Technical Report
NWU-EECS-06-20
2/21/2007

Abstraction Techniques for Model-Checking Parameterized Systems

N. Liveris; H. Zhou; R. Dick; Y. Chen; P. Banerjee

Abstract

In this paper we present a new abstraction technique that enables the usage of model checking for the verification of parameterized systems. The technique targets asynchronous systems. Compared to previous approaches the application of the proposed technique imposes fewer restrictions on the correctness property. Moreover, it can be applied to a class of parameterized systems for which other abstraction methods may not work. We demonstrate the effectiveness of the abstraction technique by applying it on a self-stabilizing spanning tree construction algorithm.

Sponsors: NSF, Motorola

Keywords: program abstraction, verification, parameterized systems

Abstraction Techniques for Model-Checking Parameterized Systems

N. Liveris[†], H. Zhou[†], R. Dick[†], Y. Chen[†], and P. Banejee[⋆]

[†]EECS Department, Northwestern University, Evanston IL 60208
[⋆]College of Engineering, University of Illinois, Chicago IL 60607

Abstract. In this paper we present a new abstraction technique that enables the usage of model checking for the verification of parameterized systems. The technique targets asynchronous systems. Compared to previous approaches the application of the proposed technique imposes fewer restrictions on the correctness property. Moreover, it can be applied to a class of parameterized systems for which other abstraction methods may not work. We demonstrate the effectiveness of the abstraction technique by applying it on a self-stabilizing spanning tree construction algorithm.

1 Introduction

This paper describes a new abstraction technique for enabling the use of automatic verification tools to check the correctness of parameterized systems.

There are two kinds of methods that are used for the verification of asynchronous systems: deductive verification and model checking. Deductive verification is an interactive verification method. The user is required to provide properties that facilitate the proof by the tool. Model checking is an automatic method. However, it is efficient only when applied to relatively small finite state space systems. Therefore, abstraction is used to transform infinite or large state space systems to smaller finite state space systems, thereby enabling the use of model checking for their verification [9]. In this paper we present an abstraction technique that enables the use of model checking for the verification of asynchronous parameterized systems.

A parameterized system is built by the parallel composition of N processes, where N can be any natural number greater than a minimum value. Model checking can be used for the verification of instances of a parameterized system with few processes, i.e., small N. Ideally we would like to prove the correctness of a parameterized system for any number of processes. However, the number of those systems is infinite when there is no upper bound for N. Moreover, even if an upper bound exists, verification may be intractable for large N because the number of states increases exponentially with the number of components in the system. There are two ways to overcome this difficulty: one is to use control abstraction [12, 9], and the other is to use the methods of invisible ranking [17, 2, 7].

The idea behind control abstraction is to abstract away an arbitrary number of symmetric processes by using a network invariant I. Then the correctness property can be proved for the abstract system, which is composed of a finite number of processes and the network invariant [9]. A difficulty with this approach is that the network invariant needs to have the same set of observable variables as the system of symmetric processes abstracted by it. Therefore, if each of the N processes has one observable variable and we use a network invariant for these processes, the network invariant needs to have N observable variables. This problem has restricted the application of control abstraction to specific classes of distributed algorithms. Control abstraction has successfully been applied on ring topologies of processes [11], in which every process has only two neighbors and, therefore, the number of input/output variables for each process is independent of N. It has also been successfully applied on systems for which the number of shared variables does not increase with the number of processes. An example is a mutual exclusion algorithm, in which all processes share only one semaphore [9].

An alternative approach is the method of invisible invariants [17]. The method can be used to bound the number of processes needed to prove a correctness property for a class of parameterized systems. The approach can be used for the verification of safety properties [2] and response liveness properties [7]. Response liveness properties are properties of the form $\Box(p \rightarrow \Diamond r)$, i.e., for every state satisfying assertion p there is a future state satisfying assertion r. The paper does not discuss how other liveness properties can be checked using this method. Moreover, the method imposes a number of restrictions on the structure of the next state relation and the initial condition of the system. The method can be automated, so the user does not have to observe the invariant being used. In our work we target general liveness properties. More specifically, the correctness property we prove for the spanning tree algorithm is a persistence property ($\Diamond\Box p$). This type of property can be used to describe the correctness of self-stabilizing systems.

In this paper we present an abstraction technique that builds on the theory of control abstraction. We target structures of processes, for which the number of observable variables is a parameter. These structures are very common. One example is proving a property for a process that is connected in a graph of arbitrary topology. Then the number of neighbors with which the process interacts is a parameter. For this type of structure we provide an abstraction that reduces the number of observable variables to a small finite number. Then model checking is used to check the correctness property on the abstract system. If the property holds for the abstract system, then it holds for the parameterized system for any valid value of the parameter. The proposed technique can be used for checking general temporal properties. Moreover, it can be applied to a class of parameterized systems for which other methods may not work.

The method is sufficient to prove that the correctness property is valid for the parameterized system. However, like control abstraction and invisible ranking, the proposed method is incomplete. If a behavior of the abstract system does not satisfy the correctness property, no conclusion can be drawn for the parameterized system. Additionally, a number of restrictions must hold for the parameterized system to guarantee the soundness of our approach. Most of these restrictions concern the atomicity of the actions of the parameterized systems.

In Section 2 we survey related work in this field and indicate our contributions. Section 3 describes the systems we consider. Section 4 gives an overview of the proposed technique. We demonstrate the application of the technique to a Spanning-Tree construction algorithm in Section 5. The soundness of the technique is proved in the appendix.

2 Related work

Instead of control abstraction, the method of invisible invariants can be used for the verification of parameterized systems [17]. Arons et al. [2] present the method for the verification of safety properties. Later the work was extended to include verification of response liveness properties [7]. The authors describe a method to bound the number of processes needed to prove a correctness property for a parameterized system. However, their method is applicable to a restricted class of distributed algorithms. One of the restrictions is that a variable h taking values in $1..N$, can appear in two kinds of expressions: "h" and "$array[h]$". These expressions can only appear in a formula that compares expressions of the same type. This means that the only operators that can be applied to h are indexing $[]$ and relational operators $=, \neq, \leq, <, \geq, >$. When operators $\oplus_N 1$ and $\ominus_N 1$ are included, the size of the abstract system increases significantly. Only response liveness property are discussed in that work. Our approach can be applied to other temporal properties, for example persistence.

Other works on the abstraction of parameterized systems can only be applied to high-atomicity distributed algorithms [5, 18, 4, 6], in which a process can read the values of all its N neighbors in one atomic step. Since this assumption may not be realistic for large N, in our work we target low atomicity distributed algorithms.

Kurshan and McMillan presented a structural induction theorem for processes [12] defining sufficient properties for the network invariants. The method was applied on two examples; in the first the processes formed a complete graph and in the second a ring. However, for the case of the complete graph the authors consider the observable behavior as the sequence of actions taken. More specifically, in that example it is the sequence of messages sent between the system and its environment. Therefore, the problem we deal with in this paper was not discussed.

Kesten and Pnueli define a sound data abstraction method [10]. Their method is useful for reducing the range of the data variables of the system. The user needs to invent the abstraction function and in some case the progress monitor for a specific system. Their method can be used as a preprocessing step to replace variables ranging over parameterized or infinite domains to abstract variables, which range over finite sets. Then our technique provides the user with an abstraction function to reduce the number of variables in the system.

Other works discuss only safety properties for the verification of parameterized systems [8].

The abstraction technique presented in this paper is similar to temporal case splitting [15, 16] in that it reduces a vector of unbounded size to a vector with a fixed small number of elements. With temporal case splitting the proof is decomposed to a large number of proof subgoals, which become a fixed number of problems using symmetry properties. Our technique includes the abstraction of the fairness conditions of the concrete system, especially on actions accessing or modifying the elements of the abstracted vector. Moreover, we define fairness conditions for the abstract system based on the existence of constant values stored in the vector. As we will show, these fairness conditions are necessary, e.g., to prove the correctness of a spanning tree algorithm. Additionally, constant values of the index type do not increase the size of the abstract system in our approach.

3 Systems we consider - Notation

In this section we describe the types of systems we consider.

We deal with the verification of closed parameterized systems. A closed parameterized system can be defined as

$$\mathcal{T}(N) = (P(1)\|P(2)\|...\|P(N))_R$$

In the above formula $P(1), P(2), ..., P(N)$ are identical processes up to renaming. The operator $\|$ denotes parallel asynchronous composition and $()_R$ restriction. Both operators are defined in the literature [9]. In the above definition of the parameterized system it is assumed that each process $P(i)$, for $i \in 1..N$, is independent of the number N of processes in the system. For example, the N processes may be used to specify a mutual exclusion algorithm with only one shared variable for the whole system [9]. However, there are cases in which the number of observable variables of each process depends on N. In those cases, the definition of the system can be written as

$$\mathcal{Q}(N) = (P(1,N)\|P(2,N)\|...\|P(N,N))_R \tag{1}$$

While $P(i,N)$ and $P(j,N)$ with $i \neq j$ differ only in their id values, $P(i,N)$ and $P(i,M)$ with $N \neq M$ have a different number of observable variables.

We are interested in proving the correctness of a parameterized systems described by (1).

In this paper we follow a similar notation that is used by Abadi and Lamport [1] for describing systems. Each system, which can be composed of one or more processes, is represented by its specification $S = \langle \Sigma, F, \mathcal{N}, L \rangle$. Σ is the state space of the specification, $F \subseteq \Sigma$ is the set of initial states, $\mathcal{N} \subseteq \Sigma \times \Sigma$ is the next state relation, and L is the supplementary property defined over Σ. The state machine $\langle \Sigma, F, \mathcal{N} \rangle$ defines the machine property of S. For all systems we consider, S is machine closed and L specifies a liveness property. The definition of machine closure can be found in the literature [1, 13].

The next state relation is defined using a set of atomic actions A. Each action α has a precondition (or enable condition) prec(α), which is a state function, and an effect part eff(α), which describes the values of the variables in the next state s', as a function of the current state s. Therefore, α can be described as the conjunction of its precondition and its effect[1]

[1] Actions also contain conjucts of the form $m' = m$ for each variable m that must remain unchanged. Therefore, the effect of an action may be considered as the conjunct of $\varepsilon(\alpha) \wedge \text{unch}(\alpha)$. In the last formula $\varepsilon(\alpha)$ is a boolean combination of predicates of the form $m' = g(s)$, and unch(α) is the conjunction of predicates of the form $m' = m$. We say that an action "reads" a variable n, when n appears in a predicate $m' = g(s)$ in $\varepsilon(\alpha)$. An action "modifies" or "writes" a variable m, when there is a predicate $m' = g(s)$ in $\varepsilon(\alpha)$ and $g(s) \neq m$. This classification is based on the syntax and can be performed by static analysis.

$\alpha \triangleq \wedge\ \mathrm{prec}(\alpha)$
$\quad\ \wedge\ \mathrm{eff}(\alpha)$

A state pair $\langle s, s' \rangle \in \mathcal{N}$, if and only if there exists $\alpha \in A$, such that $\mathrm{prec}(\alpha)$ is true for s and the pair of states $\langle s, s' \rangle$ satisfies $\mathrm{eff}(\alpha)$. Then the triple (s, α, s') is called a transition of the system. We assume there is a stuttering step $\tau \in A$ and for all states $s \in \Sigma$, $\langle s, s \rangle$ belongs to $\mathcal{N}$.

The liveness property L is a restriction imposed on the infinite behaviors of the system. It can include the conjunction of strong and weak fairness properties specified on some of the actions. We use $\mathcal{W}$ and $\mathcal{S}$ to represent the sets of actions with weak and strong fairness properties respectively. The sets $\mathcal{W}$ and $\mathcal{S}$ are disjoint and subsets of A. Then $L \rightarrow \bigwedge_{\alpha \in \mathcal{W}} wf(\alpha) \wedge \bigwedge_{\alpha \in \mathcal{S}} sf(\alpha)$. The weak and strong fairness properties are defined as

$$wf(\alpha) \triangleq (\Box\Diamond\neg \mathrm{prec}(\alpha)) \vee (\Box\Diamond(\langle \mathrm{eff}(\alpha) \rangle))$$

$$sf(\alpha) \triangleq (\Diamond\Box\neg \mathrm{prec}(\alpha)) \vee (\Box\Diamond(\langle \mathrm{eff}(\alpha) \rangle))$$

The expression $\langle \mathrm{eff}(\alpha) \rangle$ evaluates to true when action α is executed and the system's state changes. Therefore, for a pair of states $\langle s, s' \rangle$

$$\langle s, s' \rangle \models \langle \mathrm{eff}(\alpha) \rangle \;\Leftrightarrow\; \langle s, s' \rangle \models \mathrm{eff}(\alpha) \wedge s' \neq s$$

For a more detailed description of weak (wf) and strong (sf) fairness properties, the reader should refer to the literature [13].

Moreover, L can include justice and compassion requirements on sets of states. Justice requirements have the form $\Box\Diamond p$ and compassion requirements can be represented as $\Box\Diamond q \rightarrow \Box\Diamond r$, where p,q, and r are atomic predicates which specify sets of states [9]. In this paper we assume that p, q, r are not specified on the shared variables of the system, which are going to be abstracted, i.e., the variables in SV_I (see Section 4 for the definition of this set).

A sequence σ of states, with $\sigma \in \Sigma^\omega$, is a behavior of S if σ satisfies the specification S. More specifically, it must hold that $(\sigma, 0) \in F$, $\forall i \geq 0 : \langle (\sigma, i), (\sigma, i+1) \rangle \in \mathcal{N}$, and $\sigma \models L$, where (σ, i) is the ith state in sequence σ.

We use this operator $\models$ to denote that an assertion is valid for a state or a set of states. We extend its usage to temporal properties and sequences or sets of sequences. When a specification is used at the LHS and a temporal formula at the RHS, the temporal formula is valid for all behaviors of the specification.

For a state $s = (e, i) \in E \times F$, where $e \in E$ and $i \in F$, we call e the E-part of the state or e-part of the state. The e-part of the state includes only the variables in e and their values.

The operator Π denotes projection. For a state $s = (e, i)$, it holds $\Pi_e(s) = e$ and $\Pi_{[e]}(s) = i$. That means $\Pi_e(s)$ is the projection of s on the e-part of the state and $\Pi_{[e]}(s)$ is the projection of s on the part of the state that is not included in e. We use the projection operator on sets of states as well.

Another operator that is commonly used is the property operator $\mathcal{P}$. Operator $\mathcal{P}$ is defined on a temporal property χ, which could be a system specification, and denotes all behaviors that satisfy χ.

Some variables of the system are local to a specific process, while others are observable to all processes. We call the observable variables "shared variables". We consider vectors of shared variables of the form

$$\begin{aligned}
sv_1 &\in [1..N \rightarrow \mathrm{FS}_1] \\
sv_2 &\in [1..N \rightarrow \mathrm{FS}_2] \\
&\dots \\
sv_k &\in [1..N \rightarrow \mathrm{FS}_k]
\end{aligned}$$

where FS_i is a finite set for all i in $1..k$. For simplicity, we restrict our discussions to systems with only one parameter N and only one vector of shared variables sv. Then the system state space can be represented as $\Sigma = \Sigma_{nsv} \times [1..N \rightarrow \mathrm{FS}]$, where Σ_{nsv} is the state space without the sv vector. In the rest of the paper we refer to each $sv[j]$ with $j \in 1..N$ as a shared variable.

Besides vectors of shared variables, the system can have a finite set of variables that are observable to all processes. The cardinality of the set must be independent of N. We consider these variables as part of Σ_{nsv} and we restrict the usage of the term "shared variable" only for an element of the vector sv.

If the effect of one action α can be obtained from the effect of another action β by replacing any appearance of one shared variable $sv[k]$ with another shared variable $sv[j]$, then $\mathrm{eff}(\alpha)$ and $\mathrm{eff}(\beta)$ are called syntactically equivalent. For example, in Figure 1 only $\mathrm{eff}(\alpha)$ and $\mathrm{eff}(\beta)$ are syntactically equivalent.

$$\boxed{\begin{array}{llll}
\alpha \triangleq \wedge\ \mathrm{prec}(\alpha) & \beta \triangleq \wedge\ \mathrm{prec}(\beta) & \gamma \triangleq \wedge\ \mathrm{prec}(\gamma) & \delta \triangleq \wedge\ \mathrm{prec}(\delta) \\
\quad \wedge\ r' = sv[j]+1 & \quad \wedge\ r' = sv[k]+1 & \quad \wedge\ p' = sv[j]+1 & \quad \wedge\ r' = sv[j]+2
\end{array}}$$

Fig. 1. Only the two leftmost actions have syntactically equivalent effects.

We now present our assumptions for the systems we consider. Then we elaborate on the reasons for making these assumptions and their implications.

Λ1. Although actions can read and modify a number of Σ_{nsv} variables, they can either read or write at most one shared variable in each atomic step.

Λ2. Each shared variable is a single-writer multi-reader variable. More specifically,

$$\forall j \in 1..N : sv[j] \text{ can be written only by process } j$$

Λ3. The preconditions of the actions do not depend on the values of the shared variables. Therefore, reading or writing a shared variable can only be done by the effect part of an action.

Λ4. (a) There exist no action that reads only variables in Σ_{nsv} and has the same effect at some state as an action that reads a shared variable. More specifically, if α is an action that reads a shared variable and $\alpha \in \mathcal{W} \cup \mathcal{S}$, then for any action β that does not read a shared variable

$$\forall \langle s, s' \rangle \in \mathcal{N} : \langle s, s' \rangle \not\models (\langle \mathrm{eff}(\alpha) \rangle \wedge \langle \mathrm{eff}(\beta) \rangle)$$

(b) There exist no action that modifies only variables in Σ_{nsv} and has the same effect at some state as an action that writes a shared variable. More specifically, if α is an action that writes a shared variable and $\alpha \in \mathcal{W} \cup \mathcal{S}$, then for any action β that does not write a shared variable

$$\forall \langle s, s' \rangle \in \mathcal{N} : \langle s, s' \rangle \not\models (\langle \text{eff}(\alpha) \rangle \wedge \langle \text{eff}(\beta) \rangle)$$

(c) Two actions that are not syntactically equivalent and access different shared variables cannot have the same effect at some state s, unless the shared variables accessed have the same value at s. More formally, if α and β are two actions of the system with $\text{eff}(\alpha) = h(e, e', sv[j])$ and $\text{eff}(\beta) = g(e, e', sv[k])$,then

$$\forall \langle s, s' \rangle \in \mathcal{N} : \langle s, s' \rangle \not\models (\langle \text{eff}(\alpha) \rangle \wedge \langle \text{eff}(\beta) \rangle \wedge sv[j] \neq sv[k])$$

We believe that the above constraints are common among many applications. Only $\Lambda 1$ and $\Lambda 2$ restrictions are needed when safety properties are checked. When liveness conditions are checked, $\Lambda 3$ and $\Lambda 4$ restrictions must also hold. The restriction $\Lambda 3$ has been used in other works ([14],Chapter 9). Our intuition behind this restriction is that reading a non-local variable should be an atomic action. The decision of a process to execute an action should be based on local variables only. Note that process j can maintain a local copy of $sv[j]$ and because of restriction $\Lambda 2$ the copy can be always equal to the value of the shared variable. The intuition behind $\Lambda 4$ is that we cannot satisfy the liveness requirements of an action by simulating it with a completely different action. However, syntactically equivalent actions are not restricted by $\Lambda 4$. Most systems with a program counter for each process satisfy the $\Lambda 4$ restriction. More specifically, if each instruction has a different successor, the effect of each action of one process is distinct. Since the program counter is a local variable of each process, the effect of each action cannot be simulated by an action of a different process.

The restrictions $\Lambda 1$-$\Lambda 4$ do not need to hold for the fixed set of global variables in Σ_{nsv}. Therefore, we can have a fixed finite set of multi-writer variables.

For the systems amenable to our technique the correctness property φ that we are going to check is independent of the number of processes in the system. More specifically, φ is expressed as a function of the local and shared variables of a finite set of processes B. The set B is the same for all values $N \geq N_{min}$. For simplicity in this paper we assume that $|B| = 1$. This means that the correctness property is specified on $P(1, N)$. Under symmetry conditions the property will hold $\forall N : P(N, N)$, if it holds for $P(1, N)$. Note that φ can be any LTL property that can be expressed using operators $\Box$ and $\Diamond$.

As noted before we are concerned with the verification of a closed parameterized system. In such a system there is no interaction with the environment. The property that we want to prove is described as a function of some variables of the system. These variables represent the external part of the state. While all other variables belong to the internal part of the state. The distinction of external and internal part of the state is described in the literature [1].

In this section we presented the assumptions for the systems we consider and the notation we use. In the next section we describe the proposed technique for the verification of these systems.

4 The proposed technique

In this section we describe the proposed technique for the abstraction of parameterized systems of the form of (1). We start by describing a verification framework in which this technique is useful (Section 4.1). Then we present in detail the abstraction of the technique (Section 4.2). For simplicity, in this section we assume that $|B| = 1$ and that there is only one parameter N and one shared variable *sv* with N elements. Because $|B| = 1$, the number of shared variables in the abstract system is 2 ($= |B| + 1$).

4.1 Overview of the approach

We wish to verify that property φ is valid for the closed parameterized system

$$Q(N) = (P(1,N) \| P(2,N) \| ... \| P(N,N))_R$$

for all finite $N \geq N_{min}$, where N_{min} is the minimum number of processes in the system.

We assume that the system $Q(N)$ and property φ satisfy all the assumptions described in Section 3. The property may be verified as follows:

1. The user provides a network invariant $I(N)$ [9], such that for any N

$$P(2,N) \| ... \| P(N,N) \sqsubseteq_M I(N)$$

   and the number of local variables of $I(N)$ is finite and independent of N.
2. Following the steps of our technique as described in Section 4.2, the user obtains the system S_a.
3. Model checking is used to automatically prove $S_a \models \varphi$.

Then the user concludes that $Q(N) \models \varphi$ holds for all $N \geq N_{min}$.

In order for the third step to be successful, $I(N)$ and $P(1,N)$ should be finite-state processes for any N. Note that for the modular abstraction relation of step 1 to be valid, $I(N)$ must have N observable variables.

In this paper we are dealing with the second step, which is described in the next section.

4.2 Obtaining the abstract system

In this section we describe the technique to derive S_a from $S_c = (I(N) \| P(1,N))_R$. We denote the abstract system as $S_a = \langle \Sigma_a, F_a, \mathcal{N}_a, L_a \rangle$ and explain how each of the components of the S_a can be obtained from the corresponding components of $S_c = \langle \Sigma, F, \mathcal{N}, L \rangle$.

State space: The only change in the state space is that the shared variables $sv \in [1..N \rightarrow \text{FS}]$ become $sv_a \in [1..2 \rightarrow \text{FS}]$. Let Σ be expressed as $\Sigma = \Sigma_{nsv} \times [1..N \rightarrow \text{FS}]$, where Σ_{nsv} is the state space of all variables except *sv*. Then we can formally define Σ_a as $\Sigma_a = \Sigma_{nsv} \times [1..2 \rightarrow \text{FS}]$. We denote sv_a the variable in $[1..2 \rightarrow \text{FS}]$.

Next state relation: The next state relation $\mathcal{N}_a$ of S_a is defined by a new set of actions $\tilde{A}$. We derive $\tilde{A}$ from some newly defined actions and the S_c actions. Each action of S_c is either an action of $I(N)$ or of $P(1,N)$. The actions of process $I(N)$ cannot modify $sv[1]$ because of the single-writer restriction ($\Lambda 2$). They can modify any of the $N-1$ elements $sv[j]$ with $j \in 2..N$. Let SV_I be the set of these elements, i.e.

$$\mathrm{SV}_I \triangleq \{sv[j] | j \in 2..N\}$$

On the other hand, the actions of process $P(1,N)$ can access any of the elements of sv, but can only modify $sv[1]$.

The following steps describe how we obtain $\tilde{A}$, which initially is an empty set.

T0 For each value v in FS, we define and add to $\tilde{A}$ an action α_v of the form

$$\begin{aligned}\alpha_v \triangleq\ & \wedge\ sv'_a = [sv_a \text{ EXCEPT } ![2] = v] \\ & \wedge \text{ UNCHANGED } \langle all\_other\_variables \rangle\end{aligned}$$

The precondition of α_v is TRUE in all states and its effect is to change $sv_a[2]$ to a new value in FS. All other variables remain unchanged.

T1 For any action $\alpha \in A$ that does not access or write any of the variables in SV_I, $\alpha \in \tilde{A}$[2].

T2 For any action $\alpha \in A$ that reads variable $sv[j]$, with $sv[j] \in \mathrm{SV}_I$, we replace all references to $sv[j]$ by $sv_a[2]$ to obtain $\tilde{\alpha}$. Then we add $\tilde{\alpha}$ to the set of actions $\tilde{A}$, i.e. $\tilde{\alpha} \in \tilde{A}$.

T3 For any action $\alpha \in A$ that writes to variable $sv[j]$, with $sv[j] \in \mathrm{SV}_I$, we replace all references to $sv[j]$ by $sv_a[2]$ to obtain $\tilde{\alpha}$. Then we add $\tilde{\alpha}$ to $\tilde{A}$, i.e. $\tilde{\alpha} \in \tilde{A}$.

T4 For any action $\alpha \in A$ that reads an element $sv[g(e)]$ with $g(e) \in 1..N$ and $e \in \Sigma_{nsv}$, let $\mathrm{eff}(\alpha) = h(e, e', sv[g(s)])$. We define action $\tilde{\alpha}$ as follows

$$\begin{aligned}\tilde{\alpha} \triangleq\ & \vee \wedge \mathrm{prec}(\alpha) \\ & \quad \wedge\ g(e) = 1 \\ & \quad \wedge\ h(e, e', sv_a[1]) \\ & \vee \wedge \mathrm{prec}(\alpha) \\ & \quad \wedge\ g(e) \neq 1 \\ & \quad \wedge\ h(e, e', sv_a[2])\end{aligned}$$

Action $\tilde{\alpha}$ is composed of two disjuncts, one for each possible value of $g(e)$. The first disjunct is the conjunction of $\mathrm{prec}(\alpha)$, $g(e) = 1$, and the effect expression of action α with every reference of $sv[1]$ replaced by $sv_a[1]$. The second disjunct is the conjunction of $\mathrm{prec}(\alpha)$, $g(e) \neq 1$, and the effect expression of action α with every reference to $sv[1]$ replaced by $sv_a[2]$. The new action $\tilde{\alpha}$ is added to $\tilde{A}$, i.e. $\tilde{\alpha} \in \tilde{A}$.

Note that there are no actions in A that read more than one element of sv or read and modify elements of sv because of restriction $\Lambda 1$. Furthermore, if an action writes to a variable $sv[g(s)]$, we can determine whether it is an action handled by rule T1 or T3 based on the process performing the action because of restriction $\Lambda 2$. Consequently, any action in A is handled by one of the $T1 - T4$ cases.

[2] For simplicity for this and the following types of actions we do not describe the changes in the $\mathrm{unch}(\alpha)$ part. From now on we will use $\mathrm{eff}(\alpha)$ to describe the $\varepsilon(\alpha)$ part of the action.

Initial states: The initial states F_a can be obtained by the projection of F on Σ_{nsv} and the set of initial values for the variables sv. In S_a the element $sv_a[1]$ has the same set of initial values as $sv[1]$ in the original system. For $sv_a[2]$ the set of initial values is the set of all possible initial values in the original system for the elements in SV_I.

Liveness conditions: Based on the rule used to define an action $\tilde{\alpha}$ its weak or strong fairness properties will be specified. For any $\tilde{\alpha}$ constructed based on rule $T1$ from S-action α, the action inherits the weak or strong fairness properties, if any, of α. The same happens for any $\tilde{\alpha}$ produced from α by rules $T2 - T4$. However, in this case $\tilde{\alpha}$ can be considered as being constructed from a set of actions $A_s \subset A$. For any such $\tilde{\alpha}$ the fairness property added to L_a will be the weakest property specified for any action in A_s. More formally, if $\tilde{\alpha}$ can be constructed by any $\alpha \in A_s$ using one of the rules $T1 - T4$, then

$$\begin{array}{rl} A_s \cap \mathcal{W}^c \cap \mathcal{S}^c \neq \emptyset & \Rightarrow \tilde{\alpha} \in \left(\tilde{\mathcal{W}}^c \cap \tilde{\mathcal{S}}^c\right) \\ (A_s \cap \mathcal{W}^c \cap \mathcal{S}^c = \emptyset) \wedge (A_s \cap \mathcal{W} \neq \emptyset) & \Rightarrow \tilde{\alpha} \in \tilde{\mathcal{W}} \\ (A_s \cap \mathcal{W}^c \cap \mathcal{S}^c = \emptyset) \wedge (A_s \cap \mathcal{W} = \emptyset) \wedge (A_s \cap \mathcal{S} \neq \emptyset) & \Rightarrow \tilde{\alpha} \in \tilde{\mathcal{S}} \end{array}$$

In the relations above $\mathcal{W}^c$ and $\mathcal{S}^c$ are the complements of $\mathcal{W}$ and $\mathcal{S}$, respectively.

Besides the strong and weak fairness conditions on actions, we specify some liveness conditions related to constants. More specifically, suppose for any $N \geq N_{min}$ and for all behaviors of $Q(N)$, there exists $k \in 2..N$ and $v_k \in \mathrm{FS}$, such that it holds $\Box(sv[k] = v_k)$. If there exists an action $\alpha \in \mathcal{W}$, accessing $sv[k]$, then we define condition $c(e, e', v_k)$ obtained from $\langle \mathrm{eff}(\alpha) \rangle$ by replacing each occurrence of $sv[k]$ by the value v_k. We define constraint

$$cf(\alpha) \triangleq \Box\Diamond\neg\mathrm{prec}(\alpha) \vee \Box\Diamond c(e, e', v_k)$$

For an action $\alpha \in \mathcal{S}$ accessing $sv[k]$, the corresponding constraint will be

$$cf(\alpha) \triangleq \Diamond\Box\neg\mathrm{prec}(\alpha) \vee \Box\Diamond c(e, e', v_k)$$

We denote as $\mathcal{C}$ the set of the actions that read shared variables, which for all $N \geq N_{min}$ and for all behaviors of $Q(N)$ have a constant value. Note that the k does not need to be the same specific index for all behaviors, if the fairness properties are specified on a set of syntactically equivalent actions that are defined for all $i \in 2..N$.

Finally, for any justice or compassion conditions L_e expressed on variables only on Σ_{nsv} and $sv[1]$, we require that the abstract system satisfies the conditions L_e, as well.

Then L_a can be expressed as

$$L_a = \bigwedge\nolimits_{\tilde{\alpha} \in \tilde{\mathcal{W}}} wf(\tilde{\alpha}) \wedge \bigwedge\nolimits_{\tilde{\alpha} \in \tilde{\mathcal{S}}} sf(\tilde{\alpha}) \wedge \bigwedge\nolimits_{\alpha \in \mathcal{C}} cf(\alpha) \wedge L_e$$

The example below is a demonstration of rule T2. The new action is created from $N - 1$ actions of the abstract system. If there exists a constant value in the concrete system for some $k \in 2..N$ and $\forall j \in 2..N : \mathrm{Action}(j) \in \mathcal{W} \cup \mathcal{S}$, then the new abstract action is included in $\mathcal{C}$.

```
──────────── module a ────────────
concrete version
...
$Action(j) \triangleq \land\ var1 < var2$
              $\land\ \lor\ u1 + 1 < u2$
                 $\lor\ u2 \neq var2$
              $\land\ \lor\ var2' = sv[j] + 1$
                 $\lor\ var2' = sv[j]$
              $\land$ UNCHANGED $\langle other\_variables \rangle$
...
$Next \triangleq \lor\ \exists j \in 2..N : Action(j)$
         $\lor$ ...
...
──────────────────────────────────
```

```
──────────── module b ────────────
abstract version
...
$Action \triangleq \land\ var1 < var2$
           $\land\ \lor\ u1 + 1 < u2$
              $\lor\ u2 \neq var2$
           $\land\ \lor\ var2' = sva[2] + 1$
              $\lor\ var2' = sva[2]$
           $\land$ UNCHANGED $\langle other\_variables \rangle$
...
$Next \triangleq \lor\ Action$
         $\lor$ ...
...
──────────────────────────────────
```

The following theorem states that the abstraction technique is safe.

Theorem 1. *If* $S_a \models \varphi$*, then* $\forall N \geq N_{min} : Q(N) \models \varphi$.

In the appendix we give the complete proof for the soundness of the technique. In the next section we demonstrate the usefulness of the proposed technique by applying it on a spanning-tree construction algorithm.

5 Spanning-Tree example

In this section we describe the application of the proposed technique on a variant of Arora and Gouda's Spanning-Tree (ST) construction algorithm [3]. In the appendix (Algorithm 1-3) the TLA+ descriptions of different versions of the ST algorithm are displayed. The algorithm is explained in Section 5.1. In Section 5.2 we present a version of the ST-algorithm after the application of data abstraction. In Section 5.3 we present some theoretical preliminaries. We report the results of the application of our technique in Section 5.4.

5.1 ST algorithm

In this algorithm each process executes the program displayed in appendix (Algorithm 1). The node with the greatest id, EQk, becomes the root of the tree. Eventually, every other node i stores in its $\texttt{Root}[i]$ the id of the root. Moreover, eventually $\texttt{D}[i]$ holds i's distance from the root and $\texttt{F}[i]$ its parent in the tree. Node i selects the parent node, so that $\texttt{D}[i]$ is equal to the minimum distance, $\text{dist}[i]$, from the root in the graph. Note that $\text{dist}[i]$ is not a variable of the system.

For the spanning tree construction algorithm we want to prove the convergence of any process with dist $= l$ after all its neighbors with dist $= l-1$ have converged and some general properties on the graph hold. In order to do so, we assume process $P(1,N)$ is at distance l from the root and has $N-1$ neighbors. Out of the $N-1$ neighbors at least one must be a neighbor at distance $l-1$ from the root. Some of the neighbors can have dist $= l$ or dist $= l+1$. There is no neighbor that can be at distance less than $l-1$ or more than $l+1$. Otherwise, $P(1,N)$'s distance would not be l, which is a contradiction.

The property ψ that we want to prove is

$$\psi \triangleq \Diamond\Box H \rightarrow \Diamond\Box J$$

where
$$
\begin{aligned}
H \triangleq\ & \wedge\ \forall i \in 1..N : \wedge\ \text{dist}[i] = l-1 \rightarrow \wedge\ \texttt{Root}[i] = \text{EQk} \\
& \hspace{11em} \wedge\ \texttt{D}[i] = l-1 \\
& \hspace{5em} \wedge\ (\text{dist}[i] \geq l \wedge \texttt{Root}[i] = \text{EQk}) \rightarrow \texttt{D}[i] \geq l \\
& \hspace{5em} \wedge\ \texttt{Root}[i] \leq \text{EQk} \\
& \wedge\ \exists j \in 2..N : \texttt{lsv} \ = sv[j]
\end{aligned}
$$

and
$$
\begin{aligned}
J \triangleq\ & \wedge\ \texttt{Root}[1] = \text{EQk} \\
& \wedge\ \texttt{D}[1] = l \\
& \wedge\ \texttt{F}[1] \in \{j | j \in 2..N \wedge \text{dist}[j] = l-1\}
\end{aligned}
$$

Note here that $P(1,N)$ is not symmetric to all processes $P(2,N),...,P(N,N)$. However, $P(1,N)$ cannot guess the dist values of the processes, so all processes are identical up to renaming for $P(1,N)$. Moreover, $P(1,N)$ is identical up to renaming to all other process at distance l from the root.

5.2 Data abstraction

In the variant of the ST algorithm, some of the variables range over parameterized or infinite domains. These variables include

$$
\begin{aligned}
\texttt{Root} &\in [1..N \rightarrow \mathbb{N}] \\
\texttt{D} &\in [1..N \rightarrow \mathbb{N} \cup \{0\}] \\
\texttt{F} &\in \mathbb{N}
\end{aligned}
$$

and variables used as local copies of their values (`lRoot`, `lD`, and `lF`).

We use data abstraction [10, 9] to reduce the state space of the system to a finite set, which is independent of the number of processes in the system. More specifically, data abstraction reduces the range of the above variables to a finite set. Even though the range of the variables can be reduced, the number of the shared `Root` and `D` variables still depends on N. Therefore, the abstraction technique presented in this paper is employed as a next step to reduce the number of these variables to 2. Then we can use model checking to verify the system.

Formally, the steps we followed to verify that the parameterized system $Q(N)$ satisfies property φ are

1. Abstracted the system and the property to $Q(N)^a$ and φ^a in which all variables range over finite domains
2. Used the abstraction technique to transform $Q(N)^a$ to S_a

The abstract property φ^a does not need to be abstracted in the second step because we assume it is expressed on variables not in SV_I. If model checking proves that $S_a \models \varphi^a$, we conclude that $Q(N)^a \models \varphi^a$ because of Theorem 1. Moreover, because of results presented in the literature [10], $Q(N)^a \models \varphi^a \Rightarrow Q(N) \models \varphi$.

The version of the algorithm after data abstraction can be seen in the appendix (Module Process - Algorithm 2). The variable `Root` now ranges over the set $\{$LTk,EQk$\}$. Model values LTk and EQk stand for "less than root" and "equal to root", respectively. Variable `F` is abstracted to $\{$NotNeighborNode, NeighborLorMore, NeighborLMinus1, 1$\}$. Finally, all `D` values that are greater than K are abstracted to DGTK model value. The corresponding operators for the model values are defined using the principles of data abstraction [10].

5.3 Theoretical preliminaries

The first task according to the overview of our approach (Section 4.1) is to build a network invariant $I(N)$ for all processes other than $P(1)$. The following result can help us simplify the construction of the network invariant.

Lemma 1. *If every reachable state that satisfies H is an initial state, then for any N*

$$Q(N) \models \Diamond\Box H \rightarrow \Diamond\Box J$$

if and only if

$$Q(N) \models \Box H \rightarrow \Diamond\Box J$$

PROOF:We start with the direction

$$(Q(N) \models \Box H \rightarrow \Diamond\Box J) \Rightarrow (Q(N) \models \Diamond\Box H \rightarrow \Diamond\Box J)$$

Suppose it holds $Q(N) \models \Box H \rightarrow \Diamond\Box J$ and there is a behavior σ of $Q(N)$ for which $\sigma \not\models \Diamond\Box H \rightarrow \Diamond\Box J$. Then

$$\sigma \not\models \neg(\Diamond\Box H) \vee \Diamond\Box J \Rightarrow$$

$$\sigma \models \Diamond\Box H \wedge \Box\Diamond\neg J \Rightarrow$$

$$\sigma \models \Diamond\Box H \wedge \sigma \models \Box\Diamond\neg J$$

Consequently, there exists $j \in \mathbb{N}$ such that the execution segment starting at state (σ, j) satisfies always H and has infinitely many $\neg J$ states. Since (σ, j) is also an initial state, there exists sequence τ with

$$(\tau, i) = (\sigma, j+i), \forall i \geq 0$$

Sequence τ is also a behavior of $Q(N)$ and satisfies $\Box H \wedge \Box\Diamond\neg J$. However, that means that $\tau \not\models \Box H \rightarrow \Diamond\Box J$, which implies that $Q(N) \not\models \Box H \rightarrow \Diamond\Box J$. This is a contradiction. For the direction

$$(Q(N) \models \Diamond\Box H \rightarrow \Diamond\Box J) \Rightarrow (Q(N) \models \Box H \rightarrow \Diamond\Box J)$$

we note that any behavior σ of $Q(N)$ that satisfies $\Box H$ satisfies $\Diamond\Box H$ as well. Therefore, for any σ such that

$$\sigma \models \Box H \wedge \Box\Diamond\neg J$$

the following property holds

$$\sigma \models \Diamond\Box H \wedge \Box\Diamond\neg J$$

Consequently, whenever the conclusion of the implication is false, the hypothesis is false, too. □

Lemma 1 provides us with a safety property H, which can be used to simplify the network invariant $I(N)$. More specifically, we are interested in finding a system $P(1,N)||I(N)$, which specifies at least the same behaviors that $Q(N)$ specifies. The systems $Q(N)$ and $P(1,N)||I(N)$ are not restricted, since any process adjacent to $P(j,N)$, with $j \neq 1$, is communicating with $Q(N)$, providing input values to some processes in $Q(N)$. However, for any value of these inputs a behavior σ of $Q(N)$ that violates $\Box H$ is not a property that can satisfy $\Box H \wedge \Box\Diamond\neg J$. The reason is that $\Box H$ is a safety property that is violated by a finite sequence, whereas $\Box\Diamond\neg J$ can only be satisfied by an infinite behavior.

In general to find whether ψ holds for $Q(N)$ we only need to check behaviors for which $\Box H$ holds. All other behaviors satisfy ψ trivially. Therefore, $I(N)$ for all inputs should not produce any state that violates H.

In this case we choose $I(N)$ to be the process that writes on the shared variables in SV_I any values that do not violate H. Process $I(N)$ has no inputs and no local variables. It specifies the behaviors that are defined by the projection of $\Box H$ on SV_I, $\mathcal{P}(I(N)) = \mathcal{P}(\Pi_{\mathrm{SV}_I}(\Box H))$. Therefore, all possible behaviors that the system $P(2,N)||...||P(N,N)$ specifies and which satisfy $\Box H$, are specified by $I(N)$

$$\mathcal{P}\left(\Pi_{\mathrm{SV}_I}(P(2,N)||...||P(N,N))\right) \cap \mathcal{P}\left(\Pi_{\mathrm{SV}_I}(\Box H)\right) \subseteq \mathcal{P}(I(N)) \qquad (2)$$

In this case $I(N)$ does not have any inputs. Because all inputs of $P(1,N)$ are the SV_I variables, which are outputs of $I(N)$, the system $P(1,N)||I(N)$ is restricted.

For the property at the LHS of (2) we can define a specification $R(N)$ with exactly the same sequences as behaviors. The new system will be the same as $P(2,N)||...||P(N,N)$ with additional conjuncts $\Pi_{\mathrm{SV}_I}(H)$ in the initial condition and $\Pi_{\mathrm{SV}_I}(H')$ in the next state relation. Formally,

$$\mathcal{P}(R(N)) \triangleq \mathcal{P}(P(2,N)||...||P(N,N)) \cap \mathcal{P}\left(\Pi_{\mathrm{SV}_I}(\Box H)\right)$$

Moreover, we assume for all inputs of $R(N)$ that they are local variables to each process and can take any value. Since both $R(N)$ and $I(N)$ have no inputs and have exactly the same set of observable variables

$$R(N) \sqsubseteq_M I(N)$$

and because of that

$$(P(1,N)\|R(N))_R \sqsubseteq (P(1,N)\|I(N))_R$$

The system $P(1,N)\|I(N)$ can be seen in the appendix (Algorithm 2).

5.4 Application of the technique

We apply the proposed technique on the system $P(1,N)\|I(N)$ and obtain the abstract system S_a. The abstract system has 8000 states and TLC takes 2 minutes to prove the property. The concrete system with $N = 4$ has 216800 states and TLC takes 40 minutes to prove its correctness. In the appendix (Algorithm 3) the specification of the abstract system in TLA+ is displayed. The variable `SetOfLMinus1Neighbors`, which was used by the network invariant in the concrete system to leave the nodes at distance $l-1$ unchanged, is removed using data abstraction.

6 Conclusions

In this paper we presented a new abstraction technique for the verification of parameterized systems using model checking. The technique imposes less restrictions on the correctness property. We used the technique to prove a persistence temporal property of a self-stabilizing spanning-tree construction algorithm. In the future we plan to work on ways to automate the application of the abstraction technique and remove some of the restrictions on the type of systems that this technique can be applied to.

References

1. ABADI, M., AND LAMPORT, L. The existence of refinement mappings. *Theoretical Computer Science 82*, 2 (1991).
2. ARONS, T., PNUELI, A., RUAH, S., XU, J., AND ZUCK, L. D. Parameterized verification with automatically computed inductive assertions. In *CAV '01: Proceedings of the International Conference on Computer Aided Verification* (London, UK, 2001), Springer-Verlag, pp. 221–234.
3. ARORA, A., AND GOUDA, M. Distributed reset. *IEEE Transactions on Computers 43*, 9 (1994).
4. BAUKUS, K., LAKHNECH, Y., AND STAHL, K. Verification of parameterized protocols. *Journal of Universal Computer Science 7*, 2 (2001), 141–158.
5. CLARKE, E., TALUPUR, M., AND VEITH, H. Environment abstraction for parameterized verification. In *VMCAI 2006: Proceedings of the th International Conference on Verification, Model Checking, and Abstract Interpretation* (London, UK, 2006), Springer-Verlag, pp. 126–141.
6. EMERSON, E. A., AND KAHLON, V. Reducing model checking of the many to the few. In *CADE-17: Proceedings of the 17th International Conference on Automated Deduction* (London, UK, 2000), Springer-Verlag, pp. 236–254.
7. FANG, Y., PITERMAN, N., PNUELI, A., AND ZUCK, L. Liveness with invisible ranking. *International Journal on Software Tools for Technology Transfer (STTT) 8*, 3 (2006), 261–279.

8. JHALA, R., AND MCMILLAN, K. Array abstractions from proofs. In *CAV '07, accepted for publication* (2007).
9. KESTEN, Y., AND PNUELI, A. Control and data abstraction: the cornerstones of practical formal verification. *International Journal on Software Tools for Technology Transfer (STTT) 2*, 4 (2000).
10. KESTEN, Y., AND PNUELI, A. Verification by augmented finitary abstraction. *Information and Computation 163*, 1 (2000), 203–243.
11. KESTEN, Y., PNUELI, A., SHAHAR, E., AND ZUCK, L. D. Network invariants in action. In *CONCUR '02: Proceedings of the International Conference on Concurrency Theory* (London, UK, 2002), Springer-Verlag, pp. 101–115.
12. KURSHAN, R. P., AND MCMILLAN, K. L. A structural induction theorem for processes. *Information and Computation 117*, 1 (1995), 1–11.
13. LAMPORT, L. *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers*. Addison-Wesley Professional, 2002.
14. LYNCH, N. *Distributed Algorithms*. Morgan Kaufmann Publishers, 1996.
15. MCMILLAN, K. L. Verification of an implementation of tomasulo's algorithm by compositional model checking. In *CAV* (1998), A. J. Hu and M. Y. Vardi, Eds., vol. 1427 of *Lecture Notes in Computer Science*, Springer, pp. 110–121.
16. MCMILLAN, K. L. A methodology for hardware verification using compositional model checking. *Science of Computer Programming 37*, 1–3 (2000), 279–309.
17. PNUELI, A., RUAH, S., AND ZUCK, L. D. Automatic deductive verification with invisible invariants. In *TACAS 2001: Proceedings of the 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems* (London, UK, 2001), Springer-Verlag, pp. 82–97.
18. PNUELI, A., XU, J., AND ZUCK, L. Liveness with (0,1,infinity)-counter abstraction. In *CAV '02: Proceedings of the International Conference on Computer Aided Verification* (London, UK, 2002), Springer-Verlag, pp. 107–122.

A Proof of correctness

In this section we prove the correctness of the proposed technique. It is enough to prove that $S_c \sqsubseteq S_a$. As before S_c stands for the concrete system $(P(1,N) \| I(N))_R$. We are going to use the theory of refinement mappings [1] to prove that the abstraction is correct. More specifically, we first define system S_c^π by augmenting S_c with a prophecy variable π and then we define a refinement mapping from S_c^π to S_a.

A.1 System with a prophecy variable

In this section we describe how we obtain the $S_c^\pi = \langle \Sigma^\pi, F^\pi, \mathcal{N}^\pi, \mathcal{L}^\pi \rangle$ system from $S_c = \langle \Sigma, F, \mathcal{N}, \mathcal{L} \rangle$ for each $N \in \mathbb{N}$.

$T\pi 1$ The state space $\Sigma^\pi = \Sigma \times 2..N$. This implies that $\pi \in 2..N$ in all reachable states of S_c^π.

$T\pi 2$ The set of initial states $F^\pi = F \times 2..N$. As a consequence, π can have any value in $2..N$ initially.

$T\pi 3$ From the set of actions A of S_c, based on which $\mathcal{N}$ is defined, we are going to derive the set of actions A^π, which define the next state relation $\mathcal{N}^\pi$ of S_c^π. There are four types of actions in A^π:

$T\pi 3.0$ A^π includes $N-1$ actions of the form

$$\begin{aligned} \alpha_j^\pi \triangleq\ & \wedge\ \pi' = j \\ & \wedge\ \text{UNCHANGED } \langle \mathit{all\_other\_variables} \rangle \end{aligned}$$

where $j \in 2..N$ and $sv[j]$ is the next element in SV_I that is going to be read or written. These actions are always enabled.

$T\pi 3.1$ For any action α that does not modify or read any of the variables in SV_I, a new action is defined, which has as an effect the conjunction of the effect of α and the condition $\pi' = \pi$. The new action is included in A^π:

$$\begin{aligned} \alpha^\pi \triangleq\ & \wedge\ \mathrm{prec}(\alpha) \\ & \wedge\ \mathrm{eff}(\alpha) \\ & \wedge\ \pi' = \pi \end{aligned}$$

$T\pi 3.2$ For any action α that reads one of the variables in SV_I, a new action α^π is defined and added to A^π. Let the accessed variable be $sv[j]$. The precondition of α^π is the conjunction of the precondition of α and $\pi = j$. The effect is the same as the effect of α and π is left unchanged

$$\begin{aligned} \alpha^\pi \triangleq\ & \wedge\ \mathrm{prec}(\alpha) \\ & \wedge\ \pi = j \\ & \wedge\ \mathrm{eff}(\alpha) \\ & \wedge\ \pi' = \pi \end{aligned}$$

$T\pi 3.3$ For any action α that modifies one of the variables in SV_I, a new action α^π is defined and added to A^π. Let the modified variable be $sv[j]$. The precondition of α^π is the conjunction of the precondition of α and $\pi = j$. The effect is the same as the effect of α and π is left unchanged

$$\begin{aligned}\alpha^\pi \triangleq\ & \land\ \text{prec}(\alpha)\\ & \land\ \pi = j\\ & \land\ \text{eff}(\alpha)\\ & \land\ \pi' = \pi\end{aligned}$$

$T\pi3.4$ For any action α that reads a variable $sv[g(e)]$, with $g(e) \in 1..N$ and $e \in \Sigma_{nsv}$, let $\text{eff}(\alpha) = h(e, e', sv[g(e)]$. We define action α^π as follows

$$\begin{aligned}\alpha^\pi \triangleq\ & \lor \land\ \text{prec}(\alpha)\\ & \land\ g(e) = 1\\ & \land\ h(e,e',sv[g(e)])\\ & \land\ \pi' = \pi\\ & \lor \land\ \text{prec}(\alpha)\\ & \land\ \pi = g(e)\\ & \land\ h(e,e',sv[g(e)])\\ & \land\ \pi' = \pi\end{aligned}$$

The new action $\alpha^\pi$is added to A^π.

$T\pi4$ We define liveness condition L^π built by the following algorithm:

(a) $L^\pi = \text{TRUE}$

(b) For each action α^π constructed from α using any of the rules $T\pi3.1 - T\pi3.4$,

i. if $\alpha \in \mathcal{W}$, then

$$L^\pi = L^\pi \land \left((\Box\Diamond\neg\text{prec}(\alpha)) \lor \left(\Box\Diamond\left(\text{eff}(\alpha) \land s' \neq s\right)\right)\right)$$

ii. if $\alpha \in \mathcal{S}$, then

$$L^\pi = L^\pi \land \left((\Diamond\Box\neg\text{prec}(\alpha)) \lor \left(\Box\Diamond\left(\text{eff}(\alpha) \land s' \neq s\right)\right)\right)$$

where s is the projection of state s^π on Σ.

(c) If L_e is the conjunction of all justice and compassion requirements of S_c then

$$L^\pi = L^\pi \land L_e$$

Note that $L \equiv L^\pi$ by construction.

In order to prove that S_c and S_c^π define the same externally visible property, we need to prove conditions P1-P6, as Abadi and Lamport found [1]. For convenience, we repeat the conditions P1-P6 as described in their paper

P1. $\Sigma^\pi \subseteq \Sigma \times \Sigma_\pi$ for some set Σ_π

P2. (a) $\Pi_{[\pi]}(F^\pi) \subseteq F$

(b) For all $(s,\pi) \in \Pi^{-1}_{[\pi]}(F)$ there exists $\pi_0, \pi_1, ...\pi_n = \pi$ such that $(s,\pi_0) \in F^\pi$ and, for $0 \leq i < n$, $\langle(s,\pi_i),(s,\pi_{i+1})\rangle \in \mathcal{N}^\pi$

P3. If $\langle(s,\pi),(s',\pi')\rangle \in \mathcal{N}^\pi$ then $\langle s,s'\rangle \in \mathcal{N}$ or $s = s'$.

P4. If $\langle s,s'\rangle \in \mathcal{N}$ and $(s',\pi') \in \Sigma^\pi$ then there exist $\pi, \pi'_0, \ldots, \pi'_{n-1}, \pi'_n = \pi'$ such that $\langle(s,\pi),(s',\pi'_0)\rangle \in \mathcal{N}^\pi$ and, for $0 \leq i < n$: $\langle(s',\pi'_i),(s',\pi'_{i+1})\rangle \in \mathcal{N}^\pi$.

P5. $L^\pi = \Pi^{-1}_{[\pi]}(L)$.

P6. For all $s \in \Sigma$ the set $\Pi^{-1}_{[\pi]}(s)$ is finite and nonempty.

The set $\Pi^{-1}_{[\pi]}(s)$ is defined as the set of all states $t = (s,\pi)$ for which $\Pi_{[\pi]}(t) = s$.

Lemma 2. *Systems S_c and S^{π}_c define the same externally visible property.*

PROOF SKETCH: Using the way S^{π}_c was constructed (properties $T\pi0$-$T\pi4$), we prove premises P1-P6. Based on Proposition 5 of Abadi and Lamport's paper, the lemma is implied from these premises.
PROOF:
1. $T\pi1$ implies P1
   1.1. Because of $T\pi1$ it holds that $\Sigma^{\pi} = \Sigma \times 2..N$. If we substitute Σ_{π} for $2..N$, the following condition becomes true
   $$\Sigma^{\pi} = \Sigma \times \Sigma_{\pi} \Rightarrow \Sigma^{\pi} \subseteq \Sigma \times \Sigma_{\pi}$$
   □
2. $T\pi2$ implies P2(a) and P2(b)
   PROOF SKETCH: We prove that $T\pi2$ satisfies a much stronger property, i.e.,
   $$F^{\pi} = \Pi^{-1}_{[\pi]}(F)$$
   This property is the same as P2$'$ in Abadi and Lamport's paper [1].
   PROOF:
   $$\begin{aligned}\Pi^{-1}_{[\pi]}(F) &= \{(s,\pi) | s \in F \wedge (s,\pi) \in \Sigma^{\pi} \wedge \Pi^{-1}_{[\pi]}(s,\pi) = s\}\\ &= \{(s,\pi) | s \in F \wedge s \in \Sigma \wedge \pi \in 2..N\}\\ &= F \times 2..N\\ &= F^{\pi}\end{aligned}$$
3. $T\pi3.0 - T\pi3.4$ imply P3
   PROOF SKETCH: We prove P3 by doing a case based analysis of all actions in A^{π}, using the properties $T\pi3.0 - T\pi3.4$. For each action $\alpha^{\pi} \in A^{\pi}$ we prove that all possible transitions $\langle (s,\pi), \alpha^{\pi}, (s',\pi') \rangle$ satisfy $\langle s, s' \rangle \in \mathcal{N}$ or $s = s'$.
   3.1. CASE: α^{π} is constructed by rule $T\pi3.0$, then $s = s'$
   By definition of α^{π} (rule $T\pi3.0$) for any (s,π) in Σ^{π} such that $\langle (s,\pi), \alpha^{\pi}, (s',\pi') \rangle$ is a transition of S^{π}_c, it holds that $s = s'$. This is guaranteed by the second conjunct of the definition.
   3.2. CASE: α^{π} is constructed by rule $T\pi3.1$, then $\langle s, s' \rangle \in \mathcal{N}$
   Let α^{π} be an action constructed by rule $T\pi3.1$. For any transition $\langle (s,\pi), \alpha^{\pi}, (s',\pi') \rangle$ of system S^{π}_c we know that there is an action α of S_c with the same precondition and the same effect on the Σ-part of the state. Therefore, α is enabled at s and can produce s', when executed at s. This implies $\langle s, s' \rangle \in \mathcal{N}$.
   3.3. CASE: α^{π} is constructed by rule $T\pi3.2$, then $\langle s, s' \rangle \in \mathcal{N}$
   Let α^{π} be an action constructed by rule $T\pi3.2$. For any transition $\langle (s,\pi), \alpha^{\pi}, (s',\pi') \rangle$ of system S^{π}_c we know that there is an action α of system S_c such that
   $$\text{prec}(\alpha^{\pi}) \rightarrow \text{prec}(\alpha)$$
   $$\text{eff}(\alpha^{\pi}) \rightarrow \text{eff}(\alpha)$$
   Therefore, α is enabled at s and can produce s', when executed at s. This implies $\langle s, s' \rangle \in \mathcal{N}$.
   3.4. CASE: α^{π} is constructed by rule $T\pi3.3$, then $\langle s, s' \rangle \in \mathcal{N}$
   We can apply the same reasoning as in the Case 3.3.
   3.5. CASE: α^{π} is constructed by rule $T\pi3.4$, then $\langle s, s' \rangle \in \mathcal{N}$

Let α^π be an action constructed by rule $T\pi 3.4$. For any transition $\langle (s,\pi), \alpha^\pi, (s',\pi') \rangle$ of system S_c^π we know that there is an action α of system S_c such that

$$\begin{array}{rl} \alpha^\pi & \rightarrow \quad \begin{array}{l} \lor\ \text{prec}(\alpha) \land g(e) = 1 \land h(e, e', sv[g(e)]) \land \pi' = \pi \\ \lor\ \text{prec}(\alpha) \land \pi = g(e) \land h(e, e', sv[g(e)]) \land \pi' = \pi \end{array} \\ & \xrightarrow{g(e) \in 1..N} \big(\text{prec}(\alpha) \land h(e, e', sv[g(e)])\big) \\ & \rightarrow \quad \alpha \end{array}$$

Since α is expressed only on s, we have

$$\langle (s,\pi), (s',\pi') \rangle \models \alpha \rightarrow \langle s, s' \rangle \models \alpha$$

This implies $\langle s, s' \rangle \in \mathcal{N}$.

4. $T\pi 3.0 - T\pi 3.4$ imply P4

PROOF SKETCH: We prove P4 by doing a case based analysis of all actions in A, using the properties $T\pi 3.0 - T\pi 3.4$. For each $\langle s, s' \rangle \in \mathcal{N}$, we prove based on the action α that caused the transition $\langle s, \alpha, s' \rangle$ that there exist a sequence of actions in A^π that satisfy the premises of P4.

4.1. CASE: $\exists \alpha \in A$:

1. $\langle s, \alpha, s' \rangle$ is a transition of S_c
2. α does not read or modify any element of SV_I

By rule $T\pi 3.1$ there exists action $\alpha^\pi \in A^\pi$ that is enabled at s, for any value of π, and can produce s', if executed at s. The value of π remains unchanged by the execution of α^π. This implies $\langle (s,\pi'), (s',\pi') \rangle$ belongs to $\mathcal{N}^\pi$.

4.2. CASE: $\exists \alpha \in A$:

1. $\langle s, \alpha, s' \rangle$ is a transition of S_c
2. α reads or modifies an element of SV_I

Let $sv[j]$ be the element read or modified, where $j \in 2..N$. For each $(s',\pi') \in \Sigma^\pi$ there exists transition $\langle (s',j), \alpha^\pi, (s',\pi') \rangle$. The reason is that actions constructed by rule $T\pi 3.0$ are always enabled, so the action α^π is enabled in s'. Therefore, it holds $\langle (s',j), (s',\pi') \rangle \in \mathcal{N}^\pi$. Because of rules $T\pi 3.2, T\pi 3.3$ we know that there exists action in A^π, whose precondition is the conjunction of the precondition of α and $\pi = j$. Therefore, this action is enabled in state (s,j). Moreover, its effect is the effect of α and it leaves π unchanged. This implies that $\langle (s,j), (s',j) \rangle \in \mathcal{N}^\pi$.

4.3. CASE: $\exists \alpha \in A$:

1. $\langle s, \alpha, s' \rangle$ is a transition of S_c
2. $\alpha = \text{prec}(\alpha) \land h(e, e', sv[g(e)])$, where $g(e) \in 1..N$

In this case there exists action α^π in A^π defined by rule $T\pi 3.4$. Since $\langle s, s' \rangle \models \alpha$, in state s we have that $g(e) = 1$ or $g(e) \in 2..N$. If $g(e) = 1$, then

$$\langle s, s' \rangle \models \big(g(e) = 1 \land h(e, e', sv[g(e)]) \land \text{prec}(\alpha) \big)$$

For every $(s',\pi') \in \Sigma^\pi$, state (s,π), with $\pi' = \pi$, is in Σ^π and

$$\langle (s,\pi), (s',\pi') \rangle \models \big(g(e) = 1 \land h(e, e', sv[g(e)]) \land \text{prec}(\alpha) \land \pi' = \pi \big) \overset{T\pi 3.4}{\Rightarrow} \quad (3)$$
$$\langle (s,\pi), (s',\pi') \rangle \models \alpha^\pi$$

If $g(e) \in 2..N$, then let $j = g(e)$ with $j \in 2..N$. Then for all $(s',\pi') \in \Sigma^\pi$, there is action α_0^π created by rule $T\pi 3.0$ such that $\langle (s',j), (s',\pi') \rangle \models \alpha_0^\pi$. Then state (s,j) belongs to Σ^π and satisfies $g(e) = j$. Therefore,

$$\langle (s,j), (s',j) \rangle \models \big(g(e) = j \land \pi = j \land \text{prec}(\alpha) \land h(e, e', sv[g(e)]) \land \pi' = \pi \big)$$
$$\Rightarrow \langle (s,j), (s',j) \rangle \models \alpha^\pi$$

5. $T\pi4$ implies P5

PROOF SKETCH: The argument is based on the equivalence of L and L^{π}.

The L^{π} property is expressed on variables in Σ and is equivalent to L. Therefore, for every infinite behavior σ that satisfies L^{π}, $\Pi_{[\pi]}(\sigma)$ must satisfy L. Moreover, for each behavior σ satisfying L, all corresponding behaviors produced by the machine of S_c^{π} must satisfy L^{π}.

6. P6 is valid by construction of the state space

For each N and each state s, there are at most $N-1$ values for π, so at most $N-1$ elements in the set $\Pi_{[\pi]}^{-1}(s)$. Each state s which is reachable for S_c has at least one corresponding state (s,π) which is reachable for S_c^{π}. Therefore, $\Pi_{[\pi]}^{-1}(s)$ is finite and non-empty for each N.

□

The next lemma states that there exists a refinement mapping from the system S_c^{π} with the augmented prophecy variable to the abstract system constructed using rules $T1-T4$. We represent the state $s_c=(s,\pi)\in\Sigma_c^{\pi}$ as (e,sv,π), where e is the part of the state without the shared variable sv ($e\in\Sigma_{nsv}$). We consider $(e,sv[1])$ the external part of the state, even though the external part of the state may be only a part of $(e,sv[1])$. Extending to those cases should be straightforward. The shared variable is sv. In system S_c^{π} the shared variable ranges over $[1..N\rightarrow \text{FS}]$. The state of the abstract system can be represented as $s_a=(e_a,sv_a)$, where $sv_a\in[1..2\rightarrow \text{FS}]$. The refinement mapping we consider is the function $f:\Sigma_c^{\pi}\rightarrow\Sigma_a$ defined as

$$\text{let } s_c \triangleq (e,sv,\pi) \text{ in}$$
$$\forall\ s_c\in\Sigma^{\pi}: f(s_c)\triangleq(e,\langle sv[1],sv[\pi]\rangle)$$

By definition f preserves the external part and sets the first element of sv_a to $sv[1]$ and the second to $sv[\pi]$, i.e.,

$$sv_a[1]=sv[1]$$
$$sv_a[2]=sv[\pi]$$

In order to prove that f is a refinement mapping, it is sufficient to show that f satisfies conditions $R1-R4$ as shown in Abadi and Lamport's paper [1]. We list the $R1-R4$ conditions from that paper for the reader's convenience.

R1. For all $s_c\in\Sigma^{\pi}:\Pi_E(f(s_c))=\Pi_E(s_c)$.
R2. $f(F^{\pi})\subseteq F_a$.
R3. If $\langle s_c,t_c\rangle\in\mathcal{N}^{\pi}$ then $\langle f(s_c),f(t_c)\rangle\in\mathcal{N}_a$ or $f(s_c)=f(t_c)$.
R4. $f(\mathcal{P}^{\pi})\subseteq L_a$, where $\mathcal{P}^{\pi}$ is the property defined by S_c^{π}.

Lemma 3. *Function f is a refinement mapping from S_c^{π} to S_a*

PROOF SKETCH: We prove the lemma by showing that f satisfies properties $R1-R4$.
PROOF:
1. R1 is satisfied by the definition of f.

PROOF:Let $s_c \triangleq (e, sv, \pi)$ be any element in Σ^π, then

$$\forall s_c \in \Sigma^\pi : \wedge\ \Pi_E(s_c) = (e, sv[1]) \quad (4)$$

$$\wedge\ f(s_c) = (e, \langle sv[1], sv[\pi]\rangle) \quad (5)$$

$(5) \Rightarrow \forall s_c \in \Sigma^\pi : \Pi_E(f(s_c)) = \Pi_E((e, \langle sv[1], sv[\pi]\rangle)) = (e, sv[1]) = \Pi_E(s_c)$ □

2. R2 is satisfied by construction of S_a.
PROOF:By construction $F_a = \Pi_E(F^\pi) \times \left({F^\pi}_{sv[1]} \times {F^\pi}_{sv[2..N]}\right)$, where ${F^\pi}_{sv[1]}$, ${F^\pi}_{sv[2..N]}$ are the sets of possible initial values of $sv[1]$ and possible initial values of the elements in SV_I, respectively. For any state $s_c = (e, sv, \pi)$ it holds

$$\forall s_c \in F^\pi : f(s_c) = (e, \langle sv[1], sv[\pi]\rangle)$$

Since $s_c \in F^\pi$ and $\pi \in 2..N$, $e \in \Pi_E(F^\pi)$, $sv[1] \in {F^\pi}_{sv[1]}$, and $sv[\pi] \in {F^\pi}_{sv[2..N]}$. Therefore, $f(s_c) \in F_a$. We proved that $\forall s_c \in F^\pi : f(s_c) \in F_a$. This implies that $f(F^\pi) \subseteq F_a$.□

3. Rules T0−T4 imply R3 is satisfied by f.
PROOF:For any $\langle s, t\rangle \in \mathcal{N}^\pi$ there must exist action $\alpha^\pi \in A^\pi$ created by one of the rules $T\pi3.0$-$T\pi3.4$, such that $\langle s, \alpha^\pi, t\rangle$ is a transition of S^π_c. There are four cases based on which rule was used for the creation of α^π.

3.1. CASE: Action α^π is created by rule $T\pi3.0$. Then there exists an action $\tilde{\alpha}$ created by rule $T0$, such that $\langle f(s), \tilde{\alpha}, f(t)\rangle$ is a transition of S_a.
PROOF: Action α^π modifies only the value of π (by construction). After the action $sv[\pi']$ can have any value v in FS. There is an action $\tilde{\alpha} \in \tilde{A}$ created by $T0$, which leaves all variables unchanged and sets $sv_a[2]' = v$. Since $\tilde{\alpha}$ is always enabled, it is enabled in $f(s)$. Its effect is to leave e and $sv_a[1]$ unchanged and set $sv_a[2]' = v = sv[\pi']$. Therefore, the next state is $f(t)$. Consequently, $\langle f(s), \tilde{\alpha}, f(t)\rangle$ is a transition of S_a.□

3.2. CASE: Action α^π is created by rule $T\pi3.1$. Then there exists an action $\tilde{\alpha}$ created by rule $T1$, such that $\langle f(s), \tilde{\alpha}, f(t)\rangle$ is a transition of S_a.
PROOF: Action α^π can modify only the values of e, $sv[1]$ to e', $sv[1]'$ (by construction). In order for α^π to be in A^π, there must exist α an action of S with the same precondition and effect on $e, sv[1]$. Because of rule $T1$ an action $\tilde{\alpha}$ exists in $\tilde{A}$ with the same precondition and effect on $e, sv[1]$. Therefore,

$$\langle (e, \langle sv_a[1], sv_a[2]\rangle), \tilde{\alpha}, (e', \langle sv_a[1]', sv_a[2]\rangle)\rangle = \langle f(s), \tilde{\alpha}, f(t)\rangle$$

is a transition of S_a.□

3.3. CASE: Action α^π is created by rule $T\pi3.2$. Then there exists an action $\tilde{\alpha}$ created by rule $T2$, such that $\langle f(s), \tilde{\alpha}, f(t)\rangle$ is a transition of S_a.
PROOF: In this case α^π accesses variable $sv[j]$ with $j \in 2..N$. Action α^π cannot modify any of the elements of sv because of the restriction $\Lambda1$. Moreover, by construction α^π cannot modify π. In order for α^π to be enabled in s, $\pi = j$. So, this action modifies e to e' by accessing $sv[\pi]$. For this action to be constructed by rule $T\pi3.2$, there must exist action α of S. Therefore, because of rule $T2$, there must exist $\tilde{\alpha} \in \tilde{A}$ with the same precondition and effect as α, except that $sv[j]$ is replaced by $sv_a[2]$. However, in $f(s)$ by the definition of f, $sv_a[2] = sv[\pi]$. Consequently, $\tilde{\alpha}$ has the same precondition and effect on e as α^π. Since neither $sv_a[1]$ nor $sv_a[2]$ changes, the next state after the execution of $\tilde{\alpha}$ in $f(s)$ is $f(t)$. This means that $\langle f(s), \tilde{\alpha}, f(t)\rangle$ is a transition of S_a.□

3.4. CASE: Action α^π is created by rule $T\pi 3.3$. Then there exists an action $\tilde{\alpha}$ created by rule $T3$, such that $\langle f(s), \tilde{\alpha}, f(t)\rangle$ is a transition of S_a.

PROOF: Action α^π modifies element $sv[\pi]$. Moreover, because of the restriction $\Lambda 1$, it cannot read any sv variable. Therefore, the new value is a function of the state e. It must be independent of π, since the effect of α^π is the same as the effect of α based on which α^π was created. Based on α and because of rule $T3$ $\tilde{\alpha}$ is created. Action $\tilde{\alpha}$ has the same precondition and the same effect except that instead of $sv[j]$, $sv_a[2]$ is modified. Therefore, $\tilde{\alpha}$ is enabled in $f(s)$. Moreover, for any value that α^π can assign to $sv[\pi]$ as a function of e, $\tilde{\alpha}$ can assign that value to $sv_a[2]$ as a function of e. Changes to e are the same for the two actions as they are taken from α. Consequently, $\langle (e, \langle sv[1], sv[\pi]\rangle), \tilde{\alpha}, (e', \langle sv[1], sv[\pi]'\rangle)\rangle = \langle f(s), \tilde{\alpha}, f(t)\rangle$ is a transition of S_a.□

3.5. CASE: Action α^π is created by rule $T\pi 3.4$. Then there exists an action $\tilde{\alpha}$ created by rule $T4$, such that $\langle f(s), \tilde{\alpha}, f(t)\rangle$ is a transition of S_a.

PROOF: In order for α^π to exist in A^π, there must be an action $\alpha \in A$, such that $\alpha = \text{prec}(\alpha) \wedge h(e, e', sv[g(e)])$ for a state function $g(e) \in 1..N$. Then there exists action $\tilde{\alpha} \in \tilde{A}$ defined by rule T4. For a state pair $\langle s, t\rangle$ satisfying α^π we have

$$\langle s,t\rangle \models \left(\begin{array}{l} \vee\ \text{prec}(\alpha) \wedge \pi' = \pi \wedge h(e,e',sv[1]) \wedge g(e) = 1 \\ \vee\ g(e) = \pi \wedge \text{prec}(\alpha) \wedge h(e,e',sv[g(e)]) \wedge \pi' = \pi \end{array}\right) \Rightarrow$$

$$\left(\begin{array}{l} \vee\ \langle s,t\rangle \models (\text{prec}(\alpha) \wedge \pi' = \pi \wedge h(e,e',sv[1]) \wedge g(e) = 1\,) \\ \vee\ \langle s,t\rangle \models (\,g(e) = \pi \wedge \text{prec}(\alpha) \wedge h(e,e',sv[g(e)]) \wedge \pi' = \pi\,) \end{array}\right) \overset{\Lambda 1}{\Rightarrow}$$

$$\left(\begin{array}{l} \vee\ \langle s,t\rangle \models (\text{prec}(\alpha) \wedge \pi' = \pi \wedge h(e,e',sv[1]) \wedge g(e) = 1 \wedge sv' = sv\,) \\ \vee\ \langle s,t\rangle \models \left(\begin{array}{l} \wedge\ g(e) = \pi \wedge \text{prec}(\alpha) \wedge h(e,e',sv[g(e)]) \\ \wedge\ \pi' = \pi \wedge sv' = sv \wedge sv[g(e)] = sv[\pi] \end{array}\right) \end{array}\right) \Rightarrow$$

$$\left(\begin{array}{l} \vee\ \langle s,t\rangle \models (\text{prec}(\alpha) \wedge h(e,e',sv[1]) \wedge g(e) = 1 \wedge sv[\pi']' = sv[\pi]\,) \\ \vee\ \langle s,t\rangle \models \left(\begin{array}{l} \wedge\ g(e) = \pi \wedge \text{prec}(\alpha) \wedge h(e,e',sv[g(e)]) \\ \wedge\ sv[\pi']' = sv[\pi] \wedge sv[g(e)] = sv[\pi] \end{array}\right) \end{array}\right) \Rightarrow$$

$$\left(\begin{array}{l} \vee\ \langle f(s),f(t)\rangle \models (\text{prec}(\alpha) \wedge h(e,e',sv_a[1]) \wedge g(e) = 1 \wedge sv_a[2]' = sv_a[2]\,) \\ \vee\ \langle f(s),f(t)\rangle \models (\,g(e) \neq 1 \wedge \text{prec}(\alpha) \wedge h(e,e',sv_a[2]) \wedge sv_a[2]' = sv[2]\,) \end{array}\right) \Rightarrow$$

$$\langle f(s),f(t)\rangle \models \tilde{\alpha}$$

Therefore, $\langle f(s), \tilde{\alpha}, f(t)\rangle$ is a transition of S_a.□

□

4. R4 is satisfied by construction of L_a.

PROOF:We prove that R4 holds by case analysis on the type of liveness conditions.

4.1. CASE: If σ^π belongs to $\mathcal{P}(S^\pi)$, then $f(\sigma^\pi)$ satisfies all weak fairness conditions of L_a.

PROOF:Suppose σ^π belongs to $\mathcal{P}(S^\pi)$, but $f(\sigma^\pi)$ violates the weak fairness condition of an action $\tilde{\alpha}$. We prove that this leads to contradiction. In order for $\tilde{\alpha}$ to belong to $\tilde{\mathcal{W}}$, there must exist α in A, such that $\alpha \in \mathcal{W}$. Since $\sigma^\pi \in \mathcal{P}(S^\pi)$, the behavior σ^π must satisfy the weak fairness condition of α,

$$\sigma^\pi \models (\Box\Diamond\neg\text{prec}(\alpha)) \vee (\Box\Diamond\langle\text{eff}(\alpha)\rangle)$$

4.1.1. CASE: If $\sigma^\pi \models \Box\Diamond\neg\text{prec}(\alpha)$, then $f(\sigma^\pi) \models \Box\Diamond\neg\text{prec}(\tilde{\alpha})$.

PROOF:Because of rules T1-T3 and condition $\Lambda 3$, it holds that $\text{prec}(\alpha) \leftrightarrow \text{prec}(\tilde{\alpha})$. This condition holds for any T4 action $\tilde{\alpha}$ as well, since in this case

$$\text{prec}(\tilde{\alpha}) \leftrightarrow (\text{prec}(\alpha) \wedge g(e) = 1) \vee (\text{prec}(\alpha) \wedge g(e) \neq 1) \leftrightarrow \text{prec}(\alpha)$$

Moreover, because of condition $\Lambda 3$, the assertion $\text{prec}(\alpha)$ is a function of only the e part of the state. Since the two sequences σ^π and $f(\sigma^\pi)$ agree on e in each state, for all i with $i \geq 0$

$$(\sigma^\pi, i) \models \neg\text{prec}(\alpha) \Rightarrow (f(\sigma^\pi), i) \models \neg\text{prec}(\tilde{\alpha})$$

Consequently,

$$\sigma^\pi \models \Box\Diamond\neg\text{prec}(\alpha) \Rightarrow f(\sigma^\pi) \models \Box\Diamond\neg\text{prec}(\tilde{\alpha})$$

□

4.1.2. CASE: If $\sigma^\pi \models \Box\Diamond\langle\text{eff}(\alpha)\rangle$, then $f(\sigma^\pi) \models \Box\Diamond\langle\text{eff}(\alpha)\rangle$.

PROOF:For the proof we need to consider 4 cases, depending on the rule that was used to generate $\tilde{\alpha}$ from α.

4.1.2.1. CASE: If $\tilde{\alpha}$ was generated from α by using rule T1, then $\sigma^\pi \models \Box\Diamond\langle\text{eff}(\alpha)\rangle$ implies $f(\sigma^\pi) \models \Box\Diamond\langle\text{eff}(\tilde{\alpha})\rangle$.

PROOF:In this case $\text{eff}(\alpha) \leftrightarrow \text{eff}(\tilde{\alpha})$ by construction. Moreover, since $\text{eff}(\alpha)$ is a function of only the e part of the behavior, for all $i \geq 0$ it holds

$$((\sigma^\pi, i), (\sigma^\pi, i+1)) \models \langle\text{eff}(\alpha)\rangle \Rightarrow$$
$$((f(\sigma^\pi), i), (f(\sigma^\pi), i+1)) \models \langle\text{eff}(\tilde{\alpha})\rangle$$

Therefore,

$$\sigma^\pi \models \Box\Diamond\langle\text{eff}(\alpha)\rangle \Rightarrow f(\sigma^\pi) \models \Box\Diamond\langle\text{eff}(\tilde{\alpha})\rangle$$

□

4.1.2.2. CASE: If $\tilde{\alpha}$ was generated from α by using rule T2, then $\sigma^\pi \models \Box\Diamond\langle\text{eff}(\alpha)\rangle$ implies $f(\sigma^\pi) \models \Box\Diamond\langle\text{eff}(\tilde{\alpha})\rangle$.

PROOF:Assume that $\tilde{\alpha}$ accesses variable $sv[j]$. It holds that

$$\sigma^\pi \models \Box\Diamond\langle\text{eff}(\alpha)\rangle \Rightarrow \sigma^\pi \models \Box\Diamond(\langle\text{eff}(\alpha)\rangle \wedge \pi = j) \vee \sigma^\pi \models \Box\Diamond(\langle\text{eff}(\alpha)\rangle \wedge \pi \neq j)$$

We prove that either case implies $f(\sigma^\pi) \models \Box\Diamond\langle\text{eff}(\tilde{\alpha})\rangle$.

$\langle 5\rangle 1$. CASE: $\sigma^\pi \models \Box\Diamond(\langle\text{eff}(\alpha)\rangle \wedge \pi = j) \Rightarrow f(\sigma^\pi) \models \Box\Diamond\langle\text{eff}(\tilde{\alpha})\rangle$.

From rule T2 we know that

$$(sv[j] = sv_a[2]) \rightarrow (\text{eff}(\alpha) \leftrightarrow \text{eff}(\tilde{\alpha})) \tag{6}$$

In the states of σ^π in which $\pi = j$, we have $sv[\pi] = sv[j]$. Moreover, for each $i \geq 0$, the value $sv[\pi]$ in state (σ^π, i) equals the value $sv_a[2]$ of state $(f(\sigma^\pi), i)$. Therefore, for every $i \geq 0$, for any value $v \in \text{FS}$:

$$(\sigma^\pi, i) \models (\pi = j \wedge sv[j] = v) \Rightarrow (f(\sigma^\pi), i) \models sv_a[2] = v$$

Then because of (6) and the fact that $\text{eff}(\alpha)$ is a function only of e and $sv[j]$, it holds that

$$\forall i \geq 0 : ((\sigma^\pi, i), (\sigma^\pi, i+1)) \models (\langle\text{eff}(\alpha)\rangle \wedge \pi = j) \Rightarrow$$
$$((f(\sigma^\pi), i), (f(\sigma^\pi), i+1)) \models (\langle\text{eff}(\tilde{\alpha})\rangle)$$

and because of that

$$\sigma^\pi \models \Box\Diamond(\langle\text{eff}(\alpha)\rangle \wedge \pi = j) \Rightarrow f(\sigma^\pi) \models \Box\Diamond\langle\text{eff}(\tilde{\alpha})\rangle$$

□

$\langle 5\rangle 2$. CASE: $\sigma^\pi \not\models \Box\Diamond(\langle\text{eff}(\alpha)\rangle \wedge \pi = j) \Rightarrow f(\sigma^\pi) \models \Box\Diamond\langle\text{eff}(\tilde{\alpha})\rangle$.

Since $\sigma^\pi \models \Box\Diamond\langle\text{eff}(\alpha)\rangle$, there must be actions that simulate the effect of action α infinitely often. Because of restriction $\Lambda 4$a, these actions can only be actions accessing one of the variables in SV_I. Because there is a finite number of actions, there must exist action $\beta \in A$, which accesses variable $sv[k]$, such that, $\sigma^\pi \models \Box\Diamond(\langle \text{eff}(\beta)\rangle \wedge \pi = k \wedge (\text{eff}(\beta) \rightarrow \text{eff}(\alpha)))$. Then there must exist action $\tilde{\beta}$ in $\tilde{A}$. With the same reasoning as for case $\langle 5\rangle 1$, we can prove that $f(\sigma^\pi) \models \Box\Diamond\langle \text{eff}(\tilde{\beta})\rangle$. Suppose $\text{eff}(\alpha) = h(e, e', sv[j])$ and $\text{eff}(\beta) = h(e, e', sv[k])$ are syntactically equivalent logic actions, then $\text{eff}(\tilde{\alpha}) = h(e, e', sv_a[2])$ and $\text{eff}(\tilde{\beta}) = h(e, e', sv_a[2])$. Therefore, $\text{eff}(\tilde{\alpha}) \leftrightarrow \text{eff}(\tilde{\beta})$, which implies that

$$\forall i \geq 0 : ((f(\sigma^\pi), i), (f(\sigma^\pi), i+1)) \models \langle \text{eff}(\tilde{\beta})\rangle \Rightarrow$$
$$((f(\sigma^\pi), i), (f(\sigma^\pi), i+1)) \models \langle \text{eff}(\tilde{\alpha})\rangle$$

Consequently, if α and β are syntactically equivalent

$$\sigma^\pi \models \Box\Diamond(\langle \text{eff}(\alpha)\rangle \wedge \pi \neq j) \Rightarrow f(\sigma^\pi) \models \Box\Diamond\langle \text{eff}(\tilde{\alpha})\rangle$$

What is left is the case in which α and β are not syntactically equivalent. In this case, because of restriction Λ4b, in order for $\text{eff}(\beta) \rightarrow \text{eff}(\alpha)$ to hold, the value of the variable $sv[k]$ accessed by β must be equal to the value of $sv[j]$, in all pairs of states that satisfy $\text{eff}(\beta) \rightarrow \text{eff}(\alpha)$. Because of that in the corresponding states of $f(\sigma^\pi)$, $sv_a[2] = sv[k] = sv[j]$. However, since $(sv[j] = sv_a[2]) \rightarrow (\text{eff}(\alpha) \leftrightarrow \text{eff}(\tilde{\alpha}))$ by construction of $\tilde{\alpha}$,

$$\forall i \geq 0 : ((\sigma^\pi, i), (\sigma^\pi, i+1)) \models (\langle \text{eff}(\beta)\rangle \wedge sv[k] = sv[j] \wedge \pi = k \wedge (\text{eff}(\beta) \rightarrow \text{eff}(\alpha))) \Rightarrow$$
$$((f(\sigma^\pi), i), (f(\sigma^\pi), i+1)) \models \langle \text{eff}(\tilde{\alpha})\rangle$$

Consequently, in the case α and β are not syntactically equivalent

$$\sigma^\pi \models \Box\Diamond(\langle \text{eff}(\alpha)\rangle \wedge \pi \neq j) \Rightarrow f(\sigma^\pi) \models \Box\Diamond\langle \text{eff}(\tilde{\alpha})\rangle$$

□

Since in either case $f(\sigma^\pi) \models \Box\Diamond\langle \text{eff}(\tilde{\alpha})\rangle$ holds, the proof is complete. □

4.1.2.3. CASE: If $\tilde{\alpha}$ was generated from α by using rule T3, then $\sigma^\pi \models \Box\Diamond\langle \text{eff}(\alpha)\rangle$ implies $f(\sigma^\pi) \models \Box\Diamond\langle \text{eff}(\tilde{\alpha})\rangle$.

PROOF:The proof is based again on two cases. The first is $\sigma^\pi \models \Box\Diamond(\langle \text{eff}(\alpha)\rangle \wedge \pi = j)$ and the second $\sigma^\pi \models \Box\Diamond(\langle \text{eff}(\alpha)\rangle \wedge \pi \neq j)$, where $sv[j]$ is the variable written by action α. We show that in both cases $f(\sigma^\pi) \models \Box\Diamond\langle \text{eff}(\tilde{\alpha})\rangle$.

$\langle 5\rangle 1$. CASE: $\sigma^\pi \models \Box\Diamond(\langle \text{eff}(\alpha)\rangle \wedge \pi = j) \Rightarrow f(\sigma^\pi) \models \Box\Diamond\langle \text{eff}(\tilde{\alpha})\rangle$.

From rule T3 we know that

$$\left(sv[j] = sv_a[2] \wedge sv[j]' = sv_a[2]'\right) \rightarrow (\text{eff}(\alpha) \leftrightarrow \text{eff}(\tilde{\alpha})) \qquad (7)$$

In the states of σ^π in which $\pi = j$, we have $sv[\pi] = sv[j]$ and $sv[\pi]' = sv[j]'$. Moreover, for each $i \geq 0$, the value $sv[\pi]$ in state (σ^π, i) equals the value $sv_a[2]$ of state $(f(\sigma^\pi), i)$. Therefore, for every $i \geq 0$, for any value $v \in$ FS:

$$(\sigma^\pi, i) \models (\pi = j \wedge sv[j] = v) \Rightarrow (f(\sigma^\pi), i) \models sv_a[2] = v$$

Then because of (7) and the fact that $\text{eff}(\alpha)$ is a function only of e and $sv[j]$, it holds that

$$\forall i \geq 0 : ((\sigma^\pi, i), (\sigma^\pi, i+1)) \models (\langle \text{eff}(\alpha)\rangle \wedge \pi = j) \Rightarrow$$
$$((f(\sigma^\pi), i), (f(\sigma^\pi), i+1)) \models (\langle \text{eff}(\tilde{\alpha})\rangle)$$

and because of that

$$\sigma^\pi \models \Box\Diamond(\langle \text{eff}(\alpha)\rangle \wedge \pi = j) \Rightarrow f(\sigma^\pi) \models \Box\Diamond\langle \text{eff}(\tilde{\alpha})\rangle$$

□

$\langle 5\rangle 2$. CASE: $\sigma^\pi \not\models \Box\Diamond(\langle \text{eff}(\alpha)\rangle \wedge \pi = j) \Rightarrow f(\sigma^\pi) \models \Box\Diamond\langle \text{eff}(\tilde{\alpha})\rangle$.

Since $\sigma^\pi \models \Box\Diamond\langle \text{eff}(\alpha)\rangle$, there must be actions that simulate the effect of action α infinitely often. Because of restriction Λ4a, these actions can only be actions writing on one of the variables in SV_I. Because there is a finite number of actions, there must exist action $\beta \in A$, which writes variable $sv[k]$, such that, $\sigma^\pi \models \Box\Diamond(\langle \mathrm{eff}(\beta)\rangle \wedge \pi = k \wedge (\mathrm{eff}(\beta) \rightarrow \mathrm{eff}(\alpha)))$. Then there must exist action $\tilde{\beta}$ in $\tilde{A}$. With the same reasoning as for case $\langle 5\rangle 1$, we can prove that $f(\sigma^\pi) \models \Box\Diamond\langle \mathrm{eff}(\tilde{\beta})\rangle$. Note that if $k = j$ then there is a contradiction, since we assumed that it cannot be the case that $\pi = j$ infinitely often when $\langle \mathrm{eff}(\alpha)\rangle$ is satisfied by σ^π. Therefore, $k \neq j$. However, that implies that $\sigma^\pi \models \Box\Diamond(\langle \mathrm{eff}(\alpha)\rangle \wedge \pi = k \wedge sv[k]' = sv[k] \wedge sv[j]' = sv[j] \wedge \langle \mathrm{eff}(\beta)\rangle)$. Otherwise, a change in $sv[k]$ would not satisfy $\mathrm{eff}(\alpha)$. Suppose $\mathrm{eff}(\alpha) = h(e, e', sv[j], sv[j]')$ and $\mathrm{eff}(\beta) = h(e, e', sv[k], sv[k]')$ are syntactically equivalent actions, then $\mathrm{eff}(\tilde{\alpha}) = h(e, e', sv_a[2], sv_a[2]')$ and $\mathrm{eff}(\tilde{\beta}) = h(e, e', sv_a[2], sv_a[2]')$. Therefore, $\mathrm{eff}(\tilde{\alpha}) \leftrightarrow \mathrm{eff}(\tilde{\beta})$, which implies that

$$\forall i \geq 0 : ((f(\sigma^\pi), i), (f(\sigma^\pi), i+1)) \models \langle \mathrm{eff}(\tilde{\beta})\rangle \Rightarrow$$
$$((f(\sigma^\pi), i), (f(\sigma^\pi), i+1)) \models \langle \mathrm{eff}(\tilde{\alpha})\rangle$$

Consequently, if α and β are syntactically equivalent

$$\sigma^\pi \models \Box\Diamond(\langle \mathrm{eff}(\alpha)\rangle \wedge \pi \neq j) \Rightarrow f(\sigma^\pi) \models \Box\Diamond\langle \mathrm{eff}(\tilde{\alpha})\rangle$$

What is left is the case in which α and β are not syntactically equivalent. In this case, because of restriction Λ4b, in order for $\mathrm{eff}(\beta) \rightarrow \mathrm{eff}(\alpha)$ to hold, the value of the variable $sv[k]'$, which is equal to $sv[k]$, must be equal to the value of $sv[j]'$, which is equal to $sv[j]$, in all pairs of states that satisfy $\mathrm{eff}(\beta) \rightarrow \mathrm{eff}(\alpha)$. Because of that in the corresponding states of $f(\sigma^\pi)$, $sv_a[2] = sv[k] = sv[j]$ and $sv_a[2]' = sv_a[2]$. However, since $(sv[j] = sv_a[2] \wedge sv[j]' = sv_a[2]') \rightarrow (\mathrm{eff}(\alpha) \leftrightarrow \mathrm{eff}(\tilde{\alpha}))$ by construction of $\tilde{\alpha}$,

$$\forall i \geq 0 : ((\sigma^\pi, i), (\sigma^\pi, i+1)) \models (\langle \mathrm{eff}(\beta)\rangle \wedge sv[k] = sv[j] \wedge \pi = k \wedge (\mathrm{eff}(\beta) \rightarrow \mathrm{eff}(\alpha))) \Rightarrow$$
$$((f(\sigma^\pi), i), (f(\sigma^\pi), i+1)) \models \langle \mathrm{eff}(\tilde{\alpha})\rangle$$

Consequently, in the case α and β are not syntactically equivalent

$$\sigma^\pi \models \Box\Diamond(\langle \mathrm{eff}(\alpha)\rangle \wedge \pi \neq j) \Rightarrow f(\sigma^\pi) \models \Box\Diamond\langle \mathrm{eff}(\tilde{\alpha})\rangle$$

□

The two results above complete the proof. □

4.1.2.4. CASE: If $\tilde{\alpha}$ was generated from α by using rule T4, then $\sigma^\pi \models \Box\Diamond\langle \mathrm{eff}(\alpha)\rangle$ implies $f(\sigma^\pi) \models \Box\Diamond\langle \mathrm{eff}(\tilde{\alpha})\rangle$.

PROOF:Following the same reasoning as above we assume that $\sigma^\pi \models \Box\Diamond h(e, e', sv[g(e)])$. Since the range of $g(e)$ is $1..N$, which is a finite set for all N, we have

$$\sigma^\pi \models \Box\Diamond h(e, e', sv[1]) \vee \exists j \in 2..N : \sigma^\pi \models \Box\Diamond h(e, e', sv[j])$$

In the case of $\sigma^\pi \models \Box\Diamond h(e, e', sv[1])$, we have that for all $i \geq 0$, (σ^π, i) and $(f(\sigma^\pi), i)$ agree on e and $sv[1]$. Because of that

$$\forall i \geq 0 \; : \; \langle(\sigma^\pi, i), (\sigma^\pi, i+1)\rangle \models h(e, e', sv[1]) \rightarrow$$
$$\langle(f(\sigma^\pi), i), (f(\sigma^\pi), i+1)\rangle \models h(e, e', sv_a[1])$$

Therefore,

$$\sigma^\pi \models \Box\Diamond h(e, e', sv[1]) \; \Rightarrow \; f(\sigma^\pi) \models \Box\Diamond h(e, e', sv_a[1])$$

In case $\exists j \in 2..N : \sigma^\pi \models \Box\Diamond h(e, e', sv[j])$, we can split in two cases, i.e., $\pi = j$ and $\pi \neq j$, and follow the same reasoning as in step 4.1.2.2 to prove that $f(\sigma^\pi) \models \Box\Diamond h(e, e', sv_a[2])$. □

From all the above cases, it was shown that if $\sigma^\pi \models \Box\Diamond\langle \mathrm{eff}(\alpha)\rangle$, then $f(\sigma^\pi) \models \Box\Diamond\langle \mathrm{eff}(\alpha)\rangle$. $\Box$

Since both cases, i.e., $\sigma^\pi \models \Box\Diamond\neg\mathrm{prec}(\alpha)$ and $\sigma^\pi \models \Box\Diamond\langle \mathrm{eff}(\alpha)\rangle$, lead to contradiction, we conclude that $f(\sigma^\pi)$ satisfies all weak fairness conditions of L_a, when σ^π belongs to $\mathcal{P}(S^\pi)$. $\Box$

4.2. CASE: If σ^π belongs to $\mathcal{P}(S^\pi)$, then $f(\sigma^\pi)$ satisfies all strong fairness conditions of L_a.

PROOF:This case will be proven similarly to step 4.1. More specifically, suppose that there exists σ^π in $\mathcal{P}(S^\pi)$ and there exists $\tilde{\alpha}$ in $\tilde{\mathcal{S}}$, such that $sf(\tilde{\alpha})$ is not satisfied by $f(\sigma^\pi)$. We prove that this leads to a contradiction. In order for $\tilde{\alpha}$ to belong to $\tilde{\mathcal{S}}$, there must exist α in $\mathcal{S}$, such that $\sigma^\pi \models sf(\alpha)$. Equivalently,

$$\sigma^\pi \models \Diamond\Box\neg\mathrm{prec}(\alpha) \vee \Box\Diamond\langle\mathrm{eff}(\alpha)\rangle \Rightarrow$$

$$\sigma^\pi \models \Diamond\Box\neg\mathrm{prec}(\alpha) \vee \sigma^\pi \models \Box\Diamond\langle\mathrm{eff}(\alpha)\rangle$$

For both cases we can prove that

$$f(\sigma^\pi) \models \Diamond\Box\neg\mathrm{prec}(\tilde{\alpha}) \vee \Box\Diamond\langle\mathrm{eff}(\tilde{\alpha})\rangle$$

which is a contradiction.

4.2.1. CASE: If $\sigma^\pi \models \Diamond\Box\neg\mathrm{prec}(\alpha)$, then $f(\sigma^\pi) \models \Diamond\Box\neg\mathrm{prec}(\tilde{\alpha})$.

PROOF:Because of rules T1-T3, it holds that $\mathrm{prec}(\alpha) \leftrightarrow \mathrm{prec}(\tilde{\alpha})$. Moreover, because of condition $\Lambda 3$, the assertion $\mathrm{prec}(\alpha)$ is a function of only the e part of the state. Since the two sequences σ^π and $f(\sigma^\pi)$ agree on e in each state, for all i with $i \geq 0$

$$(\sigma^\pi, i) \models \neg\mathrm{prec}(\alpha) \Rightarrow (f(\sigma^\pi), i) \models \neg\mathrm{prec}(\tilde{\alpha})$$

Consequently,

$$\sigma^\pi \models \Diamond\Box\neg\mathrm{prec}(\alpha) \Rightarrow f(\sigma^\pi) \models \Diamond\Box\neg\mathrm{prec}(\tilde{\alpha})$$

$\Box$

4.2.2. CASE: If $\sigma^\pi \models \Box\Diamond\langle\mathrm{eff}(\alpha)\rangle$, then $f(\sigma^\pi) \models \Box\Diamond\langle\mathrm{eff}(\alpha)\rangle$.

PROOF:This is already proven in step 4.1.2. $\Box$

Therefore, if σ^π satisfies $sf(\alpha)$, the sequence $f(\sigma^\pi)$ satisfies $sf(\tilde{\alpha})$, which is a contradiction. $\Box$

4.3. CASE: If σ^π belongs to $\mathcal{P}(S^\pi)$, then $f(\sigma^\pi)$ satisfies all constant value fairness conditions of L_a.

PROOF:In this case we need to prove that $\sigma^\pi \in \mathcal{P}(S^\pi)$ implies that $\forall\alpha \in \mathcal{C}$: $f(\sigma^\pi) \models cf(\alpha)$. We prove this by contradiction. Assume that $\sigma^\pi \in \mathcal{P}(S^\pi)$ and there exists $\alpha \in \mathcal{C}$ such that $f(\sigma^\pi) \not\models cf(\alpha)$. There are two cases; either α belongs to $\mathcal{W}$, or α belongs to $\mathcal{S}$.

4.3.1. CASE: $\alpha \in \mathcal{W} \Rightarrow f(\sigma^\pi) \models cf(\alpha)$

PROOF:Since σ^π is in $\mathcal{P}(S^\pi)$, it must hold that $\sigma^\pi \models wf(\alpha)$. Equivalently, it holds that

$$\sigma^\pi \models \Box\Diamond\neg\mathrm{prec}(\alpha) \vee \sigma^\pi \models \Box\Diamond\langle\mathrm{eff}(\alpha)\rangle$$

We prove that each disjunct implies $f(\sigma^\pi) \models cf(\alpha)$.

4.3.1.1. CASE: $\sigma^\pi \models \Box\Diamond\neg\mathrm{prec}(\alpha) \Rightarrow f(\sigma^\pi) \models cf(\alpha)$

PROOF:Since prec(α) is only a function of e :

$$\forall i \geq 0 : (\sigma^\pi, i) \models \neg\mathrm{prec}(\alpha) \Rightarrow (f(\sigma^\pi, i)) \models \neg\mathrm{prec}(\alpha)$$

Therefore,

$$\sigma^\pi \models \Box\Diamond\neg\mathrm{prec}(\alpha) \Rightarrow f(\sigma^\pi) \models \Box\Diamond\neg\mathrm{prec}(\alpha)$$

$$\Rightarrow f(\sigma^\pi) \models cf(\alpha)$$

□

4.3.1.2. CASE: $\sigma^\pi \models \Box\Diamond\langle \text{eff}(\alpha)\rangle \Rightarrow f(\sigma^\pi) \models cf(\alpha)$

PROOF:

$$\begin{aligned}\sigma^\pi \models \Box\Diamond\langle \text{eff}(\alpha)\rangle &\Rightarrow \sigma^\pi \models \Box\Diamond c(e,e',sv[k])\\ &\Rightarrow \sigma^\pi \models \Box\Diamond c(e,e',v_k)\end{aligned}$$

The condition $c(e,e',v_k)$ is specified only on the e part of the state. Therefore,

$$\begin{aligned}\forall i \geq 0 \,:\, ((\sigma^\pi,i),(\sigma^\pi,i+1)) \models c(e,e',v_k) \Rightarrow\\ ((f(\sigma^\pi),i),(f(\sigma^\pi),i+1)) \models c(e,e',v_k)\end{aligned}$$

Consequently,

$$\begin{aligned}\sigma^\pi \models \Box\Diamond c(e,e',v_k) &\Rightarrow f(\sigma^\pi) \models \Box\Diamond c(e,e',v_k)\\ &\Rightarrow f(\sigma^\pi) \models \Box\Diamond c(e,e',v_k)\end{aligned}$$

□

This completes the proof for $\alpha \in \mathcal{W}$. □

4.3.2. CASE: $\alpha \in \mathcal{S} \Rightarrow f(\sigma^\pi) \models cf(\alpha)$

PROOF:Since σ^π is in $\mathcal{P}(S^\pi)$, it must hold that $\sigma^\pi \models sf(\alpha)$. Equivalently, it holds that

$$\sigma^\pi \models \Diamond\Box\neg\text{prec}(\alpha) \,\vee\, \sigma^\pi \models \Box\Diamond\langle \text{eff}(\alpha)\rangle$$

For the first case we can follow a similar argument as for $\alpha \in \mathcal{W}$, and for the second the argument is identical to the case $\alpha \in \mathcal{W}$, to prove that $f(\sigma^\pi) \models cf(\alpha)$. Therefore, we have a contradiction. □

Since α must be either in $\mathcal{W}$ or in $\mathcal{S}$, the proof is completed. □

4.4. CASE: If σ^π belongs to $\mathcal{P}(S^\pi)$, then $f(\sigma^\pi)$ satisfies all justice and compassion requirements of L_a.

PROOF: Since for any justice requirement $\Box\Diamond p$, the assertion p is expressed only on variables in Σ_{nsv} and $sv[1]$, we have

$$\begin{aligned}\forall i \geq 0 : (\sigma^\pi,i) \models p \leftrightarrow (f(\sigma^\pi),i) \models p \Rightarrow\\ \left(\sigma^\pi \models \Box\Diamond p\right) \leftrightarrow \left(f(\sigma^\pi) \models \Box\Diamond p\right)\end{aligned}$$

The same result can be derived for any compassion requirement $\Box\Diamond q \rightarrow \Box\Diamond r$, since q and r are assertions expressed on Σ_{nsv} and $sv[1]$. Therefore,

$$\sigma^\pi \models L_e \Rightarrow f(\sigma^\pi) \models L_e$$

Since σ^π belongs to $\mathcal{P}(S^\pi)$, we have that $\sigma^\pi \models L_e$ and, therefore, $f(\sigma^\pi)$ satisfies L_e, which is the conjunction of all justice and compassion requirements of L_a. □

□

From Lemmas 2 and 3, Theorem 1 follows.

```
(*******************************************************************************)
(**** ALGORITHM 1 **** )
(* Simplified version of the variant of Arora and Gouda's algorithm. The algorithm *)
(* that each process executes is shown. Moreover, a part of the initial conditions *)
(* and the fairness requirements for the actions of the processes are displayed. *)
(*******************************************************************************)
```

$Root \triangleq [\, i \in 1 \,..\, N \mapsto sv[i][1] \,]$

$D \triangleq [\, i \in 1 \,..\, N \mapsto sv[i][2] \,]$

MODULE *Process*

CONSTANT id

VARIABLES $lRoot$, lF, lD

```
Init  ≜   ∧ $sv[id] \in ((Nat) \times (Nat \cup \{0\}))$   represents $Root \times D$
          ∧ $F[id] \in Nat$
          ∧ $lF \in Nat$
          ∧ $lRoot \in Nat$
          ∧ $lD \in Nat$

Action1  ≜
          ∧ $id \in NodeIdSet$
          ∧ ∨ $Root[id] < id$
            ∨ ∧ $F[id] = id$
              ∧ ∨ $Root[id] \neq id$
                ∨ $D[id] \neq 0$
            ∨ ∨ $\neg(F[id] \in (Neighbors(id) \cup \{id\}))$
              ∨ $D[id] \geq K$
          ∧ $sv' = [sv$ EXCEPT $![id] = \langle id, 0\rangle]$
          ∧ $F' = [F$ EXCEPT $![id] = id]$
          ∧ UNCHANGED $\langle en\_vars, nb\_vars\rangle$

Action2  ≜
          ∧ $id \in NodeIdSet$
          ∧ $lF = F[id]$
          ∧ $lF \in Neighbors(id)$
          ∧ $D[id] \in 0 \,..\, K-1$
          ∧ ∨ $Root[id] \neq lRoot$
            ∨ $D[id] \neq lD + 1$
          ∧ $sv' = [sv$ EXCEPT $![id] = \langle lRoot, lD+1\rangle]$
          ∧ UNCHANGED $\langle en\_vars, nb\_vars, F\rangle$

Action3(j)  ≜
          ∧ $id \in NodeIdSet$
          ∧ $lF = j$
          ∧ ∨ ∧ $Root[id] < lRoot$
              ∧ $lF \in Neighbors(id)$
            $\land\ lD \in 0\,..\,K-1$
        $\lor\ \land\ Root[id] = lRoot$
            $\land\ lF \in Neighbors(id)$
            $\land\ lD + 1 < D[id]$
    $\land\ sv' = [sv\ \text{EXCEPT}\ ![id] = \langle lRoot,\ lD + 1\rangle]$
    $\land\ F' = [F\ \text{EXCEPT}\ ![id] = lF]$
    $\land\ \text{UNCHANGED}\ \langle en\_vars,\ nb\_vars\rangle$

$Action4(j)\ \triangleq$
    $\land\ id \in NodeIdSet$
    $\land\ j \in Neighbors(id)$
    $\land\ lRoot' = sv[j][1]$
    $\land\ lD' \quad = sv[j][2]$
    $\land\ lF' \quad = j$
    $\land\ \text{UNCHANGED}\ \langle en\_vars,\ st\_vars\rangle$

$Next\ \triangleq$
    $\lor\ Action1$
    $\lor\ Action2$
    $\lor\ \exists\, j \in Neighbors(id) : Action3(j)$
    $\lor\ \exists\, j \in Neighbors(id) : Action4(j)$

$Fairness$
  $\triangleq\ \land\ \mathrm{SF}_{gvars}(Action1)$
     $\land\ \mathrm{SF}_{gvars}(Action2)$
     $\land\ \forall\, j \in Neighbors(id) : \mathrm{SF}_{gvars}(Action3(j))$
     $\land\ \forall\, j \in Neighbors(id) : \mathrm{SF}_{gvars}(Action4(j))$
```

```
(************************************************************************************ )
(**** ALGORITHM 2 **** )
(* Version of the ST algorithm after data abstraction has been applied and before *)
(* the application of our technique *)
(************************************************************************************ )
```

```
---------------------------- MODULE Spanning_Tree_c ----------------------------
EXTENDS Integers, Sequences, Naturals, TLC, FiniteSets

VARIABLES   sv, F, lRoot, lD, lF, SetOfLMinus1Neighbors

CONSTANT    N, LTk, GTk, DGTK, K, EQk, NotNeighborNode,
            NeighborLorMore, NeighborLMinus1, l

--------------------------------------------------------------------------------

en_vars  ≜ ⟨SetOfLMinus1Neighbors⟩

st_vars  ≜ ⟨sv, F⟩

nb_vars  ≜ ⟨lRoot, lD, lF⟩

gvars    ≜ ⟨en_vars, st_vars, nb_vars⟩

lsv ≜ ⟨lRoot, lD⟩

Root
     ≜  [i ∈ 1 .. N ↦ sv[i][1]]

D
     ≜  [i ∈ 1 .. N ↦ sv[i][2]]

IsLessThanR(m, n) ≜
               IF m = LTk THEN
                   TRUE
                ELSE IF n = GTk THEN
                   TRUE
                ELSE
                   FALSE


IsLessThanRCons(m, n)
                 ≜
            IF   ∧ m = LTk
                 ∧ n ∈ {EQk, GTk}
             THEN
                TRUE
             ELSE IF ∧ m = EQk
                     ∧ n = GTk
                THEN
                   TRUE
                ELSE
                   FALSE

$IsEqualToRCons(m, n)$
                        $\triangleq$
               IF  $\lor\ m \in \{LTk, GTk\}$
                   $\lor\ n \in \{LTk, GTk\}$
                THEN
                   FALSE
                ELSE IF $\land\ m = EQk$
                        $\land\ n = EQk$
                THEN
                   TRUE
                ELSE
                   FALSE


$IsNotEqualToRCons(m, n)$
                  $\triangleq\ m \neq n$

$IsLessThanEqualD(m, n)$
     $\triangleq$ IF  $\land\ m \neq DGTK$
               $\land\ n \neq DGTK$
          THEN
             $m \leq n$
          ELSE IF $n = DGTK$ THEN
             TRUE
          ELSE
             FALSE


$CheckThatDiEqDjPlus1(m, n)$
     $\triangleq$ IF  $\lor\ m = DGTK$
               $\lor\ n = DGTK$
          THEN
             FALSE
          ELSE
             $m = n + 1$

$gsv(shared\_var)$
     $\triangleq$ IF $shared\_var = \langle EQk, l - 1\rangle$
          THEN     $NeighborLMinus1$
```

```
        ELSE    $NeighborLorMore$
```

Correctness Property

```
$GPropertyForOneNode$
    $\triangleq$
      $\land\ Root[1] = EQk$
      $\land\ F = NeighborLMinus1$
      $\land\ D[1] = l$
```

MODULE *Process*

CONSTANT *id*

Actions

```
$Action1$  $\triangleq$
        $\land$  $\lor\ Root[id] = LTk$
             $\lor\ \land\ F = id$
                 $\land\ \lor\ Root[id] = GTk$
                     $\lor\ \land\ Root[id] = EQk$
                         $\land\ id \neq EQk$
                     $\lor\ Root[id] = LTk$
                     $\lor\ D[id] \neq 0$
             $\lor\ \lor\ \land\ \neg(F \in \{NeighborLMinus1\})$
                     $\land\ F \neq id$
                 $\lor\ D[id] \in \{K,\ DGTK\}$
        $\land$  $\lor\ \land\ id \neq EQk$
                 $\land\ sv' = [sv \text{ EXCEPT } ![id] = \langle LTk,\ 0\rangle]$
             $\lor\ \land\ id = EQk$
                 $\land\ sv' = [sv \text{ EXCEPT } ![id] = \langle EQk,\ 0\rangle]$
        $\land$  $F' = id$
        $\land$  UNCHANGED $\langle en\_vars,\ nb\_vars\rangle$

$Action1Cons$
        $\triangleq$
        $\land$  $\lor\ \land\ F = id$
                 $\land\ \lor\ Root[id] = GTk$
                     $\lor\ \land\ Root[id] = EQk$
                         $\land\ id\ \neq EQk$
                     $\lor\ D[id] \neq 0$
             $\lor\ \lor\ F \in \{NotNeighborNode\}$
                 $\lor\ D[id] \in \{K,\ DGTK\}$
```

```
$\land$  $\lor$ $\land$ $id \neq EQk$
          $\land$ $sv' = [sv \text{ EXCEPT } ![id] = \langle LTk, 0\rangle]$
     $\lor$ $\land$ $id = EQk$
          $\land$ $sv' = [sv \text{ EXCEPT } ![id] = \langle EQk, 0\rangle]$
$\land$  $F' = id$
$\land$  UNCHANGED $\langle en\_vars, nb\_vars\rangle$
```

```
$Action2$  $\triangleq$
    $\land$  $lF = F$
    $\land$  $D[id] \in 0 \,..\, K-1$
    $\land$  $\lor \neg($ $\land$ $Root[id] = EQk$
                  $\land$ $lRoot = EQk)$
         $\lor$  $\lor$ $lD = DGTK$
             $\lor$  $\land$ $D[id] \neq DGTK$
                 $\land$ $lD \neq DGTK$
                 $\land$ $D[id] \neq lD + 1$
    $\land$  $\lor$  $\land$ $lD \in \{K, DGTK\}$
             $\land$ $sv' = [sv \text{ EXCEPT } ![id] = \langle lRoot, DGTK\rangle]$
         $\lor$  $\land$ $lD \in 0 \,..\, K-1$
             $\land$ $sv' = [sv \text{ EXCEPT } ![id] = \langle lRoot, lD+1\rangle]$
    $\land$  UNCHANGED $\langle en\_vars, nb\_vars, F\rangle$
```

```
$Action2Cons$
    $\triangleq$
    $\land$  $lF = F$
    $\land$  $\neg(lF \in \{NotNeighborNode\})$
    $\land$  $D[id] \in 0 \,..\, K-1$
    $\land$  $\lor$  $IsNotEqualToRCons(Root[id], lRoot)$
         $\lor$  $\land$  $lD \in 0 \,..\, K$
             $\land$  $D[id] \in 0 \,..\, K$
             $\land$  $D[id] \neq lD + 1$
         $\lor$  $\land$  $lD = DGTK$
             $\land$  $D[id] \in 0 \,..\, K$
    $\land$  $\lor$  $\land$ $lD \in \{K, DGTK\}$
             $\land$ $sv' = [sv \text{ EXCEPT } ![id] = \langle lRoot, DGTK\rangle]$
         $\lor$  $\land$ $lD \in 0 \,..\, K-1$
             $\land$ $sv' = [sv \text{ EXCEPT } ![id] = \langle lRoot, lD+1\rangle]$
    $\land$  UNCHANGED $\langle en\_vars, nb\_vars, F\rangle$
```

```
$Action3(u)$  $\triangleq$
    $\land$  $lF = u$
    $\land$  $\lor$  $\land$  $IsLessThanR(Root[id], lRoot)$
             $\land$  $lD \in 0 \,..\, K-1$
```

```
        ∨ ∧ $Root[id] = lRoot$
          ∧ ∨ $D[id] = DGTK$
            ∨ ∧ $D[id] \in 0 \, .. \, K$
              ∧ $lD \in 0 \, .. \, K - 1$
              ∧ $lD + 1 < D[id]$
    ∧ ∨ ∧ $lD \in \{K, DGTK\}$
          ∧ $sv' = [sv \text{ EXCEPT } ![id] = \langle lRoot, DGTK \rangle]$
      ∨ ∧ $lD \in 0 \, .. \, K - 1$
          ∧ $sv' = [sv \text{ EXCEPT } ![id] = \langle lRoot, lD + 1 \rangle]$
    ∧ $F' = lF$
    ∧ UNCHANGED $\langle en\_vars, nb\_vars \rangle$

$Action3Cons(u)$
    $\triangleq$
    ∧ $lF = u$
    ∧ ∨ ∧ $IsLessThanRCons(Root[id], lRoot)$
        ∧ $lD \in 0 \, .. \, K - 1$
      ∨ ∧ $IsEqualToRCons(Root[id], lRoot)$
        ∧ ∨ ∧ $D[id] = DGTK$
              ∧ $lD \in 0 \, .. \, K - 1$
          ∨ ∧ $D[id] \in 0 \, .. \, K$
              ∧ $lD \in 0 \, .. \, K - 1$
              ∧ $lD + 1 < D[id]$
    ∧ ∨ ∧ $lD \in \{K, DGTK\}$
          ∧ $sv' = [sv \text{ EXCEPT } ![id] = \langle lRoot, DGTK \rangle]$
      ∨ ∧ $lD \in 0 \, .. \, K - 1$
          ∧ $sv' = [sv \text{ EXCEPT } ![id] = \langle lRoot, lD + 1 \rangle]$
    ∧ $F' = lF$
    ∧ UNCHANGED $\langle en\_vars, nb\_vars \rangle$

$Action4(j)$ $\triangleq$
    ∧ $lRoot' = sv[j][1]$
    ∧ $lD'    = sv[j][2]$
    ∧ $lF'    = gsv(sv[j])$
    ∧ UNCHANGED $\langle en\_vars, st\_vars \rangle$

$Next$
  $\triangleq$ ∨ $Action1$
     ∨ $Action1Cons$
     ∨ $Action2$
     ∨ $Action2Cons$
     ∨ $\exists \, u \in \{NeighborLMinus1, NeighborLorMore\} : Action3(u)$
    $\lor \exists u \in \{NeighborLMinus1, NeighborLorMore\} : Action3Cons(u)$
    $\lor \exists j \in 2 \,..\, N : Action4(j)$
```

MODULE *ProcessLOrMore*

Actions

```
$Action1(j)$
    $\triangleq$
        $\land \neg j \in SetOfLMinus1Neighbors$
        $\land$ LET $new\_value \triangleq$ CHOOSE $a \in (l \,..\, K \cup \{DGTK\})$ : TRUE
           IN
              $sv' = [sv$ EXCEPT $![j] = \langle EQk, new\_value \rangle]$
        $\land$ UNCHANGED $\langle en\_vars, nb\_vars, F \rangle$

$Action2(j)$
    $\triangleq$
        $\land \neg j \in SetOfLMinus1Neighbors$
        $\land$ LET $new\_value \triangleq$ CHOOSE $a \in (0 \,..\, K \cup \{DGTK\})$ : TRUE
           IN
              $sv' = [sv$ EXCEPT $![j] = \langle LTk, new\_value \rangle]$
        $\land$ UNCHANGED $\langle en\_vars, nb\_vars, F \rangle$


$Next$
    $\triangleq$  $\lor \exists j \in 2 \,..\, N : Action1(j)$
                  $\lor \exists j \in 2 \,..\, N : Action2(j)$
```

```
$FS$  $\triangleq$  $\{\langle EQk, l - 1 \rangle\} \cup \{ \langle LTk, j \rangle : j \in (0 \,..\, K \cup \{DGTK\})\}$
                    $\cup \{\langle EQk, j \rangle : j \in (l \,..\, K \cup \{DGTK\}) \}$


$P(i)$  $\triangleq$  INSTANCE $Process$ WITH $id \leftarrow i$


$PLOrMore$
        $\triangleq$  INSTANCE $ProcessLOrMore$


$Init$  $\triangleq$  $\land sv \in [1 \,..\, N \to \{LTk, EQk\} \times (0 \,..\, K \cup \{DGTK\})]$
                      $\land \forall v \in 1 \,..\, N : (Root[v] = EQk) \Rightarrow D[v] \in (l - 1 \,..\, K \cup \{DGTK\})$
                      $\land lF \in \{NeighborLorMore, NeighborLMinus1\}$
```

$\land\ lRoot \in \{LTk,\ EQk\}$
$\land\ lD \in 0\,..\,K \cup \{DGTK\}$
$\land\ lRoot = EQk \Rightarrow lD \in (l-1\,..\,K \cup \{DGTK\})$

the local copies will initially have one of the values of the neighbors

$\land\ F \in \{NotNeighborNode,\ NeighborLorMore,\ NeighborLMinus1,\ 1\}$
$\land\ \exists\, v \in 2\,..\,N : (Root[v] = EQk \land D[v] = l-1)$
$\land\ SetOfLMinus1Neighbors = \{v \in 2\,..\,N : (Root[v] = EQk \land D[v] = l-1)\}$

$Invariant$
$\triangleq\ \land \forall\, v \in 1\,..\,N : Root[v] \in \{LTk,\ EQk\}$
$\land \forall\, v \in 1\,..\,N : D[v] \in (0\,..\,K \cup \{DGTK\})$
$\land\ lRoot \in \{LTk,\ EQk\}$
$\land\ lD \in 0\,..\,K \cup \{DGTK\}$
$\land\ F \in \{NotNeighborNode,\ NeighborLorMore,\ NeighborLMinus1,\ 1\}$
$\land\ lF \in \{NeighborLorMore,\ NeighborLMinus1\}$
$\land \forall\, v \in 1\,..\,N : (Root[v] = EQk) \Rightarrow D[v] \in (l-1\,..\,K \cup \{DGTK\})$
$\land \exists\, v \in 2\,..\,N : (Root[v] = EQk \land D[v] = l-1)$

$Next \triangleq\ \lor\ P(1)!Next$
$\lor\ PLOrMore!Next$

$Fairness \triangleq\ \land\, \mathrm{SF}_{gvars}(P(1)!Action1Cons)$
$\land\, \mathrm{SF}_{gvars}(P(1)!Action2Cons)$
$\land \forall\, u \in \{NeighborLMinus1,\ NeighborLorMore\} : \mathrm{SF}_{gvars}(P(1)!Action3Cons(u))$
$\land \forall\, j \in 2\,..\,N : \mathrm{SF}_{gvars}(P(1)!Action4(j))$
$\land\, (\Box\Diamond(F = NeighborLorMore)) \Rightarrow (\Box\Diamond(lF = NeighborLorMore))$

otherwise, $F = NotNeighborNode$

$Spec \triangleq Init \land \Box[Next]_{gvars} \land Fairness$

$Correctness$
$\triangleq \Diamond\Box GPropertyForOneNode$

```
─────────────── MODULE Spanning_Tree_a ───────────────
EXTENDS Integers, Sequences, Naturals, TLC, FiniteSets

VARIABLES  sv, F, lRoot, lD, lF

CONSTANT   LTk, GTk, DGTK, K, EQk, NotNeighborNode,
           NeighborLorMore, NeighborLMinus1, l

──────────────────────────────────────────────────────

st_vars  ≜ ⟨sv, F⟩

nb_vars  ≜ ⟨lRoot, lD, lF⟩

gvars    ≜ ⟨st_vars, nb_vars⟩

lsv ≜  ⟨lRoot, lD⟩

Root
    ≜  [i ∈ 1 . . 2 ↦ sv[i][1]]

D
    ≜  [i ∈ 1 . . 2 ↦ sv[i][2]]

IsLessThanR(m, n)  ≜
                   IF  m = LTk  THEN
                       TRUE
                    ELSE  IF  n = GTk  THEN
                       TRUE
                    ELSE
                       FALSE


IsLessThanRCons(m, n)
                 ≜
             IF   ∧ m = LTk
                  ∧ n  ∈ {EQk, GTk}
              THEN
                 TRUE
              ELSE  IF  ∧ m = EQk
                        ∧ n  = GTk
              THEN
                 TRUE
              ELSE
                 FALSE

IsEqualToRCons(m, n)
```

```
            $\triangleq$
    IF  $\lor m \in \{LTk, GTk\}$
        $\lor n \in \{LTk, GTk\}$
    THEN
        FALSE
    ELSE IF $\land m = EQk$
            $\land n = EQk$
    THEN
        TRUE
    ELSE
        FALSE
```

```
$IsNotEqualToRCons(m, n)$
            $\triangleq$ $m \neq n$
```

```
$IsLessThanEqualD(m, n)$
    $\triangleq$ IF  $\land m \neq DGTK$
                $\land n \neq DGTK$
            THEN
                $m \leq n$
            ELSE IF $n = DGTK$ THEN
                TRUE
            ELSE
                FALSE
```

```
$CheckThatDiEqDjPlus1(m, n)$
    $\triangleq$ IF  $\lor m = DGTK$
                $\lor n = DGTK$
            THEN
                FALSE
            ELSE
                $m = n + 1$
```

```
$gsv(shared\_var)$
    $\triangleq$ IF $shared\_var = \langle EQk, l - 1\rangle$
            THEN    $NeighborLMinus1$
            ELSE    $NeighborLorMore$
```

Correctness Property

```
$GPropertyForOneNode$
    $\triangleq$
```

```
        $\land Root[1] = EQk$
        $\land F = NeighborLMinus1$
        $\land D[1] = l$
```

MODULE *Process*

```
CONSTANT $id$
```

Actions

```
$Action1$ $\triangleq$
    $\land$  $\lor Root[id] = LTk$
             $\lor \land F = id$
                $\land \lor Root[id] = GTk$
                   $\lor \land Root[id] = EQk$
                      $\land id \neq EQk$
                   $\lor Root[id] = LTk$
                   $\lor D[id] \neq 0$
             $\lor \lor \land \neg(F \in \{NeighborLMinus1\})$
                   $\land F \neq id$
                $\lor D[id] \in \{K, DGTK\}$
    $\land$  $\lor \land id \neq EQk$
                $\land sv' = [sv$ EXCEPT $![id] = \langle LTk, 0\rangle]$
             $\lor \land id = EQk$
                $\land sv' = [sv$ EXCEPT $![id] = \langle EQk, 0\rangle]$
    $\land$  $F' = id$
    $\land$  UNCHANGED $\langle nb\_vars\rangle$

$Action1Cons$
    $\triangleq$
    $\land$  $\lor \land F = id$
                $\land \lor Root[id] = GTk$
                   $\lor \land Root[id] = EQk$
                      $\land id \neq EQk$
                   $\lor D[id] \neq 0$
             $\lor \lor F \in \{NotNeighborNode\}$
                $\lor D[id] \in \{K, DGTK\}$
    $\land$  $\lor \land id \neq EQk$
                $\land sv' = [sv$ EXCEPT $![id] = \langle LTk, 0\rangle]$
             $\lor \land id = EQk$
                $\land sv' = [sv$ EXCEPT $![id] = \langle EQk, 0\rangle]$
    $\land$  $F' = id$
    $\land$  UNCHANGED $\langle nb\_vars\rangle$
```

```
$Action2 \triangleq$
    $\land\ lF = F$
    $\land\ D[id] \in 0\,..\,K - 1$
    $\land\ \lor \neg(\ \land Root[id] = EQk$
                $\land\ lRoot = EQk)$
        $\lor\ \lor lD = DGTK$
            $\lor\ \land D[id] \neq DGTK$
                $\land\ lD \neq DGTK$
                $\land\ D[id] \neq lD + 1$
    $\land\ \lor\ \land lD \in \{K, DGTK\}$
            $\land\ sv' = [sv \text{ EXCEPT } ![id] = \langle lRoot, DGTK \rangle]$
        $\lor\ \land lD \in 0\,..\,K - 1$
            $\land\ sv' = [sv \text{ EXCEPT } ![id] = \langle lRoot, lD + 1 \rangle]$
    $\land\ \text{UNCHANGED } \langle nb\_vars, F \rangle$

$Action2Cons$
    $\triangleq$
    $\land\ lF = F$
    $\land\ \neg(lF \in \{NotNeighborNode\})$
    $\land\ D[id] \in 0\,..\,K - 1$
    $\land\ \lor\ IsNotEqualToRCons(Root[id], lRoot)$
        $\lor\ \land\ lD \in 0\,..\,K$
            $\land\ D[id] \in 0\,..\,K$
            $\land\ D[id] \neq lD + 1$
        $\lor\ \land\ lD = DGTK$
            $\land\ D[id] \in 0\,..\,K$
    $\land\ \lor\ \land lD \in \{K, DGTK\}$
            $\land\ sv' = [sv \text{ EXCEPT } ![id] = \langle lRoot, DGTK \rangle]$
        $\lor\ \land lD \in 0\,..\,K - 1$
            $\land\ sv' = [sv \text{ EXCEPT } ![id] = \langle lRoot, lD + 1 \rangle]$
    $\land\ \text{UNCHANGED } \langle nb\_vars, F \rangle$

$Action3(u) \triangleq$
    $\land\ lF = u$
    $\land\ \lor\ \land\ IsLessThanR(Root[id], lRoot)$
            $\land\ lD \in 0\,..\,K - 1$
        $\lor\ \land\ Root[id] = lRoot$
            $\land\ \lor\ D[id] = DGTK$
                $\lor\ \land D[id] \in 0\,..\,K$
                    $\land\ lD \in 0\,..\,K - 1$
                    $\land\ lD + 1 < D[id]$
    $\land\ \lor\ \land lD \in \{K, DGTK\}$
            $\land\ sv' = [sv \text{ EXCEPT } ![id] = \langle lRoot, DGTK \rangle]$
        ∨  $\land\ lD \in 0 \mathrel{..} K-1$
           $\land\ sv' = [sv \text{ EXCEPT } ![id] = \langle lRoot,\ lD+1\rangle]$
     ∧  $F' = lF$
     ∧  UNCHANGED $\langle nb\_vars\rangle$

$Action3Cons(u)$
        $\triangleq$
        ∧  $lF = u$
        ∧  ∨  ∧  $IsLessThanRCons(Root[id],\ lRoot)$
              ∧  $lD \in 0 \mathrel{..} K-1$
           ∨  ∧  $IsEqualToRCons(Root[id],\ lRoot)$
              ∧  ∨  $\land\ D[id] = DGTK$
                    $\land\ lD \in 0 \mathrel{..} K-1$
                 ∨  $\land\ D[id] \in 0 \mathrel{..} K$
                    $\land\ lD \in 0 \mathrel{..} K-1$
                    $\land\ lD + 1 < D[id]$
        ∧  ∨  $\land\ lD \in \{K,\ DGTK\}$
              $\land\ sv' = [sv \text{ EXCEPT } ![id] = \langle lRoot,\ DGTK\rangle]$
           ∨  $\land\ lD \in 0 \mathrel{..} K-1$
              $\land\ sv' = [sv \text{ EXCEPT } ![id] = \langle lRoot,\ lD+1\rangle]$
        ∧  $F' = lF$
        ∧  UNCHANGED $\langle nb\_vars\rangle$

$Action4$    $\triangleq$
     ∧  $lRoot' = sv[2][1]$
     ∧  $lD'\ \ \ = sv[2][2]$
     ∧  $lF'\ \ \ = gsv(sv[2])$
     ∧  UNCHANGED $\langle st\_vars\rangle$

$Next$
   $\triangleq$  ∨ $Action1$
       ∨ $Action1Cons$
       ∨ $Action2$
       ∨ $Action2Cons$
       ∨ $\exists\, u \in \{NeighborLMinus1,\ NeighborLorMore\} : Action3(u)$
       ∨ $\exists\, u \in \{NeighborLMinus1,\ NeighborLorMore\} : Action3Cons(u)$
       ∨ $Action4$
```

MODULE *ProcessLOrMore*

Actions

```
$Action1$
    $\triangleq$
        $\land$ LET $new\_value \triangleq$ CHOOSE $a \in (l \,..\, K \cup \{DGTK\})$ : TRUE
            IN
                $sv' = [sv$ EXCEPT $![2] = \langle EQk, new\_value \rangle]$
        $\land$ UNCHANGED $\langle nb\_vars, F \rangle$

$Action2$
    $\triangleq$
        $\land$ LET $new\_value \triangleq$ CHOOSE $a \in (0 \,..\, K \cup \{DGTK\})$ : TRUE
            IN
                $sv' = [sv$ EXCEPT $![2] = \langle LTk, new\_value \rangle]$
        $\land$ UNCHANGED $\langle nb\_vars, F \rangle$


$Next$
    $\triangleq$  $\lor$ $Action1$
           $\lor$ $Action2$
```

```
$FS$  $\triangleq$  $\{\langle EQk, l-1 \rangle\} \cup \{\ \langle LTk, j \rangle : j \in (0 \,..\, K \cup \{DGTK\})\}$
         $\cup\, \{\langle EQk, j \rangle : j \in (l \,..\, K \cup \{DGTK\})\ \}$


$T0\_added\_action(v)$
    $\triangleq$  $\land\ sv' = [sv$ EXCEPT $![2] = v]$
         $\land$ UNCHANGED $\langle nb\_vars, F \rangle$

$P(i)$      $\triangleq$  INSTANCE $Process$ WITH $id \leftarrow i$

$PLOrMore$
          $\triangleq$  INSTANCE $ProcessLOrMore$

$Init$      $\triangleq$  $\land$  $sv \in [1 \,..\, 2 \rightarrow \{LTk, EQk\} \times (0 \,..\, K \cup \{DGTK\})]$
              $\land$  $\forall v \in 1 \,..\, 2 : (Root[v] = EQk) \Rightarrow D[v] \in (l-1 \,..\, K \cup \{DGTK\})$
              $\land$  $lF \in \{NeighborLorMore, NeighborLMinus1\}$
              $\land$  $lRoot \in \{LTk, EQk\}$
              $\land$  $lD \in 0 \,..\, K \cup \{DGTK\}$
              $\land$  $lRoot = EQk \Rightarrow lD \in (l-1 \,..\, K \cup \{DGTK\})$
                        the local copies will initially have one of the values of the neighbors
              $\land$  $F \in \{NotNeighborNode, NeighborLorMore, NeighborLMinus1, 1\}$
```

$Invariant$

$$\triangleq \begin{array}{l} \land \forall v \in 1 \mathrel{..} 2 : Root[v] \in \{LTk, EQk\} \\ \land \forall v \in 1 \mathrel{..} 2 : D[v] \in (0 \mathrel{..} K \cup \{DGTK\}) \\ \land lRoot \in \{LTk, EQk\} \\ \land lD \in 0 \mathrel{..} K \cup \{DGTK\} \\ \land F \in \{NotNeighborNode, NeighborLorMore, NeighborLMinus1, 1\} \\ \land lF \in \{NeighborLorMore, NeighborLMinus1\} \\ \land \forall v \in 1 \mathrel{..} 2 : (Root[v] = EQk) \Rightarrow D[v] \in (l - 1 \mathrel{..} K \cup \{DGTK\}) \end{array}$$

$$Next \triangleq \begin{array}{l} \lor \; P(1)!Next \\ \lor \; PLOrMore!Next \end{array}$$

$$Fairness \triangleq \begin{array}{l} \land \mathrm{SF}_{gvars}(P(1)!Action1Cons) \\ \land \mathrm{SF}_{gvars}(P(1)!Action2Cons) \\ \land \forall u \in \{NeighborLMinus1, NeighborLorMore\} : \mathrm{SF}_{gvars}(P(1)!Action3Cons(u)) \\ \land \mathrm{SF}_{gvars}(P(1)!Action4) \\ \land \Box\Diamond(lRoot = EQk \land lD = l - 1 \land lF = gsv(\langle EQk, l - 1\rangle)) \\ \qquad\qquad \text{this is the cf constraint} \\ \land (\Box\Diamond(F = NeighborLorMore)) \Rightarrow (\Box\Diamond(lF = NeighborLorMore)) \end{array}$$

$$Spec \triangleq Init \land \Box[Next]_{gvars} \land Fairness$$

$Correctness$

$$\triangleq \Diamond\Box\, GPropertyForOneNode$$