that form a strongly connected subgraph of ψ. *rem* is the set of φ-states that are not r-states. The new Y is the set of remaining states.

For each i, φ_i, f_i, δ_i where $\Delta_i(s) = \delta_i$ for $s \in \varphi_i$, are the auxiliary constructs discovered at the respective stages of the execution of the algorithm. For each strengthened compassion f_i: $\langle r, \{u_1, ..., u_k\} \rangle$, we construct φ_i (the set of states

Algorithm 1 *Auxiliary_constructs*

1: m := 0
2: $accessible_{\mathcal{D}} := E(trueS\Theta)$
3: $pend := accessible_{\mathcal{D}} \wedge E(\neg qS(p \wedge \neg q))$
4: $rank\_SC(pend, [])$

where the procedure $rank\_SC$ is defined as follows.
procedure $rank\_SC(subpart,prefix)$
d:**integer**
Y:**assertion**
5: Let $d := 0$
6: Let $Y := subpart$
7: FIX (Y)
8: **Forall** $(\langle r, \{u_1, u_2, \ldots, u_k\}\rangle \in F)$ **do**
9: Let $\psi = Y \wedge \neg(Y \wedge r \wedge EX(Y \wedge u_j))$
10: **if** $\psi \wedge r \neq \emptyset$ **then**
11: Let $\varphi = E(\psi S(\psi \wedge r))$
12: **if** $\varphi \wedge \neg E(\varphi U(\varphi \wedge r)) = \emptyset$ **then**
13: Let m=m+1
14: Let d=d+1
15: Let $\varphi_m := \varphi$
16: Let $f_m := \langle r, \{u_1, u_2, \ldots, u_k\}\rangle$
17: Let $\delta_m := prefix * [d]$
18: Let $Y := Y \wedge \neg\varphi$
19: Let $rem := \varphi \wedge \neg r$
20: **if** $(rem \neq \emptyset)$ **then**
21: $rank\_SC(rem, prefix * [d])$
22: **end if**
23: **end if**
24: **end if**
25: end for
26: **if** $(Y \neq \emptyset)$ **then**
27: report "fail"
28: **end if**
29: **end-Fix**

that formed an unfair loop that contains r-states which are not the predecessor of u_j-states for some j) with its own δ_i to measure the distance between the loop to the goal states. The construction is as follows:

- S1: To start with, we deal with the *pend* states (line 4), i.e. $Y_0 = pend$. By the definition of *pend*, we know that there are no goal states in *pend*. The first unfair loop we can find, is the one nearest to goal states (under the strengthened compassion for the transition to goal states).
- S2: For each m, after removing an unfair loop φ_m, the new set of states we will be dealing with is $Y' = Y - \varphi_m$ (line 18). In Y', by calling $rank\_SC$ (line 21) recursively, we construct φ_{m+1} and δ_{m+1} (line 15,17).

- S3: According to the definition of strengthened compassion, the reason of unfairness is the "bad"states: the r-states in the loop (line 12). Therefore we remove r-sates from each φ_i (line 19), and then we deal with the remaining part of φ_i recursively (line 20,21). The ranks of states in φ_i are with the same prefix δ_i.
- S4: If all the *pend* states consist of unfair loops which can be constrained by strengthened compassion to leave the *pend* states, then the liveness property can be guaranteed. Otherwise, the liveness property fails (line 26,27).

Additional explanation of the algorithm is as follows.

- Line 7: FIX Y terminates when Y does not change after the specified computation. After termination, it is at line 26 which prepares the final result for S4.
- Line 9: Constructing ψ by removing the r states that enable the transition $r \rightarrow u_j$ from Y. Then there are no transitions of the form $r \rightarrow u_j$ in ψ_i. Then if r is part of a loop in ψ, then this loop must be unfair violating the fairness requirement under consideration.
- Line 10: Checking whether there may exist such unfair loop by checking whether there exist r-states in ψ_i.
- Line 11: Constructing assertion φ which consists of all of the reachable states from r-states in ψ_i.
- Line 12: Checking whether φ is a loop by checking whether all of the φ-states can reach r-states in φ.
  If there is at least one state in φ that cannot reach the r-states in φ, it means that φ is not a loop. Such that the distance of the unfair part of φ might not be the right one or the constraint is not specific enough. Then we go back to line 8 and consider another strengthened compassion requirement.
  If it is a loop, then it is the unfair one that is to be denoted φ_m and to be assigned the value of rank δ_m in the subsequent actions.
- Line 13-17: Constructing the helpful assertion φ_m, the strengthened compassion constraint f_m and the distance measure δ_m, respectively.
- Line 18-22: Constructing the new Y for the use by S2, and preparing the recursive call for S3.

Validity of the Algorithm For every Y at different levels of the recursive calls of the algorithm, if there exist a fairness constraint $\langle r, \{u_1, u_2, \ldots, u_k\}\rangle$ and some j, such that there exists at least one r-state which is not the predecessor of u_j-states, then this fairness constraint is sufficient to guarantee that it is not possible to stay at φ-states (the reachable states from r-states) infinitely often.

Otherwise, if during all fairness constraints, there is no such j exist, the execution of the model are not required to leave these r-states[2], and hence the response property is not valid.

[2] Following the tradition of [10], every state is assumed to have a loop to itself, and every state must be constrained by some fairness requirement in order to force the progress of computations of such a system model.

Proof of the Completeness The above arguments implies that if the response property is valid, the algorithm will terminate properly without reporting "fail", i.e. Y is decreased to the empty set at each levels of the algorithm in the recursive computation of φ_i, f_i and Δ_i. We consider in turn each of the premises and show that it holds for the extracted constructs φ_i, f_i and Δ_i with $i \leq m$. Clearly, we have $\{f_1, ..., f_m\} \subseteq F$. If $F_l \in F$ is not in $\{f_1, ..., f_m\}$, we may add $f_{m+1} = F_l$ to the set and let φ_{m+1} be $false$ and Δ_{m+1} be the empty list. If $f_{l_1}, ..., f_{l_k}$ are the same as F_l for some l, for technical reason, we make k copies of F_l such that each corresponds to one element of $\{f_{l_1}, ..., f_{l_k}\}$. Replacing F_l with k-copies does not change the contents of the system description. Then we may assume that $m = n$ (the number of fairness requirements).

1. Premise of R_1 claims that every p-state s satisfies q or $r_j \wedge \varphi_j$, for some $j \in [1..n]$. Indeed, if s does not satisfy q, it belongs to the pending graph (i.e., the initial Y, denoted hereafter by Y_0), and since all Y_0-states are divided and removed after the algorithm, s must be a part of the removed ones, it means that s belongs to some $r_j \wedge \varphi_j$.
2. Premise of R_2 requires that every immediate successor of s that satisfies $r_i \wedge \varphi_i$ must satisfy q or $r_j \wedge \varphi_j$ for some $j \in [1..n]$. As mentioned before, s belongs to Y_0, and its successor s_b must satisfy q or belong to Y_0. Similar to the situation in premise of R_1, we can get that s_b must belong to some $r_j \wedge \varphi_j$.
3. Premise of R_3 considers a state s_a that satisfies φ_i. Consider a successor s_b. It requires that s_b satisfies q, or φ_i and has the same value as $\Delta_i(s_a)$, or satisfies $r_j \wedge \varphi_j$ for some j and has $\Delta_j(s_b) \prec \Delta_i(s_a)$. According to the construction, every φ_i-state s has a rank $\Delta_i(s)$ and φ_i-state can be reached from a r_i-state by a finite φ_i-path π.
   - If s_b is a q-state, then it is acceptable.
   - If s_b is in Y, by the definition of $Y = \varphi_i + Y'$ (Y' is not reachable from states in φ_i) and the construction of φ_i, we know that s_b is in φ_i.
   - If s_b is in Y_0 - Y, such that s_b must have been removed from Y_0 in some earlier stage, satisfy $r_j \wedge \varphi_j$ with $\Delta_j(s_b) \prec \Delta_i(s_a)$ for some $j < i$.
4. Premise of R_4 requires that at least one type of transitions from r_i to $u_{i,k}$ cannot be taken from r_i-states in φ_i. By the definition of strengthened compassion , it satisfies this condition.

The above arguments proves the completeness, i.e., whenever a response property is valid, there exist auxiliary constructs for proving the property.

3.4 Dealing with Systems with Infinite Number of States

For dealing with infinite state systems, in addition to the above algorithm for constructing helpful assertions and ranks, we have to apply abstraction and concretization. The basic steps for proving the property is as follows

- Abstracting the program to a finite state one

– Constructing the helpful assertions and ranks
– Concretizing the constructs
– Proving the property by using the proof rule

The main focus of this paper is the second and the fourth issues. An example to demonstrate the above steps is shown in the next section.

4 An Example with Non-Deterministic Choice

Consider the program (which appeared also in [15]) with a non-deterministic choice of the values of variable x in Figure 3.

$$\left[\begin{array}{l} x : \textbf{natural}; \\ \textbf{init}\ x > 2; \\ l_0 : \textbf{while}: \ (x > 0) \\ \quad l_1 : \textbf{x:=2 or x:=x-1} \\ l_2 \end{array}\right]$$

Fig. 3. Example 1

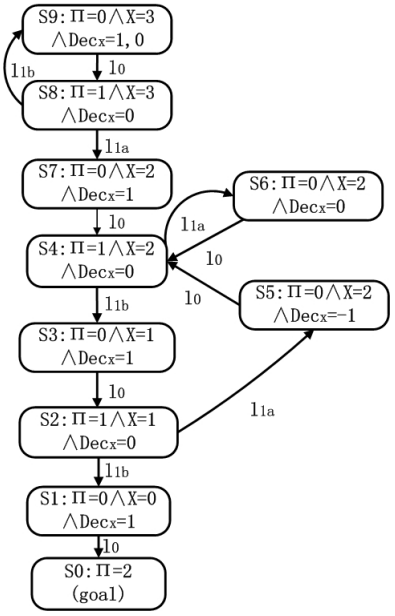

Fig. 4. Pending graph after abstraction

At location l_1 there is a non-deterministic choice, and we may denote the first transition as l_{1a} and the second as l_{1b}. Let x' denote the next state variable of x.

The property we wish to establish is $at\_l_0 \Rightarrow \Diamond at\_l_2$, and the strengthened compassion requirements are:

F_{c0}: $\langle at\_l_0, \{\neg at\_l_0\}\rangle$
F_{c1}: $\langle at\_l_1, \{at\_l_0 \wedge x = 2, at\_l_0 \wedge x' < x\}\rangle$
F_{c2}: $\langle at\_l_1 \wedge x = 1, \{at\_l_0 \wedge x = 2, at\_l_0 \wedge x' < x\}\rangle$

The basic fairness requirement of location 0 is F_{c0}. F_{c1} implies that if it is at location 1 infinitely times, the action "x=2 "must be taken from location 1 infinitely times, so does action "x=x-1". F_{c2} implies that if it is at location 1 and x equals 1 infinitely times, the action "x=2" must be taken from location 1 infinitely times, so does action "x=x-1".

Proving the property The following basic steps are explained in the subsections that follow.

- Abstracting the program to a finite state one
- Calculating the helpful assertions and ranks
- Concretizing the constructs
- Proving the property by using the proof rule

4.1 Abstraction

We apply the following abstraction α :

$$\alpha : \Pi = \pi, X = \widetilde{x}, Dec_x = dec_x$$

where the assertion $\pi = i$ stands for at program location l_i. X and dec_x are defined as follows.

$$\widetilde{x} = \begin{cases} 0 & x = 0 \\ 1 & x = 1 \\ 2 & x = 2 \\ 3 & x > 2 \end{cases} \qquad dec_x = \begin{cases} 1 & x \succ x' \\ 0 & x = x' \\ -1 & otherwise \end{cases}$$

The response property after applying the abstraction is now $\Pi = 0 \Rightarrow \Diamond \Pi = 2$, and the strengthened compassion requirements are as follows:

$$\begin{array}{l} F_0 : \langle \Pi = 0, \{\Pi \neq 0\}\rangle \\ F_1 : \langle \Pi = 1, \{\Pi = 0 \wedge (X = 2), \Pi = 0 \wedge (Dec_x = 1)\}\rangle \\ F_2 : \langle \Pi = 1 \wedge (X = 1), \{\Pi = 0 \wedge (X = 2), \Pi = 0 \wedge (Dec_x = 1)\}\rangle \end{array}$$

F_0, F_1 and F_2 are respectively the abstract version of F_{c0}, F_{c1} and F_{c2}.

4.2 Calculation

The pending graph of the finite state transition system after applying the abstraction is shown in Fig. 4. The helpful assertions and ranks we get by algorithm *Auxiliary_constructs* are in Table. 1, and the process of the calculation is explained as follows.

1. We start with $rank\_SC(pend, [])$, and set $Y = pend = \{S_1, ..., S_9\}$, d=0, m=0.
2. Neither of F_1 and F_2 satisfies the assumption $\psi \wedge r$ at line 10. This means that these two strengthened compassion requirements may be fair to Y.
3. Then F_0 is chosen at line 9, and we get $\psi = \{S_1, S_2, S_4, S_8\}$.

Φ_i	S_i	δ_i	H_i
φ_9	$\Pi = 0 \wedge X = 3 \wedge Dec_x = \{1, 0\}$	$[4, 1]$	F_0
φ_8	$\Pi = 0 \wedge X = 3 \wedge Dec_x = \{1, 0\}$ $\vee \Pi = 1 \wedge X = 3 \wedge Dec_x = 0$	$[4]$	F_1
φ_7	$\Pi = 0 \wedge X = 2 \wedge Dec_x = 1$	$[3]$	F_0
φ_6	$\Pi = 0 \wedge X = 2 \wedge Dec_x = -1$	$[2, 3]$	F_0
φ_5	$\Pi = 0 \wedge X = 2 \wedge Dec_x = 0$	$[2, 2, 1]$	F_0
φ_4	$\Pi = \{0, 1\} \wedge X = 2 \wedge Dec_x = 0$	$[2, 2]$	F_1
φ_3	$\Pi = 0 \wedge X = 1 \wedge Dec_x = 1$	$[2, 1]$	F_0
φ_2	$\Pi = \{0, 1\} \wedge X = 2 \wedge Dec_x = 0$ $\vee \Pi = 0 \wedge X = 1 \wedge Dec_x = 1$ $\vee \Pi = 0 \wedge X = 2 \wedge Dec_x = -1$	$[2]$	F_2
φ_1	$\Pi = 0 \wedge X = 0 \wedge Dec_x = 1$	$[1]$	F_0

Table 1. The Rank Table of Example 1

4. At line 10, $(Y \wedge \Pi = 0) \neq \emptyset$, and we get $\varphi = \{S_1\}$ at line 11.
5. States in φ can reach the r-state in φ (S_1 can reach itself), which means it is right to use the strengthened compassion requirement F_0.
6. Set m=1, d=1, $\varphi_1 = \{S_1\}$, $f_1 = F_0$, $\delta_1 = [1]$, $Y' = \{S_2, ..., S_9\}$ and rem = 0.
7. rem = 0 does not satisfy the condition at line 20, back to line 8 to choose another strengthened compassion requirement.
8. F_2 is chosen at line 9, and we get $\psi = Y = \{S_2, ..., S_9\}$.
9. At line 10, $(Y \wedge \Pi = 1 \wedge X = 1) \neq \emptyset$, and we get $\varphi = \{S_2, ..., S_6\}$ at line 11.
10. All of the states $\{S_2, ..., S_6\}$ in φ can reach the r-state in φ: $\{S_2\}$, which means it is right to use the strengthened compassion requirement F_2.
11. Set m=2, d=2, $\varphi_2 = \{S_2, ..., S_6\}$, $f_2 = F_2$, $\delta_2 = [2]$, $Y' = \{S_7, ..., S_9\}$ and rem $= \{S_3, ..., S_6\}$.
12. Then call $rank\_SC(\{S_3, ..., S_6\}, [2])$ in which the following is done.
    - Set $Y = \{S_3, ..., S_6\}$, d=0.
    - Choose F_0 at line 9, we get $\psi = \{S_3, ..., S_6\}$.
    - At line 10, $(Y \wedge \Pi = 0) \neq \emptyset$, and we get $\varphi = \{S_3\}$ at line 11.
    - Set m=3, d=1, $\varphi_3 = \{S_3\}$, $f_3 = F_0$, $\delta_3 = [2, 1]$, $Y' = \{S_4, ..., S_6\}$ and rem = 0.
    - This continues until we have constructed $\varphi_4, \varphi_5, \varphi_6$ and the respective ranks as shown in the table, and then Y in this recursion is empty.
13. After the recursive function call, we deal with $Y = \{S_7, ..., S_9\}$, d=2 (equals the value of d before the recursion), m=6 (equals the value of m at the end of the recursion). Choose F_0 at line 9, and we get $\psi = \{S_7, S_8\}$.
14. At line 10, $(Y \wedge \Pi = 0) \neq 0$, and we get $\varphi = \{S_7\}$ at line 11.
15. Set m=7, d=3, $\varphi_7 = \{S_7\}$, $f_7 = F_0$, $\delta_7 = [3]$, $Y' = \{S_8, S_9\}$ and rem = 0.
16. rem = 0 is does not satisfy the condition at line 20, back to line 8 to chose another strengthened compassion requirement.
17. Then choose F_1 is chosen at line 9, we get $\psi = Y = \{S, S_9\}$.
18. At line 10, $(Y \wedge \Pi = 1) \neq \emptyset$, and we get $\varphi = \{S_8, S_9\}$ at line 11.

19. All of the states $\{S_8, S_9\}$ in φ can reach the r-state in φ: $\{S_8\}$, which means it is right to use the strengthened compassion requirement F_1.
20. Set m=8, d=4, $\varphi_8 = \{S_8, S_9\}$, $f_8 = F_1$, $\delta_8 = [4]$, $Y' = \{\emptyset\}$ and rem = $\{S_9\}$.
21. Then call $rank\_SC(\{S_9\}, [4])$ in which the following is done.
    - Set $Y = \{S_9\}$, d=0.
    - Choose F_0 at line 9, we get $\psi = \{S_9\}$.
    - At line 10, $(Y \wedge \Pi = 0) \neq 0$, and we get $\varphi = S_9$ at line 11.
    - Set m=9, d=1, $\varphi_9 = \{S_9\}$, $f_9 = F_0$, $\delta_9 = [4, 1]$, $Y' = \{\emptyset\}$ and rem = 0.
    - In this recursion, Y is empty, back to the previous level.
22. Finally, Y is empty. The algorithm terminates successfully.

Merging Assertions In order to reduce the number of assertions and ranking functions, we merge assertions that differ only in their Dec variables. i.e $\varphi_5 - \varphi_7$ as φ'_5. The rank of φ' is the lowest rank of these φ_i. Then we make a re-enumeration of φ_i, and get the new helpful assertions and ranks shown in Table. 2.

Φ_i	S_i	δ_i	H_i
φ_7	$\Pi = 0 \wedge X = 3 \wedge Dec_x = \{1, 0\}$	$[4, 1]$	F_0
φ_6	$\Pi = 0 \wedge X = 3 \wedge Dec_x = \{1, 0\}$ $\vee \Pi = 1 \wedge X = 3 \wedge Dec_x = 0$	$[4]$	F_1
φ_5	$\Pi = 0 \wedge X = 2 \wedge Dec_x = \{0, 1, -1\}$	$[2, 2, 1]$	F_0
φ_4	$\Pi = \{0, 1\} \wedge X = 2 \wedge Dec_x = 0$	$[2, 2]$	F_1
φ_3	$\Pi = 0 \wedge X = 1 \wedge Dec_x = 1$	$[2, 1]$	F_0
φ_2	$\Pi = \{0, 1\} \wedge X = 2 \wedge Dec_x = 0$ $\vee \Pi = 0 \wedge X = 1 \wedge Dec_x = 1$ $\vee \Pi = 0 \wedge X = 2 \wedge Dec_x = -1$	$[2]$	F_2
φ_1	$\Pi = 0 \wedge X = 0 \wedge Dec_x = 1$	$[1]$	F_0

Table 2. The Abstract Rank

4.3 Concretization

The helpful assertions and ranks after concretization is shown in Table 3. The process of concretization is explained as follows.

Concretizing Ranks Because Dec_x is an additional variable by ranking abstraction and must be remove after concretization, we should make sure that the constraint by Dec_x in the abstract strengthened compassion requirements is reflected in the definition of ranks. The way to deal with this aspect is to append the value of variable x to corresponding ranking tuples [10].

If a rank is associated with an abstract fairness requirement involving the special variable Dec_x (for any variable x), then the rank Δ_i is to be modified as follows.

φ	s_i	$\Delta_i(s)$	H_i
φ_7	$at\_l_0 \wedge x > 2$	$[4, x_s, 1]$	F_{c0}
φ_6	$at\_l_{0,1} \wedge x > 2$	$[4, x_s, 0]$	F_{c1}
φ_5	$at\_l_0 \wedge x = 2$	$[2, 2, x_s, 1]$	F_{c0}
φ_4	$at\_l_{0,1} \wedge x = 2$	$[2, 2, x_s, 0]$	F_{c1}
φ_3	$at\_l_0 \wedge (x = 1)$	$[2, 1]$	F_{c0}
φ_2	$at\_l_{0,1} \wedge (0 < x \leq 2)$	$[2]$	F_{c2}
φ_1	$at\_l_0 \wedge (x = 0)$	$[1]$	F_{c0}

Table 3. The Concrete Rank

- If φ_i satisfies $Dec_x = -1$, then no modification is necessary, since adding the value of variable x does not help the decreasing of the rank in computations.
- Otherwise, if $\delta_i = [\beta]$ is obtained with a strengthened compassion requirement F involving Dec_x, then we insert the value of variable x and 0 after β in δ_i. Then for $\delta_j = [\beta, \gamma]$ with the same prefix β, it will be modified to $[\beta, x_s, \gamma]$, i.e., $\Delta_j(s) = [\beta, x_s, \gamma]$ where x_s denotes the value of x at state s.

For instance, in Table 1, $F_1 = \langle \Pi = 1, \{\Pi = 0 \wedge (X = 2), \Pi = 0 \wedge (Dec = 1)\}\rangle$ is used to constrain φ_4 with rank $[\beta] = [2, 2]$, and $\varphi_4 \not\models (Dec_x = -1)$. Then $\delta_4 = [\beta] = [2, 2]$ is modified to $\Delta_4(s) = [\beta, x_s, 0] = [2, 2, x_s, 0]$. For $\delta_5 = [\beta, \gamma] = [2, 2, 1]$ with the same prefix "2,2", it is modified to $\Delta_5(s) = [2, 2, x_s, 1]$.

Concretizing Assertions This process changes the abstract variables back to the concrete ones by reversing the abstraction, and removing propositions involving Dec_x.

Concretizing Requirements This process changes the abstract requirements back to the corresponding concrete ones.

4.4 Proof

We first convert the transitions of the program into formulas as follows.

$$\begin{aligned}\rho = & \; at\_l_0 \wedge x > 0 \wedge at\_l_1' \\ & \vee at\_l_1 \wedge (x' = 2 \vee x' = x - 1) \wedge at\_l_0' \\ & \vee at\_l_0 \wedge \neg(x > 0) \wedge at\_l_2'\end{aligned}$$

p: at_l_0	q: at_l_2		
$\varphi_1 : at\_l_0 \wedge x = 0$	r_1: at_l_0	$u_1 : \neg at\_l_0$	$\Delta_1(s)$: [1]
$\varphi_2 : at\_l_{0,1} \wedge 0 < x \leq 2$	r_2: $at\_l_1 \wedge x = 1$	$u_{2,1} : at\_l_0 \wedge x = 2$,$u_{2,2} : at\_l_0 \wedge x' < x$	$\Delta_2(s)$: [2]
$\varphi_3 : at\_l_0 \wedge x = 1$	r_3: at_l_0	$u_3 : \neg at\_l_0$	$\Delta_3(s)$: [2,1]
$\varphi_4 : at\_l_{0,1} \wedge x = 2$	r_4: at_l_1	$u_{4,1} : at\_l_0 \wedge x = 2$,$u_{4,2} : at\_l_0 \wedge x' < x$	$\Delta_4(s)$: [2,2,x_s,0]
$\varphi_5 : at\_l_0 \wedge x = 2$	r_5: at_l_0	$u_5 : \neg at\_l_0$	$\Delta_5(s)$: [2,2,x_s,1]
$\varphi_6 : at\_l_{0,1} \wedge x > 2$	r_6: at_l_1	$u_{6,1} : at\_l_0 \wedge x = 2$,$u_{6,2} : at\_l_0 \wedge x' < x$	$\Delta_6(s)$: [4,x_s,0]
$\varphi_7 : at\_l_0 \wedge x > 2$	r_7: at_l_0	$u_7 : \neg at\_l_0$	$\Delta_7(s)$: [4,x_s,1]

We prove that the premisses of the rule SC_RESPONSE hold when using the above auxiliary constructs as follows.

$$\begin{array}{lll} \text{R1 p} & \Rightarrow \text{q} \vee \bigvee_{j=1}^{n}(r_j \wedge \varphi_j) \\ at\_l_0 & r_7 \wedge \varphi_7 \\ & r_5 \wedge \varphi_5 \\ & r_3 \wedge \varphi_3 \\ & r_1 \wedge \varphi_1 \end{array}$$

The implication holds, since

- $p = at\_l_0$,
- at_l_0 implies the disjunction of $r_7 \wedge \varphi_7$, $r_5 \wedge \varphi_5$, $r_3 \wedge \varphi_3$ and $r_1 \wedge \varphi_1$,
- this disjunction is a part of $q \vee \bigvee_{j=1}^{n}(r_j \wedge \varphi_j)$.

In the following, for each $i \in \{1, ..., 7\}$, we prove $R2, R3, R4$, and we use the above format without further explanation.

i=1

$$\begin{array}{lll} \text{1R2} & r_1 \wedge \varphi_1 \wedge \rho & \Rightarrow q' \vee \bigvee_{j=1}^{n}((r'_j \wedge \varphi'_j) \\ & at\_l_0 \wedge x = 0 \wedge \rho & at\_l'_2 \\ \text{1R3} & \varphi_1 \wedge \rho & \Rightarrow q' \vee (\varphi_1 \wedge \Delta_1 = \Delta'_1) \vee \bigvee_{j=1}^{n}(r'_j \wedge \varphi'_j \wedge \Delta_1 \succ \Delta'_j) \\ & at\_l_0 \wedge x = 0 \wedge \rho & at\_l'_2 \\ \text{1R4} & \varphi_1 \wedge r_1 \wedge \rho \wedge \varphi'_1 & \Rightarrow \neg u'_{1,k} \\ & \emptyset & at\_l'_0 \end{array}$$

i=2

$$\begin{array}{lll} \text{2R2} & r_2 \wedge \varphi_2 \wedge \rho & \Rightarrow q' \vee \bigvee_{j=1}^{n}((r'_j \wedge \varphi'_j) \\ & at\_l_1 \wedge x = 1 \wedge \rho & r'_3 \wedge \varphi'_3 : at\_l'_0 \wedge x' = 2 \\ & & r'_1 \wedge \varphi'_1 : at\_l'_0 \wedge x' = 0 \\ \text{2R3} & \varphi_2 \wedge \rho & \Rightarrow q' \vee (\varphi'_2 \wedge \Delta_2 = \Delta'_2) \vee \bigvee_{j=1}^{n}(r'_j \wedge \varphi'_j \wedge \Delta_2 \succ \Delta'_j) \\ & at\_l_{0,1} \wedge x = \{1,2\} \wedge \rho & \varphi'_2 : at\_l'_{0,1} \wedge x' = \{1,2\} \; r'_1 \wedge \varphi'_1 : at\_l'_0 \wedge x' = 0 \\ \text{2R4} & \varphi_2 \wedge r_2 \wedge \rho \wedge \varphi'_2 & \Rightarrow \neg u'_{2,k} \\ & at\_l'_0 \wedge x' = 2 & \neg u'_{2,2} \end{array}$$

i=3

$$\begin{array}{lll} \text{3R2} & r_3 \wedge \varphi_3 \wedge \rho & \Rightarrow q' \vee \bigvee_{j=1}^{n}((r'_j \wedge \varphi'_j) \\ & at\_l_0 \wedge x = 1 \wedge \rho & r'_2 \wedge \varphi'_2 : at\_l'_1 \wedge x' = 1 \\ \text{3R3} & \varphi_3 \wedge \rho & \Rightarrow q' \vee (\varphi'_3 \wedge \Delta_3 = \Delta'_3) \vee \bigvee_{j=1}^{n}(p'_j \wedge \varphi'_j \wedge \Delta_3 \succ \Delta'_j) \\ & at\_l_0 \wedge x = 1 \wedge \rho & r'_2 \wedge \varphi'_2 : at\_l'_1 \wedge x' = 1 \\ \text{3R4} & \varphi_3 \wedge r_3 \wedge \rho \wedge \varphi'_3 & \Rightarrow \neg u'_{3,k} \\ & \emptyset & at\_l'_0 \end{array}$$

i=4

$$
\begin{array}{lll}
\text{4R2} & r_4 \wedge \varphi_4 \wedge \rho & \Rightarrow q' \vee \bigvee_{j=1}^{n}((r'_j \wedge \varphi'_j) \\
& at\_l_1 \wedge x = 2 \wedge \rho & \quad r'_3 \wedge \varphi'_3 : at\_l'_0 \wedge x' = 1 \\
& & \quad r'_5 \wedge \varphi'_5 : at\_l'_0 \wedge x' = 2 \\
\text{4R3} & \varphi_4 \wedge \rho & \Rightarrow q' \vee (\varphi'_4 \wedge \Delta_4 = \Delta'_4) \vee \quad \bigvee_{j=1}^{n}(r'_j \wedge \varphi'_j \wedge \Delta_4 \succ \Delta'_j) \\
& at\_l_{0,1} \wedge x = 2 \wedge \rho & \quad \varphi'_4 : at\_l'_{0,1} \wedge x' = 2 \; r'_3 \wedge \varphi'_3 : at\_l'_0 \wedge x' = 1 \\
\text{4R4} & \varphi_4 \wedge r_4 \wedge \rho \wedge \varphi'_4 & \Rightarrow \neg u'_{4,k} \\
& at\_l'_0 \wedge x' = 2 & \quad \neg u'_{4,2}
\end{array}
$$

i=5

$$
\begin{array}{lll}
\text{5R2} & r_5 \wedge \varphi_5 \wedge \rho & \Rightarrow q' \vee \bigvee_{j=1}^{n}((r'_j \wedge \varphi'_j) \\
& at\_l_0 \wedge x = 2 \wedge \rho & \quad r'_4 \wedge \varphi'_4 : at\_l'_1 \wedge x' = 2 \\
\text{5R3} & \varphi_5 \wedge \rho & \Rightarrow q' \vee (\varphi'_5 \wedge \Delta_5 = \Delta'_5) \vee \bigvee_{j=1}^{n}(r'_j \wedge \varphi'_j \wedge \Delta_5 \succ \Delta'_j) \\
& at\_l_0 \wedge x = 2 \wedge \rho & \quad r'_4 \wedge \varphi'_4 : at\_l'_1 \wedge x' = 2 \\
\text{5R4} & \varphi_5 \wedge r_5 \wedge \rho \wedge \varphi'_5 & \Rightarrow \neg u'_{5,k} \\
& \emptyset & \quad at\_l'_0
\end{array}
$$

i=6

$$
\begin{array}{lll}
\text{6R2} & r_6 \wedge \varphi_6 \wedge \rho & \Rightarrow q' \vee \bigvee_{j=1}^{n}((r'_j \wedge \varphi'_j) \\
& at\_l_1 \wedge x > 2 \wedge \rho & \quad r'_7 \wedge \varphi'_7 : at\_l'_0 \wedge x' > 2 \\
& & \quad r'_5 \wedge \varphi'_5 : at\_l'_0 \wedge x' = 2 \\
\text{6R3} & \varphi_6 \wedge \rho & \Rightarrow q' \vee (\varphi'_6 \wedge \Delta_6 = \Delta'_6) \vee \qquad \bigvee_{j=1}^{n}(r'_j \wedge \varphi'_j \wedge \Delta_6 \succ \Delta'_j) \\
& at\_l_{0,1} \wedge x > 2 \wedge \rho & \quad \varphi'_6 : at\_l'_{0,1} \wedge x' > 2 \wedge x = x' \; r'_5 \wedge \varphi'_5 : at\_l'_0 \wedge x' = 2 \\
& & \qquad\qquad\qquad\qquad\qquad\quad r_7 \wedge \varphi_7 : at\_l'_0 \wedge x' > 2 \wedge x > x' \\
& & \qquad\qquad\qquad\qquad\qquad\quad r'_6 \wedge \varphi'_6 : at\_l'_1 \wedge x' > 2 \wedge x > x' \\
\text{6R4} & \varphi_6 \wedge r_6 \wedge \rho \wedge \varphi'_6 & \Rightarrow \neg u'_{6,k} \\
& at\_l'_0 \wedge x' > 2 & \quad \neg u'_{6,1}
\end{array}
$$

i=7

$$
\begin{array}{lll}
\text{7R2} & r_7 \wedge \varphi_7 \wedge \rho & \Rightarrow q' \vee \bigvee_{j=1}^{n}((r'_j \wedge \varphi'_j) \\
& at\_l_0 \wedge x > 2 \wedge \rho & \quad r'_6 \wedge \varphi'_6 : at\_l'_1 \wedge x' > 2 \\
\text{7R3} & \varphi_7 \wedge \rho & \Rightarrow q' \vee (\varphi'_7 \wedge \Delta_7 = \Delta'_7) \vee \bigvee_{j=1}^{n}(r'_j \wedge \varphi'_j \wedge \Delta_7 \succ \Delta'_j) \\
& at\_l_0 \wedge x > 2 \wedge \rho & \quad r'_6 \wedge \varphi'_6 : at\_l'_1 \wedge x' > 2 \\
\text{7R4} & \varphi_7 \wedge r_7 \wedge \rho \wedge \varphi'_7 & \Rightarrow \neg u'_{7,k} \\
& \emptyset & \quad at\_l'_1
\end{array}
$$

4.5 Discussion

For proving the termination of the above program, strengthened compassion requirements are necessary. Compassion requirements are not sufficient to constrain the strongly connected component $\{S_2, S_3, S_4, S_5, S_6\}$.

The strengthened compassion requirements require that when $x = 1$ infinitely often at l_0, x must decrease right after at least once, and if this happens, the

program will terminate. However, similar compassion requirements require only that when $x = 1$ infinitely often at l_0, x must decrease some time in the future at least once, however x may increase before it decreases, and this is not sufficient for leading to termination.

Concretely, the infinite loop $(S_2S_5S_4S_3)^\omega$ in the pending graph in Fig. 4 is fair to this compassion interpretation, and is not fair to the strengthened compassion interpretation of the requirements.

5 Concluding Remarks

Strengthened compassion requirements have been studied. This kind of requirements has been compared with compassion requirements, which shows the difference between the expressive powers of these two kinds of fairness requirements. A deductive rule SC_RESPONSE for proving response properties with such requirements has been presented. Proofs of the soundness and the relative completeness of the rule have also been provided, and the application of the rule has been illustrated.

References

1. Lehmann, D.J., Pnueli, A., Stavi, J.: Impartiality, justice and fairness: The ethics of concurrent termination. In: ICALP. (1981) 264–277
2. Pnueli, A., Sa'ar, Y.: All you need is compassion. In: VMCAI. (2008) 233–247
3. Lamport, L.: Proving the correctness of multiprocess programs. IEEE Trans. Software Eng. **3**(2) (1977) 125–143
4. Fischer, M.J., Jiang, H.: Self-stabilizing leader election in networks of finite-state anonymous agents. In: OPODIS. (2006) 395–409
5. Angluin, D., Aspnes, J., Fischer, M.J., Jiang, H.: Self-stabilizing population protocols. In: OPODIS. (2005) 103–117
6. Dijkstra, E.W.: Self-stabilizing systems in spite of distributed control. Commun. ACM **17**(11) (1974) 643–644
7. Manna, Z., Pnueli, A.: Completing the temporal picture. Theor. Comput. Sci. **83**(1) (1991) 91–130
8. Sun, J., Liu, Y., Dong, J.S., Pang, J.: Pat: Towards flexible verification under fairness. In: CAV. (2009) 709–714
9. Kesten, Y., Pnueli, A.: Verification by augmented finitary abstraction. Inf. Comput. **163**(1) (2000) 203–243
10. Balaban, I., Pnueli, A., Zuck, L.D.: Modular ranking abstraction. Int. J. Found. Comput. Sci. **18**(1) (2007) 5–44
11. Long, T., Zhang, W.: Auxiliary constructs for proving liveness in compassion discrete systems. In: ATVA. (2010) 276–290
12. Balaban, I., Pnueli, A., Zuck, L.D.: Proving the refuted: Symbolic model checkers as proof generators. In: Concurrency, Compositionality, and Correctness. (2010) 221–236
13. Pnueli, A.: On the extremely fair treatment of probabilistic algorithms. In: STOC. (1983) 278–290

14. Manna, Z., Pnueli, A.: Temporal verification of reactive systems: Response. In: Essays in Memory of Amir Pnueli. (2010) 279–361
15. Main, M.G.: Complete proof rules for strong fairness and strong extreme fairness. Theor. Comput. Sci. **111**(1&2) (1993) 125–143