SRC Technical Note
1998 - 005

March 10, 1998

Reduction in TLA

Ernie Cohen and Leslie Lamport
Bellcore

Systems Research Center
130 Lytton Avenue
Palo Alto, California 94301

http://www.research.digital.com/SRC/

Reduction in TLA

Ernie Cohen
Bellcore

Leslie Lamport
Digital Equipment Corporation

10 March 1998

Abstract

Reduction theorems allow one to deduce properties of a concurrent system specification from properties of a simpler, coarser-grained version called the reduced specification. We present reduction theorems based upon a more precise relation between the original and reduced specifications than earlier ones, permitting the use of reduction to reason about a larger class of properties. In particular, we present reduction theorems that handle general liveness properties.

Contents

1 Introduction

We reason about a high-level specification of a system, with a large grain of atomicity, and hope thereby to deduce properties of a finer-grained implementation. For example, the single atomic action

$$x,\; y \;:=\; f(x, y),\; g(x, y)$$

of a high-level algorithm might be implemented by the sequence of actions

$$P(sem);\; t := x;\; x := f(x, y);\; y := g(t, y);\; V(sem) \tag{1}$$

where P and V are the usual operations on a binary semaphore sem, and t is a new variable. This process is usually justified by asserting that the two specification are, in some suitable sense, "equivalent". A *reduction theorem* is a general rule for deriving an "equivalent" higher-level specification S^R from a lower-level one S. We call S^R the *reduced* version of S. For example, S might be a multiprocess program containing critical sections, and S^R might be obtained from S by replacing each critical section with a single atomic statement.

The first reduction theorem was proposed by Lipton [10]. Several others followed [3, 5, 4, 6, 9]. In these theorems, executions of the reduced specification and of the original one are completely separate, sharing only certain properties. In the reduction theorems we present here, the original and reduced specifications "run in parallel", their executions connected by a coupling invariant [7]. Our theorems thereby provide a more precise (and hence stronger) statement of the relation between the original and the reduced specifications. This enables certain hypotheses to be stated as assumptions about a given execution, rather than in the stronger form of assumptions about all executions. In particular, we relate liveness properties of executions of the two specifications, obtaining what we believe to be the first published general reduction theorems that handle liveness. The only previous theorems we know that concern liveness are Back's [3] results for total correctness of sequential programs and a theorem in [4] showing that certain progress properties of a component are preserved under fair parallel composition with an environment.

Our theorems are stated in TLA (the Temporal Logic of Actions) [8], but they should be adaptable to other formalisms with a trace-based semantics. Space does not permit us to include examples; they will appear elsewhere.

2 The Relation Between S and S^R

We begin by examining the relation between the original specification S and the reduced version S^R. We want to infer properties of S by proving properties of S^R. For this, S and S^R needn't be equivalent; it's necessary only that S implement S^R—for some suitable notion of implementation.

Suppose S represents a multiprocess program with shared variables x and y that are accessed only in critical sections, and the reduced version S^R is obtained by replacing each critical section with a single atomic statement—for example, replacing (1) with

$$t,\ x,\ y\ :=\ x,\ f(x,y),\ g(x,y)$$

One sense in which S implements S^R is that, if we ignore the times when a process is in a critical section, S assigns the same sequences of values to all variables that S^R does. This is the notion of implementation used by Doeppner in his reduction theorem [6]. While satisfactory for many purposes, this notion of implementation is rather weak. It says nothing about what is true while a process is in its critical section, which can be a problem because assertional reasoning requires proving that an invariant holds at all times.

Let v be the tuple of all variables of S, including x and y. Our stronger notion of implementation is that there exists a tuple of "virtual variables" $\widehat{v}$ such that, as S changes the real variables v, the virtual variables $\widehat{v}$ change according to the specification $\widehat{S^R}$ obtained from S^R by replacing each real variable by its virtual counterpart. The relation between the real and virtual variables is expressed by a predicate I relating v and $\widehat{v}$. (Such a predicate is known as a "coupling invariant" [7].) This generalizes Doeppner's notion of implementation if I implies $v = \widehat{v}$ when no process is in a critical section. For example, during execution of the critical section (1), I might imply:

$$\widehat{t},\ \widehat{x},\ \widehat{y} = \begin{cases} t,\ x,\ y & \text{before executing } t := \ldots \\ t,\ f(x,y),\ g(x,y) & \text{just after executing } t := \ldots \\ t,\ x,\ g(t,y) & \text{just after executing } x := \ldots \\ t,\ x,\ y & \text{after executing } y := \ldots \end{cases}$$

All the steps of the critical section leave the virtual variables unchanged except for the assignment to t, which performs the "virtual assignment"

$$\widehat{t},\ \widehat{x},\ \widehat{y}\ :=\ \widehat{x},\ f(\widehat{x},\widehat{y}),\ g(\widehat{x},\widehat{y})$$

Expressed in temporal logic, this implementation relation is

$$S \;\Rightarrow\; \exists\, \widehat{v} \;:\; \Box I \wedge \widehat{S^R} \tag{2}$$

where $\exists$ is existential quantification over flexible[1] variables.[2] This is approximately the conclusion of our reduction theorems.

We would like to prove that S^R satisfies (implies) a property Π and deduce that S satisfies Π. By (2), all we can infer from $S^R \Rightarrow \Pi$ is $S \Rightarrow \exists\, \widehat{v} : \Box I \wedge \widehat{\Pi}$. How useful this is depends upon the nature of I and Π. Space precludes a discussion of how our reduction theorem can be applied. We just mention one important case. Suppose I implies $\widehat{z} = z$ for every variable occurring in Π. In this case, $\exists\, \widehat{v} : \Box I \wedge \widehat{\Pi}$ implies Π, so we infer $S \Rightarrow \Pi$ from $S^R \Rightarrow \Pi$. It is this result that justifies the well-known rule for reasoning about multiprocess programs that allows grouping a sequence of operations into a single atomic action if they include only a single access to a shared variable [11].

3 An Intuitive View of Reduction

We consider the situation in which one operation M is reduced to a single atomic action M^R—for example, one critical section is replaced by an atomic statement. Reduction of multiple operations can be performed by applying the theorem multiple times to reduce one operation at a time.

A single execution of the operation M consists of a sequence of M steps. These can be interleaved with other system steps, which we call E steps, as in:

$$\cdots\; s_{41} \xrightarrow{M} s_{42} \xrightarrow{E} s_{43} \xrightarrow{M} s_{44} \xrightarrow{E} s_{45} \xrightarrow{E} s_{46} \xrightarrow{M} s_{47} \xrightarrow{M} s_{48} \;\cdots \tag{3}$$

We think of E as M's environment. The idea is to construct a behavior "equivalent to" (3) by moving all the M steps together, as in

$$\cdots\; s_{41} \xrightarrow{E} u_{42} \xrightarrow{M} u_{43} \xrightarrow{M} u_{44} \xrightarrow{M} u_{45} \xrightarrow{M} u_{46} \xrightarrow{E} u_{47} \xrightarrow{E} s_{48} \;\cdots \tag{4}$$

which is then equivalent to the behavior

$$\cdots\; s_{41} \xrightarrow{E} u_{42} \xrightarrow{M^R} u_{46} \xrightarrow{E} u_{47} \xrightarrow{E} s_{48} \;\cdots \tag{5}$$

[1] In temporal logic, a flexible variable is one whose value can change over time; a rigid variable is one whose value is fixed.

[2] As with any form of implementation, this works only if S^R allows stuttering steps and $\exists$ preserves stuttering invariance [8].

$$
\begin{array}{l}
\cdots\ s_{41} \xrightarrow{R} s_{42} \xrightarrow{E} s_{43} \xrightarrow{X} s_{44} \xrightarrow{E} s_{45} \xrightarrow{E} s_{46} \xrightarrow{L} s_{47} \xrightarrow{L} s_{48}\ \cdots \\
\cdots\ s_{41} \xrightarrow{E} u_{42} \xrightarrow{R} s_{43} \xrightarrow{X} s_{44} \xrightarrow{E} s_{45} \xrightarrow{L} r_{46} \xrightarrow{E} s_{47} \xrightarrow{L} s_{48}\ \cdots \\
\cdots\ s_{41} \xrightarrow{E} u_{42} \xrightarrow{R} s_{43} \xrightarrow{X} s_{44} \xrightarrow{L} u_{45} \xrightarrow{E} r_{46} \xrightarrow{L} u_{47} \xrightarrow{E} s_{48}\ \cdots \\
\cdots\ s_{41} \xrightarrow{E} u_{42} \xrightarrow{R} s_{43} \xrightarrow{X} s_{44} \xrightarrow{L} u_{45} \xrightarrow{L} u_{46} \xrightarrow{E} u_{47} \xrightarrow{E} s_{48}\ \cdots
\end{array}
$$

Figure 1: Constructing (7) from (6).

of the reduced system.

To construct behavior (4), we restrict M so that its execution consists of a sequence of R steps, followed by an X step, followed by a sequence of L steps. We say that an execution of M is in its *first phase* before X is executed, and in its *second phase* after X is executed. (The terminology comes from the use of reduction to prove serializability of the two-phased locking discipline of database concurrency control.) Intuitively, M receives information from its environment in the first phase, and sends information to its environment in the second phase. Behaviors (3) and (4) are then

$$\cdots\ s_{41} \xrightarrow{R} s_{42} \xrightarrow{E} s_{43} \xrightarrow{X} s_{44} \xrightarrow{E} s_{45} \xrightarrow{E} s_{46} \xrightarrow{L} s_{47} \xrightarrow{L} s_{48}\ \cdots \quad (6)$$

$$\cdots\ s_{41} \xrightarrow{E} u_{42} \xrightarrow{R} u_{43} \xrightarrow{X} u_{44} \xrightarrow{L} u_{45} \xrightarrow{L} u_{46} \xrightarrow{E} u_{47} \xrightarrow{E} s_{48}\ \cdots \quad (7)$$

To obtain (7) from (6), we must move R actions to the right and L actions to the left. We say that action A *right commutes* with action B, and B *left commutes* with A, iff for any states r, s, and t such that $r \xrightarrow{A} s \xrightarrow{B} t$, there exists a state u such that $r \xrightarrow{B} u \xrightarrow{A} t$. If R actions right commute with E actions and L actions left commute with E actions, then we can obtain (7) from (6) by commuting actions as shown in Figure 1. Observe that, since we don't have to commute the X action, $u_{43} = s_{43}$ and $u_{44} = s_{44}$.

Lipton [10] was concerned with pre/postconditions, so he essentially transformed (6) to (5). Doeppner [6] transformed (6) to (7) and observed that the new behavior differs from the original only on states in which the system is in the middle of operation M. In our theorems, we use the behavior (7) to construct the virtual variables $\widehat{v}$ for the behavior (6). The value of $\widehat{v}$ in a state s_i of (6) is defined to be the value of v in a corresponding state $\nu(s_i)$ of (7), where the cor-

$$
\begin{tikzcd}
\cdots & s_{41} \arrow[r, "R"] \arrow[d] & s_{42} \arrow[r, "E"] \arrow[dl] & s_{43} \arrow[r, "X"] \arrow[dl] & s_{44} \arrow[r, "E"] \arrow[drr] & s_{45} \arrow[r, "E"] \arrow[drr] & s_{46} \arrow[r, "L"] \arrow[drr] & s_{47} \arrow[r, "L"] \arrow[dr] & s_{48} \arrow[d] & \cdots \\
\cdots & s_{41} \arrow[r, "E"] & u_{42} \arrow[r, "R"] & s_{43} \arrow[r, "X"] & s_{44} \arrow[r, "L"] & u_{45} \arrow[r, "L"] & u_{46} \arrow[r, "E"] & u_{47} \arrow[r, "E"] & s_{48} & \cdots
\end{tikzcd}
$$

Figure 2: The correspondence ν between states of (6) and of (7).

respondence is shown in Figure 2. For example, $\nu(s_{44}) = u_{46}$, so the value of $\widehat{v}$ in state s_{44} of (6) is the value of v in state u_{46} of (7). Observe that R and L steps leave $\widehat{v}$ unchanged, and the X step changes $\widehat{v}$ the way an M^R step changes v (see (5)).

For an action A, let $\xrightarrow{A^+}$ be the irreflexive transitive closure of $\xrightarrow{A}$, so $s \xrightarrow{A^+} t$ iff there exist states $r_1, \ldots, r_n$ such that $s \xrightarrow{A} r_1 \xrightarrow{A} \cdots \xrightarrow{A} r_n \xrightarrow{A} t$. There is the following relation between a state s_i and its corresponding state $\nu(s_i)$.

- If (in state s_i) M is not currently being executed—states s_{41} and s_{48} in Figure 2—then $s_i = \nu(s_i)$.
- In the first phase (execution of M begun but X not yet executed)—states s_{42} and s_{43} in Figure 2—we have $\nu(s_i) \xrightarrow{R^+} s_i$.
- In the second phase (X executed but M not terminated)—states s_{44} through s_{47} in Figure 2—we have $s_i \xrightarrow{L^+} \nu(s_i)$. (To see that $s_{45} \xrightarrow{L^+} \nu(s_{45})$, observe from Figure 1 that $s_{45} \xrightarrow{L} r_{46} \xrightarrow{L} u_{47}$.)

Observe also that:

- M is not currently being executed in a state $\nu(s_i)$.

The construction of ν described by Figure 2 works only if, once the X step has occurred, the execution of M eventually terminates. The construction can also be made to work if the entire system halts after executing X, as long as we can extend the behavior (6) by adding a finite sequence of L actions that complete the execution of M. Therefore, in the conclusion of our reduction theorems, we must replace (2) with

$$S \wedge Q \;\Rightarrow\; \exists\!\!\exists\, \widehat{v} : \Box I \wedge \widehat{S^R} \tag{8}$$

where Q asserts that, once an X step has occurred, either the execution of M eventually terminates or else the entire system halts in a state in which it is possible to complete the execution of M. Note that we allow behaviors in which execution of M remains forever in its first phase, never taking an X step.

4 Safety in TLA

In TLA, a state is an assignment of values to all flexible variables, and a behavior is a sequence of states. An action is a predicate that may contain primed and unprimed flexible variables. If A is the action $x' = 1 + y$, then $s \xrightarrow{A} t$ is true iff the value assigned to x by state t equals 1 plus the value assigned by state s to y. The canonical form of the safety[3] part of a specification is $Init \wedge \Box[N]_v$, where $Init$ is a state predicate (a formula containing only unprimed flexible variables), N is an action called the *next-state action*, v is the tuple of all flexible variables occurring in $Init$ and N, and $[N]_v$ is an abbreviation for $N \vee (v' = v)$.[4] A behavior $s_1, s_2, \ldots$ satisfies this formula iff $Init$ is true in the initial state s_1 and $s_i \xrightarrow{[N]_v} s_{i+1}$ holds for all i—that is, iff $Init$ holds initially and every step is either an N step or a stuttering step (one that leaves all the relevant variables unchanged).

From now on, we assume that v is the tuple of all flexible variables that appear in our formulas.

The next-state action N is usually written as the disjunction of all the individual atomic actions of the system. For our reduction theorems, N is defined to equal $M \vee E$, where M is the disjunction of the atomic actions of the operation being reduced, and E is the disjunction of the other system actions. We assume two state predicates $\mathcal{R}$ and $\mathcal{L}$, where $\mathcal{R}$ is true when execution of M is in its first phase (M has begun but X has not yet been executed), and $\mathcal{L}$ is true when execution of M is in its second phase (X has been executed but M has not yet terminated). We take $Init$, M, E, $\mathcal{R}$, and $\mathcal{L}$ to be parameters of the theorems. The theorems assume the following hypotheses, which assert that $\mathcal{R}$ and $\mathcal{L}$ are consistent with their interpretations as assertions about the progress of M. The hypotheses are explained below.

$$\begin{array}{llll} \text{(a)} & Init \Rightarrow \neg(\mathcal{R} \vee \mathcal{L}) & \text{(c)} & \neg(\mathcal{L} \wedge M \wedge \mathcal{R}') \\ \text{(b)} & E \Rightarrow (\mathcal{R}' \equiv \mathcal{R}) \wedge (\mathcal{L}' \equiv \mathcal{L}) & \text{(d)} & \neg(\mathcal{R} \wedge \mathcal{L}) \end{array} \tag{9}$$

(a) The system starts with M not in the middle of execution.

(b) Executing an action of the environment cannot change the phase.

[3]Any property is the conjunction of a safety property, which constrains finite behavior, and a liveness property. [2]

[4]For any expression e containing no primes, e' is the expression obtained from e by priming its flexible variables.

(c) Execution of M can't go directly from the second phase to the first phase (without completing the execution).

(d) The two phases are disjoint. This hypothesis is actually unnecessary; given predicates $\mathcal{R}$ and $\mathcal{L}$ that satisfy the other hypotheses, we can satisfy this assumption as well by replacing either $\mathcal{R}$ with $\mathcal{R} \wedge \neg\mathcal{L}$ or $\mathcal{L}$ with $\mathcal{L} \wedge \neg\mathcal{R}$.

We define the actions R, L, and X in terms of M, $\mathcal{R}$, and $\mathcal{L}$ by

$$R \;\triangleq\; M \wedge \mathcal{R}' \qquad L \;\triangleq\; \mathcal{L} \wedge M \qquad X \;\triangleq\; (\neg\mathcal{L}) \wedge M \wedge (\neg\mathcal{R}') \tag{10}$$

That is, an R step is an M step that ends in the first phase, an L step is an M step that starts in the second phase, and an X step is any other M step. Either phase can be empty. Both phases might even be empty, in which case execution of M consists of just a single X step.

We define the sequential composition $A \cdot B$ of actions A and B so that $s \xrightarrow{A \cdot B} t$ iff there exists a state u for which $s \xrightarrow{A} u \xrightarrow{B} t$. Equivalently, $A \cdot B$ equals $\exists\, \mathsf{r} : A(\mathsf{r}/v') \wedge B(\mathsf{r}/v)$, where r is a tuple of rigid variables, $A(\mathsf{r}/v')$ denotes A with each primed variable of v replaced by the corresponding component of r, and $B(\mathsf{r}/v)$ denotes B with each unprimed flexible variable of v replaced by the corresponding component of r. The equivalence of the two definitions is seen by letting r be the tuple of values assigned to the variables in v by the state u. The definition of commutativity given above can be restated as: action A *right commutes* with action B, and B *left commutes* with A, iff $A \cdot B \Rightarrow B \cdot A$. We can then state the commutativity hypotheses we used in the previous section as $R \cdot E \Rightarrow E \cdot R$ and $E \cdot L \Rightarrow L \cdot E$.

We define A^+ to equal $A \vee (A \cdot A) \vee (A \cdot A \cdot A) \vee \ldots$. This defines $s \xrightarrow{A^+} t$ to have the same meaning as above. A complete execution of M is a sequence of M steps starting and ending in states for which M is not in the middle of its execution—that is, in states satisfying $\neg(\mathcal{R} \vee \mathcal{L})$. We therefore define:

$$M^R \;\triangleq\; \neg(\mathcal{R} \vee \mathcal{L}) \wedge M^+ \wedge \neg(\mathcal{R} \vee \mathcal{L})' \tag{11}$$

We define N, N^R, S, and S^R by

$$\begin{array}{lcllcl} N & \triangleq & M \vee E \qquad & S & \triangleq & \mathit{Init} \wedge \Box[N]_v \\ N^R & \triangleq & M^R \vee E \qquad & S^R & \triangleq & \mathit{Init} \wedge \Box[N^R]_v \end{array} \tag{12}$$

Suppose $s \xrightarrow{A} t$. If the tuple of variables v has the value v_s in state s and the value v_t in state t, then the relation $A(v_s/v,\, v_t/v')$, obtained by substituting the elements of v_s for the unprimed flexible variables of A and the elements of v_t for the primed variables of A, holds. We constructed the tuple $\widehat{v}$ of virtual variables by defining a mapping ν on states of a behavior and defining the value of $\widehat{v}$ in a state s to be the tuple of values of v in the state $\nu(s)$. This means that, if $s \xrightarrow{A} \nu(s)$, then the values of v and $\widehat{v}$ in state s satisfy $A(v/v, \widehat{v}/v')$, which is just $A(\widehat{v}/v')$. If $\nu(s) \xrightarrow{A} s$, then the values of v and $\widehat{v}$ in state s satisfy $A(\widehat{v}/v,\ v/v')$. From the four observations above, based on Figure 2, about how s and $\nu(s)$ are related, we obtain the following definition of the relation I between v and $\widehat{v}$:[5]

$$\begin{array}{rcl} I & \stackrel{\Delta}{=} & \land\ \mathcal{R} \Rightarrow R^{+}(\widehat{v}/v,\ v/v') \\ & & \land\ \mathcal{L} \Rightarrow L^{+}(\widehat{v}/v') \\ & & \land\ \neg(\mathcal{R} \lor \mathcal{L}) \Rightarrow (\widehat{v} = v) \\ & & \land\ \neg(\mathcal{R} \lor \mathcal{L})(\widehat{v}/v) \end{array} \qquad (13)$$

5 Liveness in TLA

In temporal logic, $\Box$ means *always* and its dual $\Diamond$, defined to equal $\neg\Box\neg$, means *eventually*. Thus, $\Box\Diamond$ means *infinitely often* and $\Diamond\Box$ means *eventually forever*. Let σ be the behavior $s_1, s_2, \ldots$. For a predicate P, formula $\Box\Diamond P$ is true for σ iff P is true for infinitely many states s_i, and $\Diamond\Box P$ is true for σ iff P is true for all states s_i with $i > n$, for some n. For an action A, formula $\Box\Diamond A$ is true for σ iff $s_i \xrightarrow{A} s_{i+1}$ is true for infinitely many i. To maintain invariance under stuttering, we must write $\Box\Diamond\langle A\rangle_v$ rather than $\Box\Diamond A$, where $\langle A\rangle_v$ is defined to equal $A \land (v' \neq v)$. The formula $\Box\Diamond\langle A\rangle_v$ asserts of a behavior that there are infinitely many nonstuttering A steps.

We define ENABLED A to the be predicate asserting that action A is enabled. It is true of a state s iff there exists some state t such that $s \xrightarrow{A} t$. Equivalently, ENABLED A equals $\exists\, \mathsf{r} : A(\mathsf{r}/v')$, where r is a tuple of rigid variables.

We observed above that the conclusion of a reduction theorem should be (8), where Q asserts that either (i) M must eventually terminate after the X step has occurred, or (ii) the entire system halts in a state in which execution of a finite number of $\mathcal{L}$ steps can complete the execution of M.

To express (i), note that an X step makes $\mathcal{L}$ true, and $\mathcal{L}$ remains true until M terminates.[6] Thus, (i) asserts that $\mathcal{L}$ does not remain true forever, an assertion ex-

[5] We let a list of formulas bulleted with $\land$ or $\lor$ denote the conjunction or disjunction of the formulas, using indentation to eliminate parentheses.

[6] More precisely, an X step either makes $\mathcal{L}$ true or terminates the execution of M.

pressed by $\neg\Diamond\Box\mathcal{L}$, which is equivalent to $\Box\Diamond\neg\mathcal{L}$. We can weaken this condition by allowing the additional possibility that, infinitely often, it is possible to take a sequence of L steps that makes $\mathcal{L}$ false, if such a sequence can lead to only a finite number of possible values of v.

To express (ii), we note that in TLA, halting is described by a behavior that ends with an infinite sequence of stuttering steps, so eventual halting is expressed by $\Diamond\Box[\mathrm{FALSE}]_v$ (which is equivalent to $\Diamond\Box[v' = v]_v$). It is possible to complete the execution of M by taking L steps iff a sequence of L steps can make $\mathcal{L}$ false, which is true iff it is possible to take an L^+ step with $\mathcal{L}$ false in the final state. Thus, condition (ii) can be expressed as $\Diamond\Box([\mathrm{FALSE}]_v \wedge \mathrm{ENABLED}\ (L^+ \wedge \neg\mathcal{L}'))$.

Using the temporal logic tautology $\Diamond\Box(F \wedge G) \equiv (\Diamond\Box F \wedge \Diamond\Box G)$, we define Q by

$$Q \;\triangleq\; \begin{array}{l} \vee\ \Box\Diamond(\neg\mathcal{L}\ \vee\ (\exists!!\,\mathsf{r} : \mathrm{ENABLED}\ ((L^+ \wedge \neg\mathcal{L}')(\mathsf{r}/v')))) \\ \vee\ \Diamond\Box[\mathrm{FALSE}]_v\ \wedge\ \Diamond\Box\mathrm{ENABLED}\ (L^+ \wedge \neg\mathcal{L}') \end{array} \tag{14}$$

where $\exists!!\,\mathsf{r} : F$ means that there exists a finite, nonzero number of values for r for which F holds. We can now state our first reduction theorem, for specifications S that are safety properties.

Theorem 1 *Let $Init$, $\mathcal{R}$, and $\mathcal{L}$ be state predicates; let E and M be actions; and let v be the tuple of all flexible variables that occur free in these predicates and actions. Let R, L, S, S^R, I, and Q be defined by (10)–(14). If*

1. (a) $Init \Rightarrow \neg(\mathcal{R} \vee \mathcal{L})$ (c) $\neg(\mathcal{L} \wedge M \wedge \mathcal{R}')$
   (b) $E \Rightarrow (\mathcal{R}' \equiv \mathcal{R}) \wedge (\mathcal{L}' \equiv \mathcal{L})$ (d) $\neg(\mathcal{R} \wedge \mathcal{L})$
2. (a) $R \cdot E \Rightarrow E \cdot R$ (b) $E \cdot L \Rightarrow L \cdot E$

then $S \wedge Q \;\Rightarrow\; \exists\,\widehat{v} : \Box I \wedge \widehat{S^R}$, where $\widehat{v}$ is a tuple of new variables and $\widehat{\ }$ denotes substitution of the variables $\widehat{v}$ for the variables v.

The specifications S and S^R are safety properties, so it may appear that we are using the liveness property Q to prove that one safety property implies another. We need Q in general because, even though $\Box I \wedge \widehat{S^R}$ is necessarily a safety property, $\exists\,\widehat{v} : \Box I \wedge \widehat{S^R}$ need not be one. Recall that the purpose of a reduction theorem is to deduce properties of S by proving properties of S^R. For the purpose of proving safety properties, we can eliminate Q by adding the hypothesis

$$\mathcal{L} \;\Rightarrow\; \mathrm{ENABLED}\ (L^+ \wedge \neg\mathcal{L}') \tag{15}$$

which asserts that, after executing X, it is always possible to complete the execution of M. Let $\mathcal{C}(\Pi)$ be the strongest safety property implied by property Π, so Π is a safety property iff $\Pi = \mathcal{C}(\Pi)$. (The operator $\mathcal{C}$ is a topological closure operator [1].) Hypothesis (15) implies $\mathcal{C}(S \wedge Q) \equiv S$. Since $\mathcal{C}$ is monotonic ($\Pi \Rightarrow \Phi$ implies $\mathcal{C}(\Pi) \Rightarrow \mathcal{C}(\Phi)$), this proves:

Corollary 2 *With the notations and assumptions of Theorem 1, let* Π *be a safety property. If* $\mathcal{L} \Rightarrow \text{ENABLED}\ (L^+ \wedge \neg\mathcal{L}')$*, then* $(\exists\!\!\exists\, \widehat{v} : \Box I \wedge \widehat{S^R}) \Rightarrow \Pi$ *implies* $S \Rightarrow \Pi$.

6 Reducing Fairness Conditions

Most TLA specifications are of the form $S \wedge F$, where S is as in (12) and F is a liveness condition. We would like to extend the conclusion (8) to

$$S \wedge F \wedge Q \;\Rightarrow\; \exists\!\!\exists\, \widehat{v} : \Box I \wedge \widehat{S^R} \wedge \widehat{F^R} \tag{16}$$

where F^R is a suitable reduced version of F. The liveness condition F is usually expressed as a conjunction of WF (weak fairness) and/or SF (strong fairness) formulas, defined by

$$\begin{aligned} \text{WF}_v(A) &\;\triangleq\; \Diamond\Box\text{ENABLED}\,\langle A\rangle_v \;\Rightarrow\; \Box\Diamond\langle A\rangle_v \\ \text{SF}_v(A) &\;\triangleq\; \Box\Diamond\text{ENABLED}\,\langle A\rangle_v \;\Rightarrow\; \Box\Diamond\langle A\rangle_v \end{aligned}$$

Let's begin by considering the simple case where F equals $\text{WF}_v(A)$, for some action A. (The case $F = \text{SF}_v(A)$ is similar.) In this case, F^R should equal $\text{WF}_v(A^R)$, where A^R is the reduced version of action A. Reduction means replacing the given action M by M^R; it's not clear what the reduced version of an arbitrary action A should be. There are two cases in which the definition is obvious:

- If A is disjoint from M, then $A^R = A$.
- If A includes M, so $A = (A \wedge E) \vee M$, then $A^R = (A \wedge E) \vee M^R$.

We generalize these two cases by taking A^R to be $(A \wedge E) \vee A^R_M$, where an A^R_M step consists of a complete execution of M that includes at least one $A \wedge M$ step. The formal definition is:

$$A^R_M \;\triangleq\; \neg(\mathcal{R} \vee \mathcal{L}) \wedge M^* \cdot (A \wedge M) \cdot M^* \wedge \neg(\mathcal{R} \vee \mathcal{L})' \tag{17}$$

where M^* stands for $[M^+]_v$.

From the definition of WF and a little predicate logic, we see that to prove (16), it suffices to prove:

$$S \wedge Q \;\Rightarrow\; \exists\, \widehat{v} : \Box I \wedge \widehat{S^R} \wedge (\Box\Diamond\langle A\rangle_v \Rightarrow \Box\Diamond\langle \widehat{A^R}\rangle_{\widehat{v}}) \tag{18}$$

$$\Box I \;\wedge\; \Diamond\Box\text{ENABLED}\,\langle \widehat{A^R}\rangle_{\widehat{v}} \;\Rightarrow\; \Diamond\Box\text{ENABLED}\,\langle A\rangle_v \tag{19}$$

(For SF, we must replace $\Diamond\Box$ by $\Box\Diamond$ in (19).) We consider the proofs of (18) and (19) separately.

To prove (18), we must show that if a behavior contains infinitely many $\langle A\rangle_v$ steps, then it contains infinitely many $\langle \widehat{A^R}\rangle_{\widehat{v}}$ steps. To simplify this discussion, we temporarily drop the angle brackets and subscripts. We must show that infinitely many A steps imply infinitely many $\widehat{A^R}$ steps. Those infinitely many A steps must include (i) infinitely many $A \wedge E$ steps or, (ii) infinitely many $A \wedge M$ steps. We consider the two possibilities in turn.

To show that infinitely many $A \wedge E$ steps imply infinitely many $\widehat{A^R}$ steps, it suffices to construct the virtual variables so that each $A \wedge E$ step is a $\widehat{A \wedge E}$ step. We have already constructed the virtual variables so that each E step is also a $\widehat{E}$ step. We must strengthen that construction so an $A \wedge E$ step is also a $\widehat{A \wedge E}$ step. Recall that, in Figure 2, the step $s_{44} \rightarrow s_{45}$ of the top behavior is a $\widehat{E}$ step because the corresponding step $u_{46} \rightarrow u_{47}$ of the bottom behavior is an E step. We must therefore guarantee that if $s_{44} \rightarrow s_{45}$ is an $A \wedge E$ step, then $u_{46} \rightarrow u_{47}$ is also an $A \wedge E$ step. Recalling the construction of the bottom behavior, shown in Figures 1, we see that we can make $u_{46} \rightarrow u_{47}$ an $A \wedge E$ step if R right commutes with $A \wedge E$ and L left commutes with $A \wedge E$. In general, reintroducing brackets and subscripts, we can guarantee that infinitely many $\langle A \wedge E\rangle_v$ steps imply infinitely many $\langle \widehat{A^R}\rangle_{\widehat{v}}$ steps with the additional hypotheses:

$$R \cdot \langle A \wedge E\rangle_v \;\Rightarrow\; \langle A \wedge E\rangle_v \cdot R \qquad \langle A \wedge E\rangle_v \cdot L \;\Rightarrow\; L \cdot \langle A \wedge E\rangle_v$$

These hypotheses are vacuous if $A \wedge E$ equals FALSE. If $A \wedge E$ equals E, they follow from the commutativity conditions we are already assuming.

Step (ii) in proving (18) is showing that if there are infinitely many $A \wedge M$ steps, then there are infinitely many A^R_M steps. It suffices to guarantee that if one of the steps in a complete execution of M is also an A step, then the corresponding $\widehat{M^R}$ step is an A^R_M step. Figure 2 shows that an X step corresponds to a $\widehat{M^R}$ step because its starting state s satisfies $\nu(s) \xrightarrow{R^+} s$, its ending state t satisfies $t \xrightarrow{R^+} \nu(t)$, and M is not in the middle of its execution in states $\nu(s)$ and $\nu(t)$.

If the X step is an $A \wedge X$ step, then it is clear that the corresponding $\widehat{M^R}$ step is an A^R_M step. Suppose that one of the R steps is an $A \wedge R$ step, and let R^+_A equal $R^* \cdot (A \wedge R) \cdot R^*$. The $\widehat{M^R}$ step will be an A^R_M step if the starting state s of the X step satisfies $\nu(s) \xrightarrow{R^+_A} s$. Figure 1 shows that we can construct ν to satisfy this condition if we can interchange $A \wedge R$ and E actions—that is, if $A \wedge R$ (as well as R) right commutes with E. Similarly, when one of the L steps is an $A \wedge L$ step, we can guarantee that the $\widehat{M^R}$ step is an A^R_M step if $A \wedge L$ (as well as L) left commutes with E. Putting the brackets and subscripts in, we see that infinitely many $\langle A \wedge M \rangle_v$ steps imply infinitely many $\widehat{A^R}$ steps if

$$\langle A \wedge R \rangle_v \cdot E \;\Rightarrow\; E \cdot \langle A \wedge R \rangle_v \qquad E \cdot \langle A \wedge L \rangle_v \;\Rightarrow\; \langle A \wedge L \rangle_v \cdot E$$

These hypotheses are vacuous if $A \wedge M$ equals FALSE. If $A \wedge M$ equals M, they follow from the commutativity conditions we are already assuming.

The argument we just made assumes that each execution of M terminates. For example, a behavior might contain infinitely many $A \wedge R$ steps but no X steps, in which case there would be no A^R_M steps. We need the assumption that if there are infinitely many $A \wedge M$ steps, then there are infinitely many X steps. So, we replace (18) with

$$S \wedge Q \wedge O \;\Rightarrow\; \exists\, \widehat{v} : \Box I \wedge \widehat{S^R} \wedge (\Box\Diamond\langle A \rangle_v \Rightarrow \Box\Diamond\langle \widehat{A^R} \rangle_{\widehat{v}}) \tag{20}$$

where O equals $\Delta \wedge \Box\Diamond\langle A \wedge M \rangle_v \Rightarrow \Box\Diamond\langle X \rangle_v$.

Finally, we showed only that infinitely many $\langle A \rangle_v$ steps imply infinitely many $\widehat{A^R}$ steps, which are not necessarily $\langle \widehat{A^R} \rangle_{\widehat{v}}$ steps. We need to rule out the degenerate case in which those $\widehat{A^R}$ steps are stuttering steps that leave $\widehat{v}$ unchanged. We do this by assuming $(\langle A \rangle_v)^R_M \Rightarrow (v' \neq v)$. In most cases of interest, M^R implies $v' \neq v$. so $(\langle A \rangle_v)^R_M \Rightarrow (v' \neq v)$ holds for any A.

A specification can contain a (possibly infinite) conjunction of fairness properties, so we must generalize from a single action A to a collection of actions A_i, for i in some set $\mathcal{I}$. The definitions above are generalized to

$$\begin{aligned} A^R_i &\triangleq (A_i \wedge E) \vee (A_i)^R_M \\ O &\triangleq \forall\, i \in \mathcal{I} : \Box\Diamond\langle A_i \wedge M \rangle_v \Rightarrow \Box\Diamond\langle X \rangle_v \end{aligned} \tag{21}$$

The theorem whose conclusion is the generalization of (20) is:

Theorem 3 *With the notation and assumptions of Theorem 1, let A_i be an action, for all i in a finite or countably infinite set $\mathcal{I}$, and let $(A_i)^R_M$, A^R_i, and O be defined by (17) and (21). If, in addition,*

2. (c) $\forall\, i \in \mathcal{I} \,:\, R \cdot \langle A_i \wedge E \rangle_v \;\Rightarrow\; \langle A_i \wedge E \rangle_v \cdot R$
   (d) $\forall\, i \in \mathcal{I} \,:\, \langle A_i \wedge E \rangle_v \cdot L \;\Rightarrow\; L \cdot \langle A_i \wedge E \rangle_v$
   (e) $\forall\, i \in \mathcal{I} \,:\, \langle A_i \wedge R \rangle_v \cdot E \;\Rightarrow\; E \cdot \langle A_i \wedge R \rangle_v$
   (f) $\forall\, i \in \mathcal{I} \,:\, E \cdot \langle A_i \wedge L \rangle_v \;\Rightarrow\; \langle A_i \wedge L \rangle_v \cdot E$
   (g) $\forall\, i \in \mathcal{I} \,:\, (A_i)^R_M \Rightarrow (v' \neq v)$

then $S \wedge Q \wedge O \;\Rightarrow\; \exists\, \widehat{v} \,:\, \Box I \wedge \widehat{S^R} \wedge (\forall\, i \in \mathcal{I} \,:\, \Box\Diamond\langle A_i \rangle_v \Rightarrow \Box\Diamond\langle \widehat{A_i^R} \rangle_{\widehat{v}})$.

To prove (19) and its analog for SF, it suffices to prove

$$I \;\wedge\; \textsc{Enabled}\; \langle \widehat{A^R} \rangle_{\widehat{v}} \;\Rightarrow\; \textsc{Enabled}\; \langle A \rangle_v$$

This can be done with the following result, which is a simple consequence of the definition of I.

Proposition 4 *Let I be defined by (13). For any state predicates $\mathcal{P}$ and $\mathcal{Q}$, if*

(a) $\mathcal{P} \Rightarrow \mathcal{Q}$ (b) $\mathcal{Q} \wedge R \Rightarrow \mathcal{Q}'$ (c) $L \wedge \mathcal{Q}' \Rightarrow \mathcal{Q}$

then $I \wedge \widehat{\mathcal{P}} \Rightarrow \mathcal{Q}$, where $\widehat{\ }$ is defined as in Theorem 1.

Combining this proposition with the definitions of WF and SF proves the following corollary to Theorem 3.

Corollary 5 *With the notations and assumptions of Theorem 3, if*

3. (a) $\forall\, i \in \mathcal{I} \,:\, \textsc{Enabled}\; \langle A_i^R \rangle_v \Rightarrow \textsc{Enabled}\; \langle A_i \rangle_v$
   (b) $\forall\, i \in \mathcal{I} \,:\, (\textsc{Enabled}\; \langle A_i \rangle_v) \wedge R \;\Rightarrow\; (\textsc{Enabled}\; \langle A_i \rangle_v)'$
   (c) $\forall\, i \in \mathcal{I} \,:\, L \wedge (\textsc{Enabled}\; \langle A_i \rangle_v)' \;\Rightarrow\; \textsc{Enabled}\; \langle A_i \rangle_v$

then

$$\begin{aligned} &S \;\wedge\; (\forall\, i \in \mathcal{I} \,:\, \mathrm{XF}_v(A_i)) \;\wedge\; Q \;\wedge\; O \\ &\quad\Rightarrow\; \exists\, \widehat{v} \,:\, \Box I \;\wedge\; \widehat{S^R} \;\wedge\; (\forall\, i \in \mathcal{I} \,:\, \mathrm{XF}_{\widehat{v}}(\widehat{A_i^R})) \end{aligned}$$

where $\mathrm{XF}_v(A_i)$ *is either* $\mathrm{WF}_v(A_i)$ *or* $\mathrm{SF}_v(A_i)$.

Hypothesis 3(a) holds automatically for each i such that $A_i \wedge M$ equals FALSE or M, the two cases that inspired our definition of A_i^R. It is this hypothesis that most severely limits the class of actions A_i to which we can apply the corollary. In applying the theorem or the corollary, we expect the specification's fairness properties to imply $Q \wedge O$.

7 Proofs

We now briefly describe how our results are proved; complete proofs will appear elsewhere. Theorem 1 follows from Theorem 3 by letting $\mathcal{I}$ be the empty set. We already observed how Corollary 2 is proved by showing that (15) implies $\mathcal{C}(S \wedge Q) \equiv S$, a result that follows directly from the definition of $\mathcal{C}$ [1]. Proposition 4 is proved by a straightforward calculation based on the definitions of I and of the $^+$ operator; it easily proves Corollary 5. This leaves Theorem 3.

In Section 3 we sketched an intuitive proof of (8). Section 6 indicated how we can extend that proof to a proof of Theorem 3 for a single fairness condition—that is, when $\mathcal{I}$ contains a single element. We used hypotheses 2 to commute $A \wedge M$ or $A \wedge E$ steps. In the general case, we have the extra difficulty that the hypotheses do not allow us simultaneously to commute all the A_i steps. When extending the construction shown in Figure 1, we must choose a single A_i to commute at each step. The choice must be made in such a way that every A_i that is executed infinitely often is chosen infinitely often.

This proof sketch can be turned directly into a semantic proof of Theorem 3. The theorem can also be proved using only the rules of TLA, with no semantic reasoning. The key idea is to introduce a history variable that gives the value of $\widehat{v}$ when $\mathcal{R}$ is true (before X is executed) and a prophecy variable that gives the value of $\widehat{v}$ when $\mathcal{L}$ is true (after X is executed). (History and prophecy variables are explained in [1].) In addition, we need a new type of infinite prophecy variable that tells which disjunct of Q holds, as well as history and prophecy variables that choose, at each point in the construction, which A_i to commute.

8 Further Remarks

We often want to use an invariant Inv of the specification S to verify the hypotheses of the theorems. For example, when proving that R right commutes with E, we want to consider only states satisfying Inv. With TLA, it isn't necessary to weaken the hypotheses to take account of an invariant. Instead, we apply the general rule

$$\Box Inv \;\Rightarrow\; (\Box[A]_v \equiv \Box[Inv \wedge A \wedge Inv']_v)$$

Thus, if S implies $\Box Inv$, then we can replace M and E by $Inv \wedge M \wedge Inv'$ and $Inv \wedge E \wedge Inv'$.

Many TLA specifications are of the form $\exists\, w : S \wedge F$, where w is a tuple of "internal variables". Since one proves $(\exists\, w : S \wedge F) \Rightarrow \Pi$ by proving $S \wedge F \Rightarrow \Pi$

(renaming variables if necessary), it suffices to reduce $S \wedge F$. Thus, we can ignore existential quantification (hiding) when applying a reduction theorem.

References

[1] Martín Abadi and Leslie Lamport. The existence of refinement mappings. *Theoretical Computer Science*, 82(2):253–284, May 1991.

[2] Bowen Alpern and Fred B. Schneider. Defining liveness. *Information Processing Letters*, 21(4):181–185, October 1985.

[3] R. J. R. Back. Refining atomicity in parallel algorithms. Reports on Computer Science and Mathematics Ser. A, No 57, Swedish University of Åbo, February 1988.

[4] Ernie Cohen. *Compositional Proofs of Asynchronous Programs*. PhD thesis, University of Texas at Austin, May 1993.

[5] Ernie Cohen. A guide to reduction. Technical Report TM-ARH-023816, Bellcore, 1993. Available from the author at `ernie@bellcore.com`.

[6] Thomas W. Doeppner, Jr. Parallel program correctness through refinement. In *Fourth Annual ACM Symposium on Principles of Programming Languages*, pages 155–169. ACM, January 1977.

[7] David Gries and Ivan Stojmenović. A note on gramham's convex hull algorithm. *Information Processing Letters*, 25(5):323–327, July 1987.

[8] Leslie Lamport. The temporal logic of actions. *ACM Transactions on Programming Languages and Systems*, 16(3):872–923, May 1994.

[9] Leslie Lamport and Fred B. Schneider. Pretending atomicity. Research Report 44, Digital Equipment Corporation, Systems Research Center, May 1989.

[10] Richard J. Lipton. Reduction: A method of proving properties of parallel programs. *Communications of the ACM*, 18(12):717–721, December 1975.

[11] S. Owicki and D. Gries. An axiomatic proof technique for parallel programs I. *Acta Informatica*, 6(4):319–340, 1976.