Completing the Temporal Picture *†

Zohar Manna
Stanford University ‡
and
Weizmann Institute of Science§

Amir Pnueli
Weizmann Institute of Science§

Abstract. The paper presents a relatively complete proof system for proving the validity of temporal properties of reactive programs. The presented proof system improves on previous temporal systems, in that it reduces the validity of program properties into pure assertional reasoning, not involving additional temporal reasoning. The proof system is based on the classification of temporal properties according to the Borel hierarchy, providing appropriate proof rules for the classes of *safety*, *response*, and *reactivity* properties.

1 Introduction

Temporal logic is, by now, one of the acceptable and frequently used approaches to the formal specification and verification of concurrent and reactive programs. Even though we have witnessed, over the last several years, a great progress in the automatic verification of finite-state programs, the main tool for establishing that a proposed implementation satisfies its temporal specification is still that of deductive verification, using a set of axioms and inference rules.

As described in [MP83a] (see also [MP83b] and [Pnu86]), a proof system that supports the verification of temporal properties over reactive programs has to deal with three types of validity.

- $\mathcal{A}$– *Assertional* validity. This is the validity of non-temporal (state) formulae (also called *assertions*) over an arbitrary state (interpretation).

- $\mathcal{T}$– *General temporal* validity. This is the validity of temporal formulae over arbitrary sequences of states (models).

*This research was supported in part by the National Science Foundation under grants CCR-89-11512 and CCR-89-13641, by the Defense Advanced Research Projects Agency under contract N00039-84-C-0211, by the United States Air Force Office of Scientific Research under contract AFOSR-90-0057, and by the European Community ESPRIT Basic Research Action Project 3096 (SPEC).

†A preliminary version of this paper appeared in [MP89b].

‡Department of Computer Science, Stanford University, Stanford, CA 94305

§Department of Applied Mathematics, Weizmann Institute, Rehovot, Israel

- $\mathcal{P}$– *Program* validity. This is the validity of temporal formulae over sequences of states which represent computations of the analyzed program.

Corresponding to these three types of validity, the proof system may be partitioned into three parts, each providing axioms and rules for establishing the validity of the corresponding type. This is essentially the structure of the proof system presented in [MP83b], where we refer to the assertional part as the *domain* part.

The program part presents some basic proof rules and some derived rules. The derived rules provide direct support for proving some of the most frequently used temporal properties of programs.

One group of rules establishes the validity of the *invariance* formulae $\Box\, q$ and $\Box(p \rightarrow \Box\, q)$, which express the invariance of a state formula q, either throughout the computation, or triggered by the occurrence of p.

Another group of rules establishes the validity of the *eventuality* formulae $\Diamond\, q$ and $\Box(p \rightarrow \Diamond\, q)$, which express the guarantee that q will eventually happen, either once or following each occurrence of p.

These proof rules are completely satisfactory for establishing this restricted but very prevalent set of temporal formulae. The rules derive temporal conclusions from assertional premises. They have been proven relatively complete, and are the main working tools for verification of the temporal properties of programs (see, e.g., [OL82, MP84, Kro87]).

However, a question which is only partially answered in [MP83a] is how to prove all the other properties whose expression in temporal logic does not fall into the restricted class of invariance and eventuality formulae. The partial solution given there is a relative completeness result, which shows how to reduce the problem of $\mathcal{P}$-validity of a temporal formula into a set of validity problems for formulae, which are either assertional ($\mathcal{A}$-valid), or temporal but valid over arbitrary sequences of states ($\mathcal{T}$-valid).

We remind the reader that this is the general character of all *relative* completeness results for program logics such as Hoare logic [Apt81] or Dynamic Logic [Har79]. Since, as soon as we consider programs that operate over infinite domains, we lose the possibility of having true completeness, the best we can hope for is relative completeness [Coo78]. This type of completeness ensures an effective reduction from the validity of a program logic statement into the validity of a finite number of assertional statements.

Unfortunately, the reduction given in [MP83a] is not only into assertional statements, but also into generally ($\mathcal{T}$-) valid temporal statements. This requires a proof of a general program property to be based not only on assertional reasoning, but also on *temporal reasoning*, which is less familiar, even to a person who is well versed in general logic. This fact has been considered by some researchers a deficiency, and has caused them to shy away from the temporal proof system and look for alternative formalisms, in which a complete reduction into assertional statements is guaranteed ([AS89], and also see [MP87]).

In this paper we attempt to remedy the situation by providing a richer proof system for the program part, which ensures complete reduction of a general temporal formula (given in a canonical form) into a finite set of assertional statements, whose validity implies the validity of the original temporal formula.

The approach to a complete proof system is based on a classification of temporal properties according to their expression in a *canonical form*, which applies a set of restricted future modalities to arbitrary past formulae. This classification establishes a hierarchy of temporal properties [MP89], whose classes can be characterized according to three different criteria. The first characterization is in terms of the syntactic form of their canonical representation. Another characterization is semantical, looking at a property as the set of all sequences which have this property. By this view we can give a topological characterization to the classes in the hierarchy, locating it at the first two levels of the Borel hierarchy. The third characterization is in terms of structural restrictions on the Streett automaton that recognizes precisely the set of the infinite sequences which have the property.

In principle, we should provide a separate proof rule for each of the property classes in the hierarchy. In practice, we concentrate on three particular classes, which have special significance as expressing most of the interesting program properties, and which form a natural generalization of the two classes of invariance and eventuality properties considered in the previous proof systems. These are the classes of:

- *Safety properties.* These are all the properties that can be expressed by a temporal formula of the form $\square\, q$ for some *past* formula q.
- *Response properties.* These are all the properties that can be expressed by a temporal formula of the form $\square(p \rightarrow \diamond\, q)$, or alternately, $\square \diamond q$ for some *past* formulae p and q.
- *Reactivity properties.* These are all the properties that can be expressed by a conjunction of temporal formulae of the form $\square \diamond p \vee \diamond \square\, q$ for some *past* formulae p and q.

We provide complete rules for each of these classes. This provides full coverage for the entire temporal logic, since by [LPZ85] (see also [Tho81]), any property specifiable by temporal logic is a reactivity property.

2 Programs and computations

The basic computational model we use to represent programs is that of a *fair transition system.* In this model, a program P consists of the following components.

- $V = \{u_1, ..., u_n\}$: A finite list of *state variables.* Some of these variables represent *data* variables, which are explicitly manipulated by the program text. Other variables are *control* variables, which represent, for example, the location of control in each of the processes in a concurrent program. We assume each variable to be associated with a domain over which it ranges.
- Σ : A set of *states.* Each state $s \in \Sigma$ is an interpretation of V, assigning to each variable $y \in V$ a value over its domain, which we denote by $s[y]$. We denote by Σ^+ the set of all nonempty finite sequences of states.
- $\mathcal{T}$: A finite set of *transitions.* Each transition $\tau \in \mathcal{T}$ is associated with an assertion $\rho_\tau(V, V')$, called the *transition relation*, which refers to both an unprimed and a primed version of the state variables. The purpose of the transition relation ρ_τ is to express a relation between a state s and its successor s'. We use the unprimed

version to refer to values in s, and the primed version to refer to values in s'. For example, the assertion $x' = x + 1$ states that the value of x in s' is greater by 1 than its value in s.

- Θ : The *precondition*. This is an assertion characterizing all the initial states, i.e., states at which the computation of the program can start. A state is defined to be *initial* if it satisfies Θ.
- $\mathcal{J} \subseteq \mathcal{T}$: A subset of *just* transitions (also called *weakly fair* transitions). Intuitively, the requirement of justice for $\tau \in \mathcal{J}$ disallows a computation in which τ is continually enabled beyond a certain point but taken only finitely many times.
- $\mathcal{C} \subseteq \mathcal{T}$: A subset of *compassionate* transitions (also called *strongly fair* transitions). Intuitively, the requirement of compassion for $\tau \in \mathcal{C}$ disallows a computation in which τ is enabled infinitely many times but taken only finitely many times.

We define the state s' to be a τ*-successor* of the state s if

$$\langle s, s' \rangle \models \rho_\tau(V, V'),$$

where $\langle s, s' \rangle$ is the joint interpretation which interprets $x \in V$ as $s[x]$, and interprets x' as $s'[x]$. Following this definition, we can view the transition τ as a function $\tau : \Sigma \mapsto 2^\Sigma$, defined by:

$$\tau(s) = \{s' \mid s' \text{ is a } \tau\text{-successor of } s\}.$$

We say that the transition τ is *enabled* on the state s if $\tau(s) \neq \phi$. Otherwise, we say that τ is *disabled* on s. The enabledness of a transition τ can be expressed by the formula

$$En(\tau) : \; (\exists V')\rho_\tau(V, V'),$$

which is true in s iff s has some τ-successor.

We assume that one of the transitions $\tau_0 \in \mathcal{T}$ is the *idling* transition (also called *stuttering*), with the transition relation $\rho_0 : (V = V')$. Thus, s' is a τ_0-successor of s iff $s' = s$. This guarantees that every state $s \in \Sigma$ has at least one transition enabled on it.

Given a program P for which the above components have been specified, we define a *computation* of P to be an infinite sequence of states $\sigma : s_0, s_1, s_2, \ldots$, satisfying the following requirements:

- *Initiation:* s_0 is initial, i.e., $s_0 \models \Theta$.
- *Consecution:* For each $j = 0, 1, \ldots$, the state s_{j+1} is a τ-successor of the state s_j, i.e., $s_{j+1} \in \tau(s_j)$, for some $\tau \in \mathcal{T}$. In this case, we say that the transition τ is *taken* at position j in σ.
- *Justice:* For each $\tau \in \mathcal{J}$ it is not the case that τ is continually enabled beyond some point in σ but taken at only finitely many positions in σ.
- *Compassion:* For each $\tau \in \mathcal{C}$ it is not the case that τ is enabled on infinitely many states of σ but taken at only finitely many positions in σ.

For a program P, we denote by $\mathcal{C}omp(P)$ the set of all computations of P.

3 Temporal logic

We assume an underlying assertion language $\mathcal{L}$, which contains the predicate calculus, and interpreted symbols for expressing the standard operations and relations over some concrete domains. For the sake of completeness, we require that one of the domains is that of the integers, or another domain with similar expressive power.

As is known [SdRG89] (see also [SdRG84, AP86], and [Fra86]), a first-order language is not adequate to express the assertions necessary for completeness. We follow [SdRG89] and include in $\mathcal{L}$ also the fixpoints operators μ and ν. Let $Q(u_1, \ldots, u_r)$ be a predicate symbol of arity r, and $\Phi(Q, u_1, \ldots, u_r)$ be a formula whose free variables are a subset of $\{u_1, \ldots, u_r\}$, and that have only *positive* occurrence of Q, i.e., occurrences under an even number of negations. Consider the equation

$$Q(u_1, \ldots, u_r) \equiv \Phi(Q, u_1, \ldots, u_r) \tag{1}$$

Assume that all other elements, except for the predicate symbol Q and the variables $u_1, \ldots, u_r$, are interpreted. Then, (1) can be viewed as a fixpoint equation for the predicate Q. A concrete relation $\widetilde{Q}$ is said to be a *solution* of (1) if, when substituted for Q, it makes (1) valid, i.e., both sides are equal for each assignment of values to $u_1, \ldots, u_r$ over their respective domains.

Consider, for example, the equation

$$Q(u) \equiv \Big((u = 0) \vee Q(u-1)\Big) \tag{2}$$

where u ranges over the integers.

The equation (1) may have more than one solution for the predicate Q. For example, both $u \geq 0$ and $\mathbf{T}$ are solutions of (2). Under the assumption that Q appears positively in Φ, it is known that (1) always has two special solutions, called the *minimal* and *maximal* solutions and denoted by $\mu Q : \Phi$ and $\nu Q : \Phi$, respectively. These special solutions have the property that any other solution is larger than (is implied by) the minimal solution and is smaller than (implies) the maximal one. For (2), $u \geq 0$ and $\mathbf{T}$ are the minimal and maximal solutions, respectively.

Thus, the assertion language $\mathcal{L}$ contains formulae of the forms $\mu Q : \Phi$ and $\nu Q : \Phi$ for Q and Φ restricted as above. Unlike [SdRG89], we do not assume that $\mathcal{L}$ has constants or variables that range over the *ordinals*.

We refer to a formula in the assertion language $\mathcal{L}$ as a *state formula*, or simply as an *assertion*.

A *temporal formula* is constructed out of state formulae to which we apply the boolean operators $\neg$ and $\vee$ (the other boolean operators can be defined from these), and the following basic temporal operators:

$\bigcirc$ – Next $\quad \ominus$ – Previous

$\mathcal{U}$ – Until $\quad \mathcal{S}$ – Since

A *model* for a temporal formula p is an infinite sequence of states $\sigma : s_0, s_1, \ldots$, where each state s_j provides an interpretation for the variables mentioned in p.

Given a model σ, as above, we present an inductive definition for the notion of a temporal formula p holding at a position $j \geq 0$ in σ, denoted by $(\sigma, j) \models p$.

- For a state formula p,
  $(\sigma, j) \models p \iff s_j \models p$
  That is, we evaluate p locally, using the interpretation given by s_j.
- $(\sigma, j) \models \neg p \iff (\sigma, j) \not\models p$
- $(\sigma, j) \models p \vee q \iff (\sigma, j) \models p$ or $(\sigma, j) \models q$
- $(\sigma, j) \models \bigcirc p \iff (\sigma, j+1) \models p$
- $(\sigma, j) \models p \mathcal{U} q \iff$ for some $k \geq j, (\sigma, k) \models q$, and for every i such that $j \leq i < k, (\sigma, i) \models p$
- $(\sigma, j) \models \ominus p \iff j > 0$ and $(\sigma, j-1) \models p$
- $(\sigma, j) \models p \mathcal{S} q \iff$ for some $k \leq j, (\sigma, k) \models q$, and for every i such that $j \geq i > k, (\sigma, i) \models p$

Additional temporal operators can be defined as follows:

$\diamondsuit p = \mathbf{T} \mathcal{U} p$	– Eventually
$\square p = \neg \diamondsuit \neg p$	– Henceforth
$p \mathcal{W} q = \square p \vee (p \mathcal{U} q)$	– Unless (weak until)
$\diamondminus p = \mathbf{T} \mathcal{S} p$	– Sometimes in the past
$\boxminus p = \neg \diamondminus \neg p$	– Always in the past
$p \mathcal{B} q = \boxminus p \vee (p \mathcal{S} q)$	– Weak Since

Another useful derived operator is the *entailment* operator, defined by:

$$p \Rightarrow q \iff \square(p \rightarrow q).$$

A formula that contains no future operators is called a *past* formula. A formula that contains no past operators is called a *future* formula. Note that a state formula is both a past and a future formula. We refer to a past formula [future formula] that is not also a state formula, as a *strict-past* [*strict-future*, respectively] formula.

For a state formula p and a state s such that p holds on s, we say that s is a *p-state*. A state formula that holds on all states is called *assertionally valid* (*$\mathcal{A}$-valid* for short).

For a temporal formula φ and a position $j \geq 0$ such that $(\sigma, j) \models \varphi$, we say that j is a *φ-position* (in σ). Note that the satisfaction of a past formula at position $j \geq 0$ depends only on the finite prefix $s_0, \ldots, s_j$.

If $(\sigma, 0) \models p$, we say that p *holds* on σ, and denote it by $\sigma \models p$. A formula p is called *satisfiable* if it holds on some model. A formula is called *temporally valid* (*$\mathcal{T}$-valid* for short) if it holds on all models.

Two formulae p and q are defined to be *equivalent* if the formula $p \equiv q$ is valid, i.e., $\sigma \models p$ iff $\sigma \models q$, for all σ.

The notion of temporal validity requires that the formula holds over *all* models. Given a program P, we can restrict our attention to the set of models which correspond to computations of P, i.e., $\mathcal{C}omp(P)$. This leads to the notion of $\mathcal{P}$-validity, by which p is *P-valid* if it holds over all the computations of P. Similarly, we obtain the notions of P-satisfiability and P-equivalence.

3.1 Canonical form and classification

By [LPZ85] (see also [Tho81, MP90a]), every temporal formula is equivalent to a formula of the form

$$\bigwedge_{i=1}^{n} (\Box \Diamond p_i \vee \Diamond \Box q_i),$$

for some past formulae $p_i, q_i, i = 1, ..., n$.

Based on this canonical form we can classify the properties expressible by temporal logic according to their expressibility by restricted cases of this general formula. We list below the main classes in this classification, specifying their temporal characterizations. For each class we present the form of the temporal formulae that express the properties in that class, where the subformulae p, q, p_i, q_i appearing there are arbitrary past formulae. We refer the reader to [MP89, MP90b] for additional properties and characterizations of this hierarchy.

- *Safety properties* – $\Box p$.
- *Guarantee properties* – $\Diamond p$.
- *Obligation properties* – $\bigwedge_{i=1}^{n}(\Box p_i \vee \Diamond q_i)$.
- *Response properties* – $\Box \Diamond p$.
- *Persistence properties* – $\Diamond \Box p$.
- *Reactivity properties* – $\bigwedge_{i=1}^{n}(\Box \Diamond p_i \vee \Diamond \Box q_i)$.

As stated above, the reactivity class is the maximal class of properties expressible by temporal logic.

4 Rules for safety

From now on, we fix our attention on a program P, specified by the components $\langle V, \Sigma, \mathcal{T}, \Theta, \mathcal{J}, \mathcal{C} \rangle$.

In this section we consider proof rules for establishing the $\mathcal{P}$-validity of a safety formula. As we recall, a safety formula has the form $\Box p$ for some past formula p. Let us review first the appropriate rule for the simpler case that p is a state formula. This case can also be characterized by saying that $\Box p$ is a future formula.

For a transition τ, and state formulae p and q, we define the *verification condition* of τ, relative to p and q, to be the implication:

$$(\rho_\tau \wedge p) \rightarrow q', \text{ denoted by } \{p\}\tau\{q\},$$

where ρ_τ is the transition relation corresponding to τ, and q', the *primed version* of the assertion q, is obtained from q by replacing each variable occurring in q by its primed version. Since ρ_τ holds for two states s and s' iff s' is a τ-successor of s, and q' states that q holds on s', it is not difficult to see that

If the verification condition $\{p\}\tau\{q\}$ is assertionally valid, then every τ-successor of a p-state is a q-state.

For a set of transitions $T \subseteq \mathcal{T}$, we denote by $\{p\}T\{q\}$ the verification condition of T, relative to p and q, requiring that $\{p\}\tau\{q\}$ holds for *every* $\tau \in T$.

The following rule is sound and complete for establishing the P-validity of the future safety formula $\square\, q$ (i.e. when q is a state formula) over the program P.

$$\begin{array}{|lll|}
\hline
\textbf{INV} & \text{I1.} & \Theta \rightarrow \varphi \\
 & \text{I2.} & \varphi \rightarrow q \\
 & \text{I3.} & \{\varphi\}\mathcal{T}\{\varphi\} \\
\cline{2-3}
 & & \square\, q \\
\hline
\end{array}$$

This rule uses an auxiliary assertion φ which, by premise I1, holds initially, and by premise I3 is propagated from each state to its successor. This shows that φ is an invariant of the program, that is, it holds continuously over all computations of P. Since, by I2, the assertion φ implies q, it follows that q is also an invariant of the program.

4.1 General safety

Next, we extend rule **INV** to deal with safety formulae $\square\, q$, such that q is a general past formula. First, we extend the notion of the primed version of a formula, to apply also to a past formula. Recall that the intended meaning of a primed formula is to express the value of a formula in the next state, in terms of the values of the variables in the next and the current states. This is inductively defined as follows:

- For a state formula $p(V)$, we define as before

$$(p(V))' = p(V').$$

- For a *previous* formula

$$(\ominus\, p)' = p.$$

  This corresponds to our intuition that $\ominus\, p$ holds in the next state iff p holds now.

- For a *since* formula

$$(p \mathcal{S} q)' \;=\; q' \vee ((p \mathcal{S} q) \wedge p').$$

  This corresponds to the intuition that $p \mathcal{S} q$ holds in the next state if, either q holds there, or $p \mathcal{S} q$ holds now and p holds next.

With this definition, we extend the notion of the verification condition $\{p\}\tau\{q\}$ to apply also to past formulae p and q, and to mean

$$(\rho_\tau \wedge p) \Rightarrow q'.$$

Note that since we work with temporal formulae, we replaced the implication by an entailment, because we expect the implication to hold at *all* positions of the computation, not only at the first one.

As in the case of state formulae, if $\{p\}\tau\{q\}$ is valid and j is a p-position such that τ is taken at j, then $j+1$ is a q-position.

With this extension, the general single rule for establishing safety properties is given by

$$\begin{array}{|lll|}\hline \textsc{safe} & \text{S1.} & (\Theta \wedge \textbf{first}) \Rightarrow\!\!\succ \varphi \\ & \text{S2.} & \varphi \Rightarrow\!\!\succ q \\ & \text{S3.} & \{\varphi\}\mathcal{T}\{\varphi\} \\ \cline{2-3} & & \square\, q \\ \hline \end{array}$$

The implications, appearing in premises I1 and I2 of rule $\textsc{inv}$, have been replaced in rule $\textsc{safe}$ by the entailments, appearing in premises S1 and S2. In S1 we also added the conjunct **first** which is an abbreviation for the formula $\neg \ominus \textsc{t}$, characterizing the first position in the computation as the only position that has no predecessor. This conjunct is sometimes necessary to ensure that φ holds in the first position.

4.2 A minimal general part

Examining premises S1 – S3 of rule $\textsc{safe}$, we observe that they all have the form of temporal formulae, which are actually other safety formulae. How are these to be proven? Obviously, we need some additional rules, belonging to the general part. These rules enable us to prove some temporal formulae that are generally valid, i.e., hold over any sequence of states, unrelated to any particular program.

The first rule we consider is the rule of temporal instantiation, which provides a basic tool for deriving temporal validities from assertional ones. Let q be a state formula containing the propositional symbol p, and let φ be a temporal formula. We denote by $q[\varphi/p]$ the temporal formula obtained from q by replacing all occurrences of p by φ.

$$\begin{array}{|ll|}\hline \textbf{INST} & q \\ \cline{2-2} & \square\, q[\varphi/p] \\ \hline \end{array}$$

Note, in particular, that if q has the form $t \rightarrow r$ then the temporal conclusion is an entailment of the form $t[\varphi/p] \Rightarrow\!\!\succ r[\varphi/p]$. This rule is often used, without any instantiation, to derive the temporal validity of $\square\, q$ from the assertional validity of q. In these cases, it is sometimes referred to as *generalization*.

The next rule we consider can be viewed as stating the monotonicity of the temporal operator $\square$. For two temporal formulae p and q, we can interpret the entailment $p \Rightarrow\!\!\succ q$, i.e., $\square(p \rightarrow q)$, as an ordering relation between the formulae, stating that p is smaller (stronger) than q. Indeed, for a sequence σ, $p \Rightarrow\!\!\succ q$ claims that the set of positions at which p holds is contained in the set of positions at which q holds. Monotonicity of the $\square$ operator states that if $p \Rightarrow\!\!\succ q$, and $\square\, p$ is valid, then so is $\square\, q$.

$$\begin{array}{|ll|}\hline \textbf{M-SAFE} & p \Rightarrow\!\!\succ q \\ & \square\, p \\ \cline{2-2} & \square\, q \\ \hline \end{array}$$

This rule can also be viewed as a temporal version of Modus Ponens, where entailment replaces implication. In fact, the two preceding rules provide a formal support for many elementary manipulations, such as substituting equals for equals, and using any instantiation of propositional tautologies. We refer to any such manipulation as justified by *propositional reasoning*.

In addition to these general rules, we need in our general part some properties which are specific to the initial part of a sequence of states. These will enable us to draw some conclusions from the formula **first**, as is needed in premise S1 of rule **SAFE**.

These are presented by the following two axioms:

- **I-PREV**: $\textbf{first} \Rightarrow \neg \ominus p$
- **I-SINCE**: $\textbf{first} \Rightarrow \big((p\,\mathcal{S}\,q) \equiv q\big)$

Axiom **I-PREV** states that no *previous* formula can hold at the initial position of any sequence. Axiom **I-SINCE** states that the formula $p\,\mathcal{S}\,q$ can hold at the initial position iff q holds there.

4.3 Completeness of the SAFE rule

We proceed to consider the applicability of rule **SAFE** to the proofs of safety properties. First, we present an example, illustrating its use.

Example 4.1 Consider the trivial program with a single state variable x, precondition $x = 0$, and a single transition τ whose transition relation is given by $\rho_\tau : x' = x + 1$. Observe that this program has a single infinite computation, given by $\langle x : 0\rangle, \langle x : 1\rangle, \langle x : 2\rangle, \ldots$

We wish to prove for this program the trivial safety property

$$\Box((x = 10) \rightarrow \diamondminus(x = 5)).$$

This property claims that any state in which $x = 10$ must have been preceded by a state in which $x = 5$. Note that this trivial property would not be true for a program that advances in steps of 2, rather than steps of 1.

To prove this property, we identify q as $(x = 10) \rightarrow \diamondminus(x = 5)$ and intend to use rule **SAFE**. As the auxiliary formula φ, we take $(x \geq 5) \rightarrow \diamondminus(x = 5)$. The rule requires showing the following three premises:

S1. $[(x = 0)\ \wedge\ \textbf{first}]\ \Rightarrow\ ((x \geq 5) \rightarrow \diamondminus(x = 5))$

S2. $((x \geq 5) \rightarrow \diamondminus(x = 5))\ \Rightarrow\ ((x = 10) \rightarrow \diamondminus(x = 5))$

S3. $[(x' = x + 1)\ \wedge\ ((x \geq 5) \rightarrow \diamondminus(x = 5))]\ \Rightarrow\ [(x' \geq 5) \rightarrow ((x' = 5) \vee \diamondminus(x = 5))]$

In S3 we have already expanded $(\diamondminus(x = 5))'$ into $((x' = 5) \vee \diamondminus(x = 5))$. All of these apparently temporal formulae can be established by rule **INST**, using the following three valid state formulae, and their associated instantiations.

V1. $((x = 0) \wedge p) \rightarrow ((x \geq 5) \rightarrow r)$,

with the replacement of $(\mathbf{first}, \diamondminus(x = 5))$ for the proposition symbols (p, r), respectively, establishes S1.

V2. $((x \geq 5) \rightarrow p) \rightarrow ((x = 10) \rightarrow p)$,

with the replacement of $\diamondminus(x = 5)$ for the proposition symbol p, establishes S2.

V3. $[(x' = x + 1) \ \wedge \ ((x \geq 5) \rightarrow p)] \rightarrow [(x' \geq 5), \rightarrow ((x' = 5) \vee p)]$

with the replacement of $\diamondminus(x = 5)$ for the proposition symbol p, establishes S3. ∎

Theorem 8.2, presented in Section 8, establishes the adequacy of rule SAFE by stating:

> *Rule* SAFE *is complete, relative to assertional validity, for proving the $\mathcal{P}$-validity of any safety property.*

The proof of the theorem is based on the construction of a big past formula φ which relates the values of variables in an accessible state (i.e., appearing in some computation of P) to the boolean values of the past sub-formulae of the formula $\square\, q$, whose validity we wish to establish.

4.4 Incremental proofs

In the previous paragraphs, we have considered how to establish the invariance of some past formulae. Having established some basic invariants of this form, we may want to use them in order to derive more complex properties. For this purpose, we quote again rule M-SAFE, which suggests a strategy, to which we refer as *incremental verification*. According to this strategy, we establish first the validity of a simpler safety property $\square\, p$. Later, whenever we have to establish the validity (over P) of a premise that has the form $\square\, \psi$, we can instead establish the validity of $p \Rightarrow \psi$.

5 Rules for response

Response properties are those which can be expressed by a formula of the form

$$p \Rightarrow \diamond\, q, \text{ or equivalently } \square(p \rightarrow \diamond\, q)$$

for some past formulae p and q.

The rules for establishing response properties can be partitioned into *single-step* rules and *extended* rules. We consider each group in turn.

5.1 Rules for single-step response

These are the rules that establish properties that only depend on the execution of a single helpful transition (which may be selected out of several candidates) to accomplish the guaranteed response q. We have two rules in this group, which differ according to the type of *fairness* on which they rely.

The first single-step rule relies on justice to ensure that eventually a helpful transition, leading to q, will be taken. It refers to a *just* transition $\tau \in \mathcal{J}$.

$$\textbf{J-RESP}\quad \begin{array}{ll} \text{J1.} & p \Rightarrow (q \vee \varphi) \\ \text{J2.} & \{\varphi\}\mathcal{T}\{q \vee \varphi\} \\ \text{J3.} & \{\varphi\}\tau\{q\} \\ \text{J4.} & \varphi \Rightarrow (q \vee En(\tau)) \\ \hline & p \Rightarrow \Diamond q \end{array}$$

Premise J1 ensures that p entails q or φ. Premise J2 states that any transition of the program either leads from φ to q, or preserves φ. Premise J3 states that the helpful transition τ leads from φ to q. Premise J4 ensures that τ is enabled as long as φ holds and q does not occur. It is not difficult to see that if p happens, but is not followed by a q, then φ must hold continuously beyond this point, and τ is not taken. However, due to J4, this means that τ is continuously enabled but never taken beyond this point, which violates the requirement of justice with respect to τ.

The second single-step rule relies on a *compassionate* transition $\tau \in \mathcal{C}$.

$$\textbf{C-RESP}\quad \begin{array}{ll} \text{C1.} & p \Rightarrow (q \vee \varphi) \\ \text{C2.} & \{\varphi\}\mathcal{T}\{q \vee \varphi\} \\ \text{C3.} & \{\varphi\}\tau\{q\} \\ \text{C4.} & \varphi \Rightarrow \Diamond(q \vee En(\tau)) \\ \hline & p \Rightarrow \Diamond q \end{array}$$

The difference between this rule and its J-version is in the fourth premise. While J4 requires that φ entails the occurrence of q or the enabling of τ *now*, C4 requires the *eventual* occurrence of q or enabling of τ. Here, an occurrence of p not followed by a q leads, as before, to φ holding continuously, and τ never taken. However, the weaker premise C4 guarantees that τ is enabled infinitely many times, which suffices to violate the requirement of compassion with respect to τ.

In view of the fact that premise C4 appears to be of the same form as the conclusion, i.e., another response formula, one may wonder whether we may not enter a circular loop, trying to prove one response property by another. The answer to this problem is that when we prove premise C4, we actually consider a simpler program, in which τ is never used. This is because the first time τ can be taken, we have already achieved the goal of a state on which τ is enabled.

5.2 Rules for extended response

These rules combine single-step response properties to form general response properties, which need more than a single helpful transition for their achievement.

First, we list two basic rules, which express the monotonicity and transitivity of response properties. They properly belong to the general part of the proof system.

$$\textbf{M-RESP}\quad \begin{array}{c} p \Rightarrow q\ ,\quad r \Rightarrow t \\ q \Rightarrow \Diamond r \\ \hline p \Rightarrow \Diamond t \end{array} \qquad\qquad \textbf{T-RESP}\quad \begin{array}{c} p \Rightarrow \Diamond q \\ q \Rightarrow \Diamond r \\ \hline p \Rightarrow \Diamond r \end{array}$$

The most important rule for establishing extended response properties is based on well-founded induction.

We say that the binary relation $\prec$ over the set $\mathcal{A}$ (often presented as the pair $(\mathcal{A}, \prec)$) is *well-founded*, if there does not exist an infinite sequence $a_0, a_1, \ldots$, where $a_i \in \mathcal{A}$, such that $a_{i+1} \prec a_i$ for all $i = 0, 1, \ldots$.

A binary relation $\preceq$ over the set $\mathcal{A}$ (often presented as the pair $(\mathcal{A}, \preceq)$) is a *preorder* if it is reflexive and transitive. That is, for every $a, b, c \in \mathcal{A}$, $a \preceq a$ and if $a \preceq b$ and $b \preceq c$ then also $a \preceq c$.

If $a \preceq b$ and $b \preceq a$ we say that a and b are *equivalent*, and write $a \approx b$. If $a \preceq b$ but not $b \preceq a$, we denote $a \prec b$. Note that the relation $\prec$ is an ordering; that is, it is irreflexive, antisymmetric, and transitive. We refer to $\prec$ as the ordering *induced by* the preorder $\preceq$.

The preorder $(\mathcal{A}, \preceq)$ is defined to be *well-founded* if the ordering $\prec$ induced by $\preceq$ is well-founded.

For the relations $\preceq$ and $\prec$, we denote by $\succeq$ and $\succ$ the inverse relations

$$a \succeq b \iff b \preceq a \qquad \text{and} \qquad a \succ b \iff b \prec a.$$

Assume a well-founded preorder $(\mathcal{A}, \preceq)$, and a *ranking function* $\delta : \Sigma^+ \mapsto \mathcal{A}$, mapping finite sequences of states into the domain $\mathcal{A}$. The following rule uses well-founded induction to establish an extended response property.

$$\boxed{\begin{array}{lll} \textbf{W-RESP} & \text{W1.} & p \Rightarrow (q \vee \varphi) \\ & \text{W2.} & [\varphi \wedge (\delta = \alpha)] \Rightarrow \Diamond[q \vee (\varphi \wedge (\delta \prec \alpha))] \\ \hline & & p \Rightarrow \Diamond\, q \end{array}}$$

Let σ be a computation. For each position $j \geq 0$, we refer to $\delta(s_0, \ldots, s_j)$ as the *rank* at position j.

Premise W1 ensures that if p holds at some position, then either q or φ hold at that position. Premise W2 guarantees that if φ holds with a certain rank α, then eventually we will reach a position, in which either q holds, or φ is maintained but with a rank lower than α. Since a well-founded ranking cannot go on decreasing forever, we must eventually reach a q-position.

A natural question that may arise is how is premise W2, which is a response formula by its own right, to be established? In many cases this can be done by using any of the two single-step rules for response. This strategy leads to a combined rule which replaces premise W2 by the appropriate premises of **J-RESP** and **C-RESP**.

The resulting **F-RESP** rule uses the set $\mathcal{F} = \mathcal{J} \cup \mathcal{C}$, containing all just and compassionate transitions.

Without loss of generality, we may assume that $\mathcal{J} \cap \mathcal{C} = \phi$. Let $\mathcal{F} = \{\tau_1, \ldots, \tau_m\}$. Rule **F-RESP** assumes intermediate past formulae $\varphi_1, \ldots, \varphi_m$, each corresponding to some $\tau_i \in \mathcal{F}$, a well-founded preorder $(\mathcal{A}, \preceq)$, and a ranking function $\delta : \Sigma^+ \mapsto \mathcal{A}$. We denote by φ the disjunction $\varphi_1 \vee \cdots \vee \varphi_m$.

$$
\begin{array}{l}
\textbf{F-RESP} \\
\left.
\begin{array}{ll}
\text{F1.} & p \Rightarrow (q \vee \varphi) \\
\text{F2.} & \{\varphi_i \wedge (\delta = \alpha)\}\ \mathcal{T}\ \{q \vee \big(\varphi \wedge (\delta \prec \alpha)\big) \vee \big(\varphi_i \wedge (\delta \preceq \alpha)\big)\} \\
\text{F3.} & \{\varphi_i \wedge (\delta = \alpha)\}\ \tau_i\ \{q \vee \big(\varphi \wedge (\delta \prec \alpha)\big)\} \\
\multicolumn{2}{l}{\text{If } \tau_i \in \mathcal{J}, \text{ then}} \\
\text{J4.} & \varphi_i \Rightarrow (q \vee En(\tau_i)) \\
\multicolumn{2}{l}{\text{If } \tau_i \in \mathcal{C}, \text{ then}} \\
\text{C4.} & \mathcal{F} - \{\tau_i\} \vdash \varphi_i \Rightarrow \Diamond(q \vee En(\tau_i))
\end{array}
\right\} \text{for } i = 1, \ldots, m \\
\hline
\qquad p \Rightarrow \Diamond\, q
\end{array}
$$

For each intermediate past formula φ_i, the rule identifies a helpful transition $\tau_i \in \mathcal{F}$, that can be either a just or a compassionate transition. In both cases, it is required (by premise F3) that τ_i leads from each φ_i-position j with rank α to a position $j' = j + 1$, that either satisfies q, or satisfies φ with a rank strictly lower than α. Any transition different from τ_i is required (by premise F2) to lead from each φ_i-position with rank α to a position $j' = j + 1$, that either satisfies q, or satisfies φ with a rank lower than α, or satisfies φ_i with a rank not higher than α.

For the case that τ_i is a just transition, premise J4 requires that each φ_i-position j either satisfies q, or enables τ_i, i.e., τ_i is enabled on s_j. For the case that τ_i is a compassionate transition, premise C4 requires that each φ_i-position with rank α is eventually followed by a position j', that either satisfies q, or satisfies φ with a rank lower than α, or enables τ_i. To avoid circularity, premise C4 is to be proven for a simpler program, in which transition τ_i is removed from the set of fair transitions. This is feasible because when trying to achieve a state in which τ_i is enabled, we cannot be helped by τ_i itself, since its activation from a position j' implies that τ_i is already enabled on j'.

The adequacy of this set of rules for proving response properties is established in Theorem 8.3 presented in Section 8, which states:

> *The rules given above are complete, relative to assertional validity, for proving the $\mathcal{P}$-validity of any response property.*

6 Rules for reactivity

In this section we deal with *reactivity* properties, which are the properties that can be expressed by a conjunction of *simple reactivity formulae*, each having the form

$$\Box \Diamond p \vee \Diamond \Box q,$$

for some past formulae p and q. Since the validity of a conjunction follows from the validity of each of the conjuncts, we will concentrate on proving properties that are expressible by a simple reactivity formula. For brevity, we will refer to such a formula as a reactivity formula, omitting the *simple* prefix.

There are several alternative forms in which every simple reactivity property can be recast; for example

$$\Box \Diamond p \rightarrow \Box \Diamond q, \quad \text{or} \quad \Box \Diamond p \Rightarrow \Diamond q.$$

We prefer to work with an extended form of the last formula,

$$(p \wedge \Box \Diamond r) \Rightarrow \Diamond q.$$

This formula states that any occurrence of p, that is followed by infinitely many occurrences of r, must eventually be followed by an occurrence of q.

6.1 Basic reactivity

In many cases it is possible to base the proof of a reactivity property on some response properties and a well-founded argument. This is given by rule **B-REAC**.

$$\begin{array}{lll}
\textbf{B-REAC} & \text{B1.} & p \Rightarrow (q \vee \varphi) \\
 & \text{B2.} & \{\varphi \wedge (\delta = \alpha)\}\ \mathcal{T}\ \{q \vee \big(\varphi \wedge (\delta \preceq \alpha)\big)\} \\
 & \text{B3.} & [\varphi \wedge (\delta = \alpha) \wedge r] \ \Rightarrow\ \Diamond[q \vee (\delta \prec \alpha)] \\
\hline
 & & (p \wedge \Box \Diamond r) \Rightarrow \Diamond q
\end{array}$$

Premise B1 of the rule ensures that any position that satisfies p, either satisfies q or satisfies φ. Premise B2 ensures that, if j is a φ-position of rank α, then position $j+1$ either satisfies q or satisfies φ with a rank not higher than α. Premise B3 indicates that an additional occurrence of r strengthens the non-increase, guaranteed by B2, into a guaranteed eventual decrease. Thus, if there are infinitely many occurrences of r then, either δ decreases infinitely often, which is impossible due to well-foundedness, or q is eventually realized.

Corollary 8.1, presented in Section 8, precisely identifies the cases under which this rule is guaranteed to be adequate. It claims:

> *For a program with no compassionate transitions, rule* **B-REAC** *is complete, relative to assertional validity, for proving the $\mathcal{P}$-validity of any simple reactivity property.*

However, the rule may apply to many other cases in which the set of compassionate transitions is not necessarily empty.

Obviously, a reactivity property $(p \wedge \Box \Diamond r) \Rightarrow \Diamond q$ can be valid over a program due to the fact that the simpler response property $p \Rightarrow \Diamond q$ is valid. The theorem above depends on a particular mechanism to guarantee that infinitely many occurrences of r cause the eventual occurrence of q. This mechanism is based on a ranking function δ, measuring the distance away from the realization of q, such that each occurrence of an extra r causes an eventual decrease in the rank.

6.2 Reactivity in the general case

When we have compassionate transitions, a well-founded decrease is not the only mechanism by which infinitely many occurrences of r can cause the computation to progress from p to q. Another possible mechanism is based upon a compassionate transition $\tau \in \mathcal{C}$, such that τ leads from p to q, and each occurrence of r causes τ to eventually become enabled (at least once). Consequently, rule **B-REAC** is no longer adequate.

To account for this mechanism, we present first a single-step rule for reactivity under compassion. The rule concerns a *compassionate* transition $\tau \in \mathcal{C}$, and past formulae p, r, q, and φ.

$$
\begin{array}{lll}
\textbf{C-REAC} & \text{C1.} & p \Rightarrow (q \vee \varphi) \\
& \text{C2.} & \{\varphi\} \mathcal{T} \{q \vee \varphi\} \\
& \text{C3.} & \{\varphi\} \tau \{q\} \\
& \text{C4.} & (\varphi \wedge \Box \Diamond r) \Rightarrow \Diamond (q \vee En(\tau)) \\
\hline
& & (p \wedge \Box \Diamond r) \Rightarrow \Diamond q
\end{array}
$$

This rule establishes a single-step reactivity, relying on the compassionate transition $\tau \in \mathcal{C}$. Several single-step reactivity properties can be combined, using monotonicity and transitivity of reactivity formulae. Below we present two rules, properly belonging to the general part, for these two properties.

$$
\begin{array}{ll}
\textbf{M-REAC} & p' \Rightarrow p, \quad r' \Rightarrow r, \quad q \Rightarrow q' \\
& (p \wedge \Box \Diamond r) \Rightarrow \Diamond q \\
\hline
& (p' \wedge \Box \Diamond r') \Rightarrow \Diamond q'
\end{array}
\qquad
\begin{array}{ll}
\textbf{T-REAC} & (p \wedge \Box \Diamond r) \Rightarrow \Diamond q \\
& (q \wedge \Box \Diamond r) \Rightarrow \Diamond t \\
\hline
& (p \wedge \Box \Diamond r) \Rightarrow \Diamond t
\end{array}
$$

Finally, we have a well-founded rule for combining together reactivity properties using well-founded induction.

$$
\begin{array}{lll}
\textbf{W-REAC} & \text{W1.} & p \Rightarrow (q \vee \varphi) \\
& \text{W2.} & [\varphi \wedge (\delta = \alpha) \wedge \Box \Diamond r] \Rightarrow \Diamond [q \vee (\varphi \wedge (\delta \prec \alpha))] \\
\hline
& & (p \wedge \Box \Diamond r) \Rightarrow \Diamond q
\end{array}
$$

As for the case of response, we can obtain a combined rule for reactivity by replacing premise W2 by the appropriate premises of rules **B-REAC** and **C-REAC**, and adding a requirement that r holds infinitely many times.

The resulting **F-REAC** rule uses intermediate past formulae $\varphi_0, \ldots, \varphi_m$, where $\mathcal{F} = \mathcal{J} \cup \mathcal{C} = \{\tau_1, \ldots, \tau_m\}$. Each $\varphi_i, i = 1, \ldots, m$, corresponds to the case that $\tau_i \in \mathcal{F}$ is the helpful transition. The formula φ_0 corresponds to the case that the next occurrence of r brings us closer to the goal q. We denote $\varphi = \varphi_0 \vee \varphi_1 \vee \cdots \vee \varphi_m$. The rule uses also a well-founded preorder $(\mathcal{A}, \preceq)$ and a ranking function $\delta : \Sigma^+ \mapsto \mathcal{A}$.

$$
\begin{array}{ll}
\multicolumn{2}{l}{\textbf{F-REAC}} \\
\text{F1.} & p \Rightarrow (q \vee \varphi) \\
\text{F2.} & \{\varphi_i \wedge (\delta = \alpha)\} \ \mathcal{T} \ \{q \vee \big(\varphi \wedge (\delta \prec \alpha)\big) \vee \big(\varphi_i \wedge (\delta \preceq \alpha)\big)\} \quad \text{for } i = 0, \ldots, m \\
\text{F3.} & \{\varphi_0 \wedge (\delta = \alpha) \wedge r\} \ \mathcal{T} \ \{q \vee \big(\varphi \wedge (\delta \prec \alpha)\big)\} \\
& \left.\begin{array}{ll}
\text{F4.} & \{\varphi_i \wedge (\delta = \alpha)\} \ \tau_i \ \{q \vee \big(\varphi \wedge (\delta \prec \alpha)\big)\} \\
\multicolumn{2}{l}{\text{If } \tau_i \in \mathcal{J}, \text{ then}} \\
\text{J5.} & \varphi_i \Rightarrow (q \vee En(\tau_i)) \\
\multicolumn{2}{l}{\text{If } \tau_i \in \mathcal{C}, \text{ then}} \\
\text{C5.} & \mathcal{F} - \{\tau_i\} \vdash [\varphi_i \wedge \Box \Diamond r] \Rightarrow \Diamond (q \vee En(\tau_i))
\end{array}\right\} \text{for } i = 1, \ldots, m \\
\hline
& (p \wedge \Box \Diamond r) \Rightarrow \Diamond q
\end{array}
$$

The adequacy of these rules is summarized by Theorem 8.4 presented in Section 8.

The rules given above are complete, relative to assertional validity, for proving the $\mathcal{P}$-validity of any simple reactivity property.

7 Completeness for future formulae

This and the next section present the general ideas that lead to the (relative) completeness of the rules presented earlier.

In this section we show relative completeness of the rules for proving $\mathcal{P}$-validity of the future formulae $\Box p$, $p \Rightarrow \Diamond q$, and $(p \wedge \Box \Diamond r) \Rightarrow \Diamond q$. That is, for the restricted case that p, q, and r are state formulae.

These results of future completeness are not completely new and have been shown for similar rules (excluding reactivity) in several places. Completeness for future safety can be obtained by a straightforward extension of the techniques for establishing completeness of proof rules for partial correctness of sequential programs (e.g., [Coo78], [Apt81]). Completeness of rules for the non-safety classes is more involved. [LPS81] and [GFMdR85] established relative completeness of rules for future response, but only on the *semantic* level. That is, they reduced the validity of a future response formula into the validities of state formulae that use certain state predicates. However, they did not identify the precise assertion language in which these predicates can be expressed.

The first completeness proof for future response that identified the necessary assertion language was presented in [SdRG89] (an earlier version is [SdRG84]). Our presentation here uses the same assertion language.

Similar semantic completeness proofs were presented in [AS89] and [MP87].

The next section extends the above results to the case that p, q, and r are arbitrary past formulae.

7.1 Future safety

Since, as a first step, we restrict our attention to future safety formulae, it is sufficient to show that, whenever $\Box q$ is valid over the program P, we can prove this fact using rule INV. Premise I3 is proven by showing that $(\rho_\tau \wedge \varphi) \rightarrow \varphi'$ is a valid state formula for every $\tau \in \mathcal{T}$.

Lemma 7.1 (Completeness for future safety) *Rule* INV *is complete, relative to assertional validity, for proving the validity of safety formulae of the form $\Box q$, where q is a state formula.*

Proof. The basic idea of the proof is the construction of an assertion χ that holds in a state s iff s is *accessible*, i.e., appears in some computation of P. We then show *semantically* that, if $\Box q$ is indeed valid over P, then the premises of rule INV are valid when taking χ for φ. Since we are only proving *relative* completeness, it is enough to show validity of the premises, assuming an oracle that provides proofs or otherwise verifies all generally valid assertions.

We assume that our data domain is expressive enough to encode *records* (i.e., lists) of data elements, and lists of records. In the definition of the assertion, we freely use the auxiliary variable r ranging over records, and a variable λ ranging over lists of records. We are interested in records r of size $|V|$, and often write $r = V$ to denote that the record r contains the list of values of the state variables V. We use the subscripted expression $\lambda[i]$ to refer to the i-th element of λ, and the expression $last(\lambda)$ to refer to the last element of λ. For an assertion $\varphi(V)$, referring to the state variables V, and a record r of size equal to that of V, we denote by $\varphi(r)$ the assertion φ in which the value $r[i]$ is substituted for the state variable $u_i \in V$, for $i = 1, ..., |V|$.

The assertion χ is given by:

$$\chi(V) : \quad \exists\lambda : \Big(((|\lambda| > 0) \wedge \alpha \wedge \beta \wedge \gamma\Big).$$

The body of the assertion χ (to which we refer as $\Psi(V, \lambda)$) consists, in addition to the requirement that λ is nonempty, of three clauses, given by:

$$\alpha : \quad \Theta(\lambda[1])$$

$$\beta : \quad V = last(\lambda)$$

$$\gamma : \quad \forall i (1 \leq i < |\lambda|) : \bigvee_{\tau \in \mathcal{T}} \rho_\tau(\lambda[i], \lambda[i+1]).$$

The assertion χ states the existence of a list of records λ of length $n = |\lambda| > 0$. The list λ encodes a possible history of a computation from some initial state to the current state. Each element $\lambda[i]$, $i = 1, ..., n$, is a record of data elements, representing the values of the state variables V at the i-th state of the computation.

Clause α states that $\lambda[1]$ satisfies Θ, the initial assertion of the program.

Clause β states that the current values of state variables V equal $last(\lambda) = \lambda[n]$, the last record in λ.

Clause γ states that the $(i+1)$-st record of λ, for each $i = 1, ..., n-1$, is a τ-successor of the i-th record, for some transition τ, guaranteeing the correct succession from $\lambda[1]$ to $\lambda[n]$.

We will show now that χ, when substituted for φ, validates the three premises of rule **INV**.

I1. $\Theta \rightarrow \chi$

It is not difficult to see that taking λ to be $(\langle V \rangle)$, i.e., the list consisting of the single record containing the current values of $u_1, ..., u_{|V|}$, the assertion $\Theta(V)$ implies the body $\Psi(V, \lambda)$.

I2. $\chi \rightarrow q$

By our assumption that $\square\, q$ is valid over P, it follows that each accessible state satisfies q. Since χ characterizes precisely the accessible states, the premise follows.

I3. $[\rho_\tau(V, V') \wedge \exists\lambda : \Psi(V, \lambda)] \rightarrow \exists\lambda' : \Psi(V', \lambda')$, for each $\tau \in \mathcal{T}$.

We show that if V, V', and λ satisfy $\rho_\tau(V, V') \wedge \Psi(V, \lambda)$, then there exists a λ' which satisfies $\Psi(V', \lambda')$. An appropriate choice is $\lambda' : \lambda * \langle V' \rangle$, i.e., the list obtained by appending to the end of λ an additional record, consisting of the list of the values of the primed variables V'.

Since we are interested in showing completeness, relative to assertional validity, it is sufficient to show that the premises are assertionally valid, as we have done above. ▪

7.2 Future response

Assume the formula $p \Rightarrow \Diamond q$ to be valid over the program P. We have to show the existence of appropriate assertions $\varphi_1, \ldots, \varphi_m$, a well-founded preorder $(\mathcal{A}, \preceq)$, and a ranking function $\delta : \Sigma \mapsto \mathcal{A}$, such that the premises of rule **F-RESP** are valid. Note that since we are considering the future case only, it is enough to consider a function δ which ranks states (elements of Σ), rather than histories (elements of Σ^+).

As before, let χ be the relation (defined in the proof of Lemma 7.1) characterizing accessibility, and whose invariance over P has been established by the same lemma.

We define a (computation) *segment* to be a finite sequence of states $\sigma : \ s_1, s_2, ..., s_k$, for $k \geq 1$, such that for every $i = 1, ..., k-1$, s_{i+1} is a τ-successor of s_i, for some $\tau \in \mathcal{T}$. We say that the segment σ *departs* from s_1, and that it *leads from* s_1 *to* s_k. We define a segment to be *q-free* if none of the states $s_1, ..., s_k$ satisfies q.

A straightforward modification of the assertion χ leads to an assertion χ_p with free variables V which holds over a state s (i.e. χ_p is true for $s[V]$) iff there exists an accessible p-state $\hat{s}$ and a q-free segment leading from $\hat{s}$ to s. That is,

$$s \models \chi_p \iff \begin{cases} \text{There exists an accessible } p\text{-state } \hat{s} \text{ and a } q\text{-free segment,} \\ \text{leading from } \hat{s} \text{ to } s. \end{cases}$$

We take χ_p for the role of $\varphi = \varphi_1 \vee \cdots \vee \varphi_m$ in rule **F-RESP**.

It is clear that, if the state s satisfies χ_p and some computation contains s at position j, then, due to the assumed validity of $p \Rightarrow \Diamond q$, there must be a later position $r \geq j$ satisfying q.

It is also obvious that χ_p satisfies premise F1 of the rule, i.e., $p \Rightarrow (q \vee \chi_p)$. This is because, if s is an accessible p-state which does not satisfy q, then we can take $\hat{s} = s$ and the singleton segment s, leading from s to itself, as a justification for the claim that s satisfies χ_p.

Let Σ_p denote the set of all states that satisfy χ_p. Let the set of combined fair transitions $\mathcal{F}$ consist of the transitions $\tau_1, ..., \tau_m$, where each τ_i is either a just or a compassionate transition. For a segment $\sigma : s_1, ..., s_k$ and a fair transition $\tau_i \in \mathcal{F}$, we say that τ_i is *gratified* in σ if one of the following holds

- τ_i is taken in σ.
- τ_i is a just transition which is disabled on some state in σ.

For a segment σ, we define $grat(\sigma)$ to be the set of all indices $i = 1, ..., m$ such that $\tau_i \in \mathcal{F}$ is gratified in σ.

We define a binary relation $\sqsubset$ (denoting its inverse relation by $\sqsupset$) on Σ_p by:

$$s \sqsupset \tilde{s} \iff \begin{cases} \text{There exists a } q\text{-free segment } \sigma \text{ leading from } s \text{ to } \tilde{s}, \text{ such that} \\ grat(\sigma) = \{1, \ldots, m\}. \end{cases}$$

We claim that $\sqsubset$ is a well-founded relation over Σ_p. This is because an infinite sequence

$$s^1 \sqsupset s^2 \sqsupset s^3 \ldots,$$

gives rise to a computation

$$\sigma : s^0, \ldots, \hat{s}, \ldots, s^1, \ldots, s^2, \ldots, s^3, \ldots,$$

such that s^0 is initial, $\hat{s}$ satisfies p, and no state beyond $\hat{s}$ satisfies q. The existence of the segment $s^0, \ldots, \hat{s}, \ldots, s^1$ is guaranteed by the fact that s^1 satisfies χ_p. This computation obviously violates our assumption that $p \Rightarrow \Diamond q$ is valid over P. The fact that the sequence above is a computation, in particular that it satisfies all the fairness requirements, hinges on the assumption that the gratifiability set of each segment $s^i, \ldots, s^{i+1}$ is the full set $\{1, \ldots, m\}$.

It is not difficult to see that, based on a construction similar to the ones used for χ and χ_p, there exists a formula in $\mathcal{L}$ which expresses the relation $\sqsupset$.

7.2.1 Construction of a well-founded ranking

The following lemma establishes a connection between an arbitrary well-founded ordering and a well-founded ranking, such as the one required in rule **F-RESP**.

Lemma 7.2 *Let $(S, \sqsubset)$ be a well-founded ordering, with $\sqsupset$ the relation inverse to $\sqsubset$. Then there exists a ranking function $\delta_0 : S \mapsto Ordinals$, mapping each element of S into some ordinal, such that:*

a. $s \sqsupset s' \;\rightarrow\; \delta_0(s) > \delta_0(s')$.

b. If $\delta_0(s) = \beta$ and $\alpha < \beta$, then there exists an element s' such that $\delta_0(s') = \alpha$ and $s \sqsupset s'$.

c. If $s' \sqsupset s'' \;\rightarrow\; s \sqsupset s''$ for every $s'' \in S$, then $\delta_0(s) \geq \delta_0(s')$.

This lemma is proven in [LPS81], using a transfinite construction that assigns ordinal ranks $\delta_0(s)$ to the elements of S. Elements s that are $\sqsubset$-minimal. i.e., there exists no s' such that $s \sqsupset s'$, are assigned the ordinal $\delta_0(s) = 0$. All other elements s are assigned a rank which is the least upper bound of $\delta_0(s') + 1$, for all $s' \sqsubset s$.

According to the above lemma, there exists a ranking function $\delta_0 : \Sigma_p \mapsto Ordinals$, mapping states in Σ_p into the ordinals.

Let s be a φ-state and s' a successor of s. If s' does not satisfy q, then s' is also a φ-state. In this case we show that $\delta_0(s) \geq \delta_0(s')$. This inequality is ensured by clause (c) of Lemma 7.2, provided we show that for every s'', $s' \sqsupset s''$ implies $s \sqsupset s''$.

Indeed, let s'' be a state such that $s' \sqsupset s''$. By the definition there exists a segment $\sigma' : s', \ldots, s''$ leading from s' to s'', such that $grat(\sigma') = \{1, ..., m\}$. It is obvious that the segment $\sigma : s, s', \ldots, s''$, formed by appending s to the beginning of σ', leads from s to s'', and that $grat(\sigma) = \{1, ..., m\}$. This establishes $s \sqsupset s''$.

The ranking δ_0 is not fine enough to identify a helpful transition whose activation guarantees a measurable progress, i.e., leads from s to s' such that $\delta_0(s) > \delta_0(s')$. It is therefore necessary to refine it.

For a segment σ, we define the *deficit* of σ, denoted by $\Delta(\sigma)$, to be the smallest positive integer i, such that τ_i is not gratified in σ. In the case that $grat(\sigma) = \{1, ..., m\}$, $\Delta(\sigma)$ is defined to be $m + 1$. We define a q-free segment $\sigma : s_1, \ldots, s_k$ to be *leveled* if $\delta_0(s_1) = \ldots = \delta_0(s_k)$.

For every χ_p-state s, we define the *height* of s, denoted by $h(s)$ by

$$h(s) \;=\; max\{\Delta(\sigma) \mid \sigma \text{ is a leveled segment departing from } s \}.$$

The height h is expected to measure progress within segments on which δ_0 does not decrease. Clearly, any gratification of an additional transition of $\mathcal{F}$ brings us closer to the end of such a segment. This is because if all transitions of $\mathcal{F}$ have been gratified in σ, then σ cannot be leveled and its last state must have a smaller δ_0-rank than its first state.

The complete ranking function, to be used in the rule, is formed by the lexicographical pairing $\widetilde{\delta}(s) = (\delta_0(s), h(s))$. The range of the function $\widetilde{\delta}$ is defined to be $\widetilde{\mathcal{A}}$, the set of all pairs of the form (α_0, i), where α_0 is an ordinal and $i \leq m+1$.

The preorder $\widetilde{\preceq}$ over $\widetilde{\mathcal{A}}$ is defined by

$$(\alpha_0, i) \widetilde{\succeq} (\alpha_0', i') \quad \Longleftrightarrow \quad (\alpha_0 > \alpha_0') \vee \big((\alpha_0 = \alpha_0') \wedge (i \geq i')\big).$$

Clearly, this preorder is well-founded.

There are several properties these ranking functions satisfy.

P1. For every χ_p-state s, $h(s) \leq m$.
Let σ be a leveled segment leading from s to some s'. If $grat(\sigma)$ equals $\{1, ..., m\}$, then $s \sqsupset s'$ holds, which leads to $\delta_0(s) > \delta_0(s')$, contradicting the fact that σ is leveled. It follows that at least some τ_i is not gratified in σ, and therefore $h(s) \leq m$.

P2. For every χ_p-state s and its successor s', either s' satisfies q, or $\widetilde{\delta}(s) \widetilde{\succeq} \widetilde{\delta}(s')$.
Assume that s' does not satisfy q. We have already shown that $\delta_0(s) \geq \delta_0(s')$. If $\delta_0(s) > \delta_0(s')$, then clearly $\widetilde{\delta}(s) \widetilde{\succeq} \widetilde{\delta}(s')$. In the other case, i.e. $\delta_0(s) = \delta_0(s')$, let $h(s')$ be $i \leq m$. By the definition of h, there exists a leveled segment $\sigma' : s', \ldots, s''$, such that i is the smallest index of a fair transition τ_i, which is not gratified in σ'. Consider the augmented segment $\sigma : s, s', \ldots, s''$. Clearly, σ is leveled and any τ_i gratified in σ' is also gratified in σ. It follows that the deficit of σ, $\Delta(\sigma) \geq \Delta(\sigma') = i$. Since σ is only one of the leveled segments departing from s, and $h(s)$ is defined to be the maximum of the deficits of all such segments, it follows that $h(s) \geq i$. Thus, also in this case $\widetilde{\delta}(s) \widetilde{\succeq} \widetilde{\delta}(s')$.

P3. Let s be a χ_p-state, such that $h(s) = i$. Let s' be a τ_i-successor of s. Then, either s' satisfies q, or $\widetilde{\delta}(s) \widetilde{\succ} \widetilde{\delta}(s')$.
It is sufficient to consider the case that s' does not satisfy q and that $\delta_0(s) = \delta_0(s')$, and to show that $h(s) > h(s')$. Let $h(s') = i$ and $\sigma' : s', \ldots, s''$ be the segment realizing the deficit i for s'. Clearly, the augmented segment $\sigma : s, s', \ldots, s''$ gratifies all the transitions gratified by σ', and in addition also gratifies τ_i. It follows that $\Delta(\sigma) > i$, and therefore also $h(s) > i = h(s')$.

For $i = 1, \ldots, m$, we define φ_i to be $\chi_p \wedge (h = i)$. Clearly, $\chi_p \equiv \varphi_1 \vee \cdots \vee \varphi_m$.

We proceed to show that all the premises of rule **F-RESP** are satisfied by taking $\varphi, \varphi_1, \ldots, \varphi_m$ and $(\widetilde{\mathcal{A}}, \widetilde{\preceq}), \widetilde{\delta}$ for $(\mathcal{A}, \preceq), \delta$, respectively . We have already shown that F1 is valid.

F2. $\{\varphi_i \wedge (\widetilde{\delta} = \alpha)\}\ \mathcal{T}\ \{q \vee \big(\chi_p \wedge (\widetilde{\delta} \mathrel{\widetilde{\prec}} \alpha)\big) \vee \big(\varphi_i \wedge (\widetilde{\delta} \mathrel{\widetilde{\preceq}} \alpha)\big)\}$

It is straightforward to show that if s' is a successor of a φ_i-state s, then either s' satisfies q or it is also a χ_p-state, which by property P2 above satisfies $\widetilde{\delta}(s) \widetilde{\succeq} \widetilde{\delta}(s')$. If $\widetilde{\delta}(s) > \widetilde{\delta}(s')$ then $\chi_p \wedge (\widetilde{\delta} \mathrel{\widetilde{\prec}} \alpha)$ holds at s'. Otherwise, $h(s) = h(s') = i$ and s' satisfies $\varphi_i \wedge (\widetilde{\delta} \mathrel{\widetilde{\preceq}} \alpha)$.

F3. $\{\varphi_i \wedge (\widetilde{\delta} = \alpha)\}\ \tau_i\ \{q \vee \left(\chi_p \wedge (\widetilde{\delta} \mathrel{\widetilde{\prec}} \alpha)\right)\}$
Let s be a χ_p-state, such that $h(s) = i$, and s' a τ_i-successor of s. If s' does not satisfy q, then it clearly satisfies χ_p, and by the property P3 stated above, also satisfies $\widetilde{\delta}\,(s) \mathrel{\widetilde{\succ}} \widetilde{\delta}\,(s')$, i.e. s' satisfies $\alpha \mathrel{\widetilde{\succ}} \widetilde{\delta}$.

For the case that τ_i is a just transition, we proceed to show
J4. $\varphi_i \Rightarrow (q \vee En(\tau_i))$

Let s be a φ_i-state, not satisfying q. By the definition of φ_i, $h(s) = i$. Let $\sigma : s, \ldots, s''$ be the segment realizing the deficit i. If τ_i were disabled on s then, according to the definition, τ_i would have been gratified in σ. We conclude that τ_i must be enabled on s.

For the case that τ_i is a compassionate transition, we proceed to show
C4. $\mathcal{F} - \{\tau_i\} \ \vdash\ \varphi_i \Rightarrow \Diamond[q \vee En(\tau_i)]$

Let P' denote the program which is identical to P in all components, except that the compassionate transition τ_i has been removed from its combined fairness set $\mathcal{F}$. We proceed to show that $P' \models \psi$, where ψ is the formula whose validity is claimed to be provable in C4. Assume to the contrary, that ψ is not valid over P'. In that case there must exist σ, a computation of P', containing at some position j a χ_p-state s with $h(s) = i$, such that no position beyond j satisfies $q \vee En(\tau_i)$. Being a computation of P' means that it satisfies all the fairness requirements posed by P, except possibly the one of compassion with respect to τ_i. However, since $En(\tau_i)$ is one of the disjuncts excluded beyond position j, it follows that τ_i is enabled only finitely many times on σ, which implies that σ is also compassionate with respect to τ_i, and is therefore also a computation of P. This violates our original assumption that $p \Rightarrow \Diamond\, q$ is valid over P.

If we base our completeness proof on induction on the size of $\mathcal{F}$, the combined fairness set, we have just reduced the completeness problem of response properties for a program with $|\mathcal{F}| = m$, to that of program with $|\mathcal{F}| = m - 1$. By such an induction, since we have just shown that $P' \models \psi$, it follows that $P' \vdash \psi$, as is required by C4.

The base case of the induction is that of set $\mathcal{F}$ containing only just transitions. For this case, premise C4 is trivially valid since there are no compassionate transitions.

7.2.2 A well-founded ranking expressible in $\mathcal{L}$

The previous construction of the assertions $\varphi_1, \ldots, \varphi_m$, the well-founded preorder $(\widetilde{\mathcal{A}}, \mathrel{\widetilde{\preceq}})$ and the ranking function $\widetilde{\delta}$ is satisfactory in all respects, except for the fact that we do not know how to directly express $(\widetilde{\mathcal{A}}, \mathrel{\widetilde{\preceq}})$ and $\widetilde{\delta}$ in the assertional language $\mathcal{L}$. Note that $\varphi_1, \ldots, \varphi_m$ have been shown to be expressible in $\mathcal{L}$. Consequently, we replace these constructs by a new well-founded preorder $(\mathcal{A}, \preceq)$, and a new ranking function δ that are expressible in $\mathcal{L}$. As we will see, the expressible constructs are closely related to the old ones.

Consider the fixpoint equation

$$D(s_1, s_2) \ \equiv\ \chi_p(s_1) \ \wedge\ \chi_p(s_2) \ \wedge\ \forall s_2' : \left((s_2 \sqsupset s_2') \rightarrow \exists s_1' : (D(s_1', s_2') \wedge (s_1 \sqsupset s_1'))\right) \qquad (3)$$

Denote the right-hand side of this equation by $\Phi(D, s_1, s_2)$. Let $\hat{D}(s_1, s_2)$ denote $\nu D : \Phi(D, s_1, s_2)$ the maximal solution of (3). We will also write $s_1 \trianglerighteq s_2$ or $s_2 \trianglelefteq s_1$ for $\hat{D}(s_1, s_2)$. It is not difficult to see that $\trianglelefteq$ is a preorder.

The intended meaning of the preorder $\trianglerighteq$ is that it captures the preorder between states s_1 and s_2 such that $\delta_0(s_1) \geq \delta_0(s_2)$. Since the ranking $\delta_0(s)$ can be interpreted as measuring the length of chains $s = s_1 \sqsupset s_2 \sqsupset \cdots$, equation (3) claims that $\delta_0(s_1) \geq \delta_0(s_2)$ iff for every $s_2' \sqsubset s_2$ there exists an $s_1' \sqsubset s_1$ such that $\delta_0(s_1') \geq \delta_0(s_2')$. This indicates that for any chain starting at s_2 there exists a chain starting at s_1 which is at least as long as the s_2 chain.

Let $\lhd$ be the ordering induced by $\unlhd$, and $\approx$ be the equivalence relation defined by

$$s_1 \approx s_2 \iff s_1 \unlhd s_2 \text{ and } s_1 \unrhd s_2$$

This equivalence is related to the notion of *bisimulation* [Mil80].

Clearly, $\unlhd$, $\lhd$, and $\approx$ are all expressible in $\mathcal{L}$.

The following lemma relates the equivalence $\approx$ and the ordering $\lhd$ to the ranking function $\widetilde{\delta}$ and ordinal ordering.

Lemma 7.3 *For every state $s, s' \in \Sigma_p$*

a. $\delta_0(s) = \delta_0(s')$ *iff* $s \approx s'$.

b. $\delta_0(s) > \delta_0(s')$ *iff* $s \rhd s'$.

The lemma can be proven by transfinite induction on the ranking δ_0.

Following are some important properties of $\unlhd$ and $\lhd$, which follow from the above lemma.

P4. The ordering $\lhd$ (and therefore also the preorder $\unlhd$) is well-founded. This follows from the fact that the ordinal ordering $<$ is well-founded.

P5. If s and s' are χ_p-states such that $s \sqsupset s'$, then $s \rhd s'$. Consider $s \sqsupset s'$. By clause (a) of Lemma 7.2, $\delta_0(s) > \delta_0(s')$. By clause (b) of Lemma 7.3, it follows that $s \rhd s'$.

P6. If s and s' are χ_p-states, such that s' is a successor of s, then $s \unrhd s'$. This follows from the fact that, for every s'', if $s' \sqsupset s''$ then also $s \sqsupset s''$.

As the relation $\approx$ can be expressed in $\mathcal{L}$, we can capture in $\mathcal{L}$ the notion of a leveled segment. This is a segment $\sigma : s_1, \ldots, s_k$ such that $s_1 \approx \cdots \approx s_k$. Consequently, we can express in $\mathcal{L}$ the fact $h(s) = i$, for $i = 1, \ldots, m+1$.

We take

- $\mathcal{A} : \Sigma_p \cup \{\bot\}$, where $\bot$ is an element different from all other members of Σ_p.
- $\delta(s) :$ **if** $\chi_p(s)$ **then** s **else** $\bot$.
- The preorder $\preceq$ over $\mathcal{A}$ is defined to satisfy $\bot \preceq \bot$ and

  $$s \succeq s' \iff (s \rhd s') \vee \Big((s \unrhd s') \wedge (h(s) \geq h(s'))\Big),$$

  for $s, s' \in \Sigma_p$, where $\rhd$ is the inverse of $\lhd$. Clearly, this preorder is also well-founded.

It is obvious that for every two states $s, s' \in \Sigma_p$

$$\widetilde{\delta}(s) \mathrel{\widetilde{\preceq}} \widetilde{\delta}(s') \iff \delta(s) \preceq \delta(s') \tag{4}$$

This shows that the new constructs capture precisely the same relations as the previous constructs and are, in addition, expressible in $\mathcal{L}$.

In particular, we have the property

P7. If s and s' are χ_p-states, such that s' is a successor of s, then $s \succeq s'$, which can also be written as $\delta(s) \succeq \delta(s')$. If s is a φ_i-state, and s' is its τ_i-successor, then $s \succ s'$ (equivalently $\delta(s) \succ \delta(s')$).

The correspondence between the new and old constructs ensures that the new constructs also satisfy the premises of rule F-RESP.

This analysis may be summarized by the following lemma.

Lemma 7.4 (Completeness for future response) *Rule* F-RESP *is complete, relative to assertional validity, for proving the validity of response formulae of the form $p \Rightarrow \Diamond q$, where p and q are state formulae.*

7.3 Future reactivity

As the last case of future completeness, we consider proving the completeness of the presented proof system for proving formulae of the form $(p \wedge \Box \Diamond r) \Rightarrow \Diamond q$, where p, q, and r are state formulae. A helpful intuition, which will guide us in the proof, is that such a formula is valid over P iff the response formula $p \Rightarrow \Diamond q$ is valid over a program P^+ which differs from P by having an additional justice requirement, which demands that every computation contains infinitely many r-states.

With this understanding, we proceed in a route very similar to that of establishing completeness for response properties.

Lemma 7.5 (Completeness for future reactivity) *Rule* F-REAC *is complete, relative to assertional validity, for proving the validity of reactivity formulae of the form $(p \wedge \Box \Diamond r) \Rightarrow \Diamond q$, where p, q, and r are state formulae.*

Proof. As in the case of rule F-RESP, we assume the formula $(p \wedge \Box \Diamond r) \Rightarrow \Diamond q$ to be valid over the program P and show how to construct assertions $\varphi_0, \ldots, \varphi_m$, a well-founded preorder $(\mathcal{A}, \preceq)$, and a ranking function δ which satisfy the premises of rule F-RESP.

We adopt the definitions of φ, and q-free segments, from Lemma 7.4. We slightly modify the definition of gratification in a segment to read as follows:

For a segment $\sigma : s_1, \ldots, s_k$ and an index $i \in \{0, \ldots, m\}$, we say that i is *gratified* in σ if one of the following holds:

- $i > 0$ and τ_i is taken in σ.
- $i > 0$, τ_i is a just transition, and τ_i is disabled on some state in σ.
- $i = 0$ and some state in σ satisfies r.

Thus, we associate the gratification of the index 0 with the satisfaction of r. We define the set $grat(\sigma)$, for a segment σ, as before, except that its range may now be any subset of $\{0, 1, \ldots, m\}$. Similarly, we define the relation $\sqsupset$ to hold between two states, s and s', if there exists a q-free segment σ, connecting them, such that $grat(\sigma) = \{0, 1, \ldots, m\}$. The relation $\sqsubset$ (inverse of $\sqsupset$) is well-founded, because an infinite sequence of $\sqsupset$-related φ-states gives rise to a computation violating $(p \wedge \Box \Diamond r) \Rightarrow \Diamond q$. Consequently, we follow the construction of Lemma 7.4 in defining the preorder $\trianglelefteq$. The definition of the *deficit*

$\Delta(\sigma)$ of a segment σ is precisely the same as the corresponding definition in the preceding lemma, except that it now ranges over $\{0, 1, \ldots, m\}$. This leads to the height h, and to the definition of the preorder $\preceq$.

It is straightforward to verify that properties P1–P7 are still valid for $h(s) = i > 0$. A special consequence of the definitions above is that if s is a χ_p-state, which satisfies r, then $h(s) > 0$.

Take

- $\varphi_i : \chi_p \wedge (h = i)$, for $i = 0, \ldots, m$.
- $\mathcal{A} : \Sigma_p \cup \{\perp\}$ and $\prec$ as defined above.
- $\delta(s)$: **if** $\chi_p(s)$ **then** s **else** $\perp$

We now turn to establish the validity of the premises of rule F-REAC. Premises F1 and F2 follow from arguments similar to the ones presented in the case of the response rule.

Premise F3 holds trivially since, by the observation above, any state satisfying r has $h > 0$ and cannot be a φ_0-state at the same time.

Premises F4, J5, and C5 are justified by arguments similar to those of the response case. Considering C5, the inductive argument has to consider a similar reactivity property for a simpler program that has one less compassionate transition. ∎

Using the constructions employed in the proof of this theorem, it is possible to derive the following corollary.

Corollary 7.1 (Completeness of future reactivity under justice) *For a program with no compassionate transitions, rule* B-REAC *is complete, relative to assertional validity, for proving the validity of reactivity formulae of the form $(p \wedge \Box \Diamond r) \Rightarrow \Diamond q$, where p, q, and r are state formulae.*

Proof. Assume the formula $p \Rightarrow \Diamond q$ to be valid over the program P, which has no compassionate transitions. We adopt the definitions of the assertion $\varphi : \chi_p$, the domain $\mathcal{A} : \Sigma_p \cup \{\perp\}$, and the function δ from the previous lemma. As the well-founded preorder, we take $(\mathcal{A}, \trianglelefteq)$, It is not difficult to see that this choice of constructs satisfies premises B1 and B2 of the rule. In particular, property P6 guarantees, as required in B2, that a successor s' of a χ_p-state satisfies either q or $s \trianglerighteq s'$.

Consider premise B3. Assume a computation, in which the state s at position j satisfies $\chi_p \wedge r$. It is not difficult to see that there must be another state $\tilde{s}$, at position $k \geq j$, such that either $\tilde{s}$ satisfies q, or the segment $s, \ldots, \tilde{s}$ is q-free and gratifies all the indices in $\{0, 1, \ldots, m\}$. In the later case $s \sqsupset \tilde{s}$ (since s satisfies φ) and, according to property P2, this implies that $s \triangleright \tilde{s}$. This establishes premise B3. ∎

8 Completeness — the general case

This section extends the completeness results obtained in the previous section to the case that the subformulae p, q and r, are arbitrary past formulae.

The general strategy for this extension is that of reduction. We first show how to reduce a formula ψ with past subformulae, which is valid over a program P, into a formula $\hat{\psi}$ of the same class, having only state subformulae, which is valid over a program $\hat{P}$. Program $\hat{P}$ is obtained by augmenting P with additional boolean variables and extending its transition relations by clauses referring to these new variables. We then use the completeness results for state subformulae to construct a (relative) proof of the $\mathcal{P}$-validity of $\hat{\psi}$ over $\hat{P}$. We then show in three separate theorems how to transform state proofs that use rules **INV**, **F-RESP**, and **F-REAC** into past proofs by rules **SAFE**, **F-RESP**, and **F-REAC**, respectively. This will establish completeness for the general case.

8.1 Elimination of past subformulae

A temporal formula is called *stratified* if it contains no future operator within the scope of a past operator. Obviously, all formulae in canonical form are stratified, because they never apply past operators to strict-future formulae.

Let us fix our attention on a program P and a stratified formula ψ which is valid over P.

Define Φ to be the set of subformulae of ψ (possibly including ψ) whose principal operator is a past operator, i.e., $\ominus$ or $\mathcal{S}$. We define a set of new boolean variables $\mathcal{B}$ consisting of a variable b_p for each formula $p \in \Phi$. We intend to use the variable b_p to encode p, i.e., as a variable that will be true at a position in a computation iff the formula p is true there.

Let q be a subformula of ψ, and p a subformula of q. We define p to be Φ-*maximal* in q if

- $p \in \Phi$ and
- there is no r, another subformula of q, such that $r \in \Phi$ and p is a proper subformula of r, i.e., strictly contained in r.

Let $p_1, \ldots, p_n$ be all the Φ-maximal subformulae of q. We define the *statification* (i.e., encoding of past formulae as state formulae) of q, denoted by $stat(q)$ (or $\hat{q}$), to be

$$stat(q) : \; q[b_{p_1}/p_1, \ldots, b_{p_n}/p_n].$$

That is, $stat(q)$ is obtained from q by replacing all occurrences of the subformula p_i by the variable b_{p_i}, for $i = 1, \ldots, n$. It is not difficult to see that, in the special case that q is a past formula, $stat(q)$ is a state formula.

Replacing past formulae by boolean variables is obviously not enough, unless we can guarantee that in all positions of the computation the variable b_p assumes the same truth value as p. To achieve this we modify the program P, given by the system $\langle V, \Sigma, \mathcal{T}, \Theta, \mathcal{J}, \mathcal{C} \rangle$, to obtain its *statified* version $\hat{P}$, given by $\langle \hat{V}, \hat{\Sigma}, \hat{\mathcal{T}}, \hat{\Theta}, \hat{\mathcal{J}}, \hat{\mathcal{C}} \rangle$, where we define:

- $\hat{V} = V \cup \mathcal{B}$. That is, we augment V by the boolean variables $\mathcal{B}$.
- $\hat{\Sigma}$ – The set of interpretations over $\hat{V}$. Variables in $\mathcal{B}$ should be assigned boolean values.

- $\hat{\mathcal{T}}$ – Corresponding to each $\tau \in \mathcal{T}$, we place in $\hat{\mathcal{T}}$ a transition $\hat{\tau}$, whose transition relation is given by $\hat{\rho_\tau} = \rho_\tau \wedge N$. The assertion $N(\hat{V}, \hat{V}')$ controls the evolution of the variables in $\mathcal{B}$ between each state and its successor, and ensures that it corresponds to the evolution of the past formulae they stand for. The assertion N is a conjunction containing a conjunct $C(p)$ for each $p \in \Phi$. These conjuncts are given by:

  - $C(\ominus p) \ : \ b'_{\ominus p} \equiv stat(p)$.
    This conjunct guarantees that the boolean value of $b_{\ominus p}$ in the next state equals the truth-value of $stat(p)$ in the current state.
  - $C(p \mathcal{S} q) \ : \ b'_{p \mathcal{S} q} \equiv [(stat(q))' \vee (b_{p \mathcal{S} q} \wedge (stat(p))')]$.
    This conjunct guarantees that $b_{p \mathcal{S} q}$ is true in the next state iff either $stat(q)$ holds there, or $stat(p)$ holds there and $b_{p \mathcal{S} q}$ holds now.

  It is not difficult to see that N can be presented by a conjunction of the form

  $$(b'_{p_1} \equiv \pi_1) \wedge \cdots \wedge (b'_{p_n} \equiv \pi_n),$$

  where $\pi_1, \ldots, \pi_n$ are assertions that may refer to $V \cup V' \cup B$ but not to any primed variables in B'.

- $\hat{\Theta} \ : \ \Theta \wedge Init$. The assertion $Init$ ensures that the initial value of each variable $b_p \in \mathcal{B}$ matches the initial value of the past formula p. The assertion $Init$ contains a conjunct $I(p)$ for each $p \in \Phi$, given by:

  - $I(\ominus p) \ : \ \neg b_{\ominus p}$.
    This conjunct states that all *previous* formulae are initially false.
  - $I(p \mathcal{S} q) \ : \ b_{p \mathcal{S} q} \equiv stat(q)$.
    This conjunct states that the only way for $p \mathcal{S} q$ to hold at the first state in a computation is for $stat(q)$ to hold there.

  It is not difficult to see that $Init$ determines unique initial values to the variables $b_{p_1}, \ldots, b_{p_n}$, which may depend on the initial values of the variables V. It can therefore be presented as a conjunction

  $$(b_{p_1} \equiv \beta_1) \wedge \cdots \wedge (b_{p_n} \equiv \beta_n)$$

  where $\beta_1, \ldots, \beta_n$ are assertions that refer only to the variables in V.

The structure of the fairness sets $\hat{\mathcal{J}}$ and $\hat{\mathcal{C}}$ is identical to that of $\mathcal{J}$ and $\mathcal{C}$, except for the trivial renaming of each τ to $\hat{\tau}$.

Example 8.1 Consider the simple program, presented in Example 4.1 above, which was given by $V = \{x\}$, $\mathcal{T} = \{\tau\}$, where $\rho_\tau : x' = x + 1$, and $\Theta : x = 0$. The formula considered there is

$$\psi \ : \ \Box((x = 10) \rightarrow \diamondminus(x = 5)).$$

Clearly, for this case $\Phi = \{\diamondminus(x = 5)\}$, yielding a single boolean variable b, corresponding to the past formula $\diamondminus(x = 5)$, which is an abbreviation for $\mathbf{T}\mathcal{S}(x = 5)$. Consequently, we have $stat(\psi) : \square((x = 10) \rightarrow b)$, and the statified program $\hat{P}$ is given by:

- $\hat{V} = \{x, b\}$.
- $\hat{\mathcal{T}} = \{\hat{\tau}\}$, where (following some simplifications) $\hat{\rho}_\tau : (x' = x + 1) \wedge (b' \equiv [(x' = 5) \vee b])$.
- $\hat{\Theta} : (x = 0) \wedge (b \equiv (x = 5))$, which is equivalent to $(x = 0) \wedge (b \equiv \mathbf{F})$. ▪

Theorem 8.1 (Past elimination) *The formula ψ is valid over P iff $\hat{\psi} = stat(\psi)$ is valid over $\hat{P}$.*

Proof. The theorem follows from the fact that there is a one-to-one correspondence between computations of P and computations of $\hat{P}$, such that for every σ, a computation of P, and $\hat{\sigma}$, the corresponding computation of $\hat{P}$, position j, and formula $p \in \Phi$:

$$(\sigma, j) \models p \quad \Longleftrightarrow \quad (\hat{\sigma}, j) \models (b_p = \mathbf{T}).$$

This fact can be proved by induction on $j = 0, 1, \ldots$ and structural induction on $p \in \Phi$. ▪

Assume a formula ψ belonging to one of the three classes considered here, which is valid over a program P. We use the construction of Theorem 8.1 to reduce it to a future formula $\hat{\psi}$ valid over the program $\hat{P}$.

As shown by the future completeness theorems, each future safety, response, or reactivity formula, can be proven by a single application of rules **INV**, **F-RESP**, and **F-REAC**, respectively. These proofs can be presented as a sequence of lines, each corresponding to one of the premises of the relevant rule, followed by the last line which states the conclusion $\hat{\psi}$.

Assume that we have such a proof for the case that $\hat{\psi}$ is a future formula. As shown by the future completeness results, we may assume that each of the premise lines is a valid state formula. We will show how to transform each of these lines into a proof of a corresponding past formula that establishes one of the premises for a rule that proves the conclusion ψ.

Let $\hat{q}$ be a state formula, such that all occurrences of variables of B are not under the scope of any quantification or any fixpoint operator. We define the transformation inverse to $stat$ by

$$stat^{-1}(\hat{q}) \; : \; \hat{q}[p_1/b_{p_1}, \ldots, p_n/b_{p_n}].$$

To cover a more general case, it is sufficient to require that $\hat{q}$ does not quantify over B variables. This is because, if all occurrences of b in $\hat{q}(b)$ are free, we can rewrite $\hat{q}(b)$ as

$$(b \wedge \hat{q}(\mathbf{T})) \vee (\neg b \wedge \hat{q}(\mathbf{F})).$$

A similar transformation can be applied in the case that B contains several variables.

We will consider in turn each of three rules and show how to transform a future proof according to this rule into a past proof.

8.2 Transforming an INV proof

Assume that the formula $\psi : \Box\, q$, where q is some past formula is valid over the program P. Using the statification transformation described above, we construct the program $\hat{P}$ and the state formula $\hat{\psi} = \Box\, \hat{q}$ which is valid over $\hat{P}$. By the future completeness result, there exists an assertion $\hat{\varphi}$ such that the premises of rule **INV** are valid. We consider each premise and show how to transform it into a proof of a corresponding premise of rule **SAFE**.

Example 8.2 Continuing our running example, the safety property to be proven is ψ : $\Box((x = 10) \rightarrow \diamondminus(x = 5))$. The statification transformation led to $\hat{\psi} : \Box((x = 10) \rightarrow b)$. To prove its validity over $\hat{P}$ we may use the assertion $\hat{\varphi} : (x \geq 5) \rightarrow b$. ▟

I1. $\hat{\Theta} \rightarrow \hat{\varphi}$

According to the definition of $\hat{\Theta}$, this is equivalent to

$$(\Theta \wedge Init) \rightarrow \hat{\varphi}$$

Recalling that $Init$ can be represented as $(b_{p_1} \equiv \beta_1) \wedge \cdots \wedge (b_{p_n} \equiv \beta_n)$, we construct the following proof:

1. $\Theta \wedge \Big((b_{p_1} \equiv \beta_1) \wedge \cdots \wedge (b_{p_n} \equiv \beta_n)\Big) \rightarrow \hat{\varphi}$
   By assumption, this is a valid state formula.

2. $\Theta \wedge \Big((p_1 \equiv \beta_1) \wedge \cdots \wedge (p_n \equiv \beta_n)\Big) \Rightarrow \varphi$
   by rule **INST**, instantiating $b_{p_1}, \ldots, b_{p_n}$ into $p_1, \ldots, p_n$. The formula φ stands for $stat^{-1}(\hat{\varphi})$.

3. $\mathbf{first} \Rightarrow \Big((p_1 \equiv \beta_1) \wedge \cdots \wedge (p_n \equiv \beta_n)\Big)$
   Since the assertions $\beta_1, \ldots, \beta_n$ are derived from the clauses $I(p_i)$, which are identical to the relations between the value of a past formula and its past subformulae at the first position in a model, the above is a general temporal tautology. It can be proven by any complete proof system for propositional temporal logic (e.g. [MP89]).

4. $(\Theta \wedge \mathbf{first}) \Rightarrow \varphi$
   This follows by propositional temporal reasoning from lines 2 and 3.

This establishes premise S1 of rule **SAFE**.

Example 8.3 The first premise in the proof of $\hat{\psi} : (x = 10) \rightarrow b$ over $\hat{P}$ is the valid state formula

$$\Big((x = 0) \wedge (b \equiv (x = 5))\Big) \rightarrow \Big((x \geq 5) \rightarrow b\Big).$$

The above procedure uses it to derive a proof for the valid entailment

$$\Big((x = 0) \wedge \mathbf{first}\Big) \Rightarrow \Big((x \geq 5) \rightarrow \diamondminus(x = 5)\Big).$$ ▟

I2. $\hat{\varphi} \to \hat{q}$

Applying the instantiation $stat^{-1}$ to this valid state formula, we obtain

$$\varphi \Rightarrow q,$$

which is premise S2 of rule **safe**.

Example 8.4 Premise I2 of our running example is

$$\big((x \geq 5) \to b\big) \to \big((x = 10) \to b\big),$$

which, by instantiation, leads to

$$\big((x \geq 5) \to \diamondsuit(x = 5)\big) \Rightarrow \big((x = 10) \to \diamondsuit(x = 5)\big). \blacksquare$$

I3. $(\rho_\tau \wedge N \wedge \hat{\varphi}) \to \hat{\varphi}'$ for each $\tau \in \mathcal{T}$

In this form of I3, we have expanded the verification condition for the full $\mathcal{T}$ into a premise for each $\tau \in \mathcal{T}$, and replaced $\hat{\rho_\tau}$ by its definition $\rho_\tau \wedge N$.

Recall that N may be presented in the form $(b'_{p_1} \equiv \pi_1) \wedge \cdots \wedge (b'_{p_n} \equiv \pi_n)$, where $\pi_1, \ldots, \pi_n$ do not refer to variables in B'. Consequently, the formula above can be rewritten as

$$(\rho_\tau \wedge N \wedge \hat{\varphi}) \to \hat{\varphi}'[\pi_1/b'_{p_1}, \ldots, \pi_n/b'_{p_n}].$$

The formula on the right is obtained by replacing each b'_{p_i} by π_i, for $i = 1, \ldots, n$.

Because of the special form of N and the fact that $b'_{p_1}, \ldots, b'_{p_n}$ do not occur in either ρ_τ or $\hat{\varphi}'[\pi_1/b'_{p_1}, \ldots, \pi_n/b'_{p_n}]$, the above formula is (assertionally) valid iff

$$(\rho_\tau \wedge \hat{\varphi}) \to \hat{\varphi}'[\pi_1/b'_{p_1}, \ldots, \pi_n/b'_{p_n}]$$

is.

We apply the instantiation $stat^{-1}$ to this state formula. Since ρ_τ does not refer to B variables, $stat^{-1}(\rho_\tau) = \rho_\tau$. Let us denote $stat^{-1}(\hat{\varphi})$ by φ. Clearly, $stat^{-1}(\hat{\varphi})$ yields φ. Observe that $\hat{\varphi}'[\pi_1/b'_{p_1}, \ldots, \pi_n/b'_{p_n}]$ can be obtained as follows: We start with φ, replace $p_1, \ldots, p_n$ by $b_{p_1}, \ldots, b_{p_n}$, prime $b_{p_1}, \ldots, b_{p_n}$, and then replace $b'_{p_1}, \ldots, b'_{p_n}$ by $\pi_1, \ldots \pi_n$, repectively. Consequently we have

$$stat^{-1}(\hat{\varphi}'[\pi_1/b'_{p_1}, \ldots, \pi_n/b'_{p_n}]) = \varphi[stat^{-1}(\pi_1)/p_1, \ldots, stat^{-1}(\pi_n)/p_n].$$

A close comparison of the inductive definitions for p'_i, the primed version of the past formula p_i, and the corresponding clause $C(p_i)$, relating b'_{p_i} to its expression in terms of variables in $V \cup V' \cup B$, yields that p'_i is equivalent to $stat^{-1}(\pi_i)$. It follows that

$$\varphi[stat^{-1}(\pi_1)/p_1, \ldots, stat^{-1}(\pi_n)/p_n] \equiv \varphi[p'_1/p_1, \ldots, p'_n/p_n] \equiv \varphi'.$$

Thus applying the instantiation $stat^{-1}$ to the state formula above, we obtain

$$(\rho_\tau \wedge \varphi) \Rightarrow \varphi'),$$

which is premise S3 of rule **safe**.

Example 8.5 Premise I3 of the running example is

$$\Big((x' = x+1) \wedge (b' \equiv [(x'=5) \vee b]) \wedge ((x \geq 5) \rightarrow b)\Big) \rightarrow \Big((x' \geq 5) \rightarrow b'\Big).$$

The above transformation simplifies this implication to

$$\Big((x' = x+1) \wedge ((x \geq 5) \rightarrow b)\Big) \rightarrow \Big((x' \geq 5) \rightarrow [(x'=5) \vee b]\Big).$$

Applying the instantiation $stat^{-1}$ that replaces b by $\diamondminus(x=5)$, we obtain

$$\Big((x' = x+1) \wedge ((x \geq 5) \rightarrow \diamondminus(x=5))\Big) \Rightarrow \Big((x' \geq 5) \rightarrow [(x'=5) \vee \diamondminus(x=5)]\Big).$$

Observing that $(\diamondminus(x=5))'$ equals $(x'=5) \vee \diamondminus(x=5)$, this can be rewritten as

$$\Big((x' = x+1) \wedge ((x \geq 5) \rightarrow \diamondminus(x=5))\Big) \Rightarrow \Big((x \geq 5) \rightarrow \diamondminus(x=5)\Big)',$$

which is premise S3 of rule SAFE. ▪

This transformation obviously establishes the following theorem.

Theorem 8.2 (Completeness for safety) *Rule* SAFE *is complete, relative to assertional validity, for proving the validity of safety formulae of the form* $\Box\, q$.

8.3 Transforming an F-RESP proof

Here, we assume a given state proof of the formula $\hat{p} \Rightarrow \Diamond\, \hat{q}$, using rule **F-RESP**. This proof uses state formulae $\hat{\varphi}_1, \ldots, \hat{\varphi}_m$, a well-founded preorder $(\mathcal{A}, \preceq)$ and a ranking function $\hat{\delta} : \Sigma \mapsto \mathcal{A}$, mapping *states* into the domain $\mathcal{A}$.

We consider in turn each premise of the rule and show how to transform it into a corresponding past premise of rule **F-RESP**, establishing $p \Rightarrow \Diamond\, q$. Most of the transformations are similar to those used for rule **INV**.

Premises F1 and J4 are simple entailments, which for the state case must be proven as valid implications. Applying to them the instantiation $stat^{-1}$ is straightforward.

Premises F2 and F3 are verification conditions. Their instantiation by $stat^{-1}$ is very similar to the one we performed for premise I3 of rule **INV**. A new element is the ranking function $\hat{\delta}$. However, we should recall that $\hat{\delta}$ and its $stat^{-1}$-instantiation δ are just terms in the language $\mathcal{L}$, and obey precisely the same rules as the other elements of the language.

Lastly, consider premise C4. We cannot apply instantiation to this formula since rule **INST** requires a state formula as a premise, and C4, even for the case that $\hat{\varphi}_i$ are state formulae, is temporal. However, we should recall that premise C4 is no more than a recursive call to rule **F-RESP**. Unwinding the implied recursion, we obtain a tree, whose leaves are instances of premises F1–F3 and J4, its internal nodes are instances of C4 of the form

$$[\hat{\varphi}_i \wedge (\hat{\delta} = \alpha)] \Rightarrow \Diamond\Big[\hat{q} \vee \Big(\hat{\varphi} \wedge (\hat{\delta} \prec \alpha)\Big) \vee En(\tau_i)\Big],$$

and its root is the conclusion $\hat{p} \Rightarrow \Diamond\, \hat{q}$.

Since we know how to transform the leaves to past formulae, we may trace the proof upwards in the tree obtaining a valid past formula

$$[\varphi_i \wedge (\delta = \alpha)] \Rightarrow \Diamond\Big[q \vee \Big(\varphi \wedge (\delta \prec \alpha)\Big) \vee En(\tau_i)\Big]$$

for each intermediate node, and $p \Rightarrow \Diamond\, q$ for the conclusion.

We may summarize this by the following theorem.

Theorem 8.3 (Completeness for response) *Rule* **F-RESP** *is complete, relative to assertional validity, for proving the validity of response formulae of the form* $p \Rightarrow \Diamond q$.

8.4 Transforming an F-REAC proof

When we consider state proofs by rule **F-REAC**, there is no new element in their transformation into past proofs. For each premise of the rule, there exists a premise of a similar type in rules **INV** or **F-RESP**. Consequently, the relevant transformations have been described above. We may summarize this in the following theorem.

Theorem 8.4 (Completeness for reactivity) *Rule* **F-REAC** *is complete, relative to assertional validity, for proving the validity of reactivity formulae of the form* $(p \wedge \Box \Diamond r) \Rightarrow \Diamond q$.

We may apply similar transformations to Corollary 7.1 to obtain the following.

Corollary 8.1 (Completeness of B-REAC) *For a program with no compassionate transitions, rule* **B-REAC** *is complete, relative to assertional validity, for proving the validity of reactivity formulae of the form* $(p \wedge \Box \Diamond r) \Rightarrow \Diamond q$.

We should emphasize that the systematic elimination of the past from formulae and proofs, which facilitates establishing the completeness of the proof system, is not necessarily the approach we recommend for the actual verification of concrete programs. On the contrary; we strongly recommend working directly with past formulae which explicitly represent the relevant facts about the history of the computation leading to the current state. For example, we find the invariant $\Box((x = 10) \rightarrow \diamondminus(x = 5))$ much more appealing and explicit than the encoded version $\Box((x = 10) \rightarrow b)$, accompanied by the tacit understanding that $b = \mathbf{T}$ iff we have passed through a state in which $x = 5$.

Acknowledgement

We gratefully acknowledge the help of Rajeev Alur, Eddie Chang, Tom Henzinger, and Daphne Koller for their careful reading of the manuscript and thank them for many helpful suggestions.

A Properties of the Preorder $\trianglerighteq$

In Subsection 7.2.2 we defined the preorder $\trianglerighteq$ by the fixpoint equation (3), and showed that it satisfies properties P4–P6 based on Lemma 7.3 which relates the preorder to the ordinal ranking δ_0. Here, we present an alternative proof of these facts without reference to the ordinals. We first present these fact in an abstract setting. For simplicity, we present $\trianglerighteq$ as $\succeq$.

Let $\sqsupset$ be a transitive well-founded binary relation over a set $\mathcal{A}$. We write $a \sqsupseteq b$ to denote that either $a = b$ or $a \sqsupset b$. The relations $a \sqsupset b$ and $c \sqsupseteq d$ can also be written as $b \sqsubset a$ and $d \sqsubseteq c$, respectively.

The relation $\sqsupset$ can be extended to tuples over $\mathcal{A}$ by defining

$$(a_1,\ldots,a_k) \sqsupset (b_1,\ldots,b_k) \quad \text{iff} \quad \begin{cases} a_i \sqsupseteq b_i \text{ for every } i=1,\ldots,k \\ \text{and} \\ a_i \sqsupset b_i \text{ for some } i=1,\ldots,k \end{cases}$$

It can easily be shown that $\sqsupset$ is well-founded also over tuples of $\mathcal{A}$.

Define the relation $\succeq$ to be the *maximal* solution of the implication

$$a \succeq b \quad \rightarrow \quad (\forall b' \sqsubset b)\,(\exists a' \sqsubset a)\; a' \succeq b' \tag{5}$$

By Knaster–Tarski's theorem [Tar55], this implication has a maximal solution which also satisfies the equivalence

$$a \succeq b \quad \leftrightarrow \quad (\forall b' \sqsubset b)\,(\exists a' \sqsubset a)\; a' \succeq b' \tag{6}$$

Lemma A.1 (Transitivity) *Relation $\succeq$ is transitive.*

Proof. A counter example to the lemma is a triple (a,b,c) such that $a \succeq b$ and $b \succeq c$ but $a \not\succeq c$. Let (a,b,c) be a minimal counter example, i.e., a counter example such that, for all $(a',b',c') \sqsubset (a,b,c)$, (a',b',c') is not a counter example. Due to the well-foundedness of $\sqsupset$, if there exists a counter example, there exists a minimal one.

Consider any $c' \sqsubset c$. By $b \succeq c$, there exists a $b' \sqsubset b$ such that $b' \succeq c'$. By $a \succeq b$, there exists an $a' \sqsubset a$ such that $a' \succeq b'$. Since $(a',b',c') \sqsubset (a,b,c)$, (a',b',c') cannot be a counter example, and therefore $a' \succeq c'$. This shows that for every $c' \sqsubset c$ there exists an $a' \sqsubset a$ such that $a' \succeq c'$. That is, the right hand side of (6) is satisfied for a and c. It follows by (6) that $a \succeq c$ and, therefore, (a,b,c) cannot be a counter example. ∎

Lemma A.2 (Totality) *Relation $\succeq$ is total.*

Proof. Assume that (a,b) is a minimal pair of elements that are not $\succeq$-related. In particular, $a \not\succeq b$. Thus, by (6), $(\forall b' \sqsubset b)\,(\exists a' \sqsubset a)\; a' \succeq b'$ is false. This implies that there exists a $b'_0 \sqsubset b$ such that, for all $a' \sqsubset a$, $a' \not\succeq b'_0$. Consider any $a' \sqsubset a$. Since $(a',b'_0) \sqsubset (a,b)$, and (a,b) is a minimal pair of unrelated elements, a' and b'_0 must be related and therefore $b'_0 \succeq a'$. Thus for every $a' \sqsubset a$ there exists a b', namely b'_0, such that $b' \succeq a'$. This establishes the right hand side of (6) for the pair (b,a). It follows that $b \succeq a$ and, hence, a and b are related. ∎

Define $a \succ b$ as $a \succeq b \land b \not\succeq a$. Due to the fact that $\succeq$ is total, we actually have

$$a \succ b \leftrightarrow b \not\succeq a$$

Lemma A.3 (Minimal Separation)
Every a and b satisfy

$$\varphi(a,b) \quad : \quad a \succ b \quad \rightarrow \quad (\exists a' \sqsubset a)\; a' \succeq b$$

Proof. Let (a, b) be a minimal counter example to $\varphi(a, b) \wedge \varphi(b, a)$.

Assume that $\varphi(a, b)$ does not hold. In this case $a \succ b$ but $\neg(\exists a' \sqsubset a)\ a' \succeq b$. From $a \succ b$ which implies $b \not\succeq a$, it follows that there exists an $a'_0 \sqsubset a$ such that, for all $b' \sqsubset b$, $b' \not\succeq a'_0$, i.e., $a'_0 \succ b'$. Since $(a'_0, b) \sqsubset (a, b)$, (a'_0, b) is not a counter example and it should satisfy

$$\varphi(b, a'_0) \quad : \quad b \succ a'_0 \quad \rightarrow \quad (\exists b' \sqsubset b)\ b' \succeq a'_0.$$

By taking the negation of both sides and reversing the implication, this is equivalent to

$$(\forall b' \sqsubset b)\ a'_0 \succ b' \quad \rightarrow \quad a'_0 \succeq b.$$

Since we have shown that the left hand side of this implication holds, we conclude that $a'_0 \succeq b$.

From $\neg(\exists a' \sqsubset a)\ a' \succeq b$, it follows that $a' \not\succeq b$ for every $a' \sqsubset a$, in particular for $a' = a'_0$. This contradicts our previous conclusion and we therefore infer that (a, b) is not a counter example. ∎

The following lemma establishes the converse property.

Lemma A.4 (Embedding)

$$a \sqsupset b \quad \rightarrow \quad a \succ b \tag{7}$$

Proof. Let (a, b) be a minimal counter example to (7). It follows that $a \sqsupset b$ but $b \succeq a$. This implies that for every $a' \sqsubset a$, in particular for $a' = b$, there exists a $b' \sqsubset b$ such that $b' \succeq b$. We have thus identified a new pair $(b, b') \sqsubset (a, b)$ such that $b \sqsupset b'$ but $b' \succeq b$, i.e., a counter example smaller than (a, b). This leads to a contradiction. ∎

We observe the following corollary of Lemma A.1:

$$a \succeq b \wedge b \succ c \quad \rightarrow \quad a \succ c$$

This is because if c would have satisfied $c \succeq a$, we would have had, by $a \succeq b$ and transitivity, $c \succeq b$ which contradicts $b \succ c$.

Lemma A.5 (Well-Foundedness) *The relation $\succ$ is well-founded.*

Proof. Assume, to the contrary, that there existed an infinite sequence

$$a_1 \succ a_2 \succ a_3 \succ \cdots\ .$$

By Lemma A.3 and $a_1 \succ a_2$, there exists an $a'_2 \sqsubset a_1$ such that $a'_2 \succeq a_2$. By the corollary mentioned above, $a'_2 \succ a_3$. Thus, there exists an $a'_3 \sqsubset a'_2$ such that $a'_3 \succeq a_3$. In this way we construct an infinite sequence

$$a_1 \sqsupset a'_2 \sqsupset a'_3 \sqsupset \cdots\ ,$$

which contradicts the well-foundedness of $\sqsupset$. ∎

It is straightforward to verify that the relation $\succ$ satisfies properties P4–P6. In particular,

P4. $\succ$ is well-founded. This follows from Lemma A.5.

P5. $s \sqsupset s' \rightarrow s \succ s'$. This follows from Lemma A.4.

P6. If s' is a successor of s, then $s \succeq s'$. This follows from the fact that every $s'' \sqsubset s'$ satisfies $s'' \sqsubset s$. By (6) and reflexivity of $\succeq$, it follows that $s \succeq s'$.

References

[AP86] K.R. Apt and G.D. Plotkin. Countable nondeterminism and random assignment. *J. ACM*, 33,4:724–767, 1986.

[Apt81] K.R. Apt. Ten years of Hoare's logic: A survey – part I. *ACM Trans. Prog. Lang. Sys.*, 3:431–483, 1981.

[AS89] B. Alpern and F.B. Schneider. Verifying temporal properties without temporal logic. *ACM Trans. Prog. Lang. Sys.*, 11:147–167, 1989.

[Coo78] S.A. Cook. Soundness and completeness of an axiom system for program verification. *SIAM J. Comp.*, 7:70–90, 1978.

[Fra86] N. Francez. *Fairness.* Springer-Verlag, 1986.

[GFMdR85] O. Grumberg, N. Francez, J.A. Makowski, and W.-P. de Roever. A proof rule for fair termination. *Inf. and Comp.*, 66:83–101, 1985.

[Har79] D. Harel. *First-Order Dynamic Logic.* Lec. Notes in Comp. Sci. 68, Springer-Verlag, 1979.

[Kro87] F. Kröger. *Temporal Logic of Programs.* Volume 8 of *EATCS Monographs on Theoretical Computer Science*, Springer-Verlag, 1987.

[LPS81] D. Lehmann, A. Pnueli, and J. Stavi. Impartiality, justice and fairness: The ethics of concurrent termination. In *Proc. 8th Int. Colloq. Aut. Lang. Prog.*, pages 264–277, Lec. Notes in Comp. Sci. 115, Springer-Verlag, 1981.

[LPZ85] O. Lichtenstein, A. Pnueli, and L. Zuck. The glory of the past. In *Proc. Conf. Logics of Programs*, pages 196–218, Lec. Notes in Comp. Sci. 193, Springer-Verlag, 1985.

[Mil80] R. Milner. *A Calculus of Communicating Systems.* Lec. Notes in Comp. Sci. 94, Springer-Verlag, 1980.

[MP83a] Z. Manna and A. Pnueli. How to cook a temporal proof system for your pet language. In *Proc. 10th ACM Symp. Princ. of Prog. Lang.*, pages 141–154, 1983.

[MP83b] Z. Manna and A. Pnueli. Verification of concurrent programs: A temporal proof system. In J.W. de Bakker and J. Van Leuwen, editors, *Foundations of Computer Science IV, Distributed Systems: Part 2*, pages 163–255, Mathematical Centre Tracts 159, Center for Mathematics and Computer Science (CWI), Amsterdam, 1983.

[MP84] Z. Manna and A. Pnueli. Adequate proof principles for invariance and liveness properties of concurrent programs. *Sci. Comp. Prog.*, 32:257–289, 1984.

[MP87] Z. Manna and A. Pnueli. Specification and verification of concurrent programs by $\forall$-automata. In *Proc. 14th ACM Symp. Princ. of Prog. Lang.*, pages 1–12, 1987.

[MP89] Z. Manna and A. Pnueli. The anchored version of the temporal framework. In J.W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, pages 201–284, Lec. Notes in Comp. Sci. 354, Springer-Verlag, 1989.

[MP90a] O. Maler and A. Pnueli. Tight bounds on the complexity of cascaded decomposition of automata. In *Proc. 31th IEEE Symp. Found. of Comp. Sci.*, pages 672–682, 1990.

[MP90b] Z. Manna and A. Pnueli. A hierarchy of temporal properties. In *Proc. 9th ACM Symp. Princ. of Dist. Comp.*, pages 377–408, 1990.

[OL82] S. Owicki and L. Lamport. Proving liveness properties of concurrent programs. *ACM Trans. Prog. Lang. Sys.*, 4:455–495, 1982.

[Pnu86] A. Pnueli. Applications of temporal logic to the specification and verification of reactive systems: A survey of current trends. In J.W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *Current Trends in Concurrency*, pages 510–584, Lec. Notes in Comp. Sci. 224, Springer-Verlag, 1986.

[SdRG84] F.A. Stomp, W.-P. de Roever, and R.T Gerth. *The μ-Calculus as an Assertion Language for Fairness Arguments.* Technical Report 84-12, Utrecht, 1984.

[SdRG89] F.A. Stomp, W.-P. de Roever, and R.T. Gerth. The μ-calculus as an assertion language for fairness arguments. Inf. and Comp., *Inf. and Comp.* **82**, 1989.

[Tar55] A. Tarski. A lattice theoretical fixpoint theorem and its applications. *Pacific J. of Mathematics*, 5, 1955.

[Tho81] W. Thomas. A combinatorial approach to the theory of ω-automata. *Inf. and Cont.*, 48:261–283, 1981.