All You Need is Compassion*

Amir Pnueli[1,2], and Yaniv Sa'ar[2]

[1] New York University, New York, amir@cs.nyu.edu
[2] Weizmann Institute of Science, yaniv.saar@weizmann.ac.il

Abstract. The paper presents a new deductive rule for verifying response properties under the assumption of compassion (strong fairness) requirements. It improves on previous rules in that the premises of the new rule are all first order. We prove that the rule is sound, and present a constructive completeness proof for the case of finite-state systems. For the general case, we present a sketch of a relative completeness proof. We report about the implementation of the rule in PVS and illustrate its application on some simple but non-trivial examples.

1 Introduction

An important component of the formal model of reactive systems is a set of *fairness requirements*. As suggested by Lamport [13], these should come in two flavors: *weak fairness* (to which we refer as *justice requirements*) and *strong fairness* (to which we refer as *compassion*). Originally, these two distinct notions of fairness were formulated in terms of enableness and the activation of transitions within a computation, as follows:

- The requirement that transition τ is *just* implies that if τ is continuously enabled from a certain position on, then it is taken (activated) infinitely many times.
  An equivalent formulation is that every computation should contain infinitely many positions at which τ is disabled or has just been taken.
- The requirement that transition τ is *compassionate* implies that if τ is enabled infinitely many times in a computation σ, then it is taken infinitely many times.

Justice requirements are used in order to guarantee that, in a parallel composition of processes, no process is neglected forever from a certain point on. Compassion, which is a more stringent requirement, is often associated with coordination statements such as semaphore *request* y (equivalently *lock* y) operations or message passing instructions. It implies fair arbitration in the allocation of an atomic resource among several competing processes.

In a more abstract setting, a justice requirement is associated with an *assertion* (first-order state formula) J, while a compassion requirement is associated with a pair of assertions $\langle p, q\rangle$. With these identifications, the requirements are:

- A computation σ is just with respect to the requirement J, if σ contains infinitely many occurrences of states that satisfy J.
- A computation σ is compassionate with respect to the requirement $\langle p, q\rangle$, if either σ contains only finitely many p-positions or σ contains infinitely many q-positions.

* This research was supported in part by ONR grant N00014-99-1-0131.

To see that these definitions are indeed generalizations of the transition-oriented definition, we observe that the requirement that transition τ be just can be expressed by the abstract justice requirement $J_\tau = (\neg En(\tau) \vee \mathit{Taken}(\tau))$, while the requirement that transition τ be compassionate can be expressed by the abstract compassion requirement $C_\tau = \langle En(\tau), \mathit{Taken}(\tau)\rangle$. In these assertions, $En(\tau)$ is true at all states on which τ is enabled. Similarly, $\mathit{Taken}(\tau)$ is true at all states that can result by taking transition τ.

An important observation is that justice is a special case of compassion. This is because the justice requirement J can also be expressed as the degenerate compassion requirement $\langle 1, J\rangle$, where we write 1 to denote the assertion *True* which holds at every state. In view of this observation, one may raise the natural question of the necessity of keeping these two separate notions of fairness.

Several answers can be given to this question. On the modeling level, the argument is that these two notions represent different phenomena. Justice represents the natural independence of parallel processes in a multi-processing system. Compassion is typically used to provide an abstract representation of queues and priorities which are installed by the operating system in order to guarantee fairness in coordination services provided to parallel processes. There is also a different cost associated with the implementation of these two notions. In a multi-processor system, justice comes for free and is a result of the independent progress of parallel processes. In a multi-programming system, where concurrency is simulated by scheduling, justice can be implemented by any scheduling scheme that gives each process a fair chance to progress, such as round-robin scheduling. Compassion, in both types of systems, is usually implemented by maintenance of queues and use of priorities.

There is also a proof-theoretic answer to this question which is based on the fact that, up to now, all the proposed deductive rules for proving properties under the assumption of compassion were significantly more complex than the rule under the assumption of justice alone. The main claim of this paper is this need not necessarily be the case, and there exist deductive rules for verification in which the price of compassion is comparable to that of justice.

1.1 The Legacy Recursive Rule

In the way of a background, we present rule F-WELL which is derived from the proof rule presented in [15] and is representative of the different prices traditionally associated with the distinct notions of fairness. It is modified in order to represent the transition from the computational model of *fair transition systems* (used in [15]) to that of *fair discrete systems* (FDS) which we use here. The rule is presented in Fig. 1.

The FDS $(\mathcal{D}\backslash\{\langle p_i, q_i\rangle\})$ is obtained by removing from $\mathcal{D}$ the compassion requirement $\langle p_i, q_i\rangle$. Thus, $(\mathcal{D}\backslash\{\langle p_i, q_i\rangle\})$ has one compassion requirement less than $\mathcal{D}$.

The rule considers a system (FDS) which has both justice requirements ($\mathcal{J}$) and compassion requirements ($\mathcal{C}$). It establishes for this system the temporal property $p \Longrightarrow \diamondsuit\, q$ claiming that every p-state is followed by a q-state. The rule relies on "helpful" fairness requirements $F_1, \ldots, F_n$ which may be either justice or compassion requirements. Premise W3 imposes different conditions on each fairness requirement F_i according to whether F_i is a compassion or a justice requirement.

Rule F-WELL

For a well-founded domain $\mathcal{A} : (W, \succ)$,
assertions $p, q, \varphi_1, \ldots, \varphi_n$,
fairness requirements $F_1, \ldots, F_n \in \mathcal{J} \cup \mathcal{C}$,
and ranking functions $\Delta_1, \ldots, \Delta_n$ where each $\Delta_i : \Sigma \mapsto W$

W1. $p \implies q \vee \bigvee_{j=1}^{n} \varphi_j$

For each $i = 1, \ldots, n$,

W2. $\varphi_i \wedge \rho \implies q' \vee (\varphi_i' \wedge \Delta_i = \Delta_i') \vee \left(\bigvee_{j=1}^{n} (\varphi_j' \wedge \Delta_i \succ \Delta_j')\right)$

W3. If $F_i = \langle p_i, q_i \rangle \in \mathcal{C}$ then

C3. $\varphi_i \implies \neg q_i$

C4. $(\mathcal{D} \backslash \{\langle p_i, q_i \rangle\}) \models (\varphi_i \implies \Diamond (p_i \vee \neg \varphi_i))$

Otherwise ($F_i = J_i \in \mathcal{J}$),

J3. $\varphi_i \implies \neg J_i$

$\mathcal{D} \models (p \implies \Diamond q)$

Fig. 1. Legacy (recursive) rule F-WELL

Consider first the special case in which all the helpful requirements are justice requirements. In this case, we only invoke premise J3 as an instance of W3. For such a case, the rule provides a real reduction by establishing a temporal property, based on premises which are all first-order.

On the other hand, if some of the helpful requirements are compassionate, then some of the premises will include instances of C3 and C4. In this case, some of the premises are temporal and have a syntactic form similar to the of the conclusion. In such a case, one may ask whether this is not a circular rule in which the premises are not necessarily simpler than the conclusion. As observed above, the rule is not really circular because the premise C4 requires the establishment of a similar temporal property but over a system with fewer compassion requirements. So while the methodology is still sound, it appears cumbersome and its application often requires explicit induction on the number of compassion requirements in the analyzed system.

This explanation serves to illustrate that the application of this rule is significantly more complex and cumbersome in the case that we have compassion requirements, and the situation is mush simpler if all the fairness requirements are of the justice type. We refer to this phenomenon by saying that the application of this rule is *recursive* in the presence of compassion requirements.

1.2 A New Flat Rule

The main result of this paper is based on a new deductive rule for response properties which does not need any recursion in order to handle compassion requirements. The rule, called RESPONSE, is presented in Fig. 2.

For simplicity, we presented the rule for the case that the system contains only compassion requirements but no justice requirements. This is not a serious restriction since any original justice requirement $J \in \mathcal{J}_{\mathcal{D}}$ can be represented by an equivalent

Rule RESPONSE

For a well-founded domain $\mathcal{A} : (W, \succ)$,
assertions $p, q, \varphi_1, \ldots, \varphi_n$,
compassion requirements $\langle p_1, q_1 \rangle, \ldots, \langle p_n, q_n \rangle \in \mathcal{C}$,
and ranking functions $\Delta_1 , \ldots , \Delta_n$ where each $\Delta_i : \Sigma \mapsto W$

R1. $p \implies q \vee \bigvee_{j=1}^{n} (p_j \wedge \varphi_j)$

For each $i = 1, \ldots, n$,

R2. $p_i \wedge \varphi_i \wedge \rho \implies q' \vee \bigvee_{j=1}^{n} (p'_j \wedge \varphi'_j)$

R3. $\varphi_i \wedge \rho \implies q' \vee (\varphi'_i \wedge \Delta_i = \Delta'_i) \vee \bigvee_{j=1}^{n} (p'_j \wedge \varphi'_j \wedge \Delta_i \succ \Delta'_j)$

R4. $\varphi_i \implies \neg q_i$

$p \implies \diamondsuit q$

Fig. 2. Deductive rule RESPONSE

compassion requirement $\langle 1, J \rangle$. Similarly to the previous version of this rule, the rule relies on a set of premises guaranteeing that a computation which contains a p-state that is not followed by a q-state leads to an infinite chain of descending ranks. Since the ranks range over a well-founded domain $\mathcal{A} : (W, \succ)$, this leads to a contradiction.

In view of the simple form of the rule, it appears that, in many cases, the study and analysis of fair discrete systems can concentrate on the treatment of compassion requirements, and deal with justice requirements as a special case of a compassion requirement. This does not imply that we suggest giving up the class of justice requirements altogether. For modeling and implementation of reactive systems, we should keep these two classes of fairness requirements distinct. However, the main message of this paper is that, when verifying temporal properties of FDS's, the treatment of compassion requirements is conceptually not more complex than the treatment of justice requirements. Computationally, though, justice is simpler in the same way that checking emptiness of generalized Büchi automata is simpler than checking emptiness of Street automata.

The new rule has been implemented in the theorem prover PVS [18]. In fact, it has been added as an additional rule (and associated strategy) within the PVS-*based temporal prover* TLPVS [19]. In order to do so, we had to prove the soundness of the RESPONSE rule within PVS.

The rest of the paper is organized as follows. In Section 2 we introduce the computational model of fair discrete systems with its related notions of fairness. We then illustrate the application of the new rule to three examples of increasing complexity in Section 3. The soundness of the rule is stated and proved in Section 4. Section 5 discusses the question of completeness. For finite-state systems, we present a constructive proof of completeness which can be used in order to obtain the auxiliary constructs required for an application of the rule. For infinite-state systems, we sketch a proof of completeness which is based on a reduction to the proof of completeness of rule F-WELL, as presented in [15]. Finally, in Section 6 we discuss some related work.

2 The Computational Model

As a computational model, we take a *fair discrete system* (FDS) $S = \langle V, \Theta, \rho, \mathcal{J}, \mathcal{C} \rangle$, where

- V — A set of *system variables*. A *state* of S provides a type-consistent interpretation of the variables V. 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. Let Σ denote the set of all states over V.
- Θ — The *initial condition*: An assertion (state formula) characterizing the initial states.
- $\rho(V, V')$ — The *transition relation*: An assertion, relating the values V of the variables in state $s \in \Sigma$ to the values V' in an $\mathcal{D}$-successor state $s' \in \Sigma$. For a subset $U \subseteq V$ we define *pres*(U) as $\bigwedge_{v \in U}(v' = v)$. We assume that the system can always *idle*, that is, we assume that ρ has the disjunct *pres*(V).
- $\mathcal{J}$ — A set of *justice* (*weak fairness*) requirements (assertions); A computation must include infinitely many states satisfying each of the justice requirements.
- $\mathcal{C}$ — A set of *compassion (strong fairness)* requirements: Each compassion requirement is a pair $\langle p, q \rangle$ of state assertions; A computation should include either only finitely many p-states, or infinitely many q-states.

For an assertion ψ, we say that $s \in \Sigma$ is a ψ-state if $s \models \psi$. A *computation* of an FDS S is an infinite sequence of states $\sigma : s_0, s_1, s_2, ...$, satisfying the requirements:

- *Initiality*: s_0 is initial, i.e., $s_0 \models \Theta$.
- *Consecution*: For each $\ell = 0, 1, ...$, state $s_{\ell+1}$ is a $\mathcal{D}$-successor of s_ℓ. I.e., $\langle s_\ell, s_{\ell+1} \rangle \models \rho(V, V')$ where, for each $v \in V$, we interpret v as $s_\ell[v]$ and v' as $s_{\ell+1}[v]$.
- *Justice* — for every $J \in \mathcal{J}$, σ contains infinitely many occurrences of J-states.
- *Compassion*: for every $\langle p, q \rangle \in \mathcal{C}$, either σ contains only finitely many occurrences of p-states, or σ contains infinitely many occurrences of q-states.

A state s is called *accessible* if there exists a path of ρ-steps leading from some initial state to s.

A system with no compassion requirements is called a *just discrete system* (JDS). A system with no justice requirements is called a *compassionate discrete system* (CDS). As previously observed, every FDS is equivalent to a CDS. Therefore, we do not lose generality if we present the verification rule RESPONSE for CDS's.

As a specification language specifying properties of systems we use linear temporal logic (LTL) as presented, for example in [16]. In particular, we are interested in response formulas of the form $p \Longrightarrow \diamondsuit\, q$ (abbreviating $\square(p \rightarrow \diamondsuit\, q)$). This formula states that every p-state is followed by a q-state. Given an FDS $\mathcal{D}$ and an LTL formula φ, we say that φ is *valid over* $\mathcal{D}$ (is $\mathcal{D}$-valid) if every computation of $\mathcal{D}$ satisfies φ.

Reconsider rule RESPONSE as presented in Fig. 2. The rule assumes a well founded domain $\mathcal{A} : (W, \succ)$. Such a domain consists of a non-empty set W and a partial order $\succ$ over W, such that there does not exists an infinite descending chain $\alpha_0 \succ \alpha_1 \succ a_2 \succ \cdots$, of elements $a_i \in W$. The rule establishes the validity of the response property $p \Longrightarrow \diamondsuit\, q$ over a CDS $\mathcal{D}$, where p and q are assertions (first-order state formulas). In order to do so, the rule focuses on a list of (not necessarily disjoint) compassion

requirements $\langle p_1, q_1\rangle, \ldots \langle p_n, q_n\rangle \in \mathcal{C}$. With each compassion requirement $\langle p_i, q_i\rangle$ we associate a *helpful assertion* φ_i and a *ranking function* $\Delta_i : \Sigma \mapsto W$, mapping states of $\mathcal{D}$ into elements of $\mathcal{A}$.

One of the differences between the application of rule RESPONSE and that of rule F-WELL is that, in applications of F-WELL, the helpful assertions are typically disjoint. That is, every state satisfies at most one helpful assertion. In applications of rule RESPONSE there is typically an overlap among the helpful assertions. Typically, for a compassion requirement $\langle p_i, q_i\rangle$, where p_i is not identically true, every state satisfying $\varphi_i \wedge \neg p_i$ satisfies at least one more helpful assertion φ_j for $j \neq i$.

3 Examples of Verification

In this section we present three examples illustrating the application of rule RESPONSE to proofs of response properties of simple programs. In all cases we assume that the programs are represented as CDS in which we encoded the justice requirements as degenerate compassion requirement of the form $\langle 1, J\rangle$.

Example 1 (Conditional Termination).
Consider program COND-TERM presented in Fig. 3.

$$
\boxed{
\begin{array}{c}
x, y : \textbf{natural init } x = 0 \\
\left[\begin{array}{l}
\ell_1 : \textbf{while } y > 0 \textbf{ do} \\
\quad \left[\begin{array}{l} \ell_2 : x := \{0, 1\} \\ \ell_3 : y := y + 1 - 2x \end{array}\right] \\
\ell_4 :
\end{array}\right]
\end{array}
}
$$

Fig. 3. Conditionally Terminating Program COND-TERM

This program has three standard justice requirements $J_1, \ldots, J_3$, associated with the statements $\ell_1, \ldots, \ell_3$. The justice requirement associated with ℓ_i is $J_i : \neg at\_\ell_i$ which requires that the program makes infinitely many visits to a location which is different from ℓ_i, thus guaranteeing that execution does not get stuck at location ℓ_i. Such a requirement is necessary in the context of concurrent programs. In addition to these three justice requirements, we also append to the system the compassion requirement $\langle at\_\ell_3 \wedge x = 0,\ 0\rangle$, requiring that there will be only finitely many states satisfying $at\_\ell_3 \wedge x = 0$. This implies that, from a certain point on, all visits to ℓ_3 must be with a positive value of x. Obviously, under these fairness requirements, program COND-TERM must terminate.

To prove this fact, using rule RESPONSE, we choose the following constructs for $n = 4$, where F_i is the i'th compassion requirement appearing in the list:

i	F_i	φ_i	Δ_i
1	$\langle 1, \neg at\_\ell_1 \rangle$	at_ℓ_1	$(1, y, 2)$
2	$\langle 1, \neg at\_\ell_2 \rangle$	$at\_\ell_2 \wedge y > 0$	$(1, y, 1)$
3	$\langle 1, \neg at\_\ell_3 \rangle$	$at\_\ell_3 \wedge y > 0 \wedge x = 1$	$(1, y, 0)$
4	$\langle at\_\ell_3 \wedge x = 0,\ 0 \rangle$	$at\_\ell_{1..3} \wedge y \geq at\_\ell_{2,3} \wedge x \in \{0, 1\}$	1

As assertions p and q we take at_ℓ_1 and at_ℓ_4, respectively. The well-founded domain is the domain of triples of natural numbers. The rank $\Delta_4 : 1$ is an abbreviation of the triple $(1, 0, 0)$. The application of this rule already demonstrates the phenomenon of overlap among the helpful assertions. For example, every state satisfying φ_4 satisfies also φ_i for some $i \in [1..3]$.

The validity of the premises of rule RESPONSE for this choice has been verified using the theorem prover PVS [18]. □

Next we consider the example of *mutual exclusion by semaphores*.

Example 2 (Muxsem).
Consider program MUX-SEM presented in Fig. 4.

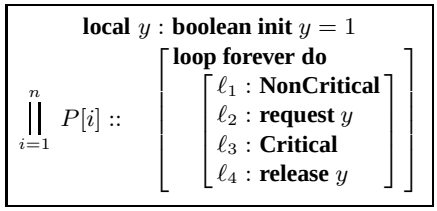

Fig. 4. Mutual Exclusion by Semaphores. Program MUX-SEM

For this program we wish to prover the response property

$$at\_\ell_2[1] \implies \diamondsuit\, at\_\ell_3[1]$$

To prove this property, using rule RESPONSE, we choose the following constructs:

i	F_i	φ_i	Δ_i
1	$\langle at\_\ell_2[1] \wedge y, at\_\ell_3[1] \rangle$	$at\_\ell_2[1]$	0
$(3, j), j > 1$	$\langle 1, \neg at\_\ell_3[j] \rangle$	$at\_\ell_2[1] \wedge at\_\ell_3[j]$	2
$(4, j), j > 1$	$\langle 1, \neg at\_\ell_4[j] \rangle$	$at\_\ell_2[1] \wedge at\_\ell_4[j]$	1

We also use the following auxiliary invariant:

$$\varphi : \quad y + \sum_{i=1}^{n} at\_\ell_{3,4}[i] \;=\; 1$$

□

Finally, we consider the example of *dining philosophers*.

Example 3 (Dining Philosophers).
Consider program MUX-SEM presented in Fig. 5.

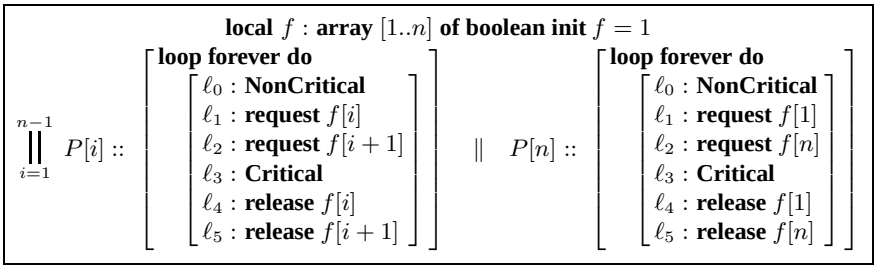

Fig. 5. Dining Philosophers. Program DINE-PHIL

For this program we wish to prover the response property

$$at\_\ell_1[2] \implies \diamondsuit\, at\_\ell_3[2]$$

which captures the property of *accessibility* for process $P[2]$. According to this property, whenever $P[2]$ exits its non-critical section ℓ_0, it will eventually enter its critical section at location ℓ_3.

In order to enumerate the fairness requirements (as well as transitions) for this system we use the indexing scheme (j, k), where $j : 1..n$ ranges over process indices, and $k : 0..5$ ranges over locations. To prove this property, using rule RESPONSE, we choose the following constructs:

(j,k)	$F_{(j,k)}$	$\varphi_{(j,k)}$	$\Delta_{(j,k)}$
$(2,2)$	$\langle at\_\ell_2[2] \wedge f[3], at\_\ell_3[2]\rangle$	$at\_\ell_2[2]$	0
$(j : [3..n], k : [3..4])$	$\langle 1, \neg at\_\ell_k[j]\rangle$	$\bigwedge_{i=2}^{j-1} at\_\ell_2[i]) \wedge at\_\ell_k[j]$	$(0, j, 5-k)$
$(j : [3..n-1], 2)$	$\langle at\_\ell_2[j] \wedge f[j+1], at\_\ell_3[j]\rangle$	$\bigwedge_{i=2}^{j} at\_\ell_2[i]$	$(0, j, 3)$
$(n, 5)$	$\langle 1, \neg at\_\ell_5[n]\rangle$	$\bigwedge_{i=2}^{n-1} at\_\ell_2[i]) \wedge at\_\ell_5[n]$	$(0, n, 0)$
$(2, 1)$	$\langle at\_\ell_1[2] \wedge f[2], at\_\ell_2[2]\rangle$	$at\_\ell_1[2]$	1
$(1, k : [3..5])$	$\langle 1, \neg at\_\ell_k[1]\rangle$	$at\_\ell_1[2] \wedge at\_\ell_k[1]$	$(1, 1, 5-k)$

We also use the following auxiliary invariant:

$$\begin{array}{ll} \bigwedge_{i=1}^{n-2}(at\_\ell_{3..5}[i] + at\_\ell_{2..4}[i+1] + f[i+1] = 1) \wedge \\ at\_\ell_{3..5}[n-1] + at\_\ell_{3..5}[n] + f[n] = 1 & \wedge \\ at\_\ell_{2..4}[1] + at\_\ell_{2..4}[n] + f[1] = 1 & \end{array}$$

The validity of the premises of rule RESPONSE for this choice has been verified using the theorem prover PVS. In Appendix A of [21] we present the program in the format accepted by TLPVS. Appendix B of [21] presents the TLPVS proof of accessibility according to the previously presented constructs. □

It is interesting to compare the proof of accessibility for the Dining Philosophers (program DINE-PHIL) to previous deductive proofs of the same property. Due to the recursiveness of the previous rule, such proofs would necessarily be based on an explicit induction, proceeding from process $P[n]$ down to lower indexed processes. Such a proof is presented, for example, in [17]. In view of this, it is fair to say that the proof presented here is the first mechanically assisted deductive proof of accessibility for program DINE-PHIL for arbitrary $n > 1$ number of processes.

Alternative formal proofs of this property can be given, based on various abstraction methods. For example, the papers [12] and [9] show how to prove this property, using the *network invariant method* of [28].

4 Soundness of the Rule

We will now prove the soundness of rule RESPONSE for proving the response property $p \Longrightarrow \diamondsuit q$.

Claim 1. Rule RESPONSE is sound. That is, if the premises of the rule are valid over an CDS $\mathcal{D}$, then so is the conclusion.

Proof

Assume, for a proof by contradiction, that the premises of the rule are valid but the conclusion is not. This 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. With no loss of generality, we can take $j = 0$.

According to premises R1 and R2 and the assumption that no state satisfies q, any state s_r satisfies $p_i \wedge \varphi_i$ for some $i \in [1..n]$. Since there are only finitely many different i's, there exists a cutoff index $t \geq 0$ such that for every i and $r \geq t$, $s_r \models p_i \wedge \varphi_i$ iff σ contains infinitely many $(p_i \wedge \varphi_i)$-positions.

Consider position $r_1 = t$. Choose i_1 to be the index such that $s_{r_1} \models p_{i_1} \wedge \varphi_{i_1}$. According to R3 and the assumption that σ contains no q-position, then either φ_{i_1} holds at all positions $r \geq r_1$, or there exists a position $r_2 \geq r_1$ and an index i_2 such that $s_{r_2} \models p_{i_2} \wedge \varphi_{i_2}$ and $\Delta_{i_1}(s_{r_1}) \succ \Delta_{i_2}(s_{r_2})$. We will show that φ_{i_1} cannot hold at all positions $r \geq r_1$.

If φ_{i_1} holds continuously beyond r_1, then due to premise R4 so does $\neg q_{i_1}$. This violates the requirement of compassion $\langle p_{i_1}, q_{i_1} \rangle$, since $p_{i_1} \wedge \varphi_{i_1}$ holding at $r_1 \geq t$ implies that $p_{i_1} \wedge \varphi_{i_1}$ (and therefore p_{i_1}) holds at infinitely many positions.

We conclude that there exists a position $r_2 \geq r_1$ and an index i_2 such that $s_{r_2} \models p_{i_2} \wedge \varphi_{i_2}$ and $\Delta_{i_1}(s_{r_1}) \succ \Delta_{i_2}(s_{r_2})$.

We now repeat the argument previously applied to i_1 in order to conclude the existence of a position $r_3 \geq r_2$ and an index i_3 such that $s_{r_3} \models p_{i_3} \wedge \varphi_{i_3}$ and $\Delta_{i_2}(s_{r_2}) \succ \Delta_{i_3}(s_{r_3})$.

Continuing in this manner, we derive an infinite sequence such that

$$\Delta_{i_1}(s_{r_1}) \succ \Delta_{i_2}(s_{r_2}) \succ \Delta_{i_3}(s_{r_3}) \succ \cdots$$

which is impossible due to the well-foundedness of $\mathcal{A}$.

We conclude that there cannot exist a computation σ violating the response property $p \Longrightarrow \diamondsuit\, q$ if the premises of rule RESPONSE are all valid. ▟

This proof of soundness has been formalized and mechanically verified, using PVS. In Appendix C of [21] we present the PVS theory of a variant of the rule. Appendix D presents the proof of soundness of this variant.

5 Completeness of the Rule

In this section we consider the completeness of rule RESPONSE. Completeness means that, whenever a response property $p \Longrightarrow \diamondsuit\, q$ is valid over a CDS $\mathcal{D}$, there exist auxiliary constructs consisting of a list of compassion requirements $\langle p_1, q_1\rangle, \ldots, \langle p_n, q_n\rangle$ and associated lists of helpful assertions $\varphi_1, \ldots, \varphi_n$ and ranking functions $\mathit{\Delta}_1, \ldots, \mathit{\Delta}_n$ which, together, satisfy the premises of rule RESPONSE.

5.1 Finite-State Systems

We consider first the case of finite-state systems. Here, we actually present an algorithm that produces the auxiliary constructs for the case that the CDS $\mathcal{D}$ satisfies the response property $p \Longrightarrow \diamondsuit\, q$.

We assume that the reader is familiar with the rudiments of model checking. In particular, we will be using the following formulas as representing elementary model-cheking computations:

- For assertions p and q, the formula $\mathbf{E}(p\,\mathcal{S}\,q)$ captures the set of all states that are reachable from a q-state by a (ρ-)path all of whose states, except possibly the first, satisfy q. A special case is $\mathbf{E} \diamondsuit\!\!\!\!-\, q \;=\; \mathbf{E}(1\,\mathcal{S}\,q)$ characterizing all states that are reachable from a q-state.
- The formula $\mathbf{E}(p\,\mathcal{U}\,q)$ captures the set of states which originate a path leading to a q-state, such that all the states in the path, except possibly the last, satisfy p.

Model Checking Response Under Compassion Consider a finite-state CDS $\mathcal{D}$ and a response property $p \Longrightarrow \diamondsuit\, q$. The following algorithm can be used in order to model check whether the response property is valid over $\mathcal{D}$.

Algorithm $mc\_resp(\mathcal{D}, p, q)$
Let $X := \mathbf{E}((\neg q)\,\mathcal{S}\,(\mathit{accessible}_{\mathcal{D}} \wedge p \wedge \neg q))$
Fix(X)
  Let $X := X \wedge \bigwedge_{i=1}^{n}(\neg p_i \;\vee\; \mathbf{E}(X\,\mathcal{U}\,(X \wedge q_i)))$
end-Fix(X)
if $(X = 0)$ **then return** 1 **else return** 0

The algorithm contains several expressions using the model-checking formulas introduced above. The expression $\mathit{accessible}_{\mathcal{D}} = \mathbf{E} \diamondsuit\!\!\!\!-\, \Theta$ captures the set of all accessible states within CDS $\mathcal{D}$. The expression $\mathbf{E}((\neg q)\,\mathcal{S}\,(\mathit{accessible}_{\mathcal{D}} \wedge p \wedge \neg q))$ describes all

states which are reachable from an accessible p-state by a finite q-free path. Finally, the expression $\mathbf{E}(X\,\mathcal{U}(X \wedge q_i))$ describes all states from which there exists an X-path within $\mathcal{D}$ leading to a q_i-state.

Thus, the algorithm places in X the set of all states which are reachable from an accessible p-state via a q-free path. Then it successively remove from X all p_i-states which do not initiate an X-path leading to a q_i-state.

Finally, the property is $\mathcal{D}$-valid iff the final value of X is empty.

Extracting the Auxiliary Constructs We will now present an algorithm which extracts a deductive proof according to rule RESPONSE from a successful model checking verification of a response property $p \implies \diamondsuit\, q$. The algorithm can be viewed as an interleaving of algorithm mc_resp interspersed with steps that incrementally construct ranks $\{1, \ldots, f\}$ and, for each rank $i \in [1..f]$, identifies an associated compassion requirement $\langle p_i, q_i \rangle$, and a helpful assertion φ_i.

Algorithm *Extract_constructs*

0 : **Let** $X := \mathbf{E}((\neg q)\,\mathcal{S}\,(accessible_{\mathcal{D}} \wedge p \wedge \neg q))$
1 : **Let** $d := 0$
2 : **fix**(X)
  3 : **For** $i \in [1...n]$ **do**
    4 : **Let** $\psi := X \ \wedge\ \neg\mathbf{E}(X\,\mathcal{U}(X \wedge q_i))$
    5 : **If** $\psi \wedge p_i \neq 0$ **then**
      6 : **Let** $d := d + 1$
      7 : **Let** $K_d := \psi \wedge p_i$
      8 : **Let** $\varphi_d := \mathbf{E}(\psi\,\mathcal{S}(\psi \wedge p_i))$
      9 : **Let** $F_d := (p_i, q_i)$
      10 : **Let** $\Delta_d := d$
      11 : **Let** $X := X \ \wedge\ \neg K_d$

In line 4 we compute in ψ the set of all X-states from which there exists no X-path leading to a q_i-state. In line 5 we check whether there exists a p_i-state belonging to ψ. If we find at least one such state then we have discovered a new assertion φ_d. In line 7 we place in K_d the set of states which form the kernel of the newly discovered assertion. In line 8 we define φ_d to consist of all the states reachable from a p_i-state by a ψ-path. In lines 9 and 10 we define F_d and Δ_d to be (p_i, q_i) and d, respectively. Finally, in line 11 we remove from X all states satisfying $\psi \wedge p_i = p_d \wedge \varphi_d = K_d$.

The following claim states the correctness of this extraction algorithm.

Claim 2. If the response property $p \implies \diamondsuit\, q$ is valid over CDS $\mathcal{D}$, then Algorithm Extract_constructs produces auxiliary constructs which satisfy the premises of rule RESPONSE.

Proof

Assume that the response property $p \implies \diamondsuit\, q$ is valid over CDS $\mathcal{D}$, and we successfully apply Algorithm *Extract_constructs*. Let us denote by $X_0 \supset X_1 \supset \cdots \supset X_{f-1} \supset X_f = 0$ the successive values assumed by variable X in the application of the algorithm. Note that we view the values of the variables appearing in the program as

assertions (represented as BDD's) as well as the sets of state defined by these assertions. For example, $X_f = 0$ means that this assertion is equivalent to *false* and also that it denotes the empty set.

Note that X_0 denotes the set of *pending states*. These are the states that can be reached from an accessible p-state by a q-free path. Observe that the successor of an X_0-state is either another X_0-state, or is a state satisfying q. Also note that, for each $r = 1, \ldots, f$, $X_r = X_{r-1} - K_r = X_{r-1} - (p_r \wedge \varphi_r)$. We will consider in turn each of the premises and show that it holds for the extracted constructs.

Premise R1 claims that every (accessible) p-state s satisfies q or $p_j \wedge \varphi_j$, for some $j \in [1..f]$. Indeed, if s does not satisfy q, it belongs to X_0, and since all X_0-states are eliminated by the algorithm, s must belong to some $K_j = (p_j \wedge \varphi_j)$.

Premise R2 requires that every successor of an (accessible) state s that satisfies $p_i \wedge \varphi_i$ must satisfy q or $p_j \wedge \varphi_j$ for some $j \in [1..f]$. Obviously state s belongs to X_0. Let s_b be any successor of s. As previously observed, s_b must satisfy q or belong to X_0. In the latter case, by an argument similar to the one presented for premise R1, s_b must belong to $K_j = (p_j \wedge \varphi_j)$, for some $j \in [1..f]$.

Premise R3 considers an (accessible) state s_a that satisfies φ_i and its successor s_b. It requires showing that s_b satisfies q, or satisfies φ_i and has the same rank as s_a, or satisfies $p_j \wedge \varphi_j$ for some j and has a rank lower than that of s_a. Since, in our construction, every state that satisfies φ_r has a rank $\Delta_r = r$, this is equivalent to requiring that s_b must satisfy q or φ_i or $p_j \wedge \varphi_j$ for some $j < i$. The fact that s belongs to φ_i implies that s can be reached from a p_i-state by a X_{i-1}-path π such that no state in π initiates an X_{i-1}-path leading to a q_i-state. Consider s_b the successor of s_a. If s_b belongs to X_{i-1}, then it satisfies φ_i. Otherwise, s_b may satisfy q which is also acceptable by R3. The remaining option is that s_b belongs to $X_0 - X_{i-1}$. In this case, s_b must have been removed from X in some earlier stage and, therefore must satisfy $p_j \wedge \varphi_j$ for some $j < i$.

Finally, we consider premise R4 which requires showing that every accessible φ_i-state s cannot satisfy q_i. By the definition of φ_i, s can be reached from a p_i-state by a X_{i-1}-path π such that no state in π initiates an X_{i-1}-path leading to a q_i-state. Since s itself is a member of π, it cannot satisfy q_i. ▟

5.2 The General Case

Next, we consider the general case of systems with potentially infinitely many states. Here, we can only claim relative completeness. Completeness is relative to assertional validity. That is, we prove that if the temporal conclusion is valid over a CDS, then there exist appropriate constructs expressible in an assertional language $\mathcal{L}$ such that the premises of the rule are assertionally valid ,i.e. state valid. Furthermore, as shown in [26], the language $\mathcal{L}$ should contain the expressive power of the μ-calculus over the underlying domain.

The approach we present here for establishing this relative completeness is based on a reduction of rule RESPONSE to the legacy rule F-WELL which, as shown in [15], is complete relative to assertional validity.

Claim 3. Rule RESPONSE is complete relative to assertional validity for proving response properties of arbitrary CDS's.

Proof Sketch
Assume that the response property $p \Longrightarrow \diamondsuit\, q$ is valid over the CDS $\mathcal{D}$. We prove that there exist constructs satisfying the premises of rule RESPONSE. The proof is by induction on the number of compassion requirements in CDS $\mathcal{D}$. Let $\mathcal{D}$ contain C compassion requirements. Assume by induction that the theorem is correct for all systems with a number of compassion requirements that is smaller than C. We invoke the completeness theorem for rule F-WELL which has been established in [15]. This yields constructs which satisfy the premises of rule F-WELL. In particular, for each $i \in [1..n]$, premise C4 guarantees the validity of the entailment $\mathcal{D}^i \models (\varphi_i \Longrightarrow \diamondsuit(p_i \vee \neg\varphi_i))$, where CDS $\mathcal{D}^i$ stands for $\mathcal{D}\backslash\{\langle p_i, q_i\rangle\}$, and is obtained by removing the compassion requirement $\langle p_i, q_i\rangle$ from $\mathcal{D}$.

Since each $\mathcal{D}^i$ has fewer compassion requirements than C, we apply the completeness claim of rule RESPONSE assumed by induction. This yields the identification of compassion requirements $F^i_1, \ldots, F^i_{n^i}$ and their associated helpful assertions $\varphi^i_1, \ldots, \varphi^i_{n^i}$ and ranking functions $\Delta^i_1, \ldots, \Delta^i_{n^i}$. We are now ready to identify the constructs necessary for the proof for CDS $\mathcal{D}$.

These consist of the compassion requirements $F_1, \ldots, F_n$ associated with the helpful assertions $\varphi_1, \ldots, \varphi_n$, and the ranking functions $(\Delta_1, 0), \ldots, (\Delta_n, 0)$. The ranking functions are obtained by lexicographic tuples formed by padding the rankings Δ_i on the right by zeros.

In addition, for each $i = 1, \ldots, n$, we add the compassion requirements $F^i_1, \ldots, F^i_{n^i}$ and their associated helpful assertions $\varphi_i \wedge \varphi^i_1, \ldots, \varphi_i \wedge \varphi^i_{n^i}$ and ranking functions $(\Delta_i, \Delta^i_1), \ldots, (\Delta_i, \Delta^i_{n^i})$. Note that the helpful assertions are obtained by conjuncting the original helpful assertions with φ_i, and the ranking functions are obtaining by a lexicographic tuple that prefixes the original ranking with Δ_i. ▟

The entire treatment in this paper focused on the special progress property of response. However, as shown in [11], once we know how to verify response properties, we can use the same methodology for proving an arbitrary LTL property. Let φ be an arbitrary LTL formula. In order to verify $\mathcal{D} \models \varphi$, it is sufficient to verify the response property $(\mathcal{D} \,|||\, T[\neg\varphi]) \models (\Theta \Longrightarrow \diamondsuit\, 0)$, where $T[\neg\varphi]$ is a *temporal tester* whose computations are all state sequences that satisfy $\neg\varphi$. This tester is composed in parallel with $\mathcal{D}$ to form a CDS whose computations are all computations of $\mathcal{D}$ which violate φ.

6 Related Work

There has been much work dealing with deductive verification of liveness properties under fairness. However, only a small fraction of this work considered compassion and suggested direct approaches for dealing with this special kind of fairness. In most cases, fairness has been reduced away, often by coupling it with the temporal property to be proven. This is the standard treatment of multiple justice requirements, and certainly that of compassion, in model checking, in particular, the automata theoretic approach in which the property is described by a non-deterministic Büchi automata. We refer the reader to [27] and [4] for a description of this reduction method.

The key reference upon which this work improves is [15] which formulated the first complete rule for liveness under compassion. Much work has been done on the deduc-

tive verification of liveness properties inspired by the STeP project [14]. Representative contributions of this work is presented in [25] and [3]. However, very little attention has been paid in this work to the special case of compassion, which has always been considered a specially difficult case.

Much work has also been invested in methods for automatically finding ranking functions for proving liveness properties and termination. Representative contributions are [8], [5], and [2]. Again, the context of this work when addressing systems with fairness has usually been that of justice.

Another approach to the verification of liveness properties can be based on abstraction. In this approach, we abstract a system into a finite-state system and verify the property on the abstract system. As pointed out in [10], the usual state abstraction if often inadequate in order to capture liveness properties. Therefore the paper introduces the notion of *ranking abstraction* which abstracts also changes in ranking. This concept has been further elaborated in [1]. Another solution to the problem has been proposed by Podelski and Rybalchenko who in [23] extend predicate abstraction ([24]) by employing predicates over program transitions, rather than states. In this way, the abstraction preserves the argument for proving termination (general liveness is handled by a reduction to fair termination). An elaboration of this work is [7], where Cook, Podelski, and Rybalchenko present a framework for verifying termination, which formalizes dual refinements – of transition predicate abstraction and of transition invariants [22]. The framework as presented in [7] lacks any notion of fairness. Therefore, [20, 6] extend it to allow for fairness requirements.

Acknowledgements We gratefully acknowledge the extensive and generous help of Tamarah Aarons, the designer of TLPVS who guided us very patiently through the growing pain of extending the system, and teaching PVS more temporal logic. We are very grateful to Lenore Zuck and Ittai Balaban for their continuous support of the development of the ideas that went into this paper. In particular, our joint work on extraction of ranking function has been very instrumental in the completeness proof of Section 5. In fact, the whole idea of the paper came out of a question asked by Dennis Dams at the defense of the thesis of Ittai Balaban. We thank Lenore, Dennis and Ittai for the insights contributed during extended discussions.

References

1. I. Balaban, A. Pnueli, and L. D. Zuck. Modular ranking abstraction. *Int. J. Found. Comput. Sci.*, 18(1):5–44, 2007.
2. A. R. Bradley, Z. Manna, and H. B. Sipma. Linear ranking with reachability. In *CAV'05*, volume 3576 of *LNCS*, pages 491–504, 2005.
3. I. Browne, Z. Manna, and H. Sipma. Generalized verification diagrams. In *FSTTCS'95*, volume 1026 of *LNCS*, pages 484–498, 1995.
4. E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 2000.
5. M. Colon and H. Sipma. Practical methods for proving program termination. In *CAV'02*, volume 2404 of *LNCS*, pages 442–454, 2002.
6. B. Cook, A. Gotsman, A. Podelski, A. Rybalchenko, and M. Y. Vardi. Proving that programs eventually do something good. In *Proc. 34th ACM Symp. Princ. of Prog. Lang.*, pages 265–276, 2007.

7. B. Cook, A. Podelski, and A. Rybalchenko. Abstraction refinement for termination. In *Static Analysis Symposium (SAS'05)*, volume 3672 of *LNCS*, pages 87–101, 2005.
8. D. Dams, R. Gerth, and O. Grumberg. A heuristic for the automatic generation of ranking functions. In G. Gopalakrishnan, editor, *Workshop on Advances in Verification*, pages 1–8, 2000.
9. Y. Kesten, N. Piterman, and A. Pnueli. Bridging the gap between fair simulation and trace inclusion. *Inf. and Comp.*, 200(1):35–61, 2005.
10. Y. Kesten and A. Pnueli. Verification by finitary abstraction. *Information and Computation, a special issue on Compositionality*, 163(1):203–243, 2000.
11. Y. Kesten and A. Pnueli. A Compositional Approach to CTL* Verification. *Theor. Comp. Sci.*, 331(2–3):397–428, 2005.
12. Y. Kesten, A. Pnueli, E. Shahar, and L. D. Zuck. Network invariants in action. In *CONCUR'02*, volume 2421 of *LNCS*, pages 101–115, 2002.
13. L. Lamport. Proving the correctness of multiprocess programs. *Trans. Soft. Eng.*, 3:125–143, 1977.
14. Z. Manna, A. Anuchitanukul, N. Bjørner, A. Browne, E. Chang, M. Colón, L. D. Alfaro, H. Devarajan, H. Sipma, and T. Uribe. STeP: The Stanford Temporal Prover. Technical Report STAN-CS-TR-94-1518, Dept. of Comp. Sci., Stanford University, Stanford, California, 1994.
15. Z. Manna and A. Pnueli. Completing the temporal picture. *Theor. Comp. Sci.*, 83(1):97–130, 1991.
16. Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems: Safety*. Springer-Verlag, New York, 1995.
17. Z. Manna and A. Pnueli. Temporal verification of reactive systems: Progress. Draft manuscript, available in http://theory.stanford.edu/ zm/tvors3.html, 1996.
18. S. Owre, N. Shankar, J. Rushby, and D. Stringer-Calvert. *PVS System Guide*. Menlo Park, CA, 2001.
19. A. Pnueli and T. Arons. TLPVS: A PVS-Based LTL Verification System. In *Verification: Theory and Practice*, volume 2772 of *LNCS*, pages 598–625, 2004.
20. A. Pnueli, A. Podelski, and A. Rybalchenko. Separating fairness and well-foundedness for the analysis of fair discrete systems. In *TACAS'05*, volume 3440 of *LNCS*, pages 124–139, 2005.
21. A. Pnueli and Y. Sa'ar. All you need is compassion. Research Report, Dept. of Computer Science, New York University Technical Report, Oct. 2007. http://www.cs.nyu.edu/acsys/pubs/permanent/all-you-need-is-compassion.pdf.
22. A. Podelski and A. Rybalchenko. Transition invariants. In *LICS'04*, pages 32–41, 2004.
23. A. Podelski and A. Rybalchenko. Transition predicate abstraction and fair termination. In *POPL'05*, pages 132–144, 2005.
24. S. Graf and H. Saidi. Construction of abstract state graphs with PVS. In *CAV'97*, volume 1254 of *LNCS*, pages 72–83, 1997.
25. H. Sipma, T. Uribe, and Z. Manna. Deductive model checking. *Formal Methods in System Design*, 15(1):49–74, 1999.
26. F. Stomp, W.-P. de Roever, and R. Gerth. The μ-calculus as an assertion language for fairness arguments. *Inf. and Comp.*, 82:278–322, 1989.
27. M. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *LICS'86*, pages 332–344, 1986.
28. P. Wolper and V. Lovinfosse. Verifying properties of large sets of processes with network invariants. In J. Sifakis, editor, *Automatic Verification Methods for Finite State Systems*, volume 407 of *LNCS*, pages 68–80, 1989.