ON A FIXPOINT SEMANTICS AND THE DESIGN OF PROOF RULES FOR FAIR PARALLEL PROGRAMS

Charanjit S. Jutla and Josyula R. Rao

Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712-1188

TR-92-23 April 1992

On a Fixpoint Semantics and the Design of Proof Rules for Fair Parallel Programs

Charanjit S. Jutla
IBM T.J. Watson Research Center
Yorktown Heights, New York 10598

Josyula R. Rao*
Department of Computer Sciences
The University of Texas at Austin
Austin, Texas 78712-1188

April 17, 1992

Abstract

In this paper, we present a predicate transformer approach to the semantics of parallel programs. The meaning of a program is given by two components – a predicate transformer **wlt** characterizing the progress properties of the program and a predicate transformer **wsafe** concerned with safety properties.

We illustrate the utility of our semantics by developing a simple and powerful framework for the systematic design of proof rules for progress for a range of fairness assumptions including pure nondeterminism, unconditional fairness, minimal progress, weak fairness and strong fairness. Beginning with an intuitive branching time temporal logic (CTL^*) formula characterizing progress for the fairness notion being considered, we obtain a simple fixpoint characterization of **wlt**. We use this fixpoint characterization to extract a simple UNITY–like proof rule for proving progress under the aforementioned fairness constraint. A key feature of our framework is that by merely checking a set of simple conditions, the soundness and completeness of the proof rule is easily guaranteed. It is to be noted that unlike previous work on proof rules for fairness, our meta–theoretic arguments are conducted without resorting to any complicated machinery such as ordinals. Further, the UNITY–like formulation of the proof rule enables one to use the UNITY theory of progress when designing programs based on different notions of fairness.

The fixpoint characterizations of **wlt** for the fairness constraints share a common structure – they are all expressed in terms of a simpler predicate transformer, called the *generalized weakest precondition* (**gwp**). We motivate a notion of completeness for progress under fairness constraints and show that it is subject to the same assumptions and constraints as Hoare triples. We define a simple notion of program composition and show that for all notions of fairness considered (excepting strong fairness) **gwp** and **wsafe** are compositional. Finally, we show that two programs have the same predicate transformer semantics (in terms of "wlt" and "wsafe") iff they agree on all formulae of a fair version of the full branching time logic CTL* without stuttering. This means that two semantically equivalent programs agree on a rich class of properties.

*Supported by the Texas Advanced Research Program under Grant No. 003658-065 and by the Office of Naval Research, Contract N00014-90-J-1640.

Contents

0 Introduction

0.1 Background

In 1967, in a pioneering paper, Floyd [Flo67] introduced the idea of inductive assertions for proving the correctness of sequential programs. In his approach, programs were specified as flowcharts and the task of verifying them was reduced to proving two kinds of properties about them — *invariance* (or partial correctness) assertions and *termination* (or total correctness) assertions. In 1969, Hoare [Hoa69] and Manna [Man69] proposed proof systems for proving partial correctness assertions of Algol–like programs. In a sense, these proof rules provided an *axiomatic* semantics of the language.

While the axiomatic definition of the programming language was adequate for the design of sequential programs, Park [Par70] suggested a simpler semantics for sequential programs : in his approach, the meaning of a program was a straightforward "set-theoretic" (fixpoint) characterization of the partial functions computed by simple recursive programs. In a similiar vein, Dijkstra [Dij75] proposed a simple and elegant semantics of while-loop programs by postulating that the performance of a mechanism S is sufficiently well-known, provided we can derive for any postcondition R the corresponding weakest precondition, $\mathbf{wp}\,.S.R$. Both authors stressed the point that an approach based on "predicate transformers" would avoid a detailed (and usually unnecessary) description of the intermediate states of program execution.

In the seventies, the realization of the importance of nondeterminism and parallelism prompted researchers to try to extend verification techniques to cover programs incorporating these features as well. Preliminary efforts [Kel76; Lam77; OG76a; OG76b] concentrated on proving partial correctness or the so-called *safety* properties. While they were successful in this respect, their achievements with regard to proving termination were modest. This was due to two reasons. Firstly, unlike sequential programs, parallel programs are ongoing and non-terminating and for such programs, general progress properties (like reachability), rather than the special case of termination, are of interest. Secondly, it was recognized that an essential ingredient of proofs of progress for parallel programs is *fairness*. Simply put, an assumption of fairness guarantees that every process is scheduled for execution *sufficiently often*, regardless of the state of the other processes. *Sufficiently often* was variously defined giving rise to differing notions of fairness. It soon came to be recognized that reasoning with fairness assumptions was akin to reasoning about unbounded nondeterminism and this meant that the standard techniques of induction with integer-valued metrics would not work in proving termination properties [AP86; LPS81].

In the early eighties, several papers proposing proof rules based on a variety of fairness notions — unconditional fairness, weak fairness, strong fairness, extreme fairness etc — appeared in the literature [Par80; GPSJ80; GFMR81; LPS81; Par81b; OL82; Pnu83; QS83; APS84; SdRG89] (see [Fra86] for a survey). The work closest in philosophy and spirit to ours, is the work of Park [Par81b] where a fixpoint characterization for proving termination under weak fairness was given.

In addition to work on proof rules for proving useful properties of parallel programs (under fairness assumptions), attention has been focussed on formulating a semantics for parallel programs. Several researchers have characterized a parallel program by a set of possible behaviors in an effort to capture its semantics. In [Par81a], Park characterized such sets for finite state programs in terms of ω-regular languages and showed that such sets are closed under fair parallel composition. In [BKP84], the authors use temporal logic to give a characterization of the semantics of parallel programs. Similarly, in [Mos89], Moschovakis gives the semantics in terms of winning strategies of players in certain games. However, in all these approaches, the idea has been to capture the set of all fair execution sequences of a program. Apart from carrying undue detail of the actual evaluating mechanism, such semantics suffer from "stuttering" [Lam77]. Attempts at avoiding stuttering have either led to complicated machinery – like the temporal logic of reals [BKP86] or improper handling of divergent processes.

0.2 The Problem

Our notion of a program is borrowed from UNITY [CM88] : a program is essentially a set of statements. Operationally speaking, program *execution* starts in *any* state and proceeds by repeatedly selecting a statement from the set and executing it with the selection being governed by some fairness constraint. This simple notion of a program has proved to be sufficiently powerful to express a wide spectrum of both terminating and reactive programs as shown in [CM88].

We are interested in two kinds of properties of our program, namely, *safety* and *progress* properties. We use the *weak-until* operator to express safety properties. For state predicates X and Y, a program satisfies X *weak-until* Y, if once X is true the program remains in a state satisfying X as long as Y is false. We use the *leads-to* operator [Lam77] to express progress properties. For state predicates X and Y, a program satisfies the progress property X *leads-to* Y, if along *all* fair execution sequences, if the program is in a state in which X is true, it is or will be in a state satisfying Y. In developing proof systems for safety and progress, we will be interested in the following questions.

- *Proof Rule*: Given that the execution of a program satisfies some fairness constraint, what is the proof rule for proving progress properties of the program ?
- *Methodology for Proof Rule Design*: Given an arbitrary fairness constraint, is there a systematic way of *deriving* a proof rule for it ? One would expect a methodology for proof-rule design to have the following characteristics.
  - *Applicability*: One should be able to design proof rules for a wide range of fairness notions.

  - *Meta–theory*: The soundness and completeness of the proof rules should be easy to guarantee. In particular, one should not redo such meta–theoretic proofs for each notion of fairness. Further the arguments for soundness and completeness should be simple and should avoid the use of complicated theoretical machinery such as ordinals.

- *Compositionality*: Is the proof rule *compositional* ? That is, given that a property has been proved for a program F, is there a simple way to infer that the same property holds when program F is composed with another program G without having to *reprove* the property for F ?

- *Methodology for Program Design*: In [CM88], Chandy and Misra introduce a small and powerful set of rules to manipulate their progress operators (**ensures** and *leads–to*) during specification refinement. A careful choice of operators and proof rules could simplify program design when using fairness notions other than the unconditional fairness of UNITY.

0.3 Contributions

In the spirit of [Par70] and [Dij75] we give a predicate transformer semantics for parallel programs. Formally the meaning of a program is given by two components – a predicate tranformer for safety and another for progress. The first component – $\mathbf{wsafe}.X.Y$ – is a function from pairs of predicates to predicates. Intuitively, $\mathbf{wsafe}.X.Y$ is the weakest predicate Z stronger than $X \vee Y$ such that Z *weak-until* Y holds. We give a simple fixpoint definition of **wsafe**. The second component of our meaning function captures progress. It is given by a predicate transformer that for a given predicate X and fairness constraint formalizes the notion of the **w**eakest predicate that **l**eads-**t**o X, also called **wlt**.X.

Beginning with an intuitively understandable definition of progress under a fairness constraint, expressed as a formula of the branching time temporal logic $CTL^{\star}$, we develop simple fixpoint characterizations of $\mathbf{wlt}.X$ for a range of fairness constraints, namely pure nondeterminism, unconditional fairness, minimal progress, weak fairness and strong fairness. We show that the various definitions of **wlt** enjoy a common structure – they can all be uniformly defined in terms of a simpler predicate transformer – $\mathbf{gwp}.s.X$ – the *generalized weakest precondition* of statement s with respect to predicate X. Our definitions are new, simpler than those known in the literature [Par80; SdRG89] and constitute a contribution by themselves.

The fixpoint characterizations of the **wlt** predicate transformer serve as a basis for the extraction of simple UNITY–style proof systems for proving progress under all the aforementioned fairness assumptions. While these proof rules are simpler than those currently known in the literature, they are (as would be expected) logically equivalent to the proof rules suggested in [MP84]. Further, they have the additional methodological advantage that their UNITY–like formulation enables one to use the UNITY theory of the *leads–to* in the design and derivation of programs based on different notions of fairness. Thus we show a *systematic* way of designing proof rules for programs executing under assumptions of unbounded nondeterminism. The rules constitute a second contribution of our paper.

A third contribution of our work is a simple framework to conduct meta–theoretic arguments about proof rules for progress. Specifically, we show that if the relations defined by the proof rules and the corresponding fixpoint characterizations meet a simple and easily–checked set of conditions, then the soundness and completeness of the proof rules are easily guaranteed. Further our proofs of soundness and completeness are conducted in a disciplined and calculational manner without resorting to any complicated machinery (including transfinite ordinals).

Note that it is well–known that the ordinal constants *cannot* be eliminated in proofs of progress properties of programs assuming unbounded nondeterminism. However, it was not known whether ordinals constants were necessary in a meta–theoretic argument about such proof rules. In particular, all arguments about the soundness and completeness of proof rules for fairness that we have seen have explicitly used ordinals. We believe that we are the first to eliminate them.

We give *two* arguments for the soundness and completeness of our proof rules. A non–constructive argument (in Sections 2 and 3) uses fixpoint operators but *no* ordinals and a constructive argument (in Section 5) that uses both fixpoint operators and ordinals (as one would expect).

In many treatments of meta–theoretic arguments about fairness, assumptions about the complexity of the assertion language, expressibility of weakest preconditions and relative completeness are tacitly made and not explicated. By drawing comparisons to Hoare triples we show that a notion of completeness of a proof rule for *leads–to* must necessarily be restricted and subject to the same constraint of *relative completeness in the sense of Cook*. We then construct a proof of completeness of *leads–to* under unconditional fairness paying special attention to the roles played by the issues mentioned above.

In the second last section of the paper, we restrict our attention to unconditional and weak fairness and show that that our semantics, in terms of **wsafe** and **gwp**$.s$ are compositional.

In the last section of the paper, we restrict our attention to unconditional and weak fairness. We show that our semantics allow us to state and prove in a relative sense the *strong equivalence* property (see [Par70]) between programs (namely, they have the same operational meaning). To this end, we define operational semantics using a version of branching time temporal logic (which is insensitive to stuttering), and show that two programs have the same predicate transformers iff they have the same operational semantics. In particular, we show that the μ-calculus of **wlt** and **wsafe** is as expressive as a fair version of a substantial subset of branching time logic CTL* without stuttering.

A restricted version of Section 2 (using ordinals and more conditions) appeared in [JKR89]. Since then we have greatly simplified and generalized our framework to deal with more general notions of fairness and have obtained a compositional semantics.

0.4 Plan of paper

The rest of the paper is organized as follows. After introducing our notational conventions and proof format in section 1, we develop a general framework for fairness arguments in section 2. In section 3, we outline a methodology for the design of proof rules. In the following section, we introduce our computational model and show how the framework of section 3 can be applied to develop sound and relatively complete proof rules for pure nondeterminism, unconditional fairness, minimal progress, weak fairness and strong fairness. In section 5, we show that a notion of completeness for *leads-to* must be subject to the same assumptions and constraints as a system for proving Hoare triples. We present a completeness proof for unconditional fairness with special attention to these details. After introducing the predicate transformer **wsafe** in section 6, we present results on compositionality in section 7. In the last section the paper, we define an operational semantics for our programs using a fair version of branching time temporal logic (which is insensitive to stuttering), and show that two programs have the same predicate transformers iff they have the same operational semantics. Detailed proofs of all theorems appear in the appendix.

1 Preliminaries

1.1 Notational Conventions

We will use the following convention : the expression

$$\langle \underline{Q} x : r.x : t.x \rangle$$

where $\underline{Q}$ denotes either universal ($\forall$) or existential ($\exists$) quantification, x is called the *dummy*, $r.x$ the *range* and $t.x$ the term of the quantification. The expression denotes quantification over all terms for which the dummy satisfies the range.

Universal quantification over all the program variables is denoted by the square brackets ([], read *everywhere*). This operator takes a predicate as an argument and returns a boolean value. It enjoys all the properties of universal quantification over a non-empty domain. The operator was introduced in [DS90] and the interested reader is referred to it for a detailed discussion.

We assume familiarity with the usual boolean and arithmetic operators. The operators we use are summarized below, ordered by increasing binding power.

$$\equiv, \not\equiv$$
$$\Leftarrow, \Rightarrow$$
$$\models\!\!\Rightarrow, \mapsto, \leadsto, \text{ etc.}$$
$$\wedge, \vee$$
$$\neg$$
$$=, \neq, \leq, <, \geq, >$$
$$+, -$$
$$\text{“.” (function application)}$$

For binary relations **R** and **S** on predicates, we say that **R** is *stronger* than **S** (or **S** is *weaker* than **R**) if and only if $\langle \forall X, Y :: X\,\mathbf{R}\,Y \Rightarrow X\,\mathbf{S}\,Y \rangle$ For unary predicate transformers f and g, we say that f is *stronger* than g (or g is *weaker* than f) if and only if $\langle \forall X :: [f.X \Rightarrow g.X] \rangle$

1.2 On Extreme Solutions of Equations

For a set of equations E in the unknown x, we write $x : E$ to explicate the dependence on the unknown x. Given an ordering $\Rightarrow$ on the solutions of E, y is the *strongest solution* (*or the least fixpoint of* E) if and only if

1. y solves E
2. $\langle \forall z : z \,\text{solves}\, E : y \Rightarrow z \rangle$

The *weakest solution* (*or the greatest fixpoint*) of E can be defined in a similiar manner.

We extend the notation for quantification, introduced earlier, to cover extremal solutions as follows. We allow expressions of the form

$$\langle \underline{Q}x :: E \rangle$$

where $\underline{Q}$ can be one of μ and ν to denote the strongest and weakest solution of $x : E$ respectively.

Finally, we will use the following form of the Theorem of Knaster–Tarski [DS90], to prove properties of extremal solutions.

Theorem 0 (Knaster–Tarski) *For monotonic f, the equation*

$$Y : [f.Y \equiv Y]$$

has a strongest and a weakest solution. Furthermore, it has the same strongest solution as

$$Y : [f.Y \Rightarrow Y]$$

and the same weakest solution as

$$Y : [f.Y \Leftarrow Y].$$

Theorem 1 *For monotonic f, let g.X be the strongest solution and h.X the weakest solution of*

$$Y : [f.X.Y \equiv Y].$$

Then both g and h are monotonic.

1.3 Proof Format

We render our proofs in a format introduced in [DS90]. In this format, each step of the proof consists of a syntactic transfomation and is recorded in detail with a hint justifying the transformation. Thus the structure of the proof encourages a calculational style of reasoning.

For example, a proof of $[A \equiv D]$ would be presented as

$$\begin{array}{ll} & A \\ = & \{\text{hint why } [A \equiv B]\} \\ & B \\ = & \{\text{hint why } [B \equiv C]\} \\ & C \\ = & \{\text{hint why } [C \equiv D]\} \\ & D \end{array}$$

We also allow other transitive operators in the leftmost column. Among these are the more traditional *implies* ($\Rightarrow$) but also for reasons of symmetry, *follows-from* ($\Leftarrow$). For a more treatment of this subject the reader is referred to [DS90].

2 A Framework for Fairness Arguments

2.1 A Relational Approach

Let $\mathcal{E}$ be a given binary relation on state predicates; $\mathcal{E}$ describes the basic progress properties of the program. Assume that $\mathcal{E}$ satisfies the following basic requirement.

$$\text{(E0)} \qquad [X \Rightarrow Y] \;\Rightarrow\; (X \;\mathcal{E}\; Y)$$

Consider now the set L of equations L0–L2 in the unknown relation $\triangleright$:

$$\begin{array}{ll} \text{(L0)} & (X \;\mathcal{E}\; Y) \;\Rightarrow\; (X \triangleright Y) \\ \text{(L1)} & (X \triangleright Y) \wedge (Y \triangleright Z) \;\Rightarrow\; (X \triangleright Z) \\ \text{(L2)} & \langle \forall X : X \in W : X \triangleright Y \rangle \;\Rightarrow\; (\langle \exists X : X \in W : X \rangle \triangleright Y) \end{array}$$

L0 states that the $\mathcal{E}$ relation is a subset of the relation $\triangleright$. Transitivity of $\triangleright$ is expressed by L1. L2 means that $\triangleright$ is disjunctively closed over W, where W is an arbitrary set of predicates.

With "$\Rightarrow$" as a partial order on relations we show that the set L of equations in the unknown relation $\triangleright$ has a strongest solution. The proof of the Lemma is due to Edsger W. Dijkstra.

Lemma 0 *There is a unique strongest solution of* $\triangleright : L$.

Proof (of Lemma 0): We show that the conjunction of all solutions of L is a solution of L. Obviously this is the strongest solution of L, since it implies all solutions. (End of Proof)

Definition 0 *The unique strongest solution of L is called* leads-to ($\leadsto$).

Remark : Notice that the $\leadsto$ only depends on $\mathcal{E}$. Since this is the only parameter, giving an interpretation for $\mathcal{E}$ completely determines $\leadsto$. (*End of Remark*)

Using E0 and L0 we now observe:

Lemma 1 $\quad [X \Rightarrow Y] \;\Rightarrow\; X \leadsto Y$

As a Corollary of Lemma 1 we get the reflexivity of $\leadsto$.

Corollary 0 $\quad X \leadsto X$

2.2 A Predicate Transformer Approach

Assume that we are given a unary predicate transformer $\mathbf{gwp}.s.X$ for each statement s of a parallel program. Intuitively, the predicate transformer captures the g*eneralized* w*eakest* p*recondition that establishes predicate* X, by a single helpful transition of the program, in the presence of other (possibly conflicting) transitions.

Given this interpretation, it is natural to expect the predicate transformer $\mathbf{gwp}.s$ to be monotonic. That is,

$$\text{(P0)} \qquad [X \;\Rightarrow\; Y] \Rightarrow [\mathbf{gwp}.s.X \;\Rightarrow\; \mathbf{gwp}.s.Y]$$

Now consider the equation :

$$Z : [Z \;\equiv\; Y \vee \langle \exists s :: \mathbf{gwp}.s.Z \rangle]$$

Note that from the monotonicity of $\mathbf{gwp}.s$, $\exists$ and $\vee$, the right-hand side of the equivalence is monotonic. Thus by the Knaster-Tarski Theorem, the equation has a strongest solution. Let this solution be called $\mathbf{wlt}.Y$. Since $\mathbf{wlt}.Y$ is the strongest solution, it satisfies the following conditions :

$$\begin{array}{ll} \text{(W0)} & [\,\mathbf{wlt}.Y \;\equiv\; Y \vee \langle \exists s :: \mathbf{gwp}.s.(\mathbf{wlt}.Y)\rangle\,] \\ \text{(W1)} & [Y \vee \langle \exists s :: \mathbf{gwp}.s.Z \rangle \;\Rightarrow\; Z] \;\Rightarrow\; [\,\mathbf{wlt}.Y \;\Rightarrow\; Z\,] \end{array}$$

Furthermore, since the equation defining $\mathbf{wlt}.Y$ is monotonic, by Theorem 1, **wlt** is monotonic as well.

Remark : Notice that the **wlt** only depends on $\mathbf{gwp}.s$. Since this is the only parameter, giving an interpretation for $\mathbf{gwp}.s$ completely determines **wlt**. (*End of Remark*)

We now enumerate, without proof, some useful properties of **wlt**.

$$
\begin{array}{ll}
\text{(W2)} & [Y \Rightarrow \mathbf{wlt}.Y] \\
\text{(W3)} & [\langle \exists s :: \mathbf{gwp}.s.(\mathbf{wlt}.Y)\rangle \Rightarrow \mathbf{wlt}.Y] \\
\text{(W4)} & [X \Rightarrow Y] \Rightarrow [\mathbf{wlt}.X \Rightarrow \mathbf{wlt}.Y] \\
\text{(W5)} & [\mathbf{wlt}.(\mathbf{wlt}.Y) \equiv \mathbf{wlt}.Y]
\end{array}
$$

Having finished our investigation into the properties of **wlt**, we are now in a position to relate it to $\mathbf{gwp}.s$.

Lemma 2 $\langle \forall X :: [\langle \exists s :: \mathbf{gwp}.s.X\rangle \Rightarrow \mathbf{wlt}.X]\rangle$

Proof (of 2):

$$
\begin{array}{cl}
 & [\langle \exists s :: \mathbf{gwp}.s.X\rangle \Rightarrow \mathbf{wlt}.X] \\
\Leftarrow & \{\text{property W3}\} \\
 & [\langle \exists s :: \mathbf{gwp}.s.X\rangle \Rightarrow \langle \exists s :: \mathbf{gwp}.s.(\mathbf{wlt}.X)\rangle] \\
\Leftarrow & \{\text{predicate calculus}\} \\
 & \langle \forall s :: [\mathbf{gwp}.s.X \Rightarrow \mathbf{gwp}.s.(\mathbf{wlt}.X)]\rangle \\
\Leftarrow & \{\text{monotonicity of } \mathbf{gwp}.s\} \\
 & \langle \forall s :: [X \Rightarrow \mathbf{wlt}.X]\rangle \\
= & \{\text{property W2}\} \\
 & \mathit{true}
\end{array}
$$

(End of Proof)

2.3 Constraints on *leads-to* and wlt

Our goal is to relate the relation $\leadsto$ to the predicate transformer **wlt**. To attain this end, we require that $\leadsto$ and **wlt** respect certain consistency constraints. With $\mathcal{E}$ and $\leadsto$ as defined in Section 2.1 and $\mathbf{gwp}.s$ and **wlt** as defined in Section 2.2, these constraints can be stated as

$$
\begin{array}{ll}
\text{(C0)} & (X \;\mathcal{E}\; Y) \Rightarrow [X \Rightarrow \mathbf{wlt}.Y] \\
\text{(C1)} & \langle \exists s :: \mathbf{gwp}.s.X\rangle \leadsto X
\end{array}
$$

2.4 The predicate transformer led

In order to illustrate the connection between the relation $\leadsto$ and the predicate transformer **wlt** we need one more concept from [Kna88]. We introduce this using the following lemma.

Lemma 3 *The equation $X{:}\, X \leadsto Y$ has a weakest solution.*

Proof (of Lemma 3): Abbreviate $X{:}\, X \leadsto Y$ by C. We show that the disjunction of all solutions of C solves C. Clearly this is the weakest solution, as it is implied by all the solutions of C.

$$
\begin{array}{cl}
 & \langle \exists X : X \text{ solves } C : X\rangle \leadsto Y \\
\Leftarrow & \{\text{L2}\} \\
 & \langle \forall X : X \text{ solves } C : X \leadsto Y\rangle \\
= & \{\text{definition } C \text{ and predicate calculus}\} \\
 & \mathit{true}
\end{array}
$$

(End of Proof)

The weakest solution of $X{:}\, X \leadsto Y$ is called the $\mathbf{led}.Y$ (read, *led-from* Y). The predicate transformer **led** enjoys the following properties.

$$
\begin{array}{ll}
\text{(LF0)} & \mathbf{led}.Y \leadsto Y \\
\text{(LF1)} & (X \leadsto Y) \Rightarrow [X \Rightarrow \mathbf{led}.Y]
\end{array}
$$

2.5 Relating *leads-to* and wlt

We now present the main Theorem of this section, which shows the close relationship between the relation $\leadsto$ and the predicate transformer **wlt**.

Theorem 2 $[X \Rightarrow \mathbf{wlt}.Y]$ *iff* $(X \leadsto Y)$

Proof (of Theorem 2): The proof of the theorem is by mutual implication.

($\Leftarrow$) Since *leads-to* ($\leadsto$) is the strongest solution of the equations L0, L1 and L2, it is sufficient to show that the relation **R** defined as $(X \;\mathbf{R}\; Y) \;\equiv\; [X \Rightarrow \mathbf{wlt}.Y]$ solves the same equations. The result follows.

Equation L0 :

$$\begin{array}{ll}
 & (X \;\mathcal{E}\; Y) \\
\Rightarrow & \{\text{C0}\} \\
 & [X \Rightarrow \mathbf{wlt}.Y] \\
= & \{\text{definition of } \mathbf{R}\} \\
 & (X \;\mathbf{R}\; Y)
\end{array}$$

Equation L1 (transitivity):

$$\begin{array}{ll}
 & (X \;\mathbf{R}\; Y) \wedge (Y \;\mathbf{R}\; Z) \\
= & \{\text{definition of } \mathbf{R}\} \\
 & [X \Rightarrow \mathbf{wlt}.Y] \wedge [Y \Rightarrow \mathbf{wlt}.Z] \\
\Rightarrow & \{\text{monotonicity of } \mathbf{wlt}\} \\
 & [X \Rightarrow \mathbf{wlt}.Y] \wedge [\mathbf{wlt}.Y \Rightarrow \mathbf{wlt}.(\mathbf{wlt}.Z)] \\
\Rightarrow & \{\text{transitivity of } \Rightarrow\} \\
 & [X \Rightarrow \mathbf{wlt}.(\mathbf{wlt}.Z)] \\
\Rightarrow & \{\text{W5 with Y := Z}\} \\
 & [X \Rightarrow \mathbf{wlt}.Z] \\
= & \{\text{definition of } \mathbf{R}\} \\
 & (X \;\mathbf{R}\; Z)
\end{array}$$

Equation L2 (disjunction):

$$\begin{array}{ll}
 & \langle \forall X :: X \;\mathbf{R}\; Y\rangle \\
= & \{\text{definition of } \mathbf{R}\} \\
 & \langle \forall X :: [X \Rightarrow \mathbf{wlt}.Y]\rangle \\
= & \{\text{interchange quantifications}\} \\
 & [\langle \forall X :: X \Rightarrow \mathbf{wlt}.Y\rangle] \\
= & \{\text{predicate calculus}\} \\
 & [\langle \exists X :: X\rangle \Rightarrow \mathbf{wlt}.Y] \\
= & \{\text{definition of } \mathbf{R}\} \\
 & \langle \exists X :: X\rangle \;\mathbf{R}\; Y
\end{array}$$

($\Rightarrow$)

0. $\langle \exists s :: \mathbf{gwp}.s.(\mathbf{led}.Y)\rangle \rightsquigarrow \mathbf{led}.Y$
   ,Constraint C1
1. $\mathbf{led}.Y \rightsquigarrow Y$
   ,Property LF0 of the **led**
2. $\langle \exists s :: \mathbf{gwp}.s.(\mathbf{led}.Y)\rangle \rightsquigarrow Y$
   ,From 0, 1 and the transitivity of $\rightsquigarrow$
3. $Y \rightsquigarrow Y$
   ,Corollary 0
4. $(\langle \exists s :: \mathbf{gwp}.s.(\mathbf{led}.Y)\rangle \vee Y) \rightsquigarrow Y$
   ,From 2, 3 and the disjunctivity of $\rightsquigarrow$
5. $[\langle \exists s :: \mathbf{gwp}.s.(\mathbf{led}.Y)\rangle \vee Y \ \Rightarrow\ \mathbf{led}.Y]$
   ,From 4 and Property LF1 of **led**
6. $[\mathbf{wlt}.Y \ \Rightarrow\ \mathbf{led}.Y]$
   ,From 5 and Property W1 with Z := **led**.Y
7. $\mathbf{wlt}.Y \rightsquigarrow \mathbf{led}.Y$
   ,6 and Lemma 1
8. $X \rightsquigarrow \mathbf{wlt}.Y$
   ,From hypothesis $[X \Rightarrow \mathbf{wlt}.Y]$ and Lemma 1
9. $X \rightsquigarrow Y$
   ,From 8, 7, 1 and transitivity of $\rightsquigarrow$

(End of Proof)

2.6 Summary

We recapitulate our main result and summarize the results of this section. We are given a binary relation $\mathcal{E}$ on predicates that satisfies E0 i.e.,

$$[X \Rightarrow Y] \ \Rightarrow\ (X\, \mathcal{E}\, Y).$$

This fixes the interpretation of $\rightsquigarrow$. We are also given a unary predicate transformer **gwp**.s that satisfies P0 i.e.,

$$[X \Rightarrow Y] \ \Rightarrow\ [\mathbf{gwp}.s.X \Rightarrow \mathbf{gwp}.s.Y].$$

This fixes the interpretation of **wlt**. If the definitions of $\mathcal{E}$ and **gwp**.s satisfy the consistency conditions C0 and C1 i.e.,

$$(X\, \mathcal{E}\, Y) \ \Rightarrow\ [X \Rightarrow \mathbf{wlt}.Y].$$

$$\langle \exists s :: \mathbf{gwp}.s.Y\rangle \rightsquigarrow Y$$

then our theorem states that,

$$(X \rightsquigarrow Y) \text{ iff } [X \Rightarrow \mathbf{wlt}.Y].$$

3 Methodology for the Design of Proof Rules

We propose the following methodology for the design of proof rules for progress using fairness assumptions.

- For each fairness assumption, propose a CTL* formula that expresses progress under the prescribed fairness notion.
- Find a μ–calculus formula that is logically equivalent to this CTL* formula.
- Using the μ–calculus formula as a guide, propose a definition for the predicate transformer **gwp**.s. Show that the predicate transformer **wlt**, based on this definition of **gwp**.s, is equivalent to the μ–calculus formula and therefore completely captures progress under the fairness notion assumed.
- As a final step, use the definition of the **gwp**.s predicate transformer and the conditions of the framework of Section 2, to propose a definition for the relation $\mathcal{E}$. This guarantees that the proof rules for $\rightsquigarrow$ based on this definition of $\mathcal{E}$ are sound and relatively complete.

4 Applications

In this section, we introduce our computational model and apply the methodology proposed above to the design of proof rules for progress under a variety of fairness assumptions including pure nondeterminism (no fairness), unconditional fairness, minimal progress, weak fairness and strong fairness.

4.1 The Computational Model

Our notion of a program is based on ideas drawn from Dijkstra's guarded command language [Dij75] and Chandy and Misra's UNITY [CM88]. A *program* consists of two parts : a collection of variable declarations and a finite and non–empty set of *guarded* statements. We call these sections, **declare** and **assign** respectively. Intuitively, a guarded statement, s, is of the form $B \rightarrow S$. The guard, B, is a boolean predicate on the state space of the program. We shall also refer to it as *grd.s*.

The semantics of a guarded statement $B \rightarrow S$ is given as follows :

$$[\mathbf{wp}.``B \rightarrow S".Y \;\; \equiv \;\; (B \Rightarrow \mathbf{wp}.S.Y) \wedge (\neg B \Rightarrow Y)]$$

We assume that our statements are *deterministic* and *terminating*. That is, for all statements s,

$$\begin{array}{ll} \text{(S0)} & [\mathbf{wp}.s.(\neg X) \equiv \neg(\mathbf{wlp}.s.X)] \\ \text{(S1)} & [\mathbf{wp}.s.X \equiv \mathbf{wlp}.s.X] \end{array}$$

We also restrict our attention to statements, s, whose weakest precondition semantics, $\mathbf{wp}.s$ is monotonic. That is,

$$\text{(S2)} \qquad [X \Rightarrow Y] \Rightarrow [\mathbf{wp}.s.X \Rightarrow \mathbf{wp}.s.Y]$$

Indeed, these conditions constitute a specification of the statements that we permit in our programs.

A *state* of the program is an assignment of values to the variables of the program. A statement s is said to be *enabled* in a state if *grd.s* is true in that state.

The operational interpretation of our programs is as follows. Program execution is allowed to begin in any state: there are no specific initial conditions. We have chosen to do this to keep our treatment simple [Mis90]. Intuitively speaking, an execution of a program begins in any state and proceeds by repeatedly selecting a (guarded) statement from the **assign** section of the program and executing it with the selection being subject to a fairness constraint. For example, a simple fairness constraint may require the eventual execution of each statement whose guard is continuously enabled. Thus, in our model, parallelism is captured by a fair, interleaved execution of the statements of the program with each statement being executed atomically.

More formally, the intended model of a program is a forest of infinite computation trees with the nodes of the trees labelled with program states and the edges with program statements. The root node of the tree is labelled with the state in which program execution is initiated. For each statement s of our program, there is an edge labelled s from node v to node w if and only if execution of s transforms the state of the node v to the state of node w. An *execution sequence* of the program is any path in a tree that originates at the root node: an execution sequence is said to be *fair* if it satisfies some fairness constraint. Apart from the usual predicates, we also have auxiliary propositions $\overline{s}$, for each statement s; $\overline{s}$ holds at a node if and only if the edge directed into that node is labelled s.

To get a handle on the properties of the computation trees, we employ the syntax of a branching time temporal logic, commonly known as CTL* [EH83]. In addition to the usual first–order connectives and quantifiers, the syntax of CTL* allows two path quantifiers – **A** ("for all paths") and **E** ("exists a path") – and the usual linear time operators – **G** ("always"), **F** ("eventually"), **X** ("next time"), **W** ("weak until") and **U** ("strong until"). We use CTL* to formally express properties of computation trees and manipulate them. The interested reader is referred to [EH83] for more details.

4.2 Minimal Progress

The constraint of minimal progress is defined as follows : *in an execution sequence, if some statement is enabled then some (possibly different) enabled statement is executed.*

We define an execution sequence to be *maximal* if it satisfies the minimal progress condition. A program executed under the assumption of *minimal progress* reaches a state satisfying Y if and only if along all *maximal* execution sequences a state satisfying Y is reached. This notion of fairness is captured by the CTL* formula

$$\mathbf{A}(\mathbf{G}(\langle \exists s :: grd.s\rangle \Rightarrow \langle \exists s :: grd.s \wedge \mathbf{X}\,\overline{s}\rangle) \Rightarrow \mathbf{F}\,Y).$$

We prove the following lemma.

Lemma 4

$$\mathbf{A}(\mathbf{G}(\langle \exists s :: grd.s\rangle \Rightarrow \langle \exists s :: grd.s \wedge \mathbf{X}\,\overline{s}\rangle) \Rightarrow \mathbf{F}\,Y) \equiv$$

$$\langle \mu X :: (\langle \exists s :: grd.s \wedge \langle \forall t : grd.t : \mathbf{wp}\,.t.X\rangle\rangle) \vee Y\rangle$$

To relate this notion of progress to $\mathbf{wlt}\,.Y$, we need to define the predicate transformer $\mathbf{gwp}\,.s$. The proof of the equivalence of the two definitions suggests the following definition of $\mathbf{gwp}\,.s$.

$$[\mathbf{gwp}\,.s.X \;\equiv\; grd.s. \wedge \langle \forall t : grd.t : \mathbf{wp}\,.t.X\rangle]$$

This definition allows us to prove the following theorem which will be crucial in guaranteeing that our proof system for minimal progress is sound and relatively complete.

Theorem 3

$$[\mathbf{A}(\mathbf{G}(\langle \exists s :: grd.s\rangle \Rightarrow \langle \exists s :: grd.s \wedge \mathbf{X}\,\overline{s}\rangle) \Rightarrow \mathbf{F}\,Y) \;\equiv\; \mathbf{wlt}\,.Y]$$

To be able to define a proof system we need to define the relation $\mathcal{E}$. Such a definition, motivated by the definition of the predicate transformer $\mathbf{gwp}\,.s$ and the proofs of constraints E0, P0, C0 and C1 is,

$$(X \;\mathcal{E}\; Y) \;\equiv\; \langle \forall s :: [X \wedge \neg Y \wedge grd.s \;\Rightarrow\; \mathbf{wp}\,.s.Y]\rangle$$

$$\wedge \langle \exists t :: [X \wedge \neg Y \;\Rightarrow\; grd.t]\rangle$$

We show in the appendix that these definitions satisfy the conditions E0, P0, C0 and C1.

Remark on Pure Nondeterminism: The case of pure nondeterminism (that is, no fairness constraint at all) is a special case of minimal progress where the guard of every statement $grd.s$ is true. By doing this uniformly for the CTL* formula, Lemma 4, definition of $\mathbf{gwp}\,.s$, Theorem 3 and definition of $\mathcal{E}$ we obtain the following.

A program executed with no fairness constraint reaches a state satisfying predicate Y if and only if along *all* execution sequences a state satisfying Y is eventually reached.

- CTL* formula:

$$\mathbf{AF}Y.$$

- Equivalent μ–calculus formula:

$$[\mathbf{AF}Y \;\equiv\; \langle \mu X :: \mathbf{AX}X \vee Y\rangle]$$

- Definition of $\mathbf{gwp}\,.s$:

$$[\mathbf{gwp}\,.s.Y \;\equiv\; \langle \forall t :: \mathbf{wp}\,.t.Y\rangle]$$

- Definition of $\mathcal{E}$:

$$(X \;\mathcal{E}\; Y) \;\equiv\; \langle \forall s :: [X \wedge \neg Y \;\Rightarrow\; \mathbf{wp}\,.s.Y]\rangle$$

(*End of Remark*)

4.3 Weak Fairness

Weak fairness is defined as follows : *an execution sequence is weakly-fair if and only if each statement that is eventually enabled continuously is infinitely often executed.*

A program which is executed under the constraint of *weak fairness* reaches a state satisfying Y if and only if along *all weakly-fair* execution sequences a state satisfying Y is reached. This notion is accurately captured by the CTL* formula

$$\mathbf{A}(\langle \forall s :: \mathbf{F}\,\mathbf{G}\, grd.s \Rightarrow \mathbf{G}\,\mathbf{F}(grd.s \wedge \mathbf{X}\,\overline{s})\rangle \;\Rightarrow\; \mathbf{F}\,Y).$$

We prove the following lemmata.

Lemma 5

$$[\mathbf{A}(\langle \forall s :: \mathbf{F}\,\mathbf{G}\, grd.s \Rightarrow \mathbf{G}\,\mathbf{F}(grd.s \wedge \mathbf{X}\,\overline{s})\rangle \;\Rightarrow\; \mathbf{F}\,Y) \;\equiv\; \langle \mu Z :: Y \vee \langle \exists s :: \mathbf{A}((\mathbf{wp}\,.s.Z \wedge grd.s)\;\mathbf{W}\;Z)\rangle\rangle]$$

Lemma 6

$$[\mathbf{A}(Y \;\mathbf{W}\; Z) \;\equiv\; \langle \nu V :: Z \vee (Y \wedge \mathbf{A}\,\mathbf{X}\,V)\rangle]$$

To relate this notion of progress to **wlt**$.Y$, we need to define the predicate transformer **gwp**$.s$. The proof of the equivalence of the two definitions suggests the following definition of **gwp**$.s$.

$$[\mathbf{gwp}\,.s.X \;\equiv\; \langle \nu Y :: (\langle \forall t :: \mathbf{wp}\,.t.Y\rangle \wedge \mathbf{wp}\,.s.X \wedge grd.s) \vee X\,\rangle]$$

This definition allows us to prove the following theorem which will be crucial in guaranteeing that our proof system for weak fairness is sound and relatively complete.

Theorem 4 (Completeness of wlt)

$$[\mathbf{A}(\langle \forall s :: \mathbf{F}\,\mathbf{G}\, grd.s \Rightarrow \mathbf{G}\,\mathbf{F}(grd.s \wedge \mathbf{X}\,\overline{s})\rangle \;\Rightarrow\; \mathbf{F}\,Y) \;\equiv\; \mathbf{wlt}\,.Y]$$

To be able to define a proof system, we need to define the predicate transformer $\mathcal{E}$. Such a definition, motivated by the definition of the predicate transformer **gwp**$.s$ and the proofs of the constraints E0, P0, C0 and C1. is,

$$\begin{array}{rcl}(X\;\mathcal{E}\;Y) & \equiv & \langle \forall s :: [X \wedge \neg Y \;\Rightarrow\; \mathbf{wp}\,.s.(X \vee Y)]\rangle \\ & & \wedge\, \langle \exists s :: [X \wedge \neg Y \;\Rightarrow\; \mathbf{wp}\,.s.Y \wedge grd.s]\rangle\end{array}$$

We show in the appendix that these definitions satisfy the conditions E0, P0, C0 and C1.

Remark on Unconditional Fairness: The case of unconditional fairness is a special case of weak fairness where the guard of every statement $grd.s$ is true. By doing this uniformly for the CTL* formula, Lemma 5, definition of **gwp**$.s$, Theorem 4 and definition of $\mathcal{E}$ we obtain the following.

Unconditional fairness is defined as follows : *an execution sequence is unconditionally fair if and only if each statement is executed infinitely often.* This notion of fairness has received considerable publicity since the advent of UNITY [CM88]: using a simple notion of programs (similiar to ours) based on unconditional fairness, Chandy and Misra show that it is possible to encode a wide spectrum of parallel programs.

A program which is executed under the constraint of *unconditional fairness* reaches a state satisfying Y if and only if along *all unconditionally fair* execution sequences a state satisfying Y is reached.

- CTL* formula:

$$\mathbf{A}(\langle \forall s :: \mathbf{G}\,\mathbf{F}\,\overline{s}\rangle \;\Rightarrow\; \mathbf{F}\,Y).$$

- Equivalent μ–calculus formula:

$$[\mathbf{A}(\langle \forall s :: \mathbf{G}\,\mathbf{F}\,\overline{s}\rangle \;\Rightarrow\; \mathbf{F}\,Y) \;\equiv\; \langle \mu Z :: Y \vee \langle \exists s :: \mathbf{A}(\mathbf{wp}\,.s.Z\;\mathbf{W}\;Z)\rangle\rangle]$$

- Definition of **gwp**$.s$:

$$[\mathbf{gwp}\,.s.X \;\equiv\; \langle \nu Y :: (\langle \forall t :: \mathbf{wp}\,.t.Y\rangle \wedge \mathbf{wp}\,.s.X) \vee X\,\rangle]$$

- Definition of $\mathcal{E}$:

$$\begin{array}{rcl}(X\;\mathcal{E}\;Y) & \equiv & \langle \forall s :: [X \wedge \neg Y \;\Rightarrow\; \mathbf{wp}\,.s.(X \vee Y)]\rangle \\ & & \wedge\, \langle \exists s :: [X \wedge \neg Y \;\Rightarrow\; \mathbf{wp}\,.s.Y]\rangle\end{array}$$

(*End of Remark*)

4.4 Strong Fairness

Strong fairness is defined as follows : *an execution sequence is strongly-fair if and only if each statement that is enabled infinitely often is executed infinitely often.*

We have investigated the applicability of our framework to strong fairness. It turns out that the proof obligations for strong fairness are more complicated than all the cases previously considered. To prove a property of a program F, one has to prove properties of programs which are related to F : specifically, one has to consider programs that have one statement less than F. We shall denote this by $F - \{s\}$ where s is the omitted statement. Towards this end, we introduce a new notational convention: properties of programs will be parameterized by the name of the program. Till now, the name of the program was implicit in our properties.

A program which is executed under the constraint of *strong fairness* reaches a state satisfying Y if and only if along *all strongly-fair* execution sequences a state satisfying Y is reached. This notion is accurately captured by the CTL* formula

$$\mathbf{A}(\langle \forall s :: \mathbf{G}\,\mathbf{F}\,grd.s \Rightarrow \mathbf{G}\,\mathbf{F}(grd.s \wedge \mathbf{X}\,\overline{s})\rangle \Rightarrow \mathbf{F}\ Y).$$

In the following the program being considered is named F and quantifiers over statements will be assumed to range over the statements of F.

Definition 1 *For a program F, define Θ_F as follows.*

$$[\Theta_F \equiv \langle \forall s :: \mathbf{G}\,\mathbf{F}\,grd.s \ \Rightarrow\ \mathbf{G}\,\mathbf{F}(grd.s \wedge \mathbf{X}\,\overline{s})\rangle].$$

We prove the following lemma.

Lemma 7 *Abbreviate by $\Phi.s.F.Z$,*

$$[\Phi.s.F.Z \ \equiv\ \mathbf{A}(grd.s \Rightarrow \mathbf{X}(\neg\overline{s} \vee Z)) \ \wedge\ \mathbf{A}(\Theta_{F-\{s\}} \Rightarrow \mathbf{F}(grd.s \vee Z))].$$

Then,

$$[\mathbf{A}(\Theta_F \ \Rightarrow\ \mathbf{F}\,Y) \ \equiv\ \langle \mu Z :: Y \vee \langle \exists s :: \mathbf{A}(\Phi.s.F.Z\ \mathbf{W}\ Z\,)\rangle\rangle]$$

To relate this notion of progress to $\mathbf{wlt}\,.Y$, we need to define the predicate transformer $\mathbf{gwp}\,.s$. The proof of the equivalence of the two definitions suggests the following definition of $\mathbf{gwp}\,.s$.

$$[\mathbf{gwp}\,.s.F.X \ \equiv\ \langle \nu Y :: (\langle \forall t :: \mathbf{wp}\,.t.Y\rangle \wedge (grd.s \Rightarrow \mathbf{wp}\,.s.X) \wedge \mathbf{wlt}\,.(F\ -\ \{s\}).(grd.s \vee X)) \vee X\rangle].$$

This definition allows us to prove the following theorem which will be crucial in guaranteeing that our proof system for strong fairness is sound and relatively complete.

Theorem 5 (Completeness of wlt)

$$[\mathbf{A}(\langle \forall s :: \mathbf{G}\,\mathbf{F}\,grd.s \Rightarrow \mathbf{G}\,\mathbf{F}(grd.s \wedge \overline{s})\rangle \Rightarrow \mathbf{F}\ Y) \ \equiv\ \mathbf{wlt}\,.Y]$$

To be able to define a proof system, we need to define the predicate transformer $\mathcal{E}$. Such a definition, motivated by the definition of the predicate transformer $\mathbf{gwp}\,.s$ and the proofs of the constraints E0, P0, C0 and C1 is,

$$\begin{array}{rcll}(X\ \mathcal{E}\ Y\,\mathrm{in\,F}) & \equiv & \langle \forall s :: & [X \wedge \neg Y \Rightarrow \mathbf{wp}\,.s.(X \vee Y)]\rangle \wedge \\ & & \langle \exists s :: & (X \wedge \neg Y \leadsto (grd.s \vee Y)\,\mathrm{in\,F} - \{\mathrm{s}\}) \wedge \\ & & & [X \wedge \neg Y \wedge grd.s \Rightarrow \mathbf{wp}\,.s.Y]\rangle\end{array}$$

We show in the appendix that these definitions satisfy the conditions E0, P0, C0 and C1.

5 On a notion of completeness of *leads–to*

In most of the rules for proving progress under assumptions of fairness, the expressiveness of the assertion language and the nature of the completeness proven is not very clear. The assertion language is usually assumed to be a first–order language augmented with fixpoint operators, a sort denoting the ordinals, constants for all the recursive ordinals and an ordering relation on the ordinals. With respect to completeness, most authors ([GP88], [San90], [Kna90]) prove completeness *relative* to the completeness of some existing temporal logic: often the temporal logic chosen is too powerful and once again the assumptions underlying the completeness result are not clear.

In this section, we briefly review the literature pertaining to the completeness of Hoare triples and show that the same problems apply to the completeness of the $\leadsto$. Thus the best that one can hope for is *completeness in the sense of Cook.* We then illustrate, how our completeness result for unconditional fairness can be used to *construct* a proof of the progress property that holds in the model. Our careful treatment clearly illustrates the roles played by the complexity of the assertion language, relative completeness, expressibility of weakest preconditions and the nature of the sets used in the disjunction axiom L2.

5.1 Reviewing Completeness

As the first step of our investigation, we trace the problems that were encountered in defining the notion of completeness for Hoare triples [Apt81] and draw parallels to the $\leadsto$ relation.

Let AL denote a suitable language for assertions and PL denote a language for programming constructs. Let H denote a proof system for Hoare logic, T denote a proof system for assertions of AL and L denote a proof system for the $\leadsto$ relation.

- The systems H and L, by themselves, cannot guarantee completeness for Hoare triples and the $\leadsto$ relation. Specifically, $\{true\}\, x := t\, \{true\}$ and $true \leadsto true$ are true of all models but are unprovable in H and L respectively. One needs an axiomatic theory for the language of assertions to prove $true \Rightarrow true$. The solution lies in augmenting H and L with a proof system for assertions, namely T.

- While adding an axiomatic system for AL allows one to prove some additional assertions, it does not solve the problem of incompleteness. If AL can express statements of arithmetic, then by Gödel's incompleteness theorem the set of validities of AL is not recursively enumerable. That is there exist true assertions of the AL that are unprovable. Such an assertion, say P, can be expressed by the Hoare triple $\{true\}\, skip\, \{P\}$ or as the property $true \leadsto P$ for the program containing only the skip statement. Clearly, the Hoare triple and the progress property hold in the model but *cannot* be proved in H and L, even when the latter are augmented with T. This source of incompleteness stems from the complexity of the assertion language AL. One possible solution is to restrict it, so that Gödel assertions cannot be expressed.

- Consider a very simple assertion language consisting of the two assertions $true$ and $false$. The set of programs for which the Hoare triple $\{true\}\, S\, \{false\}$ holds are precisely those programs that diverge on all possible inputs. It is known that this set is not recursively enumerable [HU79]. Thus there must be a valid Hoare triple that cannot be derived in H. One can construct a similiar example for the $\leadsto$ operator as well. In a recent paper [Pac90], Jan Pachl has shown that a non-recursively enumerable set, namely, the set of instances of the Post's Correspondence Problem that don't have a solution can be encoded as a set of $\leadsto$ properties of a program.

  This example shows that curtailing the expressive power of the assertion language alone is not a solution to the incompleteness of Hoare logic. To factor out the incompleteness that is due to the assertion language, one introduces the notion of *relative completeness.* This can be informally stated as follows: For all models M and Hoare triples Φ, if $\models_M \Phi$ then $Tr_M \vdash_H \Phi$ where Tr_M are all the assertions that are true in the model M and we are permitted to use these assertions in the proof of Φ.

- Introducing relative completeness does not eliminate the problem of incompleteness. An additional source of incompleteness is the inability to express (or name) sets of states which occur during program execution by predicates. One such example is given by Wand [Wan78]. Wand exhibits a simple while program and a partial correctness Hoare triple corresponding to it. While the Hoare triple is valid in the model described, it cannot be derived in a proof system for Hoare triples even by assuming relative completeness. This is because it is impossible to express the invariant of the while program using the assertion language described.

  While Wand's example was a Hoare triple for a while program, the exact same argument can be readily extended to a $\leadsto$ property of the same while program recast as a program admissible in our formalism. This shows that relative completeness alone is not enough to remedy the incompleteness of the $\leadsto$ operator of UNITY.

- As a means of incorporating expressibility of assertions that occur during program execution, Cook [Coo78] introduced the notion of expressibility of the *weakest precondition*[0]. This was used to define a new notion of

[0]Actually Cook introduced expressibility of the strongest postcondition but showed that the concepts were equivalent

completeness – *completeness in the sense of Cook.* A proof system H for PL is *complete in the sense of Cook,* if for every model M in which the weakest pre-condition is expressible and for every Hoare triple Φ, if $\models_M \Phi$, then $Tr_M \vdash_G \Phi$. Notice that this notion of completeness uses relative completeness as well.

To overcome the problems associated with the completeness of $\leadsto$ it is necessary to define a notion of completeness akin to that of Cook. In the following derivation, for the case of unconditional fairness, we show exactly where it is necessary to appeal to expressibility of weakest preconditions.

5.2 Constructing a Proof of Progress

In this section we give a *constructive* proof of completeness of the proof system for unconditional fairness. Since the proof is constructive, we have to use ordinals unlike the proof of Section 2.

Assume that a property $X \leadsto Y$ holds in the intended model of execution for a program F executed on the assumption of unconditional fairness. On the basis of our framework, we may assert that

$$[\mathbf{A}(\langle\forall s :: \mathbf{G}\,\mathbf{F}\,\overline{s}\rangle \Rightarrow \mathbf{F}\,Y) \;\equiv\; \mathbf{wlt}.Y]$$

where $\mathbf{wlt}.Y$ is given by,

$$[\mathbf{wlt}.Y \;\equiv\; \langle\mu Z :: Y \vee \langle\exists s :: \mathbf{gwp}.s.Z\rangle\rangle]$$

and $\mathbf{gwp}.s.Z$ is given by,

$$[\mathbf{gwp}.s.Z \;\equiv\; \langle\nu V :: (\langle\forall t :: \mathbf{wp}.t.V\rangle \wedge \mathbf{wp}.s.Z) \vee Z\rangle].$$

Furthermore, if $X \leadsto Y$ holds in the intended model of computation trees, then using the above, we may assert that $[X \Rightarrow \mathbf{wlt}.Y]$. We now show how to construct a derivation of $X \leadsto Y$ explicating all our assumptions as we go along.

1. To express the predicate transformer $\mathbf{gwp}.s.Y$, the assertion language should contain fixpoint operators and propositional variables which can be bound by these fixpoint operators. Furthermore, it requires the expressibility of the weakest preconditions ($\mathbf{wp}.s$) of all the statements. This is where completeness in the sense of Cook comes in.

2. The first step of our derivation shows that
$$\mathbf{gwp}.s.Y \;\mathcal{E}\; Y.$$
While this proof has been omitted in the interest of brevity, it can be found in [JKR89].

3. From 2, using L0, we can infer that
$$\mathbf{gwp}.s.Y \leadsto Y.$$

4. From 3, using the disjunction axiom of $\leadsto$ L2, over a *finite* set of program statements, we can infer that,
$$\langle\exists s :: \mathbf{gwp}.s.Y\rangle \leadsto Y.$$

5. Since $\mathbf{wlt}.Y$ is expressed as the strongest solution of a monotonic predicate transformer, we know by the Theorem of Knaster–Tarski, that it can be expressed as
$$\mathbf{wlt}.Y \;\equiv\; \langle\exists\alpha :: f^{\alpha}.Y\rangle$$
where the predicate transformer f is defined as,
$$[f^0.Y \;\equiv\; false]$$
$$[f^{i+1}.Y \;\equiv\; (Y \vee \langle\exists s :: \mathbf{gwp}.s.(f^i.Y)\rangle)]$$
$$[f^{\alpha}.Y \;\equiv\; \langle\exists\beta : \beta < \alpha : f^{\beta}.Y]$$
The expression of each of the iterates of f upto the closure ordinal, requires the assertion language to contain a sort corresponding to the ordinals and an ordering relation $<$ on the ordinals. Using these we can show that each of the $f^{\alpha}.Y$ is expressible in the assertion language. Using this definition of $\mathbf{wlt}$ and 4 we show that,
$$\langle\forall\alpha :: f^{\alpha}.Y \leadsto Y\rangle.$$

The proof of this fact requires us to use the disjunction axiom L2 over a set of ordinals less than a fixed ordinal α. Application of the disjunction axiom once again over the set of ordinals less than the closure ordinal yields,

$$\mathbf{wlt}\,.Y \rightsquigarrow Y.$$

Notice that all applications of the disjunction axiom were confined to well–ordered sets.

6. We have not yet made use of the fact that in the model $[X \Rightarrow \mathbf{wlt}\,.Y]$ holds. Since $\mathbf{wlt}\,.Y$ is expressible in the assertion language (see 5) $[X \Rightarrow \mathbf{wlt}\,.Y]$ is a truth of the assertion language in the model of computation trees. It is at this point that *relative* completeness enters the picture. If we could assume the existence of a derivation for such an assertion, then by Lemma 1, we have $X \rightsquigarrow \mathbf{wlt}\,.Y$. This along with the conclusion of 5, permits us to deduce,
$$X \rightsquigarrow Y$$

6 The predicate transformer wsafe

Consider the following predicate transformer introduced in [JKR89].

$$[\,\mathbf{wsafe}\,.F.X.Y \;\equiv\; \langle \nu Z :: (\langle \forall s : s \,\mathrm{in}\, \mathrm{F} : \mathbf{wp}\,.\mathrm{s.Z}\rangle \wedge \mathrm{X}) \vee \mathrm{Y}\rangle]$$

The importance of **wsafe** lies in the fact that it is the basis for defining both the *safety* and *progress* properties of programs. It has been shown in [JKR89] that

$$(X \;\mathbf{unless}\; Y \,\mathrm{in}\, \mathrm{F}) \;\text{ iff }\; [\mathrm{X} \Rightarrow \mathbf{wsafe}\,.\mathrm{F.X.Y}]$$

where **unless** is the safety operator of UNITY [CM88] is defined as

$$X \;\mathbf{unless}\; Y \,\mathbf{in}\, F \;\equiv\; \langle \forall s : s\,\mathbf{in}\,F : [X \wedge \neg Y \Rightarrow \mathbf{wp}\,.s.(X \vee Y)]\rangle.$$

7 Compositionality

We illustrate the compositionality of our semantics for the case of *unconditional fairness.* The result can be easily extended to the case of *weak fairness* as well. Similiar results hold for *minimal progress* and *pure nondeterminism.*

One of the desirable characteristics of a semantics for parallel programs is *compositionality.* Suppose that we are given two programs F and G and an operator $\odot$ to compose them. Then the semantics is said to be compositional if the semantic function of $F \odot G$ can be inferred from the semantic functions of F and G.

We define a simple notion of composition ($\odot$) of two programs F and G and show how $\mathbf{wsafe}\,.(F \odot G)$ and $\mathbf{gwp}\,.s.(F \odot G)$ can be inferred from the corresponding properties of F and G.

Definition 2 (Composition : $\odot$) *Let F and G be two programs. Then we denote by $F \odot G$, the program obtained taking the union of the* **assign** *sections of the two programs.*

We are now ready to present our theorems on composition.

Theorem 6 (Compositionality of $\mathcal{E}$) *Given predicates X and Y,*

$$\begin{aligned}(X\;\mathcal{E}\;Y\,\mathbf{in}\,F \odot G) \;\;\equiv\;\; & (X\;\mathcal{E}\;Y\,\mathbf{in}\,F \;\wedge\; X\;\mathbf{unless}\;Y\,\mathbf{in}\,G)\\ & \vee\,(X\;\mathbf{unless}\;Y\,\mathbf{in}\,F \;\wedge\; X\;\mathcal{E}\;Y\,\mathbf{in}\,G)\end{aligned}$$

The proof of Theorem 6 follows directly from the definition of $\mathcal{E}$ for unconditional fairness.

Theorem 7 (Compositionality of wsafe) *Given predicates X and Y,*

$$[\,\mathbf{wsafe}\,.(F \odot G).X.Y \equiv \langle \nu Z :: \mathbf{wsafe}\,.G.(\mathbf{wsafe}\,.F.(X \wedge Z).Y).Y\rangle]$$

Theorem 8 (Compositionality of gwp .s) *Let the predicate transformer* we .$s.F.X.Y$ *be defined by*

$$[\mathbf{we}.s.F.X.Y \;\equiv\; \langle \nu Z :: (\langle \forall s : s \,\mathbf{in}\, F : \mathbf{wp}.s.Z\rangle \wedge X \wedge \mathbf{wp}.s.Y) \vee Y\rangle].$$

where

$$[\mathbf{we}.s.F.true.Y \;\equiv\; \mathbf{gwp}.s.F.Y].$$

Then,

$$[\mathbf{we}.s.(F \odot G).X.Y \;\equiv\; \langle \nu W :: \mathbf{we}.s.G.(X \wedge \mathbf{we}.s.F.W.Y).Y\rangle]$$

The proof of Theorem 7 appears in the appendix. The proof of Theorem 8 is very similiar to that Theorem 7 and is being omitted.

8 "Strong" Equivalence

Let $\mathcal{L}$ be a set of state predicates of programs F and G. We say that $F \cong_{\mathcal{L}} G$ iff

- $\forall X \in \mathcal{L}$: $[\mathbf{wlt}.F.X \equiv \mathbf{wlt}.G.X]$ and,
- $\forall X, Y \in \mathcal{L}$: $[\mathbf{wsafe}.F.X.Y \equiv \mathbf{wsafe}.G.X.Y]$.

This is the notion of "strong"equivalence, and is second order expressible as in [Par70]. However, unlike [Par70] where only termination was of concern, the above equivalence is provable only in special cases: in particular when $\mathcal{L}$ is finite. Note that $\mathcal{L}$ can be finite even though the state space of the programs may be infinite. It may represent a finite set of observable predicates.

We next show that strong equivalence of two programs proves that they agree on a rich class of properties. We say, $F \sim_{\mathcal{L}} G$ iff they agree on all formulae of the following Branching time Temporal Logic FwCTL*, which is insensitive to stuttering. The branching modality is: along all fair paths ($\mathbf{A}_\phi$, where ϕ is the fairness constraint of interest), and the path modality is reflexive strong until ($\mathbf{U}$) with the following restriction. For any path formula $\mathbf{G}\,p$ inside an $\mathbf{E}_\phi$, p should not contain $\mathbf{U}$ unless it is nested inside another branching modality $\mathbf{A}_\phi$ or $\mathbf{E}_\phi$. However, untils which are of the form $true\,\mathbf{U}\,p$ (i.e. $\mathbf{F}\,p$) are allowed. A similar restriction holds for the dual case, i.e. for formula $\mathbf{F}\,p$ inside an $\mathbf{A}_\phi$. In many cases, it is easy to remove untils, e.g. $[\mathbf{G}(p\,\mathbf{U}\,q) \equiv \mathbf{G}(p \vee q)]$. Thus, $\mathbf{E}_\phi\,\mathbf{G}(p\,\mathbf{U}\,q)$ has an equivalent formula in FwCTL*. However, $\mathbf{E}_\phi\,\mathbf{G}(p\,\mathbf{U}(q\,\mathbf{U}\,r))$ is in general not reducible to FwCTL*.

Given $\mathcal{L}$ a set of state predicates, let $\mathcal{L}^* \supseteq \mathcal{L} \cup \{true, false\}$, be the set closed under conjunctions, disjunctions, negation and predicate transformers **wlt** and **wsafe**.

Theorem 12

$$F \sim_{\mathcal{L}} G \text{ iff } F \cong_{\mathcal{L}^*} G$$

Remark : The above theorem holds when **wlt** is replaced by **we** in the definition of $\cong_{\mathcal{L}}$ provided $\mathcal{L}$ contains all the propositions $\overline{s}$ introduced in Section 3.0. (*End of Remark*)

When fairness just means pure non-determinism, the above result follows from the translation of CTL* to μ-calculus (see [EL86]). Although FwCTL* is a subset of CTL*, translating FwCTL* to the μ-calculus of **wsafe** and **wlt** requires a new and direct approach as opposed to [EL86] and [Niw88].

9 Acknowledgement

We would like to thank Jayadev Misra for his continuing encouragement and support. We also thank Tony Hoare, Edsger Dijkstra and the Austin Tuesday Afternoon Club for their insightful comments. The second author would like to thank T. K. Prasad for many stimulating discussions.

References

[AP86] Krzysztof Apt and Gordon Plotkin. Countable nondeterminism and random assignment. *Journal of the ACM*, 33(4):724–767, 1986.

[APS84] Krzysztof Apt, Amir Pnueli, and J. Stavi. Fair termination revisited with delay. *Theoretical Computer Science*, 33:65–84, 1984.

[Apt81] Krzysztof Apt. Ten years of Hoare logic – part 1. *ACM Transactions on Programming Languages and Systems*, 3(4):431–483, 1981.

[BKP84] Howard Barringer, Ruuard Kuiper, and Amir Pnueli. Now you may compose temporal logic specifications. In *Proceedings of the 16th Annual ACM Symposium on the Theory of Computing*, pages 51–64, 1984.

[BKP86] Howard Barringer, Ruuard Kuiper, and Amir Pnueli. A really abstract concurrent model and its temporal logic. In *Proceedings of the 12th Annual ACM Symposium on the Principles of Programming Languages*, pages 173–183, 1986.

[CM88] K. Mani Chandy and Jayadev Misra. *Parallel Program Design: A Foundation.* Addison-Wesley, 1988.

[Coo78] Stephen Cook. Soundness and completeness of an axiom system for program verification. *SIAM Journal of Computing*, 7(1):70–90, 1978.

[Dij75] Edsger W. Dijkstra. Guarded commands, nondeterminacy and the formal derivation of programs. *Communications of the ACM*, 18:453–457, August 1975.

[DS90] Edsger W. Dijkstra and Carel S. Scholten. *Predicate Calculus and Program Semantics.* Springer-Verlag, 1990.

[EH83] Alan Emerson and Joseph Y. Halpern. "sometimes" and "not never" revisited: On branching versus linear time. In *Proceedings of the 10th Annual ACM Symposium on the Principles of Programming Languages*, Austin, Texas, January 1983.

[EL86] Alan Emerson and D. L. Lei. Model–checking in the propositional μ-calculus. In *Proceedings of the Fist Annual IEEE Symposium on Logic in Computer Science*, 1986.

[Flo67] Robert W. Floyd. Assigning meanings to programs. In *Proceedings of the American Mathematical Society's Symposia in Applied Mathematics*, volume 19, pages 19–31, 1967.

[Fra86] Nissim Francez. *Fairness.* Springer-Verlag, New York, 1986.

[GFMR81] O. Grumberg, N. Francez, J. A. Makowsky, and W-P. De Roever. A proof rule for the fair termination of guarded commands. In *Proceedings of the International Symposium on Algorithmic Languages*, Amsterdam, The Netherlands, October 1981.

[GP88] Robert Gerth and Amir Pnueli. Rooting unity. In *Proceedings of the Fifth International Workshop on Software Specification and Design*, May 1988.

[GPSJ80] D. Gabbay, Amir Pnueli, S. Shelah, and J.Stavi. On the temporal analysis of fairness. In *Proceedings of the Seventh Annual ACM Symposium on the Principles of Programming Languages*, Las Vegas, Nevada, January 1980.

[Hoa69] C. A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580,583, 1969.

[HU79] John E. Hopcroft and Jeffrey D. Ullman. *Intoduction to Automata Theory, Languages and Computation.* Addison-Wesley, 1979.

[JKR89] Charanjit S. Jutla, Edgar Knapp, and Josyula R. Rao. A predicate transformer approach to the semantics of parallel programs. In *Proceedings of the Eighth Annual ACM Symposium on the Principles of Distributed Computing*, pages 249–263, 1989.

[Kel76] R. M. Keller. Formal verification of parallel programs. *Communications of the ACM*, 19(7):371–384, 1976.

[Kna88] Edgar Knapp. A comparison of the *led-from* and *leads-to*. Technical Report TR–88–35, The University of Texas at Austin, Department of Computer Sciences, 1988.

[Kna90] Edgar R. Knapp. Soundness and relative completeness of unity. Submitted to JACM, 1990.

[Lam77] Leslie Lamport. Proving the correctness of multiprocess programs. *IEEE Transactions on Software Engineering*, 3(2):125–143, 1977.

[LPS81] Daniel Lehmann, Amir Pnueli, and J. Stavi. Impartiality, justice and fairness : The ethics of concurrent termination. In O. Kariv and S. Even, editors, *Lecture Notes in Computer Science 115 : Proceedings of the Eighth ICALP*. Springer-Verlag, New York, 1981.

[Man69] Zohar Manna. Properties of programs and first order predicate calculus. *Journal of the ACM*, 16:244–255, 1969.

[Mis90] Jayadev Misra. Soundness of the substitution axiom. Notes on UNITY 14-90, Department of Computer Science, The University of Texas at Austin, Austin TX 78712, 1990.

[Mos89] Yiannis Moschovakkis. A game–theoretic modelling of concurrency. In *Proceedings of the Fourth Anuual IEEE Symposium on Logic in Computer Science*, 1989.

[MP84] Zohar Manna and Amir Pnueli. Adequate proof principles for invariance and liveness properties of concurrent programs. *Science of Computer Programming*, 4:257–289, 1984.

[Niw88] Damien Niwinski. Fixed points versus infinite generation. In *Proceedings of the Third Anuual IEEE Symposium on Logic in Computer Science*, pages 402–409, 1988.

[OG76a] Susan Owicki and David Gries. An axiomatic proof technique for parallel programs. *Acta Informatica*, 5:319–339, 1976.

[OG76b] Susan Owicki and David Gries. Verifying properties of parallel programs : An axiomatic approach. *Communications of the ACM*, 19(5):279–286, 1976.

[OL82] Susan Owicki and Leslie Lamport. Proving liveness properties of concurrent programs. *ACM Transactions on Programming Languages and Systems*, 4(3):455–495, July 1982.

[Pac90] Jan Pachl. Three definitions of *leads-to* for unity. Notes on UNITY 23–90, 1990.

[Par70] David Park. Fixpoint induction and proofs of program properties. In D. Mitchie, editor, *Machine Intelligence 5*. Edinburgh University Press, 1970.

[Par80] David Park. On the semantics of fair parallelism. In D. Biorner, editor, *Lecture Notes in Computer Science 86 : Proceedings of the Winter School on Formal Software Specification*. Springer-Verlag, 1980.

[Par81a] David Park. Concurrency and automata on infinite sequences. In *Proceedings of the GI Conference on Theoretical Computer Science*, Karlsruhe, West Germany, 1981.

[Par81b] David Park. A predicate transformer for weak fair iteration. In *Proceedings of the Sixth IBM Symposium on Mathematical Foundations of Computer Science (Hakone)*, IBM, New York, 1981.

[Pnu83] Amir Pnueli. On the extremely fair treatment of probabilistic algorithms. In *Proceedings of the 15th Annual Symposium on the Theory of Computing*, pages 278–290, 1983.

[QS83] J. P. Queille and J. Sifakis. Fairness and related properties in transition systems – a temporal logic to deal with fairness. *Acta Informatica*, 19:195–220, 1983.

[San90] Beverly Sanders. Eliminating the substitution axiom from unity. Technical Report 128, Departement Informatik Institut für Computersysteme, Eidgenössische Technische Hochschule, Zürich, Switzerland, 1990.

[SdRG89] F. A. Stomp, W-P. de Roever, and R. T. Gerth. The μ-calculus as an assertion language for fairness arguments. *Information and Computation*, 82(3):278–322, September 1989.

[Wan78] Mitchell Wand. A new incompleteness result for Hoare's system. *Journal of the ACM*, 25(1):168–175, 1978.

Proofs of Theorems

Minimal Progress

Lemma 4

$$\mathbf{A}(\mathbf{G}(\langle \exists s :: grd.s\rangle \Rightarrow \langle \exists s :: grd.s \wedge \mathbf{X}\,\overline{s}\rangle) \Rightarrow \mathbf{F}\,Y) \equiv$$
$$\langle \mu X :: \langle \exists s :: grd.s \wedge \langle \forall s : grd.s : \mathbf{wp}\,.s.X\rangle\rangle \vee Y\rangle$$

Proof (of 4): We prove the contrapositive; specifically, we show that,

$$\begin{array}{ll}
 & \neg\,\mathbf{A}(\mathbf{G}(\langle \exists s :: grd.s\rangle \Rightarrow \langle \exists s :: grd.s \wedge \mathbf{X}\,\overline{s}\rangle) \Rightarrow \mathbf{F}\,Y) \\
= & \{\text{temporal logic and predicate calculus}\} \\
 & \mathbf{E}(\mathbf{G}(\langle \exists s :: grd.s\rangle \Rightarrow \langle \exists s :: grd.s \wedge \mathbf{X}\,\overline{s}\rangle) \wedge \mathbf{G}\,\neg Y)) \\
= & \{\text{temporal logic}\} \\
 & \mathbf{E}\,\mathbf{G}((\langle \exists s :: grd.s\rangle \Rightarrow \langle \exists s :: grd.s \wedge \mathbf{X}\,\overline{s}\rangle) \wedge \neg Y) \\
= & \{\text{Proof Obligation}\} \\
 & \langle \nu X :: \langle \forall s :: \neg grd.s \vee \langle \exists t : grd.t : \mathbf{wp}\,.t.X\rangle\rangle \wedge \neg Y\rangle \\
= & \{\text{property S0 and S1}\} \\
 & \langle \nu X :: \langle \forall s :: \neg grd.s \vee \langle \exists t : grd.t : \neg\,\mathbf{wp}\,.t.(\neg X)\rangle\rangle \wedge \neg Y\rangle \\
= & \{[\neg\langle \mu X :: f.X\rangle \;\equiv\; \langle \nu X :: \neg f.(\neg X)\rangle]\} \\
 & \neg\langle \mu X :: \langle \exists s :: grd.s \wedge \langle \forall t : grd.t : \mathbf{wp}\,.t.X\rangle\rangle \vee Y\rangle
\end{array}$$

Abbreviating $\mathbf{E}\,\mathbf{G}((\langle \exists s :: grd.s\rangle \Rightarrow \langle \exists s :: grd.s \wedge \mathbf{X}\,\overline{s}\rangle) \wedge \neg Y)$ by Z, our proof obligation is two-fold. We first have to show that Z is a solution of the equation $X : [X \equiv \langle \forall s :: \neg grd.s \vee \langle \exists t : grd.t : \mathbf{wp}\,.t.X\rangle\rangle \wedge \neg Y\rangle]$ and secondly, that it is the weakest solution.

Ad 0. To show that Z solves the equation, it is sufficient (by Theorem 0) to show that

$$[Z \Rightarrow \langle \forall s :: \neg grd.s \vee \langle \exists t : grd.t : \mathbf{wp}\,.t.Z\rangle\rangle \wedge \neg Y\rangle].$$

The following derivation proves this.

$$\begin{array}{ll}
 & Z \\
= & \{\text{definition of } Z\} \\
 & \mathbf{E}\,\mathbf{G}((\langle \exists s :: grd.s\rangle \Rightarrow \langle \exists s :: grd.s \wedge \mathbf{X}\,\overline{s}\rangle) \wedge \neg Y) \\
\Rightarrow & \{[\mathbf{E}\,\mathbf{G}\,X \Rightarrow \mathbf{E}(X \wedge \mathbf{X}\,\mathbf{E}\,\mathbf{G}\,X)]\} \\
 & \mathbf{E}((\langle \exists s :: grd.s\rangle \Rightarrow \langle \exists s :: grd.s \wedge \mathbf{X}\,\overline{s}\rangle) \wedge \neg Y \wedge \mathbf{X}\,Z) \\
= & \{\text{predicate calculus}\} \\
 & \mathbf{E}((\langle \forall s :: \neg grd.s\rangle \vee \langle \exists s :: grd.s \wedge \mathbf{X}\,\overline{s}\rangle) \wedge \neg Y \wedge \mathbf{X}\,Z) \\
\Rightarrow & \{\text{temporal logic and predicate calculus}\} \\
 & \mathbf{E}((\langle \forall s :: \neg grd.s\rangle \vee \langle \exists s :: grd.s \wedge \mathbf{X}\,\overline{s} \wedge \mathbf{X}\,Z\rangle) \wedge \neg Y) \\
= & \{\text{temporal logic}\} \\
 & \mathbf{E}((\langle \forall s :: \neg grd.s\rangle \vee \langle \exists s :: grd.s \wedge \mathbf{wp}\,.s.Z\rangle) \wedge \neg Y) \\
\Rightarrow & \{\text{temporal logic}\} \\
 & ((\langle \forall s :: \neg grd.s\rangle \vee \langle \exists s :: grd.s \wedge \mathbf{wp}\,.s.Z\rangle) \wedge \neg Y)
\end{array}$$

Ad 1. To show that Z is the weakest solution, we have to show that any solution of equation $X : [X \equiv \langle \forall s :: \neg grd.s \vee \langle \exists t : grd.t : \mathbf{wp}\,.t.X\rangle\rangle \wedge \neg Y\rangle]$ implies Z.

Assume that X is a solution of the equation and that X holds at a node u. Since X is a solution of the equation, we can conclude that $\neg Y$ holds at u. At this point two cases arise.

0. $\langle \forall s :: \neg grd.s\rangle$ holds at u. Since all the statements are disabled, effectively there is only one execution, namely the one in which the state of u is repeated infinitely. Furthermore since the state along this execution does not change, $\langle \forall s :: \neg grd.s\rangle \wedge \neg Y$ holds at each node of the path beginning at u, that is, Z holds at u.

1. $\langle \exists s :: grd.s\rangle$ holds at u. From the definition of X this means that $\langle \exists t : grd.t : \mathbf{wp}\,.t.X\rangle$ holds at u. Thus there exists a child (call it v) of u, such that the state of v is obtained by executing statement t in the state of u. Furthermore, from our convention, $\overline{t} \wedge X$ will be true in v. Since X holds at v, the above argument can be repeated with v in place of u. Thus, a simple induction establishes the existence of a path from u along which $\mathbf{E}\,\mathbf{G}((\langle \exists s :: grd.s\rangle \Rightarrow \langle \exists s :: grd.s \wedge \mathbf{X}\,\overline{s}\rangle) \wedge \neg Y)$ holds, that is, Z holds. This concludes our proof. (End of Proof)

Theorem 3

$$[\mathbf{A}(\mathbf{G}(\langle\exists s :: grd.s\rangle \Rightarrow \langle\exists s :: grd.s \wedge \mathbf{X}\,\overline{s}\rangle) \Rightarrow \mathbf{F}\,Y) \quad\equiv\quad \mathbf{wlt}\,.Y]$$

Proof (of 3):

$$\begin{array}{ll}
 & \mathbf{A}(\mathbf{G}(\langle\exists s :: grd.s\rangle \Rightarrow \langle\exists s :: grd.s \wedge \mathbf{X}\,\overline{s}\rangle) \Rightarrow \mathbf{F}\,Y) \\
= & \{\text{Lemma 4}\} \\
 & \langle\mu X :: \langle\exists s :: grd.s \wedge \langle\forall s : grd.s : \mathbf{wp}\,.s.X\rangle\rangle \vee Y\rangle \\
= & \{\text{definition of } \mathbf{gwp}\,.s\} \\
 & \langle\mu X :: \langle\exists s :: \mathbf{gwp}\,.s.X\rangle \vee Y\rangle \\
= & \{\text{definition of } \mathbf{wlt}\} \\
 & \mathbf{wlt}\,.Y
\end{array}$$

(End of Proof)

We now show that the definitions of the predicate transformer **gwp** $.s$ and the relation $\mathcal{E}$ satisfy conditions E0, P0, C0 and C1.

Proof (of E0):

$$\begin{array}{ll}
 & [X \Rightarrow Y] \\
\Rightarrow & \{[X \wedge \neg Y \equiv false]\} \\
 & \langle\forall s :: [X \wedge \neg Y \wedge grd.s \Rightarrow \mathbf{wp}\,.s.Y]\rangle \wedge \\
 & \langle\exists t :: [X \wedge \neg Y \Rightarrow grd.t]\rangle \\
= & \{\text{definition of } \mathcal{E}\} \\
 & (X\ \mathcal{E}\ Y)
\end{array}$$

(End of Proof)

Proof (of P0):

$$\begin{array}{ll}
 & [X \Rightarrow Y] \\
\Rightarrow & \{\text{property S2; monotonicity of } \forall\} \\
 & [grd.s. \wedge \langle\forall t : grd.t : \mathbf{wp}\,.t.X\rangle \Rightarrow \\
 & grd.s. \wedge \langle\forall t : grd.t : \mathbf{wp}\,.t.Y\rangle] \\
= & \{\text{definition of } \mathbf{gwp}\,.s\} \\
 & [\mathbf{gwp}\,.s.X \Rightarrow \mathbf{gwp}\,.s.Y]
\end{array}$$

(End of Proof)

Proof (of C0):

$$\begin{array}{ll}
 & [X \Rightarrow \mathbf{wlt}\,.Y] \\
\Leftarrow & \{\text{Lemma 2, definition of } \mathbf{wlt}\} \\
 & [X \Rightarrow Y \vee \langle\exists s :: \mathbf{gwp}\,.s.Y\rangle] \\
= & \{\text{predicate calculus}\} \\
 & [X \wedge \neg Y \Rightarrow \langle\exists s :: \mathbf{gwp}\,.s.Y\rangle] \\
= & \{\text{definition of } \mathbf{gwp}\,.s\} \\
 & [X \wedge \neg Y \Rightarrow \langle\exists s :: grd.s \wedge \langle\forall t : grd.t : \mathbf{wp}\,.t.Y\rangle\rangle] \\
= & \{\text{predicate calculus}\} \\
 & [X \wedge \neg Y \Rightarrow \langle\exists s :: grd.s\rangle \wedge \langle\forall t : grd.t : \mathbf{wp}\,.t.Y\rangle] \\
= & \{\text{predicate calculus}\} \\
 & [X \wedge \neg Y \Rightarrow \langle\exists s :: grd.s\rangle] \wedge \\
 & [X \wedge \neg Y \Rightarrow \langle\forall t : grd.t : \mathbf{wp}\,.t.Y\rangle] \\
\Leftarrow & \{\text{predicate calculus}\} \\
 & \langle\forall s :: [X \wedge \neg Y \wedge grd.s \Rightarrow \mathbf{wp}\,.s.Y]\rangle \wedge \\
 & \langle\exists t :: [X \wedge \neg Y \Rightarrow grd.t]\rangle \\
= & \{\text{definition of } \mathcal{E}\} \\
 & (X\ \mathcal{E}\ Y)
\end{array}$$

(End of Proof)

Proof (of C1):

$$\begin{array}{cl}
 & \langle \exists s :: \mathbf{gwp}.s.X\rangle \leadsto X \\
= & \{\text{definition of } \mathbf{gwp}.s\} \\
 & \langle \exists s :: grd.s \wedge \langle \forall t : grd.t : \mathbf{wp}.t.X\rangle\rangle \leadsto X \\
\Leftarrow & \{\text{property L2 of } \leadsto\} \\
 & \langle \forall s :: grd.s \wedge \langle \forall t : grd.t : \mathbf{wp}.t.X\rangle \leadsto X\rangle \\
\Leftarrow & \{\text{property L0 of } \leadsto\} \\
 & \langle \forall s :: grd.s \wedge \langle \forall t : grd.t : \mathbf{wp}.t.X\rangle\ \mathcal{E}\ X\rangle \\
\Leftarrow & \{\text{definition of } \mathcal{E}\} \\
 & true
\end{array}$$

(End of Proof)

Weak Fairness

Lemma 5

$$[\mathbf{A}(\langle \forall s :: \mathbf{F}\,\mathbf{G}\,grd.s \Rightarrow \mathbf{G}\,\mathbf{F}(grd.s \wedge \mathbf{X}\,\overline{s})\rangle \ \Rightarrow\ \mathbf{F}\,Y) \ \equiv$$

$$\langle \mu Z :: Y \vee \langle \exists s :: \mathbf{A}((\mathbf{wp}.s.Z \wedge grd.s)\ \mathbf{W}\ Z)\rangle\rangle]$$

Proof (of 5): We prove the contrapositive, i.e,

$$\begin{array}{cl}
 & \neg\,\mathbf{A}(\langle \forall s :: \mathbf{F}\,\mathbf{G}\,grd.s \Rightarrow \mathbf{G}\,\mathbf{F}(grd.s \wedge \mathbf{X}\,\overline{s})\rangle \ \Rightarrow\ \mathbf{F}\,Y) \\
= & \{\text{temporal logic and predicate calculus}\} \\
 & \mathbf{E}\langle \forall s :: (\mathbf{G}\,\mathbf{F}\,\neg grd.s \vee \mathbf{G}\,\mathbf{F}(grd.s \wedge \mathbf{X}\,\overline{s})) \wedge \mathbf{G}\,\neg Y\rangle \\
= & \{\text{Proof Obligation}\} \\
 & \langle \nu X :: \neg Y \wedge \langle \forall s :: \mathbf{E}(X\ \mathbf{U}\,(X \wedge (grd.s \Rightarrow \mathbf{wlp}.s.X)))\rangle\rangle \\
= & \{\neg(X\,\mathbf{W}\,Y)\ \equiv\ \neg Y\,\mathbf{U}(\neg X \wedge \neg Y)\} \\
 & \langle \nu X :: \neg Y \wedge \langle \forall s :: \mathbf{E}\,\neg((\neg\,\mathbf{wlp}.s.X \wedge grd.s)\ \mathbf{W}\ \neg X)\rangle\rangle \\
= & \{\text{property S0}\} \\
 & \langle \nu X :: \neg Y \wedge \langle \forall s :: \mathbf{E}\,\neg((\mathbf{wp}.s.(\neg X) \wedge grd.s)\ \mathbf{W}\ \neg X)\rangle\rangle \\
= & \{[\neg\langle \mu X :: f.X\rangle\ \equiv\ \langle \nu X :: \neg f.(\neg X)\rangle]\} \\
 & \neg\langle \mu Z :: Y \vee \langle \exists s :: \mathbf{A}((\mathbf{wp}.s.Z \wedge grd.s)\ \mathbf{W}\ Z)\rangle\rangle
\end{array}$$

Abbreviating the expression $\mathbf{E}\langle \forall s :: (\mathbf{G}\,\mathbf{F}\,\neg grd.s \vee \mathbf{G}\,\mathbf{F}(grd.s \wedge \mathbf{X}\,\overline{s})) \wedge \mathbf{G}\,\neg Y\rangle$ by Z, our proof obligation is to show that Z is the weakest solution of $X : [X \equiv \neg Y \wedge \langle \forall s :: \mathbf{E}(X\ \mathbf{U}\,(X \wedge (grd.s \Rightarrow \mathbf{wlp}.s.X)))\rangle]$. We first show that Z is a solution and then show that it is the weakest solution.

Ad 0. To show that Z solves the equation, it is sufficient to show (by Theorem 0) that

$$[Z \Rightarrow \neg Y \wedge \langle \forall s :: \mathbf{E}(Z\ \mathbf{U}\,(Z \wedge (grd.s \Rightarrow \mathbf{wlp}.s.Z)))\rangle].$$

We show the two conjuncts separately.

$$\begin{array}{cl}
 & Z \\
= & \{\text{definition of } Z\} \\
 & \mathbf{E}\langle \forall s :: (\mathbf{G}\,\mathbf{F}\,\neg grd.s \vee \mathbf{G}\,\mathbf{F}(grd.s \wedge \mathbf{X}\,\overline{s})) \wedge \mathbf{G}\,\neg Y\rangle \\
\Rightarrow & \{\text{temporal logic and predicate calculus}\} \\
 & \mathbf{E}\,\mathbf{G}\,\neg Y \\
\Rightarrow & \{\text{temporal logic}\} \\
 & \neg Y
\end{array}$$

Z

$=$ {definition of Z}

$\mathbf{E}\langle\forall t :: (\mathbf{G}\,\mathbf{F}\,\neg grd.t \vee \mathbf{G}\,\mathbf{F}(grd.t \wedge \mathbf{X}\,\overline{t})) \wedge \mathbf{G}\,\neg Y\rangle$

$=$ {$\mathbf{G}$ idempotent}

$\mathbf{E}\langle\forall t :: \mathbf{G}(\mathbf{G}\,\mathbf{F}\,\neg grd.t \vee \mathbf{G}\,\mathbf{F}(grd.t \wedge \mathbf{X}\,\overline{t})) \wedge \mathbf{G}\,\neg Y\rangle$

$=$ {interchange quantification}

$\mathbf{E}\,\mathbf{G}\langle\forall t :: (\mathbf{G}\,\mathbf{F}\,\neg grd.t \vee \mathbf{G}\,\mathbf{F}(grd.t \wedge \mathbf{X}\,\overline{t})) \wedge \mathbf{G}\,\neg Y\rangle$

$=$ {temporal logic and predicate calculus}

$\mathbf{E}(\mathbf{G}\langle\forall t :: (\mathbf{G}\,\mathbf{F}\,\neg grd.t \vee \mathbf{G}\,\mathbf{F}(grd.t \wedge \mathbf{X}\,\overline{t})) \wedge \mathbf{G}\,\neg Y\rangle \wedge$
$\langle\forall s :: \mathbf{G}\,\mathbf{F}\,\neg grd.s \vee \mathbf{G}\,\mathbf{F}(grd.s \wedge \mathbf{X}\,\overline{s})\rangle)$

$\Rightarrow$ {temporal logic}

$\mathbf{E}(\mathbf{G}\langle\forall t :: (\mathbf{G}\,\mathbf{F}\,\neg grd.t \vee \mathbf{G}\,\mathbf{F}(grd.t \wedge \mathbf{X}\,\overline{t})) \wedge \mathbf{G}\,\neg Y\rangle \wedge$
$\langle\forall s :: \mathbf{G}\,\mathbf{F}(\neg grd.s \vee (grd.s \wedge \mathbf{X}\,\overline{s}))\rangle)$

$\Rightarrow$ {temporal logic}

$\mathbf{E}(\mathbf{G}\langle\forall t :: (\mathbf{G}\,\mathbf{F}\,\neg grd.t \vee \mathbf{G}\,\mathbf{F}(grd.t \wedge \mathbf{X}\,\overline{t})) \wedge \mathbf{G}\,\neg Y\rangle \wedge$
$\langle\forall s :: \mathbf{F}(grd.s \Rightarrow (grd.s \wedge \mathbf{X}\,\overline{s}))\rangle)$

$\Rightarrow$ {predicate calculus}

$\mathbf{E}(\mathbf{G}\langle\forall t :: (\mathbf{G}\,\mathbf{F}\,\neg grd.t \vee \mathbf{G}\,\mathbf{F}(grd.t \wedge \mathbf{X}\,\overline{t})) \wedge \mathbf{G}\,\neg Y\rangle \wedge$
$\langle\forall s :: \mathbf{F}(grd.s \Rightarrow \mathbf{X}\,\overline{s})\rangle)$

$=$ { $\wedge$ over $\forall$; non–empty programs}

$\mathbf{E}\langle\forall s :: \mathbf{G}\langle\forall t :: (\mathbf{G}\,\mathbf{F}\,\neg grd.t \vee \mathbf{G}\,\mathbf{F}(grd.t \wedge \mathbf{X}\,\overline{t})) \wedge \mathbf{G}\,\neg Y\rangle$
$\wedge\, \mathbf{F}(grd.s \Rightarrow \mathbf{X}\,\overline{s})\rangle$

$\Rightarrow$ {interchange quantification}

$\langle\forall s :: \mathbf{E}(\mathbf{G}\langle\forall t :: (\mathbf{G}\,\mathbf{F}\,\neg grd.t \vee \mathbf{G}\,\mathbf{F}(grd.t \wedge \mathbf{X}\,\overline{t})) \wedge \mathbf{G}\,\neg Y\rangle$
$\wedge\, \mathbf{F}(grd.s \Rightarrow \mathbf{X}\,\overline{s}))\rangle$

$\Rightarrow$ {temporal logic}

$\langle\forall s :: \mathbf{E}(\mathbf{G}\,\mathbf{E}\langle\forall t :: (\mathbf{G}\,\mathbf{F}\,\neg grd.t \vee \mathbf{G}\,\mathbf{F}(grd.t \wedge \mathbf{X}\,\overline{t}))$
$\wedge\, \mathbf{G}\,\neg Y\rangle \;\wedge \mathbf{F}(grd.s \Rightarrow \mathbf{X}\,\overline{s}))\rangle$

$=$ {definition of Z}

$\langle\forall s :: \mathbf{E}(\mathbf{G}\,Z \wedge \mathbf{F}(grd.s \Rightarrow \mathbf{X}\,\overline{s}))\rangle$

$\Rightarrow$ {temporal logic}

$\langle\forall s :: \mathbf{E}(\mathbf{G}\,Z \wedge \mathbf{F}(grd.s \Rightarrow \mathbf{wlp}\,.s.Z))\rangle$

$\Rightarrow$ {$(\mathbf{G}\,X \wedge \mathbf{F}\,Y \Rightarrow X\,\mathbf{U}(X \wedge Y))$}

$\langle\forall s :: \mathbf{E}(Z\;\mathbf{U}\,(Z \wedge (grd.s \Rightarrow \mathbf{wlp}\,.s.Z)))\rangle$

Ad 1. To show that Z is the weakest solution, we have to show that any solution of the equation $X : [X \equiv \neg Y \wedge \langle\forall s :: \mathbf{E}(X\;\mathbf{U}\,(X \;\wedge\; (grd.s \Rightarrow \mathbf{wlp}\,.s.X)))\rangle]$ implies Z.

Assume that X is a solution of the equation and that X holds at a node u. We will show that there exists a path beginning at u such that each statement s whose guard is eventually continuously enabled is executed infinitely often along the path and that at each state along the path $\neg Y$ holds. This would imply that for all statements s, $(\mathbf{F}\,\mathbf{G}\,grd.s \Rightarrow \mathbf{G}\,\mathbf{F}(grd.s \wedge \mathbf{X}\,\overline{s})) \wedge \mathbf{G}\,\neg Y$, that is, Z holds at u.

So assume that X holds at u. Since X is a solution of the equation, $\neg Y \wedge \langle\forall s :: \mathbf{E}(X\;\mathbf{U}\,(X \wedge (grd.s \Rightarrow \mathbf{wlp}\,.s.X)))\rangle$ holds at u. Then $\neg Y$ holds at u and there exists a path from u for which $\langle\forall s :: X\,\mathbf{U}(X \wedge (grd.s \Rightarrow \mathbf{wlp}\,.s.X))\rangle$ holds. Now assume that $\mathbf{F}\,\mathbf{G}\,grd.t$ holds along the same path from u, in particular, $\mathbf{G}\,grd.t$ holds, say from node v. Now, from $X\,\mathbf{U}(X \wedge (grd.t \Rightarrow \mathbf{wlp}\,.t.X))$, X holds at every node on the $u, \ldots, v$ path and further, the path can be extended to a node w, at which $X \wedge (grd.t \Rightarrow \mathbf{wlp}\,.t.X)$ is true. Since X holds at all the nodes on the $u, \ldots, v, \ldots w$ path $\neg Y$ holds at all the nodes on the path as well. Since $\mathbf{G}\,grd.t$ holds at v, $grd.t$ holds at w. Since $X \wedge (grd.t \Rightarrow \mathbf{wlp}\,.t.X)$ holds at w, there is a child of node w (call it x), such that the state of node x is reached by executing statement t in the state of of node w. Since $\mathbf{wlp}\,.t.X$ holds at w, X holds at x. So the same argument can be repeated to extend the path from x so that some statement whose guard is eventually continuously enabled is eventually selected for execution. That is to say, we can construct a path from u such that for all statements s, $(\mathbf{F}\,\mathbf{G}\,grd.s \Rightarrow \mathbf{G}\,\mathbf{F}(grd.s \wedge \mathbf{X}\,\overline{s})) \wedge \mathbf{G}\,\neg Y$ holds for the path. In other words, Z holds at u. (End of Proof)

Lemma 6

$$[\mathbf{A}(Y\ \mathbf{W}\ Z)\ \equiv\ \langle \nu V :: Z \vee (Y \wedge \mathbf{A}\,\mathbf{X}\,V)\rangle]$$

Proof (of 6): We abbreviate $\mathbf{A}(Y\ \mathbf{W}\ Z)$ by X. We prove the lemma in two steps. We first show that X solves the equation $V : [V \equiv Z \vee (Y \wedge \mathbf{A}\,\mathbf{X}\,V)]$ and we then show that it is the weakest solution.

Ad 0. To show that X is a solution, it is enough to show (by Theorem 0) that

$$[X \Rightarrow Z \vee (Y \wedge \mathbf{A}\,\mathbf{X}\,X)].$$

Suppose X holds at a node u. Then, $\mathbf{A}(Y\ \mathbf{W}\ Z)$ holds at u. This means that along any path beginning at u, either Y holds indefinitely or at every node of the path Y holds till a node at which Z holds is reached. That is, either Z holds at u or Y holds at u and X holds at all the children of u. That is $Z \vee (Y \wedge \mathbf{A}\,\mathbf{X}\,X)$ holds at u.

Ad 1. Suppose V is a solution of the equation $V : [V \equiv Z \vee (Y \wedge \mathbf{A}\,\mathbf{X}\,V)]$. To show that X is the weakest solution, we have to show that $[V \Rightarrow X]$. We demonstrate this by proving the contrapositive.

Suppose $\neg X$ holds at a node u. Then there exists a path originating at u such that, $\neg(Y\ \mathbf{W}\ Z)$ holds. That is, there is a sequence of nodes $u, v, \ldots, w, x$ such that $Y \wedge \neg Z$ holds at all the nodes on the segment $u, v, \ldots, w$ and $\neg Y \wedge \neg Z$ holds at x. Since $[\neg V \equiv \neg Z \wedge (\neg Y \vee \mathbf{E}\,\mathbf{X}\,\neg V)]$, we may assert that $\neg V$ holds at x. From the definition of $\neg V$, this means that $\neg V$ holds at w and by a simple induction $\neg V$ holds at u. That is $[\neg X \Rightarrow \neg V]$. (End of Proof)

Theorem 4 (Completeness of wlt)

$$[\mathbf{A}(\langle \forall s :: (\mathbf{F}\,\mathbf{G}\,grd.s \Rightarrow \mathbf{G}\,\mathbf{F}(grd.s \wedge \mathbf{X}\,\overline{s}))\rangle\ \Rightarrow\ \mathbf{F}\ Y)\ \equiv\ \mathbf{wlt}.Y]$$

Proof (of 4):

$$\begin{array}{cl}
 & \mathbf{A}(\langle \forall s :: (\mathbf{F}\,\mathbf{G}\,grd.s \Rightarrow \mathbf{G}\,\mathbf{F}(grd.s \wedge \mathbf{X}\,\overline{s}))\rangle \Rightarrow \mathbf{F}\,Y) \\
= & \{\text{Lemma 5}\} \\
 & \langle \mu Z :: Y \vee \langle \exists s :: \mathbf{A}((\mathbf{wp}.s.Z \wedge grd.s)\ \mathbf{W}\ Z)\rangle\rangle \\
= & \{\text{Lemma 6}\} \\
 & \langle \mu Z :: Y\ \vee \langle \exists s :: \langle \nu V :: Z \vee ((\mathbf{wp}.s.Z \wedge grd.s) \wedge \mathbf{A}\,\mathbf{X}\,V)\rangle\rangle\rangle \\
= & \{[\mathbf{A}\,\mathbf{X}\,V\ \equiv\ \langle \forall t :: \mathbf{wp}.t.V\rangle]\} \\
 & \langle \mu Z :: Y\ \vee \langle \exists s :: \langle \nu V :: Z \vee (\mathbf{wp}.s.Z \wedge grd.s \wedge \langle \forall t :: \mathbf{wp}.t.V\rangle)\rangle\rangle\rangle \\
= & \{\text{definition of } \mathbf{gwp}.s\} \\
 & \langle \mu Z :: Y \vee \langle \exists s :: \mathbf{gwp}.s.Z\rangle\rangle \\
= & \{\text{definition of } \mathbf{wlt}\} \\
 & \mathbf{wlt}.Y
\end{array}$$

(End of Proof)

We now show that the definitions of the predicate transformer $\mathbf{gwp}.s$ and the relation $\mathcal{E}$ satisfy conditions E0, P0, C0 and C1.

Proof (of E0):

$$\begin{array}{cl}
 & [X \Rightarrow Y] \\
\Rightarrow & \{[X \wedge \neg Y \equiv false]\} \\
 & \langle \forall s :: [X \wedge \neg Y\ \Rightarrow\ \mathbf{wp}.s.(X \vee Y)]\rangle \wedge \\
 & \quad \langle \exists s :: [X \wedge \neg Y\ \Rightarrow\ grd.s \wedge \mathbf{wp}.s.Y]\rangle \\
= & \{\text{definition of } \mathcal{E}\} \\
 & (X\ \mathcal{E}\ Y)
\end{array}$$

(End of Proof)

Proof (of P0):

$[X \Rightarrow Y]$
$\Rightarrow$ {property S2; predicate calculus}
$[(\langle\forall t :: \mathbf{wp}.t.Z\rangle \wedge \mathbf{wp}.s.X \wedge grd.s) \vee X$
$\quad \Rightarrow (\langle\forall t :: \mathbf{wp}.t.Z\rangle \wedge \mathbf{wp}.s.Y \wedge grd.s) \vee Y]$
$\Rightarrow$ {Theorem 1}
$[\langle\nu Z :: (\langle\forall t :: \mathbf{wp}.t.Z\rangle \wedge \mathbf{wp}.s.X \wedge grd.s) \vee X\rangle$
$\quad \Rightarrow \langle\nu Z :: (\langle\forall t :: \mathbf{wp}.t.Z\rangle \wedge \mathbf{wp}.s.Y \wedge grd.s) \vee Y\rangle]$
$=$ {definition of $\mathbf{gwp}.s$}
$[\mathbf{gwp}.s.X \Rightarrow \mathbf{gwp}.s.Y]$

(End of Proof)

Proof (of C0):

$(X \,\mathcal{E}\, Y)$
$=$ {definition of $\mathcal{E}$}
$\langle\forall s :: [X \wedge \neg Y \;\Rightarrow\; \mathbf{wp}.s.(X \vee Y)]\rangle \wedge$
$\langle\exists s :: [X \wedge \neg Y \;\Rightarrow\; grd.s \wedge \mathbf{wp}.s.Y]\rangle$
$=$ {interchange quantification}
$[\langle\forall s :: X \wedge \neg Y \;\Rightarrow\; \mathbf{wp}.s.(X \vee Y)\rangle] \wedge$
$\langle\exists s :: [X \wedge \neg Y \;\Rightarrow\; grd.s \wedge \mathbf{wp}.s.Y]\rangle$
$=$ {$\Rightarrow$ over $\forall$}
$[X \wedge \neg Y \;\Rightarrow\; \langle\forall s :: \mathbf{wp}.s.(X \vee Y)\rangle] \wedge$
$\langle\exists s :: [X \wedge \neg Y \;\Rightarrow\; grd.s \wedge \mathbf{wp}.s.Y]\rangle$
$=$ {$\wedge$ over $\exists$}
$\langle\exists s :: [X \wedge \neg Y \;\Rightarrow\; \langle\forall s :: \mathbf{wp}.s.(X \vee Y)\rangle] \wedge$
$[X \wedge \neg Y \;\Rightarrow\; grd.s \wedge \mathbf{wp}.s.Y]\rangle$
$=$ {predicate calculus}
$\langle\exists s :: [X \wedge \neg Y \;\Rightarrow\; \langle\forall s :: \mathbf{wp}.s.(X \vee Y)\rangle \wedge grd.s \wedge \mathbf{wp}.s.Y]\rangle$
$=$ {predicate calculus}
$\langle\exists s :: [X \vee Y \;\Rightarrow\; (\langle\forall s :: \mathbf{wp}.s.(X \vee Y)\rangle \wedge grd.s \wedge \mathbf{wp}.s.Y) \vee Y]\rangle$
$\Rightarrow$ {μ–calculus}
$\langle\exists s :: [X \vee Y \;\Rightarrow\; \langle\nu Z :: (\langle\forall s :: \mathbf{wp}.s.Z\rangle \wedge grd.s \wedge \mathbf{wp}.s.Y) \vee Y\rangle]\rangle$
$\Rightarrow$ {interchange quantification}
$[\langle\exists s :: X \vee Y \;\Rightarrow\; \langle\nu Z :: (\langle\forall s :: \mathbf{wp}.s.Z\rangle \wedge grd.s \wedge \mathbf{wp}.s.Y) \vee Y\rangle\rangle]$
$=$ {$\Rightarrow$ over $\exists$; non–empty programs}
$[X \vee Y \;\Rightarrow\; \langle\exists s :: \langle\nu Z :: (\langle\forall s :: \mathbf{wp}.s.Z\rangle \wedge grd.s \wedge \mathbf{wp}.s.Y) \vee Y\rangle\rangle]$
$=$ {definition of $\mathbf{gwp}.s$}
$[X \vee Y \;\Rightarrow\; \langle\exists s :: \mathbf{gwp}.s.Y\rangle]$
$\Rightarrow$ {Lemma 2}
$[X \vee Y \;\Rightarrow\; \mathbf{wlt}.Y]$
$\Rightarrow$ {predicate calculus}
$[X \;\Rightarrow\; \mathbf{wlt}.Y]$

(End of Proof)

Proof (of C1): The proof will be in two steps. In the first step, we show that $\mathbf{gwp}.s.X \;\mathcal{E}\; X$ holds. In the second step, we use this to show that $\langle\exists s :: \mathbf{gwp}.s.X\rangle \leadsto X$.

$$\begin{array}{ll}
 & \mathbf{gwp}.s.X\ \mathcal{E}\ X \\
= & \{\text{definition of } \mathcal{E}\} \\
 & \langle \forall t :: [\mathbf{gwp}.s.X \wedge \neg X \ \Rightarrow\ \mathbf{wp}.t.(\mathbf{gwp}.s.X \vee X)]\rangle \\
 & \wedge\langle \exists t :: [\mathbf{gwp}.s.X \wedge \neg X \ \Rightarrow\ grd.t \wedge \mathbf{wp}.t.X]\rangle \\
= & \{\text{definition of } \mathbf{gwp}\} \\
 & \langle \forall t :: [\langle \forall u :: \mathbf{wp}.u.(\mathbf{gwp}.s.X)\rangle \wedge grd.s \wedge \mathbf{wp}.s.X \wedge \neg X \\
 & \ \Rightarrow\ \mathbf{wp}.t.(\mathbf{gwp}.s.X \vee X)]\rangle \wedge \\
 & \langle \exists t :: [\langle \forall u :: \mathbf{wp}.u.(\mathbf{gwp}.s.X)\rangle \wedge \mathbf{wp}.s.X \wedge grd.s \wedge \neg X \\
 & \ \Rightarrow\ grd.t \wedge \mathbf{wp}.t.X]\rangle \\
= & \{\text{predicate calculus}\} \\
 & true
\end{array}$$

$$\begin{array}{ll}
 & \langle \exists s :: \mathbf{gwp}.s.X\rangle \leadsto X \\
\Leftarrow & \{\text{property L2}\} \\
 & \langle \forall s :: \mathbf{gwp}.s.X \leadsto X\rangle \\
\Leftarrow & \{\text{property L0}\} \\
 & \langle \forall s :: \mathbf{gwp}.s.X\ \mathcal{E}\ X\rangle \\
= & \{\text{First step of proof}\} \\
 & true
\end{array}$$

(End of Proof)

Strong Fairness

Definition 1 *For a program F, define Θ_F as follows.*

$$[\Theta_F \equiv \langle \forall s :: \mathbf{G}\,\mathbf{F}\,grd.s \ \Rightarrow\ \mathbf{G}\,\mathbf{F}(grd.s \wedge \mathbf{X}\,\overline{s})\rangle].$$

Lemma 7 *For all programs F,*

$$[\Theta_F \ \equiv\ \mathbf{G}\,\Theta_F]$$

and

$$[\Theta_F \ \equiv\ \mathbf{F}\,\Theta_F].$$

Proof (of 7): We demonstrate each of the equivalences by a derivation.

$$\begin{array}{ll}
 & \mathbf{G}\,\Theta_F \\
= & \{\text{definition of } \Theta_F\} \\
 & \mathbf{G}\langle \forall s :: \mathbf{G}\,\mathbf{F}\,grd.s \Rightarrow \mathbf{G}\,\mathbf{F}(grd.s \wedge \mathbf{X}\,\overline{s})\rangle \\
= & \{\text{interchange quantification}\} \\
 & \langle \forall s :: \mathbf{G}(\mathbf{G}\,\mathbf{F}\,grd.s \Rightarrow \mathbf{G}\,\mathbf{F}(grd.s \wedge \mathbf{X}\,\overline{s}))\rangle \\
= & \{\text{predicate calculus}\} \\
 & \langle \forall s :: \mathbf{G}(\mathbf{F}\,\mathbf{G}\,\neg grd.s \vee \mathbf{G}\,\mathbf{F}(grd.s \wedge \mathbf{X}\,\overline{s}))\rangle \\
= & \{\ [\mathbf{F}\,\mathbf{G}\,X \equiv \mathbf{G}\,\mathbf{F}\,\mathbf{G}\,X]\} \\
 & \langle \forall s :: \mathbf{G}(\mathbf{G}\,\mathbf{F}\,\mathbf{G}\,\neg grd.s \vee \mathbf{G}\,\mathbf{F}(grd.s \wedge \mathbf{X}\,\overline{s}))\rangle \\
= & \{\ [\mathbf{G}\,\mathbf{F}\,X \vee \mathbf{G}\,\mathbf{F}\,Y \equiv \mathbf{G}\,\mathbf{F}(X \vee Y)]\} \\
 & \langle \forall s :: \mathbf{G}\,\mathbf{G}\,\mathbf{F}(\mathbf{G}\,\neg grd.s \vee (grd.s \wedge \mathbf{X}\,\overline{s}))\rangle \\
= & \{\mathbf{G} \text{ idempotent}\} \\
 & \langle \forall s :: \mathbf{G}\,\mathbf{F}(\mathbf{G}\,\neg grd.s \vee (grd.s \wedge \mathbf{X}\,\overline{s}))\rangle \\
= & \{\ [\mathbf{G}\,\mathbf{F}\,X \vee \mathbf{G}\,\mathbf{F}\,Y \equiv \mathbf{G}\,\mathbf{F}(X \vee Y)]\} \\
 & \langle \forall s :: \mathbf{G}\,\mathbf{F}\,\mathbf{G}\,\neg grd.s \vee \mathbf{G}\,\mathbf{F}(grd.s \wedge \mathbf{X}\,\overline{s})\rangle \\
= & \{\ [\mathbf{F}\,\mathbf{G}\,X \equiv \mathbf{G}\,\mathbf{F}\,\mathbf{G}\,X]\} \\
 & \langle \forall s :: \mathbf{F}\,\mathbf{G}\,\neg grd.s \vee \mathbf{G}\,\mathbf{F}(grd.s \wedge \mathbf{X}\,\overline{s})\rangle \\
= & \{\text{predicate calculus}\} \\
 & \langle \forall s :: \mathbf{G}\,\mathbf{F}\,grd.s \Rightarrow \mathbf{G}\,\mathbf{F}(grd.s \wedge \mathbf{X}\,\overline{s})\rangle \\
= & \{\text{definition of } \Theta_F\} \\
 & \Theta_F
\end{array}$$

The proof of $[\Theta_F \equiv \mathbf{F}\,\Theta_F]$ is by mutual implication. One direction, $[\Theta_F \Rightarrow \mathbf{F}\,\Theta_F]$ follows trivially from temporal logic ($[X \Rightarrow \mathbf{F}\,X]$). The other direction is given by the following.

$$
\begin{array}{cl}
 & \mathbf{F}\,\Theta_F \\
= & \{\text{definition of } \Theta_F\} \\
 & \mathbf{F}\langle \forall s :: \mathbf{G}\,\mathbf{F}\,grd.s \Rightarrow \mathbf{G}\,\mathbf{F}(grd.s \wedge \mathbf{X}\,\overline{s})\rangle \\
\Rightarrow & \{\text{interchange quantification}\} \\
 & \langle \forall s :: \mathbf{F}(\mathbf{G}\,\mathbf{F}\,grd.s \Rightarrow \mathbf{G}\,\mathbf{F}(grd.s \wedge \mathbf{X}\,\overline{s}))\rangle \\
= & \{\text{predicate calculus}\} \\
 & \langle \forall s :: \mathbf{F}(\mathbf{F}\,\mathbf{G}\,\neg grd.s \vee \mathbf{G}\,\mathbf{F}(grd.s \wedge \mathbf{X}\,\overline{s}))\rangle \\
= & \{\ [\mathbf{F}\,\mathbf{G}\,X \equiv \mathbf{G}\,\mathbf{F}\,\mathbf{G}\,X]\} \\
 & \langle \forall s :: \mathbf{F}(\mathbf{G}\,\mathbf{F}\,\mathbf{G}\,\neg grd.s \vee \mathbf{G}\,\mathbf{F}(grd.s \wedge \mathbf{X}\,\overline{s}))\rangle \\
= & \{\ [\mathbf{G}\,\mathbf{F}\,X \vee \mathbf{G}\,\mathbf{F}\,Y \equiv \mathbf{G}\,\mathbf{F}(X \vee Y)]\} \\
 & \langle \forall s :: \mathbf{F}\,\mathbf{G}\,\mathbf{F}(\mathbf{G}\,\neg grd.s \vee (grd.s \wedge \mathbf{X}\,\overline{s}))\rangle \\
= & \{\ [\mathbf{F}\,\mathbf{G}\,\mathbf{F}\,X \equiv \mathbf{G}\,\mathbf{F}\,X]\} \\
 & \langle \forall s :: \mathbf{G}\,\mathbf{F}(\mathbf{G}\,\neg grd.s \vee (grd.s \wedge \mathbf{X}\,\overline{s}))\rangle \\
= & \{\ [\mathbf{G}\,\mathbf{F}\,X \vee \mathbf{G}\,\mathbf{F}\,Y \equiv \mathbf{G}\,\mathbf{F}(X \vee Y)]\} \\
 & \langle \forall s :: \mathbf{G}\,\mathbf{F}\,\mathbf{G}\,\neg grd.s \vee \mathbf{G}\,\mathbf{F}(grd.s \wedge \mathbf{X}\,\overline{s})\rangle \\
= & \{\ [\mathbf{F}\,\mathbf{G}\,X \equiv \mathbf{G}\,\mathbf{F}\,\mathbf{G}\,X]\} \\
 & \langle \forall s :: \mathbf{F}\,\mathbf{G}\,\neg grd.s \vee \mathbf{G}\,\mathbf{F}(grd.s \wedge \mathbf{X}\,\overline{s})\rangle \\
= & \{\text{predicate calculus}\} \\
 & \langle \forall s :: \mathbf{G}\,\mathbf{F}\,grd.s \Rightarrow \mathbf{G}\,\mathbf{F}(grd.s \wedge \mathbf{X}\,\overline{s})\rangle \\
= & \{\text{definition of } \Theta_F\} \\
 & \Theta_F
\end{array}
$$

(End of Proof)

Corollary 1

$$[\Theta_F \ \equiv\ \mathbf{F}\,\mathbf{G}\,\Theta_F].$$

Lemma 7 *Abbreviate by $\Phi.s.F.X$,*

$$[\Phi.s.F.X \ \equiv\ \mathbf{A}(grd.s \Rightarrow \mathbf{X}(\neg\overline{s} \vee X)) \ \wedge\ \mathbf{A}(\Theta_{F-\{s\}} \Rightarrow \mathbf{F}(grd.s \vee X))].$$

Then,

$$[\mathbf{A}(\Theta_F \ \Rightarrow\ \mathbf{F}\,Y) \ \equiv\ \langle \mu X :: Y \vee \langle \exists s :: \mathbf{A}(\Phi.s.F.X\ \mathbf{W}\ X)\rangle\rangle]$$

Proof (of 7): First, we derive an expression for $\neg\Phi.s.F.\neg X$.

$$
\begin{array}{cl}
 & \neg\Phi.s.F.\neg X \\
= & \{\text{definition of } \Phi\} \\
 & \neg(\mathbf{A}(grd.s \Rightarrow \mathbf{X}(\neg\overline{s} \vee \neg X)) \wedge \\
 & \qquad \mathbf{A}(\Theta_{F-\{s\}} \Rightarrow \mathbf{F}(grd.s \vee \neg X))) \\
= & \{\text{predicate calculus and temporal logic}\} \\
 & \mathbf{E}(grd.s \wedge \mathbf{X}(\overline{s} \wedge X)) \vee \mathbf{E}(\Theta_{F-\{s\}} \wedge \mathbf{G}(\neg grd.s \wedge X))
\end{array}
$$

We discharge our proof obligation by showing the contrapositive, that is,

$$
\begin{array}{cl}
 & \neg\,\mathbf{A}(\Theta_F \Rightarrow \mathbf{F}\,Y) \\
= & \{\text{temporal logic}\} \\
 & \mathbf{E}(\Theta_F \wedge \mathbf{G}\,\neg Y) \\
= & \{\text{proof obligation}\} \\
 & \langle \nu X :: \neg Y \wedge \langle \forall s :: \mathbf{E}(X\ \mathbf{U}\,(X \wedge \neg\Phi.s.F.\neg X))\rangle\rangle \\
= & \{[\neg(X\ \mathbf{W}\ Y) \equiv \neg Y\ \mathbf{U}\,(\neg X \wedge \neg Y)]\ \} \\
 & \langle \nu X :: \neg Y \wedge \langle \forall s :: \mathbf{E}\,\neg(\Phi.s.F.\neg X\ \mathbf{W}\ \neg X\,)\rangle\rangle \\
= & \{[\neg\langle \mu X :: f.X\rangle \equiv \langle \nu X :: \neg f.(\neg X)\rangle]\} \\
 & \neg\langle \mu X :: Y \vee \langle \exists s :: \mathbf{A}(\Phi.s.F.X\ \mathbf{W}\ X\,)\rangle\rangle
\end{array}
$$

We abbreviate $\mathbf{E}(\Theta_F \wedge \mathbf{G}\,\neg Y)$ by Z. We discharge our proof obligation in two steps. We first show that Z solves the equation $X : [X \equiv \neg Y \wedge \langle \forall s :: \mathbf{E}(X\,\mathbf{U}(X \wedge (\neg \Phi.s.F.\neg X)))\rangle]$ and we then show that it is the weakest solution.

Ad 0. To show that Z is a solution, it is enough to show (by Theorem 0) that

$$[Z \Rightarrow \neg Y \wedge \langle \forall s :: \mathbf{E}(Z\ \mathbf{U}\,(Z \wedge \neg \Phi.s.F.\neg Z))\rangle].$$

We prove each conjunct separately.

$$\begin{array}{ll} & Z \\ = & \{\text{definition of } Z\} \\ & \mathbf{E}(\Theta_F \wedge \mathbf{G}\,\neg Y) \\ \Rightarrow & \{\text{temporal logic}\} \\ & \mathbf{E}\,\mathbf{G}\,\neg Y \\ \Rightarrow & \{\text{predicate calculus}\} \\ & \neg Y \end{array}$$

$$
\begin{array}{ll}
 & Z \\
= & \{\text{definition of } Z\} \\
 & \mathbf{E}(\Theta_F \wedge \mathbf{G}\,\neg Y) \\
= & \{\text{predicate calculus}\} \\
 & \mathbf{E}(\Theta_F \wedge \Theta_F \wedge \mathbf{G}\,\neg Y) \\
= & \{\text{Lemma 7}\} \\
 & \mathbf{E}(\mathbf{G}\,\Theta_F \wedge \Theta_F \wedge \mathbf{G}\,\neg Y) \\
= & \{\text{temporal logic and predicate calculus}\} \\
 & \mathbf{E}(\mathbf{G}(\Theta_F \wedge \mathbf{G}\,\neg Y) \wedge \Theta_F) \\
\Rightarrow & \{\text{temporal logic}\} \\
 & \mathbf{E}(\mathbf{G}\,\mathbf{E}(\Theta_F \wedge \mathbf{G}\,\neg Y) \wedge \Theta_F) \\
= & \{\text{definition of } Z\} \\
 & \mathbf{E}(\mathbf{G}\,Z \wedge \Theta_F) \\
= & \{\text{definition of } \Theta_F\} \\
 & \mathbf{E}(\mathbf{G}\,Z \wedge \langle \forall s :: \mathbf{G}\,\mathbf{F}\,grd.s \Rightarrow \mathbf{G}\,\mathbf{F}(grd.s \wedge \mathbf{X}\,\overline{s})\rangle \wedge \Theta_F) \\
= & \{\text{temporal logic; Corollary 1}\} \\
 & \mathbf{E}(\mathbf{G}\,Z \wedge \langle \forall s :: \mathbf{G}\,\mathbf{F}\,grd.s \Rightarrow \mathbf{G}\,\mathbf{F}(grd.s \wedge \mathbf{X}\,\overline{s})\rangle \wedge \\
 & \qquad \mathbf{F}\,\mathbf{G}\,\Theta_F \wedge \mathbf{F}\,\mathbf{G}\,Z \wedge \mathbf{G}\,Z) \\
= & \{\wedge \text{ over } \forall\text{; non-empty programs}\} \\
 & \mathbf{E}(\mathbf{G}\,Z \wedge \langle \forall s :: (\mathbf{G}\,\mathbf{F}\,grd.s \Rightarrow \mathbf{G}\,\mathbf{F}(grd.s \wedge \mathbf{X}\,\overline{s})) \wedge \\
 & \qquad \mathbf{F}\,\mathbf{G}\,\Theta_F \wedge \mathbf{F}\,\mathbf{G}\,Z \wedge \mathbf{G}\,Z\rangle) \\
\Rightarrow & \{\text{predicate calculus}\} \\
 & \mathbf{E}(\mathbf{G}\,Z \wedge \langle \forall s :: (\mathbf{F}\,\mathbf{G}\,\neg grd.s \wedge \mathbf{F}\,\mathbf{G}\,Z \wedge \mathbf{F}\,\mathbf{G}\,\Theta_F) \\
 & \qquad \vee (\mathbf{G}\,\mathbf{F}(grd.s \wedge \mathbf{X}\,\overline{s}) \wedge \mathbf{G}\,Z)\rangle) \\
\Rightarrow & \{\text{definition of } \Theta\} \\
 & \mathbf{E}(\mathbf{G}\,Z \wedge \langle \forall s :: (\mathbf{F}\,\mathbf{G}\,\neg grd.s \wedge \mathbf{F}\,\mathbf{G}\,Z \wedge \mathbf{F}\,\mathbf{G}\,\Theta_{F-\{s\}}) \vee \\
 & \qquad (\mathbf{G}\,\mathbf{F}(grd.s \wedge \mathbf{X}\,\overline{s}) \wedge \mathbf{G}\,Z)\rangle) \\
\Rightarrow & \{\text{temporal logic}\} \\
 & \mathbf{E}(\mathbf{G}\,Z \wedge \langle \forall s :: \mathbf{F}\,\mathbf{G}(\neg grd.s \wedge Z \wedge \Theta_{F-\{s\}}) \vee \\
 & \qquad \mathbf{G}\,\mathbf{F}(grd.s \wedge \mathbf{X}(\overline{s} \wedge Z))\rangle) \\
\Rightarrow & \{\text{temporal logic}\} \\
 & \mathbf{E}(\mathbf{G}\,Z \wedge \langle \forall s :: \mathbf{F}(\mathbf{G}(\neg grd.s \wedge Z) \wedge \Theta_{F-\{s\}}) \vee \\
 & \qquad \mathbf{F}(grd.s \wedge \mathbf{X}(\overline{s} \wedge Z))\rangle) \\
\Rightarrow & \{\text{temporal logic}\} \\
 & \mathbf{E}(\mathbf{G}\,Z \wedge \langle \forall s :: \mathbf{F}\,\mathbf{E}(\mathbf{G}(\neg grd.s \wedge Z) \wedge \Theta_{F-\{s\}}) \vee \\
 & \qquad \mathbf{F}\,\mathbf{E}(grd.s \wedge \mathbf{X}(\overline{s} \wedge Z))\rangle) \\
= & \{\text{temporal logic}\} \\
 & \mathbf{E}(\mathbf{G}\,Z \wedge \langle \forall s :: \mathbf{F}(\mathbf{E}(\mathbf{G}(\neg grd.s \wedge Z) \wedge \Theta_{F-\{s\}}) \vee \\
 & \qquad \mathbf{E}(grd.s \wedge \mathbf{X}(\overline{s} \wedge Z)))\rangle) \\
= & \{\text{predicate calculus}\} \\
 & \mathbf{E}(\langle \forall s :: \mathbf{G}\,Z \wedge \mathbf{F}(\mathbf{E}(\mathbf{G}(\neg grd.s \wedge Z) \wedge \Theta_{F-\{s\}}) \vee \\
 & \qquad \mathbf{E}(grd.s \wedge \mathbf{X}(\overline{s} \wedge Z)))\rangle) \\
\Rightarrow & \{\text{predicate calculus}\} \\
 & \langle \forall s :: \mathbf{E}(\mathbf{G}\,Z \wedge \mathbf{F}(\mathbf{E}(\mathbf{G}(\neg grd.s \wedge Z) \wedge \Theta_{F-\{s\}}) \vee \\
 & \qquad \mathbf{E}(grd.s \wedge \mathbf{X}(\overline{s} \wedge Z))))\rangle \\
\Rightarrow & \{[\mathbf{G}\,X \wedge \mathbf{F}\,Y \Rightarrow X\,\mathbf{U}(X \wedge Y)]\} \\
 & \langle \forall s :: \mathbf{E}(Z\ \mathbf{U}(Z \wedge (\mathbf{E}(\mathbf{G}(\neg grd.s \wedge Z) \wedge \Theta_{F-\{s\}}) \vee \\
 & \qquad \mathbf{E}(grd.s \wedge \mathbf{X}(\overline{s} \wedge Z)))))\rangle
\end{array}
$$

Ad 1. To show that Z is the weakest solution, we have to show that any solution of the equation $X : [X \equiv \neg Y \wedge \langle \forall s :: \mathbf{E}(X\,\mathbf{U}(X \wedge \neg\Phi.s.F.\neg X))\rangle]$ implies Z.

Assume that X is a solution of the equation and that X holds at a node u. From the equation, $\neg Y$ holds at u. Choose t to be any statement of the program. Corresponding to t, there exists a path from u on which $X\,\mathbf{U}(X \wedge \neg\Phi.t.F.\neg X)$ holds. From the definition of $\mathbf{U}$, there is a path $u, \ldots, v$, such that at all the nodes on the path X holds and $X \wedge \neg\Phi.t.F.\neg X$ holds at v. Since X holds at all the nodes of the $u, \ldots, v$ path, $\neg Y$ holds at every node of the path as well. From the definition of $\neg\Phi.t.F.\neg X$, two cases can arise.

0. If the first disjunct is true at v, then there exists a path from v, such that $grd.t$ is true in v and $\bar{t} \wedge X$ is true in the second node (call it x) of the path. Thus t can be executed at node v and since X is true at x, the above argument can be repeated with each of the remaining statements of the program, thereby constructing a path from u on which each statement is infinitely often executed. Further, since each X holds at every node, $\neg Y$ holds at every node. That is Z holds at u.

1. If the second disjunct is true at v, then there exists a path from v along which t is only finitely often enabled and $\Theta_{F-\{t\}}$ holds. That is, the path is strongly–fair with respect to F. Further since $\mathbf{G}\,X$ holds along the path, $\mathbf{G}\,\neg Y$ holds as well. Thus we have demonstrated the existence of a strongly–fair path from u along which $\neg Y$ holds. That is, Z holds at u. This concludes our proof obligation. (End of Proof)

Theorem 5 (Completeness of wlt)

$$[\mathbf{A}(\langle \forall s :: \mathbf{G}\,\mathbf{F}\, grd.s \Rightarrow \mathbf{G}\,\mathbf{F}(grd.s \wedge \bar{s})\rangle \Rightarrow \mathbf{F}\ Y) \quad \equiv \quad \mathbf{wlt}.Y]$$

Proof (of 5):

$$\begin{array}{cl}
 & \mathbf{A}(\langle \forall s :: (\mathbf{F}\,\mathbf{G}\, grd.s \Rightarrow \mathbf{G}\,\mathbf{F}(grd.s \wedge \mathbf{X}\,\bar{s}))\rangle \Rightarrow \mathbf{F}\,Y) \\
= & \{\text{Lemma 7}\} \\
 & \langle \mu X :: Y \vee \langle \exists s :: \mathbf{A}(\Phi.s.F.X\ \mathbf{W}\ X)\rangle\rangle \\
= & \{[\mathbf{A}(Y\,\mathbf{W}\,Z) \equiv \langle \nu V :: Z \vee (Y \wedge \mathbf{A}\,\mathbf{X}\,V)\rangle]\} \\
 & \langle \mu X :: Y \vee \langle \exists s :: \langle \nu Z :: (\langle \forall t :: \mathbf{wp}.t.Z\rangle \wedge \Phi.s.F.X) \vee X\,\rangle\rangle \\
= & \{\text{temporal logic and definition of } \mathbf{gwp}\} \\
 & \langle \mu X :: Y \vee \langle s :: \mathbf{gwp}.s.F.X\rangle\rangle \\
= & \{\text{definition of } \mathbf{wlt}\} \\
 & \mathbf{wlt}.F.Y
\end{array}$$

(End of Proof)

We now have to show that the definitions of the predicate transformer $\mathbf{gwp}.s$ and the relation $\mathcal{E}$ satisfy conditions E0, P0, C0 and C1. Unlike the cases that we have considered thusfar, the recursive nature of the definitions of $\mathbf{gwp}.s$ and $\mathcal{E}$ require the use of induction on the size of the programs to prove properties C0 and C1.

Proof (of E0):

$$\begin{array}{cl}
 & [X \Rightarrow Y] \\
\Rightarrow & \{[X \wedge \neg Y \equiv \mathit{false}]\} \\
 & \langle \forall s :: [X \wedge \neg Y \Rightarrow \mathbf{wp}.s.(X \vee Y)]\rangle \wedge \\
 & \langle \exists s :: (X \wedge \neg Y \rightsquigarrow (grd.s \vee Y)\,\mathrm{in}\,\mathrm{F} - \{\mathrm{s}\}) \wedge \\
 & \qquad [X \wedge \neg Y \wedge grd.s \Rightarrow \mathbf{wp}.s.Y]\rangle \\
= & \{\text{definition of } \mathcal{E}\} \\
 & (X\ \mathcal{E}\ Y)
\end{array}$$

(End of Proof)

Proof (of P0):

$[X \Rightarrow Y]$

$\Rightarrow$ {predicate calculus and monotonicity of **wlt**}

$[((\langle \forall t :: \mathbf{wp}.t.Z\rangle \wedge (grd.s \Rightarrow \mathbf{wp}.s.X) \wedge$
$\quad \mathbf{wlt}.(F - \{s\}).(grd.s \vee X)) \vee X) \Rightarrow$
$((\langle \forall t :: \mathbf{wp}.t.Z\rangle \wedge (grd.s \Rightarrow \mathbf{wp}.s.Y) \wedge$
$\quad \mathbf{wlt}.(F - \{s\}).(grd.s \vee Y)) \vee Y)]$

$\Rightarrow$ {Theorem 1}

$[\langle \nu Z :: ((\langle \forall t :: \mathbf{wp}.t.Z\rangle \wedge (grd.s \Rightarrow \mathbf{wp}.s.X) \wedge$
$\quad \mathbf{wlt}.(F - \{s\}).(grd.s \vee X)) \vee X\rangle \Rightarrow$
$\langle \nu Z :: ((\langle \forall t :: \mathbf{wp}.t.Z\rangle \wedge (grd.s \Rightarrow \mathbf{wp}.s.Y) \wedge$
$\quad \mathbf{wlt}.(F - \{s\}).(grd.s \vee Y)) \vee Y\rangle]$

$\Rightarrow$ {Monotonicity of $\exists$, definition of $\mathbf{gwp}.s$}

$[\mathbf{gwp}.s.X \Rightarrow \mathbf{gwp}.s.Y]$

(End of Proof)

As mentioned before we shall prove the remaining properties by induction on the size of the program.

Proof (of C0): Base case : Assume that program F has one statement. This is consistent with our assumption that our programs are non–empty.

$(X \;\mathcal{E}\; Y \,\mathrm{in\,F})$

$=$ {definition of $\mathcal{E}$, one–point rule}

$[X \wedge \neg Y \Rightarrow \mathbf{wp}.s.(X \vee Y)] \wedge$
$[X \wedge \neg Y \wedge grd.s \Rightarrow \mathbf{wp}.s.Y] \wedge$
$(X \wedge \neg Y \leadsto grd.s \vee Y \,\mathrm{in}\, \{\})$

$\Rightarrow$ {predicate calculus}

$[X \wedge \neg Y \;\Rightarrow \mathbf{wp}.s.(X \vee Y) \wedge (grd.s \Rightarrow \mathbf{wp}.s.Y)]$

$=$ {predicate calculus}

$[X \vee Y \;\Rightarrow (\mathbf{wp}.s.(X \vee Y) \wedge (grd.s \Rightarrow \mathbf{wp}.s.Y)) \vee Y]$

$\Rightarrow$ {definition of ν}

$[X \vee Y \;\Rightarrow \langle \nu Z :: (\mathbf{wp}.s.Z \wedge (grd.s \Rightarrow \mathbf{wp}.s.Y)) \vee Y\rangle]$

$=$ {definition of $\mathbf{gwp}.s$ for a single statement program}

$[X \vee Y \;\Rightarrow \mathbf{gwp}.s.F.Y]$

$\Rightarrow$ {Lemma 2}

$[X \vee Y \;\Rightarrow \mathbf{wlt}.F.Y]$

$\Rightarrow$ {predicate calculus}

$[X \;\Rightarrow \mathbf{wlt}.F.Y]$

Induction step : We assume that the theorem holds for all programs with less than k statements. This means that if program G contained less than k statements then from the proof of Theorem 2,

$$(\star) \qquad (X \leadsto Y \,\mathrm{in\,G}) \Rightarrow [\mathrm{X} \Rightarrow \mathbf{wlt}.\mathrm{G.Y}].$$

Consider a program F having k statements. We have the following derivation.

$(X \ \mathcal{E}\ Y \text{ in F})$
$=$ {definition of $\mathcal{E}$}
$\langle \forall s :: [X \wedge \neg Y \Rightarrow \mathbf{wp}.s.(X \vee Y)]\rangle \wedge$
$\langle \exists s :: (X \wedge \neg Y \leadsto (grd.s \vee Y) \text{ in F} - \{\text{s}\}) \wedge$
$[X \wedge \neg Y \wedge grd.s \Rightarrow \mathbf{wp}.s.Y]\rangle$
$\Rightarrow$ {induction hypothesis and $(\star)$}
$\langle \forall s :: [X \wedge \neg Y \Rightarrow \mathbf{wp}.s.(X \vee Y)]\rangle \wedge$
$\langle \exists s :: [X \wedge \neg Y \Rightarrow \mathbf{wlt}.(F - \{s\}).(grd.s \vee Y)] \wedge$
$[X \wedge \neg Y \wedge grd.s \Rightarrow \mathbf{wp}.s.Y]\rangle$
$=$ {interchange quantification, $\Rightarrow$ over $\forall$}
$[X \wedge \neg Y \Rightarrow \langle \forall s :: \mathbf{wp}.s.(X \vee Y)\rangle] \wedge$
$\langle \exists s :: [X \wedge \neg Y \Rightarrow \mathbf{wlt}.(F - \{s\}).(grd.s \vee Y)] \wedge$
$[X \wedge \neg Y \wedge grd.s \Rightarrow \mathbf{wp}.s.Y]\rangle$
$=$ {$\wedge$ over $\exists$}
$\langle \exists s :: [X \wedge \neg Y \Rightarrow \langle \forall s :: \mathbf{wp}.s.(X \vee Y)\rangle] \wedge$
$[X \wedge \neg Y \Rightarrow \mathbf{wlt}.(F - \{s\}).(grd.s \vee Y)] \wedge$
$[X \wedge \neg Y \wedge grd.s \Rightarrow \mathbf{wp}.s.Y]\rangle$
$=$ {predicate calculus}
$\langle \exists s :: [X \wedge \neg Y \Rightarrow (\langle \forall s :: \mathbf{wp}.s.(X \vee Y)\rangle \wedge$
$(grd.s \Rightarrow \mathbf{wp}.s.Y) \wedge \mathbf{wlt}.(F - \{s\}).(grd.s \vee Y))]\rangle$
$\Rightarrow$ {predicate calculus}
$\langle \exists s :: [X \vee Y \Rightarrow (\langle \forall s :: \mathbf{wp}.s.(X \vee Y)\rangle \wedge$
$(grd.s \Rightarrow \mathbf{wp}.s.Y) \wedge \mathbf{wlt}.(F - \{s\}).(grd.s \vee Y)) \vee Y]\rangle$
$\Rightarrow$ {definition of ν}
$\langle \exists s :: [X \vee Y \Rightarrow \langle \nu Z :: (\langle \forall s :: \mathbf{wp}.s.Z\rangle \wedge$
$(grd.s \Rightarrow \mathbf{wp}.s.Y) \wedge \mathbf{wlt}.(F - \{s\}).(grd.s \vee Y)) \vee Y\rangle]\rangle$
$\Rightarrow$ {interchange quantification}
$[X \vee Y \Rightarrow \langle \exists s :: \langle \nu Z :: (\langle \forall s :: \mathbf{wp}.s.Z\rangle \wedge$
$(grd.s \Rightarrow \mathbf{wp}.s.Y) \wedge \mathbf{wlt}.(F - \{s\}).(grd.s \vee Y)) \vee Y\rangle\rangle]$
$=$ {definition of $\mathbf{gwp}.s$}
$[X \vee Y \Rightarrow \langle \exists s :: \mathbf{gwp}.s.F.Y\rangle]$
$\Rightarrow$ {Lemma 2}
$[X \vee Y \Rightarrow \mathbf{wlt}.F.Y]$
$\Rightarrow$ {predicate calculus}
$[X \Rightarrow \mathbf{wlt}.F.Y]$

(End of Proof)

Proof (of 1): Base Case : Assume that program F has one statement.

$(\langle \exists s :: \mathbf{gwp}.s.F.X\rangle \leadsto X)$
$=$ {definition of $\mathbf{gwp}.s$ for a single statement program, $[\mathbf{wlt}.\{\}.Y \equiv Y]$}
$\langle \nu Z :: (\mathbf{wp}.s.Z \wedge (grd.s \Rightarrow \mathbf{wp}.s.X) \wedge (grd.s \vee X)) \vee X\rangle \leadsto X$
$\Leftarrow$ {property L0}
$\langle \nu Z :: (\mathbf{wp}.s.Z \wedge (grd.s \Rightarrow \mathbf{wp}.s.X) \wedge (grd.s \vee X)) \vee X\rangle \ \mathcal{E}\ X$
$=$ {definition of $\mathcal{E}$, abbreviate $\mathbf{gwp}.s.F.X$}
$\langle \forall t :: [\mathbf{gwp}.s.F.X \wedge \neg X \Rightarrow \mathbf{wp}.t.(\mathbf{gwp}.s.F.X \vee X)]\rangle \wedge$
$\langle \exists t :: (\mathbf{gwp}.s.F.X \wedge \neg X \leadsto (grd.t \vee X) \text{ in F} - \{\text{t}\}) \wedge$
$[\mathbf{gwp}.s.F.X \wedge \neg X \wedge grd.t \Rightarrow \mathbf{wp}.t.X]\rangle$
$\Leftarrow$ {definition of $\mathbf{gwp}.s.F.X$}
true

Induction step : We assume that the theorem holds for all programs with less than k statements. This means that if program G has less than k statements then we can assert from the proof of Theorem 2,

$$(\star\star) \qquad [X \Rightarrow \mathbf{wlt}.G.Y] \Rightarrow (X \leadsto Y \text{ in G}).$$

Consider program F to have k statements.

$$\begin{array}{ll}
 & \langle \exists s :: \mathbf{gwp}.s.F.X\rangle \leadsto X \\
\Leftarrow & \{\text{property L2}\} \\
 & \langle \forall t :: \mathbf{gwp}.t.F.X \leadsto X\rangle \\
\Leftarrow & \{\text{property L0}\} \\
 & \langle \forall t :: \mathbf{gwp}.t.F.X\ \mathcal{E}\ X\rangle \\
= & \{\text{definition of } \mathcal{E}\} \\
 & \langle \forall t :: \langle \forall s :: [\mathbf{gwp}.t.F.X \wedge \neg X \Rightarrow \mathbf{wp}.s.(\mathbf{gwp}.t.F.X \vee X)]\rangle \wedge \\
 & \langle \exists s :: (\mathbf{gwp}.t.F.X \wedge \neg X \leadsto (grd.s \vee X)\,\mathrm{in\,F} - \{\mathrm{s}\}) \wedge \\
 & \qquad [\mathbf{gwp}.t.F.X \wedge \neg X \wedge grd.s \Rightarrow \mathbf{wp}.s.X]\rangle\langle \\
= & \{\text{induction hypothesis and } (\star\star)\} \\
 & \langle \forall t :: \langle \forall s :: [\mathbf{gwp}.t.F.X \wedge \neg X \Rightarrow \mathbf{wp}.s.(\mathbf{gwp}.t.F.X \vee X)]\rangle \wedge \\
 & \langle \exists s :: [\mathbf{gwp}.t.F.X \wedge \neg X \Rightarrow \mathbf{wlt}.(F - \{s\}).(grd.s \vee X)] \wedge \\
 & \qquad [\mathbf{gwp}.t.F.X \wedge \neg X grd.s \Rightarrow \mathbf{wp}.s.X]\rangle\rangle \\
= & \{\text{definition of } \mathbf{gwp}.t.F.X\} \\
 & \mathit{true}
\end{array}$$

(End of Proof)

Compositionality

Theorem 7 (Compositionality of wsafe) *Given arbitrary predicates X and Y,*

$$\begin{array}{l}[\mathbf{wsafe}.(F \odot G).X.Y \equiv \\ \quad \langle \nu Z :: \mathbf{wsafe}.G.(\mathbf{wsafe}.F.(X \wedge Z).Y).Y\rangle]\end{array}$$

Proof (of 7): In the following the statement variables s, t and u are assumed to range over programs F, G and $F \odot G$ respectively.

We abbreviate $\mathbf{wsafe}.(F \odot G).X.Y$ by A. From the definition of **wsafe**, this means that A satisfies the following conditions.

(0) $\quad [A \equiv (\langle \forall u :: \mathbf{wp}.u.A\rangle \wedge X) \vee Y]$

(1) $\quad [W \Rightarrow (\langle \forall u :: \mathbf{wp}.u.W\rangle \wedge X) \vee Y] \Rightarrow [W \Rightarrow A]$

We abbreviate $\mathbf{wsafe}.F.(X \wedge A).Y$ by B. From the definition of **wsafe**, this means that B satisfies the following conditions.

(2) $\quad [B \equiv (\langle \forall s :: \mathbf{wp}.u.B\rangle \wedge X \wedge A) \vee Y]$

(3) $\quad \begin{array}{l}[W \Rightarrow (\langle \forall s :: \mathbf{wp}.s.W\rangle \wedge X \wedge A) \vee Y] \\ \qquad\qquad \Rightarrow [W \Rightarrow B]\end{array}$

We abbreviate $\mathbf{wsafe}.G.B.Y$ by C. From the definition of **wsafe**, this means that C satisfies the following conditions.

(4) $\quad [C \equiv (\langle \forall t :: \mathbf{wp}.t.C\rangle \wedge B) \vee Y]$

(5) $\quad [W \Rightarrow (\langle \forall t :: \mathbf{wp}.t.W\rangle \wedge B) \vee Y] \Rightarrow [W \Rightarrow C]$

Our proof obligation requires us to show that A is the weakest solution of the equation $Z : [Z \equiv \mathbf{wsafe}.G.(\mathbf{wsafe}.F.(X \wedge Z).Y).Y]$. We show this in two steps. We first show that A solves the equation and we then show that it is the weakest solution.

Ad 0. To show that A solves the equation, it is enough to show that

$$[A \Rightarrow \mathbf{wsafe}.G.(\mathbf{wsafe}.F.(X \wedge A).Y).Y].$$

That is, we are required to show that $[A \Rightarrow C]$. We first show that $[A \Rightarrow B]$ and use that to show $[A \Rightarrow C]$.

$$\begin{array}{ll}
 & [A \Rightarrow B] \\
\Leftarrow & \{\text{property } (3)\} \\
 & [A \Rightarrow ((\forall s :: \mathbf{wp}.s.A) \wedge X \wedge A) \vee Y] \\
\Leftarrow & \{\text{predicate calculus}\} \\
 & [A \Rightarrow ((\forall u :: \mathbf{wp}.u.A) \wedge X \wedge A) \vee Y] \\
\Leftarrow & \{\text{property } (0)\} \\
 & \mathit{true}
\end{array}$$

$$\begin{array}{ll}
 & [A \Rightarrow C] \\
\Leftarrow & \{\text{property } (5)\} \\
 & [A \Rightarrow ((\forall t :: \mathbf{wp}.t.A) \wedge B) \vee Y] \\
\Leftarrow & \{\text{predicate calculus}\} \\
 & [A \Rightarrow ((\forall u :: \mathbf{wp}.u.A) \wedge B) \vee Y] \\
= & \{\text{property } (0) \text{ and above derivation}\} \\
 & \mathit{true}
\end{array}$$

Ad 1. To show that A is the weakest solution of the equation, we have to show that,

$$[D \equiv \mathbf{wsafe}.G.(\mathbf{wsafe}.F.(X \wedge D).Y).Y] \Rightarrow [D \Rightarrow A].$$

Towards this end, abbreviate $\mathbf{wsafe}.F.(X \wedge D).Y$ by E. From the definition of **wsafe**, this means that,

(6) $[E \equiv ((\forall s :: \mathbf{wp}.s.E) \wedge X \wedge D) \vee Y]$

(7) $[Z \Rightarrow ((\forall s :: \mathbf{wp}.s.Z) \wedge X \wedge D) \vee Y] \Rightarrow [Z \Rightarrow E]$

(8) $[D \equiv ((\forall t :: \mathbf{wp}.t.D) \wedge E) \vee Y]$

(9) $[Z \Rightarrow ((\forall t :: \mathbf{wp}.t.Z) \wedge E) \vee Y] \Rightarrow [Z \Rightarrow D]$

To show that A is the weakest solution, we have to show $[D \Rightarrow A]$. To do this, we first show that $[E \Rightarrow D]$ and use that in showing $[D \Rightarrow A]$.

$$\begin{array}{ll}
 & [E \Rightarrow D] \\
= & \{\text{property } (6)\} \\
 & [(((\forall s :: \mathbf{wp}.s.E) \wedge X \wedge D) \vee Y) \Rightarrow D] \\
= & \{\text{predicate calculus}\} \\
 & [(\forall s :: \mathbf{wp}.s.E) \wedge X \wedge D \Rightarrow D] \wedge [Y \Rightarrow D] \\
= & \{\text{predicate calculus, property } (8)\} \\
 & \mathit{true}
\end{array}$$

$[D \Rightarrow A]$

$\Leftarrow$ {property (1)}

$[D \Rightarrow (\langle\forall u :: \mathbf{wp}.u.D\rangle \wedge X) \vee Y]$

$=$ {property (8)}

$[(\langle\forall t :: \mathbf{wp}.t.D\rangle \wedge E) \vee Y$
$\quad \Rightarrow (\langle\forall u :: \mathbf{wp}.u.D\rangle \wedge X) \vee Y]$

$=$ {property (6)}

$[(\langle\forall t :: \mathbf{wp}.t.D\rangle \wedge ((\langle\forall s :: \mathbf{wp}.s.E\rangle \wedge X \wedge D) \vee Y)) \vee Y$
$\quad \Rightarrow (\langle\forall u :: \mathbf{wp}.u.D\rangle \wedge X) \vee Y]$

$=$ {predicate calculus}

$[(\langle\forall t :: \mathbf{wp}.t.D\rangle \wedge \langle\forall s :: \mathbf{wp}.s.E\rangle \wedge X \wedge D) \vee Y$
$\quad \Rightarrow (\langle\forall u :: \mathbf{wp}.u.D\rangle \wedge X) \vee Y]$

$\Leftarrow$ {monotonicity of **wp**, $[E \Rightarrow D]$}

$[(\langle\forall t :: \mathbf{wp}.t.D\rangle \wedge \langle\forall s :: \mathbf{wp}.s.D\rangle \wedge X \wedge D) \vee Y$
$\quad \Rightarrow (\langle\forall u :: \mathbf{wp}.u.D\rangle \wedge X) \vee Y]$

$=$ {predicate calculus}

true

This concludes the proof of compositionality of **wsafe**. (End of Proof)

"Strong" Equivalence

Theorem 12

$$F \sim_{\mathcal{L}} G \text{ iff } F \cong_{\mathcal{L}^*} G$$

Proof (of 12): We give a sketch of the proof of the difficult direction, that is we show that $F \cong_{\mathcal{L}^*} G$ implies $F \sim_{\mathcal{L}} G$.

Consider the following language: predicate transformers **wlt** and **wsafe**, propositions from $\mathcal{L}$, propositional variables, the usual boolean operators, and the least and greatest fixpoint operators. We will call this language the μ-calculus of **wlt** and **wsafe**, henceforth referred to as the μ-calculus.

Recall that FwCTL* is composed of branching modalities: along all fair paths ($\mathbf{A}_\phi$), and exists a fair path ($\mathbf{E}_\phi$), and path modality: reflexive strong until (**U**), with the aforementioned restrictions on path formulae.

Given a formula in FwCTL*, we will show that there is an equivalent formula in the above μ-calculus. To convert an FwCTL* formula to μ-calculus it suffices to convert a formula of the following form: exists a fair path such that a certain weak Propositional Linear Temporal Logic (wPLTL) formula holds. This follows because $[\mathbf{A}_\phi\, p \equiv \neg\, \mathbf{E}_\phi\, \neg p]$ and the μ-calculus has all the boolean operators.

The wPLTL formulae, mentioned above, will have modalities **G** and **U** (i.e strong until), and negation will only be allowed on propositions. Moreover, for any subformula of the form $\mathbf{G}\, p$, p should be devoid of **U** or should be convertable to a formula which only has **G** and **F** modalities (note that $[\mathbf{F}\, q \equiv \mathit{true}\, \mathbf{U}\, q]$). Such p will be referred to as **U**-*Free*. The negations can be pushed in to be applied only to the propositions because $[\neg(p\, \mathbf{U}\, q) \equiv \mathbf{G}(\neg q) \vee (\neg q\, \mathbf{U}\, \neg p)]$.

Our first step is to convert any wPLTL formula to a normal form. We first show that any wPLTL formulae is equivalent to a ***disjunctive formula (d-formula)*** in which the conjunctions are applied only to propositions. The following facts are used in the proof.

- (1) $(p\, \mathbf{U}\, q) \wedge (r\, \mathbf{U}\, s) \quad \equiv \quad ((p \wedge r)\, \mathbf{U}(s \wedge (p\, \mathbf{U}\, q))) \vee ((p \wedge r)\, \mathbf{U}(q \wedge (r\, \mathbf{U}\, s)))$
- (1′) $\mathbf{G}\, p \wedge (r\, \mathbf{U}\, s) \quad \equiv \quad (p \wedge r)\, \mathbf{U}(s \wedge \mathbf{G}\, p)$

The above formula reduces the number of **U**s in every conjunct occuring on the RHS, and the claim follows by induction.

Next we convert these simplified d-formulae to a normal form in which we have fixpoint operators. The normal form will have the following characteristic.

- (2) In a formula of the form $\langle \mu x :: f(x)\rangle$, if $g \wedge h$ or $g\, \mathbf{U}\, h$ is a subformula of $f(x)$, then x will not appear in g.

In the following P, P_i will denote propositions (*Prop*), x will denote a propositional variable (*PV*) and p, q, r, p_i, q_i will denote arbitrary formulae. We use the following facts:

- (3) $(P \wedge \mathbf{G}\, p)\, \mathbf{U}\, r \;\equiv\; \mathbf{G}\, p \wedge (P\, \mathbf{U}\, r)$
- (4) $(P \wedge (p\, \mathbf{U}\, q))\, \mathbf{U}\, r \;\equiv\; \langle \mu x :: (r \vee ((P \wedge p)\, \mathbf{U}((P \wedge q \wedge x) \vee ((P \wedge q)\, \mathbf{U}\, x) \vee (r \wedge (p\, \mathbf{U}\, q)))))\rangle$
- (5) $(\bigvee_i P_i \wedge (p_i\, \mathbf{U}\, q_i))\, \mathbf{U}\, r \;\equiv\; \langle \mu x :: (r \bigvee_i ((P_i \wedge p_i)\, \mathbf{U}((P_i \wedge q_i \wedge x) \vee ((P_i \wedge q_i)\, \mathbf{U}\, x) \vee (r \wedge (p_i\, \mathbf{U}\, q_i)))))\rangle$

Since we have introduced propositional variables, we must show how to reduce conjunctions of wPLTL formulae and propositional variables. We use the following rule:

- (6) $q \wedge \langle \mu x :: f(x)\rangle \;\equiv\; \langle \mu x :: f(x) \wedge q\rangle$, with $x \wedge q$ replaced by x.

When applying these rules we ensure the following.

- (7) To maintain (2) as an invariant, we do not apply rules (4) or (5) to a formula if it is a subformula of the left operand of another **U** (until).

Also, we do not apply these rules to any subformula inside a **G**, because some of these rules might introduce a **U** inside a **G**, e.g. by rule (1'): $\mathbf{G}\, p \wedge (\mathit{true}\, \mathbf{U}\, s) \;\equiv\; p\, \mathbf{U}(s \wedge \mathbf{G}\, p)$. Formulae inside **G**s will be dealt with later.

The above rules and observations can be used to convert any d-formula to a formula in the class $\mathcal{F}$, whose syntax is given by the rules:

$$\mathcal{F}\ :-\ Prop \ \|\ PV \ \|\ Prop \wedge \mathcal{F} \ \|\ Prop\,\mathbf{U}\,\mathcal{F} \ \|\ \mathbf{G}(\mathbf{U}\text{-Free})\mathcal{F} \ \|\ \mathcal{F}_1 \vee \mathcal{F}_2 \ \|\ \langle \mu x :: \mathcal{F}\rangle, \text{where}\,\mathcal{F}\,\text{has}\,x\,\text{free}$$

We prove this using the following metric ρ. If p is a subformula of the left operand of another until then $\rho = \rho 2$, otherwise $\rho = \rho 1$. The rules for $\rho 1$, and $\rho 2$ are identical, except the last one (i.e. for until)

1. $\rho 1(Prop) = 0$.
2. $\rho 1(PV) = 0$.
3. $\rho 1(p) = \max\,(\rho 1(p1), \rho 1(p2))$, if $p = p1 \vee p2$ or $p = p1 \wedge p2$.
4. $\rho 1(p) = \rho 1(p1)$, if $p = \langle \mu x :: p1\rangle$ or $p = \mathbf{G}\,p1$.
5. $\rho 1(p) = \max(\rho 2(p1)+1, \rho 1(p2))$, if $p = p1\,\mathbf{U}\,p2$.
6. $\rho 2(p) = \max(\rho 2(p1)+1, \rho 2(p2)+1)$, if $p = p1\,\mathbf{U}\,p2$.

The metric ρ is designed to capture the maximum depth of nesting of untils, ignoring the depth along the rightmost (modulo disjunction) chain of operands.

We show that the above rules when properly applied reduce the metric ρ until the formulae are in $\mathcal{F}$ (that is, $\rho \leq 1$). It is an easy exercise to check that (1) and (1′) maintain ρ, i.e, $\rho(p \wedge q) = \max(\rho(p), \rho(q))$. It then follows that rules (3) and (6) also maintain ρ. Also, if we make sure that when applying rules (4) or (5), we inductively first reduce r into $\mathcal{F}$ (so that $\rho(r) \leq 1$), we can show that rule (4) and (5) reduce ρ. We show this for rule (4).

Abbreviating the left and right hand sides of (4) by LHS and RHS respectively, we have,

$$\begin{array}{cl}
 & \rho(RHS) \\
= & \{\text{definition of } \rho\text{; LHS is not in the left operand of } \mathbf{U}\} \\
 & max\{\rho 1(r), \rho 2(p)+1, \rho 1(q \wedge x), \rho 2(q)+1, \rho 1(p\,\mathbf{U}\,q))\} \\
= & \{\rho 1(q \wedge x) = \rho 1(q \wedge \langle \mu x : f(x)\rangle) \text{ and}\} \\
 & \{\text{from (6), } \rho 1(q \wedge \langle \mu x : f(x)\rangle) = max(\rho 1(q), \rho 1(f(x))), \text{ because } x \wedge q \text{ is replaced by } x\} \\
= & \max\{\rho 1(r), \rho 2(p)+1, \rho 1(q), \rho 2(q)+1, \rho 1(p\,\mathbf{U}\,q)\} \\
\leq & \{\text{arithmetic}\} \\
 & \max\{\rho 2(p)+1, 1, \rho 2(q)+1\} \\
< & \{\text{arithmetic}\} \\
 & \max\{\rho 2(p)+2, \rho 2(q)+1\} \\
= & \{\text{definition of } \rho\} \\
 & \rho(LHS)
\end{array}$$

Now, we switch our attention to the formulae of the kind $\mathbf{G}\,p$, where p is $\mathbf{U}$-Free. Since $\mathbf{G}$ distributes over $\wedge$ and $\mathbf{F}$ distributes over $\vee$, every formula $\mathbf{G}\,p$ can be reduced to the form $\mathbf{G}(\vee_i p_i)$, where each p_i is of the PGF form, i.e. $\wedge_m P_m \wedge \mathbf{G}\,p \wedge_n \mathbf{F}\,p_n$, and each p_n is inductively of the PGF form.

$$\begin{array}{l}
\mathbf{G}((\wedge_m P_m \wedge \mathbf{G}\,p \wedge_{n\in N} \mathbf{F}\,p_n) \bigvee_i p_i) \equiv \\
\quad \mathbf{G}(\wedge_{n\in N} \mathbf{F}\,p_n) \wedge \mathbf{G}((\wedge_m P_m \wedge \mathbf{G}\,p) \bigvee_i p_i) \\
\quad \vee \vee_{n'\in N} (((\wedge_m P_m \wedge p) \bigvee_i p_i)\,\mathbf{U}(p_{n'} \wedge_m P_m \wedge \mathbf{G}\,p \wedge_{n\in N-n'} \mathbf{F}\,p_n))\,\mathbf{U}\,G(p \wedge \bigvee_i p_i) \\
\quad \vee G(\bigvee_i p_i)
\end{array}$$

A formula is of the PG form if it is $\wedge_m P_m \wedge \mathbf{G}\,p$. Now, we show how to simplify formulae of the kind $\mathbf{G}(\vee_i PG_i)$.

$$\begin{array}{l}
\mathbf{G}((\wedge_m P_m \wedge \mathbf{G}\,p) \bigvee_i p_i) \equiv \\
\quad (\bigvee_i p_i)\,\mathbf{U}(\mathbf{G}\,p \wedge\ G(\wedge_m P_m \bigvee_i p_i)) \vee \mathbf{G}(\bigvee_i p_i)
\end{array}$$

Finally, we reduce $\mathbf{G}\,\mathbf{F}\,p$, where p is of the PGF form itself.

$$\mathbf{G}\,\mathbf{F}(\wedge_m P_m \wedge \mathbf{G}\,p \wedge_n \mathbf{F}\,p_n) \equiv$$
$$\mathbf{F}\,\mathbf{G}\,p \wedge \mathbf{G}\,\mathbf{F}(\wedge_m P_m) \wedge GF(\wedge_n p_n)$$

Thus, in the original formula, starting with the outermost **G** we can push **G** inside using the above three rules, till **G** occurs in only three forms: $\mathbf{G}(\vee_i \wedge_j P_{ij})$, and $\mathbf{G}\,\mathbf{F}(\vee_i \wedge_j P_{ij})$. Since, we are only going to allow conjunctions with propositions in our normal form, we will keep the **G** in the following form, $\mathbf{G}(CNFPROP \wedge_k \mathbf{F}(CNFPROP_k))$, where $CNFPROP$ is of the form $\vee_i \wedge_j P_{ij}$.

With the normal form $\mathcal{F}$ being a chain of **U**s, and μs, and with the fact that the propositional variables are in conjunction with only propositions, it can be proved by simple structural induction that:

$\mathbf{E}\,f = f$, if f is $Prop$ or PV,
$= Prop \wedge \mathbf{E}\,f1$, if $f = Prop \wedge f1$,
$= \mathbf{E}(P\ \mathbf{U}\ \mathbf{E}\,f1)$, if $f = P\ \mathbf{U}\,f1$,
$= \mathbf{E}\,f1 \vee \mathbf{E}\,f2$, if $f = f1 \vee f2$,
$= \langle \mu x :: \mathbf{E}\,f \rangle$, if $f = \langle \mu x :: f \rangle$.

Finally, $\mathbf{E}\,\mathbf{G}(CNFPROP \wedge_k \mathbf{F}(CNFPROP_k)) = \langle \nu x :: \mathbf{E}\,\mathbf{G}((CNFPROP \wedge_k \mathbf{E}\,\mathbf{F}(CNFPROP_k \wedge x)) \rangle$.

Thus we have managed to push the modality **E** such that it is applied only to **U** and **G** with arguments as state predicates only.

Finally, the above distribution of the modality **E** also holds if **E** is replaced by $\mathbf{E}_\phi$, i.e. exists a fair path, where ϕ is the fairness condition (unconditional or weak, say $\mathbf{G}\,\mathbf{F}\,p$). This follows because, the only case in which an infinite fair path has to be demonstrated is in the case of formulae $\mathbf{G}\,p$, because only such subformulae give the possibility of an infinite path. However, $\mathbf{E}\,\mathbf{G}$ in the fixpoint characterization allows us to take $\mathbf{E}\,\mathbf{F}(CNFPROP_k)$ arbitrarily later, i.e. after an occurence of p, so that if a path is constructed by infinite concatenations, it will have infinitely many p's and hence be fair, and still have the property $\mathbf{G}(CNFPROP \wedge_k \mathbf{F}(CNFPROP_k))$. This only holds for weak PLTL, and hence the restriction on the language FwCTL*.

Since, $\mathbf{E}_\phi\,\mathbf{G}$, and $\mathbf{E}_\phi\,p\,\mathbf{U}\,q$ can be represented in terms of **wlt** and **wsafe** (**wlt** is $\neg\,\mathbf{E}_\phi\,\mathbf{G}$ and **wsafe**$.p.q$ is $\neg\,\mathbf{E}_\phi\,\neg q\,\mathbf{U}\,\neg p$), we have converted FwCTL* to mu-calculus of **wlt** and **wsafe**. Then by Knaster-Tarski Theorem, and since $\mathcal{L}^*$ has $true$, $false$, and is closed under conjunction, disjunction, negation, and under predicate transformers **wlt** and **wsafe**, F and G being "strongly" equivalent imply they are operationally equivalent, i.e. $F \sim_{\mathcal{L}} G$.

Also, **wlt** can be given as a fixpoint of **we**, and hence this direction also holds for **wlt** replaced by **we** in the definition of $\cong_{\mathcal{L}}$. (End of Proof)