Separating Fairness and Well-Foundedness for the Analysis of Fair Discrete Systems*

Amir Pnueli[1], Andreas Podelski[2], and Andrey Rybalchenko[2]

[1] New York University, New York
[2] Max-Planck-Institut für Informatik, Saarbrücken

Abstract. Fair discrete systems (FDSs) are a computational model of concurrent programs where fairness assumptions are specified in terms of sets of states. The analysis of fair discrete systems involves a non-trivial interplay between fairness and well-foundedness (ranking functions). This interplay has been an obstacle for automation. The contribution of this paper is a new analysis of temporal properties of FDSs. The analysis uses a domain of binary relations over states labeled by sets of indices of fairness requirements. The use of labeled relations separates the reasoning on well-foundedness and fairness.

1 Introduction

Fair discrete systems provide a computational model of concurrent programs where fairness assumptions are specified in terms of sets of states [8]. The analysis of fair discrete systems involves a non-trivial interplay between fairness and well-foundedness (ranking functions). Its automation is a difficult task. One particular difficulty is the design of an abstract domain that accounts for well-foundedness and fairness.

We propose an analysis that avoids such an interplay by separating the reasoning on well-foundedness and fairness. The analysis is based on binary relations over states that are labeled by sets of indices of fairness requirements. We design an operator F_{FDS} on a (concrete) domain D_{FDS} of such labeled relations. We use least fixed points of F_{FDS} to establish the validity of temporal FDS properties. Furthermore, we design an abstract domain $D^{\#}_{\mathsf{FDS}}$ on which approximations of least fixed points of F_{FDS} are effectively computable. The formalization of our analysis follows the framework of abstract interpretation [3].

The starting point for the design of our analysis is a domain D that consists of binary relations over states, together with an operator F that composes relations with the transition relation of the system. This domain allows us to reason about well-foundedness [18], but it does not account for justice and compassion requirements of FDSs. We extend the domain D to account for fairness by labeling its elements with sets of indices of fairness requirements. We extend the composition operator F by taking the labeling into account, and obtain the operator F_{FDS}, whose least fixed points allow

* The second and third author were supported in part by the German Research Foundation (DFG) as a part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems" (SFB/TR 14 AVACS), by the German Federal Ministry of Education and Research (BMBF) in the framework of the Verisoft project under grant 01 IS C38.

us to reason about well-foundedness and fairness. Given a set of labeled relations L that constitute the least fixed point of F_{FDS}, we account for well-foundedness by considering the relations that appear in the elements of L. We reason about fairness by considering the sets of labels.

We provide an abstract domain $D^{\#}_{\mathsf{FDS}}$ on which approximations of least fixed points of F_{FDS} can be computed. We abstract the part of labeled relations that may cause the iterative fixed point computation to diverge. This means that the abstract domain $D^{\#}_{\mathsf{FDS}}$ consists of abstractions of binary relations over states labeled by sets of indices of fairness requirements. We assume that the correspondence between the domain of relations and their abstractions is given by a Galois connection, which is left as a parameter of our analysis.

Our analysis accounts for general temporal properties by applying the automata-theoretic framework for the verification of concurrent programs [23]. We encode the temporal property into a specification automaton. We translate the acceptance condition of the product of the automata-theoretic construction into additional fairness requirements, which we handle in the same way as the fairness requirements of the FDS. Then, we apply our analysis on the product FDS.

For proving the soundness and partial completeness [2] of our analysis we develop a corresponding proof rule whose auxiliary assertions are labeled relations.

We have implemented the analysis in a prototype tool, and applied it on interesting examples of concurrent programs. We proved an eventual reachability property for a concurrent program that evolves the inter-process communication via an asynchronous, lossy, and corrupting channel. The property relies on the eventual reliability of the channel, which we model by a compassion requirement. We also considered the mutual exclusion protocols BAKERY and TICKET. For each protocol, we proved the non-starvation property, *i.e.* the accessibility of the critical section, for the first process. Justice requirements are needed to deal with the process idling in all examples.

Our main contribution is the analysis of temporal properties of fair discrete systems, and the proof of its soundness and partial completeness. The analysis is based on separating the reasoning on well-foundedness and fairness, which facilitates its automation. We achieve the separation by building the analysis on the domain of binary relations labeled by sets of indices of fairness requirements.

Related Work The framework of abstract interpretation provides a basis for the systematic design of a program analysis [3]. It is difficult to integrate fairness into the definition of abstract domains.

There exist verification methods for finite-state systems with state-based fairness requirements that account for justice and compassion on the algorithmic level, *e.g.* [8, 13]. Experimental evaluation has confirmed the advantage of the direct treatment of fairness (as opposed to the automata-theoretic translation into a Büchi acceptance condition).

For dealing with infinite-state systems, there exist proof rules for the verification of termination [12] and general temporal properties [14] under justice and compassion requirements that account for the fairness requirements without applying the automata-theoretic encoding. The proof rules rely on well-founded orderings, which must be supplied by the user. Justice requirements are handled directly by the proof rules; ver-

ification under compassion requirements is done by recursive application of the proof rule to a transformed program. Our proof rule treats justice and compassion in a uniform way, without program transformation.

The uniform liveness-verification of parameterized FDSs [5, 6] requires construction of auxiliary assertions that account for well-foundedness and fairness. The construction of such assertions can be effectively automated by applying "instantiate-project-and-generalize" heuristic, which allows for the treatment of several classes of parameterized communication protocols. Our approach relies on least fixed point computations, where heuristics can be applied to find abstractions.

Transition invariants provide a basis for reasoning about well-foundedness [18]. They can account for the fairness given by a Büchi accepting condition. Labeled relations extend transition invariant to account for justice and compassion requirements imposed on sets of states, *i.e.*, for the generalized Büchi and Streett acceptance conditions.

Abstract-transition programs, introduced in [19], provide a basis for an automated method for the verification of programs with the *transition*-based fairness requirements. Its accounting for well-foundedness is similar to the one via labeled relations, whereas the treatment of fairness is based on graphs underlying abstract-transition programs.

The automata-theoretic framework of [23] is the basis of our analysis for the verification of general temporal properties. For infinite-state concurrent programs, the Büchi and the Streett acceptance conditions are translated to the Wolper (*i.e.* all states are accepting) acceptance condition. Thus, a proof of fair termination is reduced to a proof of termination of a program obtained from the original one by a transformation that encodes the fairness requirements into the state space. This approach is converse to ours.

The stack assertions based method of [9] for proving fair termination accounts for justice and compassion requirements directly. The method requires identification of tuples of well-founded mappings (stacks assertions), one element for each fairness requirement, which must by supplied by the user. The method keeps track of fairness through the tuple structure. No automation is described.

2 Preliminaries

Fair Discrete Systems Following [8], a fair discrete system (FDS) $\mathcal{S} = \langle \Sigma, \Theta, \mathcal{T}, \mathcal{J}, \mathcal{C} \rangle$ consists of:

- Σ: a set of *states*,
- Θ: a set of *initial* states such that $\Theta \subseteq \Sigma$,
- $\mathcal{T}$: a finite set of *transitions* such that each transition $\tau \in \mathcal{T}$ is associated with a *transition relation* $\rho_\tau \subseteq \Sigma \times \Sigma$,
- $\mathcal{J} = \{J_1, \ldots, J_k\}$: a set of *justice* requirements, such that $J_i \subseteq \Sigma$ for each $i \in \{1, \ldots, k\}$,
- $\mathcal{C} = \{\langle p_1, q_1 \rangle, \ldots, \langle p_m, q_m \rangle\}$: a set of *compassion* requirements, such that $p_i, q_i \subseteq \Sigma$ for each $i \in \{1, \ldots, m\}$.

A *computation* σ is a maximal sequence of states $s_1, s_2, \ldots$ such that s_1 is an *initial* state, *i.e.* $s_1 \in \Theta$, and for each $i \geq 1$ there exists a transition $\tau \in \mathcal{T}$ such that s_i goes to s_{i+1} under ρ_τ, *i.e.* $(s_i, s_{i+1}) \in \rho_\tau$. A finite segment $s_i, s_{i+1}, \ldots, s_j$ of a computation where $i < j$ is called a *computation segment*. The set Acc of *accessible states* consists of all states that appear in computations of $\mathcal{S}$.

A computation $\sigma = s_1, s_2, \ldots$ satisfies the set of justice requirements $\mathcal{J}$ when for each $J \in \mathcal{J}$ there exist infinitely many positions i in σ such that $s_i \in J$. The computation σ satisfies the set of compassion requirements $\mathcal{C}$ when for each $\langle p, q \rangle \in \mathcal{C}$ either σ contains only finitely many positions i such that $s_i \in p$, or σ contains infinitely many positions j such that $s_j \in q$.

We observe that justice requirements can be translated into compassion requirements as follows. For every justice requirement J we extend the set of compassion requirements by the pair $\langle \Sigma, J \rangle$. We assume that all justice requirements are translated into the compassion requirements, and that the set of compassion requirements $\mathcal{C}$ contains the translated justice requirements. A specialization of the analysis presented in this paper for an explicit treatment of justice requirements is straightforward.

Automata-Theoretic Approach to Temporal Verification Given a FDS $\mathcal{S}$, we verify a temporal property Ψ under the compassion requirements $\mathcal{C}$ by applying the automata-theoretic framework [23]. We assume that the property is given by a (possibly infinite-state) specification automaton $\mathcal{A}_\Psi$ that accepts exactly the infinite sequences of states that violate the property Ψ. We do not encode the compassion requirements into the automaton.

Let $\mathcal{A}_\Psi$ be a Büchi automaton with the set of states Q and the acceptance condition $F \subseteq Q$. Let the FDS $\mathcal{S} ||| \mathcal{A}_\Psi$ be the product of the synchronous parallel composition of $\mathcal{S}$ and $\mathcal{A}_\Psi$.

Remark 1. The FDS $\mathcal{S}$ with the compassion requirements $\mathcal{C}$ satisfies the property Ψ given by the Büchi automaton $\mathcal{A}_\Psi$ if and only if the FDS $\mathcal{S} ||| \mathcal{A}_\Psi$ terminates under the compassion requirements $\mathcal{C}_{|||}$ shown below.

$$\mathcal{C}_{|||} \;=\; \{\langle p \times Q, q \times Q \rangle \mid \langle p, q \rangle \in \mathcal{C}\} \cup \{\langle \Sigma \times Q, \Sigma \times F \rangle\}$$

Domain of Transition Invariants Following [18], we define a domain $D = 2^{\Sigma \times \Sigma}$ of binary relations over states ordered by the subset inclusion ordering $\subseteq$. On this domain, we define an operator $F_\tau : D \to D$, where $\tau \in \mathcal{T}$ and the symbol $\circ$ denotes the relational composition (*i.e.* $R_1 \circ R_2 = \{(s, s'') \mid (s, s') \in R_1 \text{ and } (s', s'') \in R_2\}$):

$$F_\tau(T) \;=\; T \circ \rho_\tau \; .$$

We will use the domain D and the operator F_τ as a starting point for the development our analysis.

3 Analysis

We fix a FDS $\mathcal{S}$ with the set of compassion requirements $\mathcal{C}$. We define an analysis that allows one to prove that $\mathcal{S}$ terminates under $\mathcal{C}$. We apply the Galois connection

approach for abstract interpretation [3] as a basis for our analysis. We assume an abstract domain $D^{\#}$, partially ordered by a relation $\sqsubseteq$, that contains abstractions of elements (binary relations) from the domain D. Let a Galois connection (α, γ) formalize the correspondence between the domains D and $D^{\#}$, formally:

$$\forall T \in D\ \forall T^{\#} \in D^{\#}.\ \alpha(T) \sqsubseteq T^{\#} \Leftrightarrow T \subseteq \gamma(T^{\#})\ .$$

Let $|\mathcal{C}|$ be the set of the indices of all compassion requirements:

$$|\mathcal{C}| \ =\ \{1, \ldots, m\}\ .$$

We obtain a domain D_{FDS} that accounts for compassion requirements by an extension of D with sets of compassion requirements:

$$D_{\mathsf{FDS}} \ =\ D \times 2^{|\mathcal{C}|} \times 2^{|\mathcal{C}|}\ .$$

We define an ordering $\subseteq_{\mathsf{FDS}}$ on elements (T_1, P_1, Q_1) and (T_2, P_2, Q_2) of D_{FDS}:

$$(T_1, P_1, Q_1) \subseteq_{\mathsf{FDS}} (T_2, P_2, Q_2) \ =\ T_1 \subseteq T_2 \text{ and } P_1 \subseteq P_2 \text{ and } Q_1 \subseteq Q_2\ .$$

We define the following auxiliary functions that map sets of states into sets of indices of compassion requirements. For a set of states $S \subseteq \Sigma$ we have

$$\mathsf{None}(S) = \{j \in |\mathcal{C}| \mid S \cap p_j = \emptyset\}\ , \qquad \mathsf{Some}(S) = \{j \in |\mathcal{C}| \mid S \cap q_j \neq \emptyset\}\ .$$

We extend the functions None and Some to binary relations. Given a relation $T \subseteq \Sigma \times \Sigma$, we have

$$\mathsf{None}(T) = \bigcup_{(s_1,s_2)\in T} \mathsf{None}(\{s_1, s_2\})\ , \qquad \mathsf{Some}(T) = \bigcup_{(s_1,s_2)\in T} \mathsf{Some}(\{s_1, s_2\})\ .$$

We define an operator $F_{\mathsf{FDS},\tau} : D_{\mathsf{FDS}} \to D_{\mathsf{FDS}}$, which is an extension of the operator F_τ that accounts for compassion requirements, as follows:

$$F_{\mathsf{FDS},\tau}(T, P, Q) \ =\ (F_\tau(T), P \cap \mathsf{None}(F_\tau(T)), Q \cup \mathsf{Some}(F_\tau(T)))\ .$$

Theorem 1. *The operator $F_{\mathsf{FDS},\tau}$ is monotonic. Formally,*

$$(T_1, P_1, Q_1) \subseteq_{\mathsf{FDS}} (T_2, P_2, Q_2) \implies F_{\mathsf{FDS},\tau}(T_1, P_1, Q_1) \subseteq_{\mathsf{FDS}} F_{\mathsf{FDS},\tau}(T_2, P_2, Q_2)\ .$$

Proof. Let (T_1, P_1, Q_1) and (T_2, P_2, Q_2) be two elements of D_{FDS} such that $(T_1, P_1, Q_1) \subseteq_{\mathsf{FDS}} (T_2, P_2, Q_2)$. Since $T_1 \subseteq T_2$ we have

$$\bigcup_{(s,s')\in T_1 \circ \rho_\tau} \mathsf{None}(\{s, s'\}) \subseteq \bigcup_{(s,s')\in T_2 \circ \rho_\tau} \mathsf{None}(\{s, s'\})\ ,$$

i.e., we have $\mathsf{None}(F_\tau(T_1)) \subseteq \mathsf{None}(F_\tau(T_2))$. Analogously, we have $\mathsf{Some}(F_\tau(T_1)) \subseteq \mathsf{Some}(F_\tau(T_2))$. We conclude $F_{\mathsf{FDS},\tau}(T_1, P_1, Q_1) \subseteq_{\mathsf{FDS}} F_{\mathsf{FDS},\tau}(T_2, P_2, Q_2)$. □

We define an abstract counterpart $D^{\#}_{\mathsf{FDS}}$ of the domain D_{FDS} as follows:

$$D^{\#}_{\mathsf{FDS}} \;=\; D^{\#} \times 2^{|\mathcal{C}|} \times 2^{|\mathcal{C}|} \,.$$

We define an ordering $\subseteq^{\#}_{\mathsf{FDS}}$ on elements $(T^{\#}_1, P_1, Q_1)$ and $(T^{\#}_2, P_2, Q_2)$ of $D^{\#}_{\mathsf{FDS}}$:

$$(T^{\#}_1, P_1, Q_1) \subseteq^{\#}_{\mathsf{FDS}} (T^{\#}_2, P_2, Q_2) \;=\; T^{\#}_1 \sqsubseteq T^{\#}_2 \text{ and } P_1 \subseteq P_2 \text{ and } Q_1 \subseteq Q_2 \,.$$

Note that we only abstract a component of D_{FDS}-elements that may potentially allow for infinite, strictly increasing chains $(T_1, P_1, Q_1) \subset_{\mathsf{FDS}} (T_2, P_2, Q_2) \subset_{\mathsf{FDS}} \ldots$. We define a pair of functions $(\alpha_{\mathsf{FDS}}, \gamma_{\mathsf{FDS}})$ that connect the domains D_{FDS} and $D^{\#}_{\mathsf{FDS}}$:

$$\alpha_{\mathsf{FDS}}(T, P, Q) = (\alpha(T), P, Q) \,, \qquad \gamma_{\mathsf{FDS}}(T^{\#}, P, Q) = (\gamma(T^{\#}), P, Q) \,.$$

Lemma 1. *The pair of functions $(\alpha_{\mathsf{FDS}}, \gamma_{\mathsf{FDS}})$ is a Galois connection between D_{FDS} and $D^{\#}_{\mathsf{FDS}}$.*

Proof. From the monotonicity of γ and α follows that α_{FDS} and γ_{FDS} are monotonic. We carry out the following transformations:

$$\begin{aligned} \alpha_{\mathsf{FDS}}(\gamma_{\mathsf{FDS}}(T^{\#}, P, Q)) &= \alpha_{\mathsf{FDS}}(\gamma(T^{\#}), P, Q) \\ &= (\alpha(\gamma(T^{\#})), P, Q) \,. \end{aligned}$$

Since (α, γ) is a Galois connection, by Theorem 5.3.0.4 in [4], we have that $\alpha(\gamma(T^{\#})) \subseteq T^{\#}$ and hence $\alpha_{\mathsf{FDS}}(\gamma_{\mathsf{FDS}}(T^{\#}, P, Q)) \subseteq_{\mathsf{FDS}} (T^{\#}, P, Q)$. Similarly, we obtain $(T, P, Q) \subseteq_{\mathsf{FDS}} \gamma_{\mathsf{FDS}}(\alpha_{\mathsf{FDS}}(T, P, Q))$. By Theorem 5.3.0.4 in [4], we conclude that $(\alpha_{\mathsf{FDS}}, \gamma_{\mathsf{FDS}})$ is a Galois connection. □

The abstract operator $F^{\#}_{\mathsf{FDS},\tau} : D^{\#}_{\mathsf{FDS}} \to D^{\#}_{\mathsf{FDS}}$ is defined below:

$$F^{\#}_{\mathsf{FDS},\tau}(T^{\#}, P, Q) \;=\; \alpha_{\mathsf{FDS}}(F_{\mathsf{FDS},\tau}(\gamma_{\mathsf{FDS}}(T^{\#}, P, Q))) \,.$$

We extend $F^{\#}_{\mathsf{FDS},\tau}$ to the full set of transitions $\mathcal{T}$:

$$F^{\#}_{\mathsf{FDS}}(T^{\#}, P, Q) \;=\; \{F^{\#}_{\mathsf{FDS},\tau}(T^{\#}, P, Q) \mid \tau \in \mathcal{T}\} \,.$$

The monotonicity of the fixed point operator $F^{\#}_{\mathsf{FDS}}$ is a direct consequence of Theorem 1 and the monotonicity of the abstraction/concretization functions. By Tarski's fixed point theorem, the least fixed point of $F^{\#}_{\mathsf{FDS}}$ exists. We denote the least fixed point of $F^{\#}_{\mathsf{FDS}}$ above the basis $\{(\alpha(\rho_\tau), \mathsf{None}(\rho_\tau), \mathsf{Some}(\rho_\tau)) \mid \tau \in \mathcal{T}\}$ by $\mathsf{lfp}(F^{\#}_{\mathsf{FDS}}, \mathcal{T})$. We compute $\mathsf{lfp}(F^{\#}_{\mathsf{FDS}}, \mathcal{T})$ in the usual fashion. If the range of the abstraction function α does not allow infinite, strictly increasing chains then the fixed point computation always terminates after finitely many iterations.

We show our analysis for termination of the FDS $\mathcal{S}$ under the compassion requirements $\mathcal{C}$ on Figure 1. For proving the soundness and partial completeness of the analysis we will develop a corresponding proof rule, whose auxiliary assertions denote elements of the domain D_{FDS}. The partial completeness property, following [2], requires that the analysis gives a positive answer for a FDS that terminates under compassion requirements $\mathcal{C}$ in case the abstract domain $D^{\#}_{\mathsf{FDS}}$ satisfies the following property. The domain $D^{\#}_{\mathsf{FDS}}$ contains an abstract value $L^{\#}$ that satisfies the condition imposed by the analysis.

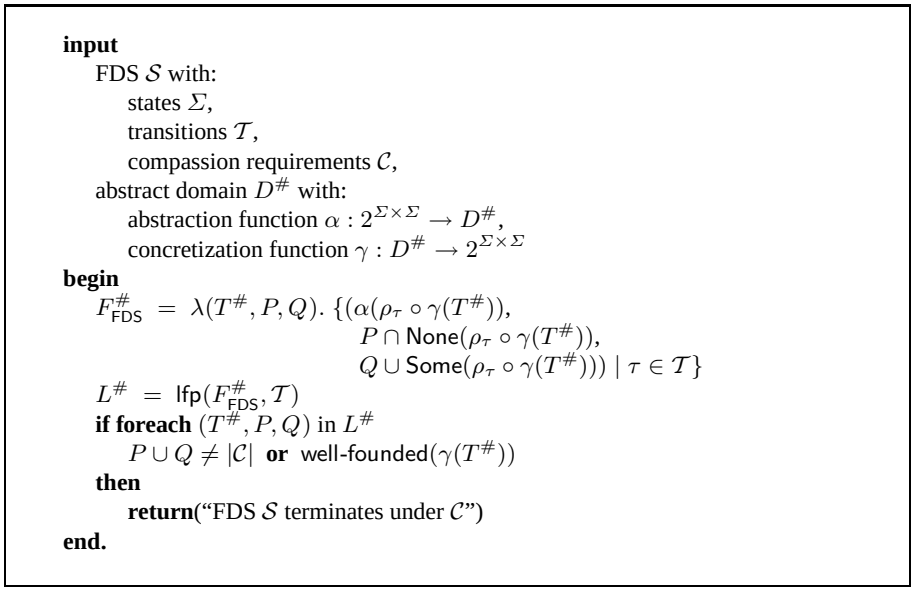

Fig. 1. Analysis of termination for a fair discrete system $\mathcal{S}$ under compassion requirements $\mathcal{C}$.

Theorem 2. *The analysis shown on Figure 1 is sound and partially complete.*

Proof. See Section 5. □

We apply our analysis on general temporal properties of fair discrete systems as follows. Let Ψ be a temporal property given by a Büchi automaton $\mathcal{A}_\Psi$. Note that we do not encode the fairness requirements into $\mathcal{A}_\Psi$. Following Remark 1, we construct a FDS $\mathcal{S}|||\mathcal{A}_\Psi$ together with the set of compassion requirements $\mathcal{C}_{|||}$. For proving that the FDS $\mathcal{S}$ satisfies the property Ψ under the compassion requirements $\mathcal{C}$ we apply our analysis on $\mathcal{S}|||\mathcal{A}_\Psi$ (with compassion requirements $\mathcal{C}_{|||}$).

We account for temporal properties given by generalized Büchi and Streett automaton in a straightforward way. For this purpose, we use a direct translation of the generalized Büchi and Streett acceptance conditions into compassion requirements, following the lines of the translation shown in Remark 1.

4 Proof Rule

In this section, we show a proof rule for the verification of fair discrete systems. The auxiliary assertions of the proof rule, called *labeled relations*, denote elements of the domain D_{FDS} used by our analysis. The correspondence between the proof rule and the analysis will allow us to prove Theorem 2, which states the analysis' correctness.

Informally, a labeled relation is a triple (T, P, Q) consisting of a binary relation T over states together with two sets of compassion requirements P and Q. Labeled

relations capture sets of computations segments. A computation segment $s_1, \ldots, s_n$ is captured by (T, P, Q) if the pair (s_1, s_n) is an element of T, and the infinite sequence $(s_1, \ldots, s_n)^\omega$, *i.e.* the infinite concatenation of the segment with itself, satisfies only those compassion requirements whose indices are in the set $P \cup Q$. We give a formal definition of labeled relations below.

Definition 1 (Labeled Relation). *A* labeled relation (T, P, Q) *consists of a binary relation* $T \subseteq \Sigma \times \Sigma$ *and two sets of indices (*labels*)* $P, Q \subseteq |\mathcal{C}|$*. The labeled relation* (T, P, Q) captures *a computation segment* $s_1, \ldots, s_n$ *if the following conditions hold:*

$$(s_1, s_n) \in T\ , \qquad \mathsf{None}(\{s_1, \ldots, s_n\}) \subseteq P\ , \qquad \mathsf{Some}(\{s_1, \ldots, s_n\}) \subseteq Q\ .$$

We write $\mathsf{seg}(T, P, Q)$ *for the set of all computation segments that are captured by the labeled relation* (T, P, Q)*.*

The following theorem allows us to separate the reasoning about well-foundedness and fairness.

Theorem 3. *The FDS* $\mathcal{S}$ *terminates under the set of compassion requirements* $\mathcal{C}$ *if and only if there exist labeled relations* $(T_1, P_1, Q_1), \ldots, (T_n, P_n, Q_n)$ *such that 1) every computation segment of* $\mathcal{S}$ *is captured by some labeled relation from the set, and 2) for every labeled relation* (T, P, Q) *in the set either* $|\mathcal{C}| \neq P \cup Q$ *or the relation* T *is well-founded.*

Proof. (**if**-direction) For a proof by contraposition, assume that 1) a finite set L of labeled relations captures every computation segment, 2) for each $(T, P, Q) \in L$ holds that either $|\mathcal{C}| \neq P \cup Q$ or the relation T is well-founded, and 3) $\mathcal{S}$ does not terminate under the compassion requirements $\mathcal{C}$. We will show that there exists a labeled relation (T, P, Q) in L such that the relation T is not well-founded and $|\mathcal{C}| = P \cup Q$.

By the assumption that $\mathcal{S}$ does not terminate under $\mathcal{C}$, there exists an infinite computation $\sigma = s_1, s_2, \ldots$ that satisfies all compassion requirements.

We partition the set $|\mathcal{C}|$ of indices of compassion requirements into two subsets $|\mathcal{C}|^p$ and $|\mathcal{C}|^q$ as follows. An index j (of the compassion requirement $\langle p_j, q_j \rangle$) is an element of the subset $|\mathcal{C}|^p$ if there exist only finitely many positions i in σ such that $s_i \in p_j$; otherwise, j is an element of the subset $|\mathcal{C}|^q$. There exists a position r such that for each $i \geq r$ and for each $j \in |\mathcal{C}|^p$ we have $s_i \notin p_j$.

Let $H = h_1, h_2, \ldots$ be an infinite ordered set of positions in σ such that $h_1 = r$ and for each $i \geq 1$ and for each $j \in |\mathcal{C}|^q$ there exist a position h between the positions h_i and h_{i+1} with $s_h \in q_j$. Since σ satisfies all compassion requirements such a set H exists.

For the fixed σ and the fixed H, we choose a function f that maps an ordered pair (k, l), where $k < l$, of indices in H to one of the labeled relations in L as follows:

$$f(k, l) \ = \ (T, P, Q) \in L \quad \text{ such that } (s_k, \ldots, s_l) \in \mathsf{seg}(T, P, Q)\ .$$

Such a function f exists since every computation segment is captured by some labeled relation in L. The function f induces an equivalence relation $\sim$ on ordered pairs of elements from H:

$$(k, l) \sim (k', l') \ = \ f(k, l) = f(k', l')\ .$$

The equivalence relation $\sim$ has finite index since the range of f is finite.

By Ramsey's theorem [20], there exists an infinite ordered set of positions $K = k_1, k_2, \ldots$, where $k_i \in H$ for all $i \geq 1$, with the following property. All pairs of elements in K belong to the same equivalence class, say $[(m,n)]_\sim$ with $m, n \in K$. That is, for all $k, l \in K$ such that $k < l$ we have $(k,l) \sim (m,n)$. We fix m and n. Let (T_{mn}, P_{mn}, Q_{mn}) denote the labeled relation $f(m,n)$.

Since for all $i \geq 1$ we have $(k_i, k_{i+1}) \sim (m,n)$, the function f maps the pair (k_i, k_{i+1}) to (T_{mn}, P_{mn}, Q_{mn}) for all $i \geq 1$. Hence, the infinite sequence $s_{k_1}, s_{k_2}, \ldots$ is induced by the relation T_{mn}, *i.e.,* for all $i \geq 1$ we have $(s_{k_i}, s_{k_{i+1}}) \in T_{mn}$. We conclude that the relation T_{mn} is not well-founded.

By the choice of H the following claims hold. For every $i \geq k_1$ and for every $j \in |\mathcal{C}|^p$ the state s_i is not an element of p_j. For every $i \geq 1$ and for every $j \in |\mathcal{C}|^q$ there exists a position k between the positions k_i and k_{i+1} such that $s_k \in q_j$. Hence, for every $i \geq 1$ the infinite sequence $(s_{k_i}, \ldots, s_{k_{i+1}})^\omega$ satisfies all compassion requirements. We conclude $|\mathcal{C}| = P_{mn} \cup Q_{mn}$.

(**only if**-direction) is shown after the proof of Theorem 4 in this section. □

We formalize the correspondence between labeled relations and the ingredients of our analysis by the lemmas below. The ordering $\subseteq_{\mathsf{FDS}}$ approximates the subset inclusion ordering between the sets of computation segments captured by labeled relations, as shown in Lemma 2.

Lemma 2. *The relation $\subseteq_{\mathsf{FDS}}$ is an approximation of the entailment relation between the sets of computation segments that are captured by two labeled relations. Formally,*

$$(T_1, P_1, Q_1) \subseteq_{\mathsf{FDS}} (T_2, P_2, Q_2) \implies \mathsf{seg}(T_1, P_1, Q_1) \subseteq \mathsf{seg}(T_2, P_2, Q_2)\ .$$

Proof. Let the computation segment $s_1, \ldots, s_n$ be captured by the labeled relation (T_1, P_1, Q_1). From $(T_1, P_1, Q_1) \subseteq_{\mathsf{FDS}} (T_2, P_2, Q_2)$ and the definition of labeled relations, we directly obtain $(s_1, \ldots, s_n) \in \mathsf{seg}(T_2, P_2, Q_2)$. □

The operator $F_{\mathsf{FDS},\tau}$ is 'compatible' with the composition of computation segments, as formalized in Lemma 3.

Lemma 3. *Every extension of a computation segment that is captured by a labeled relation (T, P, Q) by a segment consisting of a pair of states in a transition relation ρ_τ is captured by the application of the operator $F_{\mathsf{FDS},\tau}$ on (T, P, Q). Formally,*

$$(s_1, \ldots, s_n) \in \mathsf{seg}(T, P, Q) \text{ and } (s_n, s_{n+1}) \in \rho_\tau \implies (s_1, \ldots, s_n, s_{n+1}) \in \mathsf{seg}(F_{\mathsf{FDS},\tau}(T, P, Q))\ .$$

Proof. Let $s_1, \ldots, s_n$ be a computation segment that is captured by the labeled relation (T, P, Q), and let (s_n, s_{n+1}) be an element of the transition relation ρ_τ. By the definition of labeled relations, for the set of indices of compassion requirements $P_n = \mathsf{None}(\{s_1, \ldots, s_n\})$ we have $P_n \subseteq P$. Furthermore, for the set of indices $P_{n+1} = \mathsf{None}(\{s_1, \ldots, s_n, s_{n+1}\})$ holds $P_{n+1} \subseteq \mathsf{None}(\{s_1, s_{n+1}\}) \subseteq \mathsf{None}(T \circ \rho_\tau)$

and $P_{n+1} \subseteq P_n$. Hence, we have $P_{n+1} \subseteq P$ and $P_{n+1} \subseteq \mathsf{None}(T \circ \rho_\tau)$. We conclude $P_{n+1} \subseteq P \cap \mathsf{None}(F_\tau(T))$.

Analogously, we have $\mathsf{Some}(\{s_1, \ldots, s_n\}) \subseteq Q$, and, hence, for the set of indices $Q_{n+1} = \mathsf{Some}(\{s_1, \ldots, s_n, s_{n+1}\})$ holds $Q_{n+1} \subseteq Q \cup \mathsf{Some}(F_\tau(T))$.

The pair of states (s_1, s_{n+1}) is an element of the relational composition $T \circ \rho_\tau$, since (s_1, s_n) is an element of the relation T. We conclude that $s_1, \ldots, s_n, s_{n+1}$ is captured by $F_{\mathsf{FDS},\tau}(T, P, Q)$. □

We canonically extend the ordering $\subseteq_{\mathsf{FDS}}$ to sets of labeled relations:

$$L \subseteq_{\mathsf{FDS}} M = \forall (T_1, P_1, Q_1) \in L\, \exists (T_2, P_2, Q_2) \in M.\, (T_1, P_1, Q_1) \subseteq_{\mathsf{FDS}} (T_2, P_2, Q_2)\ .$$

We canonically extend the operator $F_{\mathsf{FDS},\tau}$ to sets of labeled transitions:

$$F_{\mathsf{FDS},\tau}(L) \;=\; \{F_{\mathsf{FDS},\tau}(T, P, Q) \mid (T, P, Q) \in L\}\ .$$

We show a proof rule COMP-TERM for verifying the termination of fair discrete systems under compassion requirements on Figure 2. By applying Theorem 3, we reduce this termination proof to the problem of identifying of a set of labeled relation that captures every computation segment of $\mathcal{S}$. The premises P1 and P2 identify such sets of labeled relations, which is justified by Lemma 4. The premise P3 accounts for well-foundedness and fairness.

Lemma 4. *A set of labeled relations L for the FDS $\mathcal{S}$ that satisfies the premises* P1 *and* P2 *of the proof rule* COMP-TERM *captures every computation segment.*

Proof. Given a set of labeled relations L that satisfies the premises P1 and P2 of the proof rule COMP-TERM, we show that every computation segment is captured by some labeled relation in L by the induction over the segment length.

Let s_1, s_2 such that $(s_1, s_2) \in \rho_\tau$, where τ is a transition, be a computation segment. From $\mathsf{None}(\{s_1, s_2\}) \subseteq \mathsf{None}(\rho_\tau)$ and $\mathsf{Some}(\{s_1, s_2\}) \subseteq \mathsf{Some}(\rho_\tau)$ follows directly that the segment s_1, s_2 is captured by the labeled relation $(\rho_\tau, \mathsf{None}(\rho_\tau), \mathsf{Some}(\rho_\tau))$. By Lemma 2 and the premise P1, the segment s_1, s_2 is captured by some labeled relation in L, which is $\subseteq_{\mathsf{FDS}}$-greater than $(\rho_\tau, \mathsf{None}(\rho_\tau), \mathsf{Some}(\rho_\tau))$.

The induction assumption is that the computation segment $s_1, \ldots, s_n$ is captured by a labeled relation (T, P, Q) from L. Let (s_n, s_{n+1}) be an element of ρ_τ. By Lemma 3, we have $(s_1, \ldots, s_n, s_{n+1}) \in \mathsf{seg}(F_{\mathsf{FDS},\tau}(T, P, Q))$. Analogously to the base case, the segment $s_1, \ldots, s_n, s_{n+1}$ is captured by some labeled relation in L, which is $\subseteq_{\mathsf{FDS}}$-greater than $F_{\mathsf{FDS},\tau}(T, P, Q)$. □

Theorem 4. *The proof rule* COMP-TERM *is sound and complete.*

Proof. The soundness of the proof rule follows directly from the **if**-direction of Theorem 3, and Lemma 4.

For proving completeness, we assume that the FDS $\mathcal{S}$ terminates under the compassion requirements $\mathcal{C}$. We construct a set L of labeled relations that satisfies all premises of

FDS $\mathcal{S}$ with:
states Σ,
compassion requirements $\mathcal{C}$,
transitions $\mathcal{T}$,
Set of labeled relations $L = \{(T_1, P_1, Q_1), \ldots, (T_n, P_n, Q_n)\}$ such that:
$T_i \subseteq \Sigma \times \Sigma$ and $P_i, Q_i \subseteq |\mathcal{C}|$ for all $i \in \{1, \ldots, n\}$

P1:	$(\rho_\tau, \mathsf{None}(\rho_\tau), \mathsf{Some}(\rho_\tau)) \subseteq_{\mathsf{FDS}} L$	for each $\tau \in \mathcal{T}$
P2:	$F_{\mathsf{FDS},\tau}(L) \subseteq_{\mathsf{FDS}} L$	for each $\tau \in \mathcal{T}$
P3:	$P_i \cup Q_i \neq \|\mathcal{C}\|$ or T_i well-founded	for each $i \in \{1, \ldots, n\}$

FDS $\mathcal{S}$ terminates under compassion requirements $\mathcal{C}$

Fig. 2. Proof rule COMP-TERM.

the proof rule COMP-TERM. Let L be the set of labeled relations defined as follows. For each pair of sets of indices $P \subseteq |\mathcal{C}|$ and $Q \subseteq |\mathcal{C}|$ let (T, P, Q) be a labeled relation in L such that a pair of states (s, s') is an element of the relation T if there exists a computation segment $s_1, \ldots, s_n$ such that $s_1 = s$, $s_n = s'$, $P = \mathsf{None}(\{s_1, \ldots, s_n\})$, and $Q = \mathsf{Some}(\{s_1, \ldots, s_n\})$.

We prove that L satisfies all premises of the proof rule COMP-TERM. We make the following assumptions on the transition relations ρ_τ, where $\tau \in \mathcal{T}$.

Assumption 1 *For every pair (s, s') of states in the transition relation ρ_τ, where $\tau \in \mathcal{T}$, the sequence s, s' is a computation segment.*

This assumption is not a proper restriction. We can assume that the transition relations are restricted to the accessible states. Alternatively, we may use a weaker version of the proof rule that restricts the transition relations ρ_τ in the premise P1 to the accessible states Acc.

Assumption 2 *For each transition $\tau \in \mathcal{T}$ there exists two sets of indices P and Q of compassion requirements such that for every pair (s, s') of states in ρ_τ we have $P = \mathsf{None}(\{s, s'\})$ and $Q = \mathsf{Some}(\{s, s'\})$.*

This assumption can be fulfilled by splitting every transition relation according to the sets that appear in the fairness requirements. Now we prove that L satisfies every premise of the proof rule.

Premise P1**:** We show that for every program transition $\tau \in \mathcal{T}$ the condition $(\rho_\tau, \mathsf{None}(\rho_\tau), \mathsf{Some}(\rho_\tau)) \subseteq_{\mathsf{FDS}} (T, P, Q)$ holds for the labeled relation $(T, P, Q) \in L$ such that $P = \mathsf{None}(\rho_\tau)$ and $Q = \mathsf{Some}(\rho_\tau)$. We need to prove $\rho_\tau \subseteq T$. For every pair of states (s, s') in ρ_τ the sequence s, s' is a computation segment, by Assumption 1. Furthermore, we have $\mathsf{None}(\{s, s'\}) = P$ and $\mathsf{Some}(\{s, s'\}) = Q$, by Assumption 2. Hence, by construction of the labeled relation (T, P, Q), the pair (s, s') is an element of the relation T.

Premise P2**:** We show that for every labeled relation $(T_1, P_1, Q_1) \in L$ and for every transition $\tau \in \mathcal{T}$ it holds $F_{\mathsf{FDS},\tau}(T_1, P_1, Q_1) \subseteq_{\mathsf{FDS}} (T_2, P_2, Q_2)$, where (T_2, P_2, Q_2)

is the labeled relation in L such that $P_2 = P_1 \cap \mathsf{None}(F_\tau(T_1))$ and $Q_2 = Q_1 \cup \mathsf{Some}(F_\tau(T_1))$. We need to prove $T_1 \circ \rho_\tau \subseteq T_2$.

We note the following auxiliary statement. For every pair (s, s') of states in T_1 we have $P_1 \subseteq \mathsf{None}(\{s\})$, $\mathsf{Some}(\{s\}) \subseteq Q_1$, $P_1 \subseteq \mathsf{None}(\{s'\})$, and $\mathsf{Some}(\{s'\}) \subseteq Q_1$. To justify the statement above for the pair $(s, s') \in T_1$, we consider a computation segment $s, \ldots, s'$ that is captured by (T_1, P_1, Q_1) such that $\mathsf{None}(\{s, \ldots, s'\}) = P_1$ and $\mathsf{Some}(\{s, \ldots, s'\}) = Q_1$, which exists by construction of (T_1, P_1, Q_1). From the definitions of None and Some, our auxiliary statement follows directly.

Now we are ready to prove $T_1 \circ \rho_\tau \subseteq T_2$. For a pair of states $(s_1, s_n) \in T_1$ there exists a computation segment $s_1, \ldots, s_n$ that is captured by the labeled relation (T_1, P_1, Q_1) such that $\mathsf{None}(\{s_1, \ldots, s_n\}) = P_1$ and $\mathsf{Some}(\{s_1, \ldots, s_n\}) = Q_1$, by construction of (T_1, P_1, Q_1). By Lemma 3, for a pair of states $(s_n, s_{n+1}) \in \rho_\tau$ the computation segment $s_1, \ldots, s_n, s_{n+1}$ is captured by the labeled relation $F_{\mathsf{FDS},\tau}(T_1, P_1, Q_1)$. Next, we prove the equalities

$$\mathsf{None}(\{s_1, \ldots, s_n, s_{n+1}\}) = P_1 \cap \mathsf{None}(F_\tau(T_1)) \,,$$
$$\mathsf{Some}(\{s_1, \ldots, s_n, s_{n+1}\}) = Q_1 \cup \mathsf{Some}(F_\tau(T_1)) \,,$$

from which $(s_1, s_{n+1}) \in T_2$ follows directly, by construction of (T_2, P_2, Q_2). We follow the chain of observations below:

$$\begin{aligned}
&\mathsf{None}(\{s_1, \ldots, s_n, s_{n+1}\}) \\
&= P_1 \cap \mathsf{None}(\{s_n, s_{n+1}\}) \\
&= P_1 \cap \mathsf{None}(\{s_n, s_{n+1}\}) \cap \bigcup_{(s,s') \in T_1, (s', s'') \in \rho_\tau} \mathsf{None}(\{s\}) && \text{since } P_1 \subseteq \mathsf{None}(\{s\}) \\
&= P_1 \cap \bigcup_{(s,s') \in T_1, (s', s'') \in \rho_\tau} (\mathsf{None}(\{s\}) \cap \mathsf{None}(\{s_n, s_{n+1}\})) \\
&= P_1 \cap \bigcup_{(s,s') \in T_1, (s', s'') \in \rho_\tau} (\mathsf{None}(\{s\}) \cap \mathsf{None}(\{s', s''\})) && \text{by Assumption 2} \\
&= \bigcup_{(s,s') \in T_1, (s', s'') \in \rho_\tau} (\mathsf{None}(\{s, s''\}) \cap \mathsf{None}(\{s'\}) \cap P_1) \\
&= \bigcup_{(s,s') \in T_1, (s', s'') \in \rho_\tau} (\mathsf{None}(\{s, s''\}) \cap P_1) && \text{since } P_1 \subseteq \mathsf{None}(\{s'\}) \\
&= P_1 \cap \mathsf{None}(F_\tau(T_1)) \,.
\end{aligned}$$

The proof of $\mathsf{Some}(\{s_1, \ldots, s_n, s_{n+1}\}) = Q_1 \cup \mathsf{Some}(F_\tau(T_1))$ is analogous.

Premise P3**:** We show, by contraposition, that for every labeled relation (T, P, Q) in L such that $P \cup Q = |\mathcal{C}|$ we have that the relation T is well-founded.

Assume that there exists an infinite sequence of states $s^1, s^2, \ldots$ such that for all $i \geq 1$ the pair (s^i, s^{i+1}) is an element of T, *i.e.*, the relation T is not well-founded. By construction of (T, P, Q), the state s^1 is accessible from some initial state $s_1 \in \Theta$. Furthermore, for all $i \geq 1$ there exists a computation segment $(s^i, \ldots, s^{i+1}) \in \mathsf{seg}(T, P, Q)$ that connects the states s^i and s^{i+1}. For connecting the states s^i and s^{i+1} we choose a computation segment such that $\mathsf{None}(\{s^i, \ldots, s^{i+1}\}) = P$ and

$\mathsf{Some}(\{s^i, \ldots, s^{i+1}\}) = Q$. Such a segment exists by construction of (T, P, Q). We conclude that there exists an infinite computation $\sigma = s_1, \ldots, s^1, \ldots, s^2, \ldots$. Next, we prove that σ satisfies all compassion requirements.

For each $j \in P$ we have that p_j-states does not appear in σ after the state s^1. For each $j \in Q$ and each $i \geq 1$ we have that some q_j-state appear in the segment $s^i, \ldots, s^{i+1}$. Since $P \cup Q = |\mathcal{C}|$, the computation σ satisfies all compassion requirements.

There is a contradiction to our assumption that $\mathcal{S}$ terminates under the compassion requirements $\mathcal{C}$. □

Proof. Theorem 3 (**only if**-direction) The set of labeled relations constructed in the completeness part of the proof of Theorem 4 satisfies all premises of the proof rule COMP-TERM. By Lemma 4, such a set captures all computation segments. Hence, whenever the FDS $\mathcal{S}$ terminates under the compassion requirements $\mathcal{C}$, there exists a set of labeled relations L such that for each $(T, P, Q) \in L$ we have either $P \cup Q \neq |\mathcal{C}|$ or the relation T is well-founded. □

We obtain a proof rule for the verification of general temporal properties of fair discrete systems by following Remark 1. We account for temporal properties given by a generalized Büchi automaton or a Streett automaton in a straightforward way, since generalized Büchi and Streett acceptance conditions are directly expressible as compassion requirements.

5 Correctness of the Analysis

We prove the soundness of the analysis as follows. First, we observe that the abstract least fixed point $\mathsf{lfp}(F^{\#}_{\mathsf{FDS}}, \mathcal{T})$ represents a finite set L of labeled relations:

$$L \;=\; \{(\gamma(T), P, Q) \mid (T, P, Q) \in \mathsf{lfp}(F^{\#}_{\mathsf{FDS}}, \mathcal{T})\}\ .$$

that satisfies the premises P1 and P2 of the proof rule COMP-TERM. This observation holds since the operator $F^{\#}_{\mathsf{FDS}}$ is a conservative approximation of the operator F_{FDS}, due to the Galois connection $(\alpha_{\mathsf{FDS}}, \gamma_{\mathsf{FDS}})$. Hence, whenever the analysis gives the positive answer then the set L satisfies all premises of the proof rule. By Theorem 4, we conclude the analysis is sound.

The partial completeness of the analysis follows from the completeness of the proof rule. We assume that the FDS $\mathcal{S}$ terminates under the compassion requirements $\mathcal{C}$. Let L be a finite set of labeled relations that satisfies all premises of the proof rule. Such a set exists, by Theorem 4. Let the abstract domain $D^{\#}_{\mathsf{FDS}}$ contains an abstract value $L^{\#}$ such that $L = \gamma_{\mathsf{FDS}}(L^{\#})$. By Theorem 13 in [2], we conclude that the least fixed point $\mathsf{lfp}(F^{\#}_{\mathsf{FDS}}, \mathcal{T})$ computed on $D^{\#}_{\mathsf{FDS}}$ satisfies the condition that leads to the positive answer.

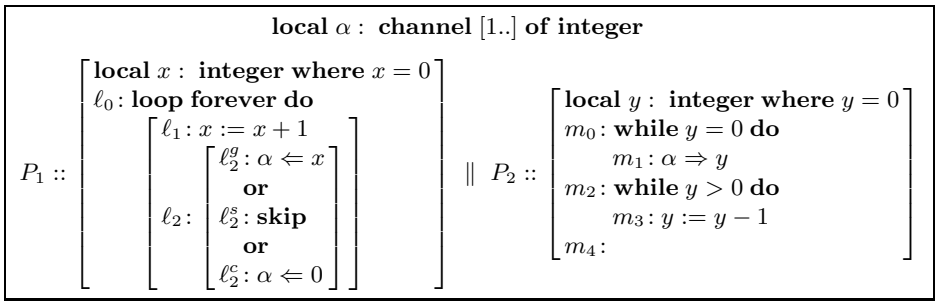

Fig. 3. Program CORR-ANY-DOWN.

6 Applications

We have implemented the analysis in a prototype tool using SICStus Prolog [10] and its built-in solver for linear arithmetic [7]. We applied the tool on several examples, described below.

In our implementation, we have instantiated the abstract domain $D^{\#}_{\mathsf{FDS}}$ by a set of abstract transitions. Abstract transitions are conjunctions built from some fixed, finite set of transition predicates [19]. A transition predicate denotes a binary relation over states, and is represented by an atomic assertion over unprimed and primed program variables, *e.g.* $x' \leq x - 1$. The abstraction of a relation T is the abstract transition $T^{\#}$ such that T entails the relation denoted by $T^{\#}$. The meaning of the concretization function γ is identity. We represent the relations $\gamma(T^{\#})$ by a 'simple' program that consists of a single while loop with only update statements in the loop body, following [18, 19]. There exist a number of well-foundedness tests for the class of simple while programs that are built using linear arithmetic expressions [1, 17, 22]. Our tool implements the test described in [17].

We give a brief description of the example programs. We start with the program CORR-ANY-DOWN, shown on Figure 3. The communication between the processes of the program CORR-ANY-DOWN takes place over an asynchronous channel α. The channel α is unreliable. Messages sent over the channel can be transmitted correctly, get lost or corrupted during the transmission. The transition $\alpha \Leftarrow x$ models a correct transmission, **skip** models the message loss, and $\alpha \Leftarrow 0$ models the message corruption [16]. We prove the eventual reachability of the location m_4.

This property relies on the assumption that the value of the variable x is eventually communicated to the variable y, *i.e.*, that the channel α is eventually reliable. We model the eventual reliability by a compassion requirement $\langle at\_\ell_1, at\_\ell_2^g \rangle$ that ensures a successful transmission if there are infinitely many attempts to send a message.

The eventual reliability of the communication channel is in fact not sufficient for proving termination. We also need to exclude computations in which one of the processes idles forever in some location. Hence, we introduce a justice requirement for each location, *e.g.* $\neg at\_\ell_1$ and $\neg(at\_m_0 \wedge y = 0)$.

	I	II	III
Number of justice requirements	10	5	5
Number of compassion requirements	1	0	0
Number of transition predicates	19	7	11
Least fixed point computation, sec	363.2	2.7	3.4
Well-foundedness tests, sec	0.5	0.03	0.04

Fig. 4. Analysis of the programs CORR-ANY-DOWN (I), BAKERY (II), and TICKET (III).

We model the asynchronous communication channel α by an integer array of infinite size. We keep track of the positions in the array at which the read and write operations take place, as well as the position at which the first successfully transmitted value is written.

The program BAKERY is a simplified version [15] of the Bakery mutual exclusion protocol [11] for two processes. We verify the starvation freedom for the first process. This means that whenever it leaves the non-critical section, it will eventually reach the critical section. The property relies on justice assumptions that none of the processes idles forever in some location.

The program TICKET is another mutual exclusion protocol. We verify the starvation freedom property for the first process. It requires the same kind of fairness requirements as the program BAKERY.

Figure 4 shows the collected statistics. For each program we give the number of justice and compassion requirements that were necessary to prove the property, and the number of transition predicates that induce the abstract domain $D^\#$. We measured the time spent on the fixed point computation $\mathsf{lfp}(F^\#_{\mathsf{FDS}}, \mathcal{T})$, and the well-foundedness checks $\mathsf{well\text{-}founded}(\gamma(T^\#))$ (see the analysis on Figure 1).

7 Conclusion

We have presented an analysis of temporal properties of fair discrete systems. Our analysis relies on the domain of labeled relations, which provides the separation of well-foundedness and fairness. We have successfully applied our analysis to verify temporal properties of interesting programs. The verified properties rely on justice and compassion requirements.

References

1. M. Colón and H. Sipma. Synthesis of linear ranking functions. In *Proc. of TACAS'2001: Tools and Algorithms for the Construction and Analysis of Systems*, volume 2031 of *LNCS*, pages 67–81. Springer, 2001.
2. P. Cousot. Partial completeness of abstract fixpoint checking. In *Proc. of SARA'2000: Abstraction, Reformulation, and Approximation*, volume 1864 of *LNCS*, pages 1–15. Springer, 2000.

3. P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proc. of POPL'1977: Principles of Programming Languages*, pages 238–252. ACM Press, 1977.
4. P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In *Proc. of POPL'1979: Principles of Programming Languages*, pages 269–282. ACM Press, 1979.
5. Y. Fang, N. Piterman, A. Pnueli, and L. D. Zuck. Liveness with incomprehensible ranking. In *Proc. of TACAS'2004: Tools and Algorithms for the Construction and Analysis of Systems*, volume 2988 of *LNCS*, pages 482–496. Springer, 2004.
6. Y. Fang, N. Piterman, A. Pnueli, and L. D. Zuck. Liveness with invisible ranking. In Steffen and Levi [21], pages 223–238.
7. C. Holzbaur. *OFAI clp(q,r) Manual, Edition 1.3.3*. Austrian Research Institute for Artificial Intelligence, Vienna, 1995. TR-95-09.
8. Y. Kesten, A. Pnueli, and L. Raviv. Algorithmic verification of linear temporal logic specifications. In *Proc. of ICALP'1998: Int. Colloq. on Automata, Languages and Programming*, volume 1443 of *LNCS*, pages 1–16. Springer, 1998.
9. N. Klarlund. Progress measures and stack assertions for fair termination. In *Proc. of PODC'1992: Principles of Distributed Computing*, pages 229–240. ACM Press, 1992.
10. T. I. S. Laboratory. *SICStus Prolog User's Manual*. Swedish Institute of Computer Science, PO Box 1263 SE-164 29 Kista, Sweden, October 2001. Release 3.8.7.
11. L. Lamport. A new solution of Dijkstra's concurrent programming problem. *Communications of the ACM*, 17(8):453–455, 1974.
12. D. Lehmann, A. Pnueli, and J. Stavi. Impartiality, justice and fairness: The ethics of concurrent termination. In *Proc. of ICALP'1981: Int. Colloq. on Automata, Languages and Programming*, volume 115 of *LNCS*, pages 264–277. Springer, 1981.
13. O. Lichtenstein and A. Pnueli. Checking that finite state concurrent programs satisfy their linear specification. In *Proc. of POPL'1985: Principles of Programming Languages*, pages 97–107. ACM Press, 1985.
14. Z. Manna and A. Pnueli. Completing the temporal picture. *Theoretical Computer Science*, 83(1):91–130, 1991.
15. Z. Manna and A. Pnueli. *Temporal verification of reactive systems: Safety*. Springer, 1995.
16. Z. Manna and A. Pnueli. Temporal verification of reactive systems: Progress. Draft, 1996.
17. A. Podelski and A. Rybalchenko. A complete method for the synthesis of linear ranking functions. In Steffen and Levi [21], pages 239–251.
18. A. Podelski and A. Rybalchenko. Transition invariants. In *Proc. of LICS'2004: Logic in Computer Science*, pages 32–41. IEEE, 2004.
19. A. Podelski and A. Rybalchenko. Transition predicate abstraction and fair termination. In *Proc. of POPL'2005: Principles of Programming Languages*. ACM Press, 2005. To appear.
20. F. P. Ramsey. On a problem of formal logic. In *Proc. London Math. Soc.*, volume 30, pages 264–285, 1930.
21. B. Steffen and G. Levi, editors. *Proc. of VMCAI'2004: Verification, Model Checking, and Abstract Interpretation*, volume 2937 of *LNCS*. Springer, 2004.
22. A. Tiwari. Termination of linear programs. In *Proc. of CAV'2004: Computer Aided Verification*, volume 3114 of *LNCS*, pages 70–82. Springer, 2004.
23. M. Y. Vardi. Verification of concurrent programs — the automata-theoretic framework. *Annals of Pure and Applied Logic*, 51:79–98, 1991.