REASONING ABOUT PROBABILISTIC ALGORITHMS

Josyula R. Rao

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

TR-90-07 April 1990

Contents

0 Introduction

0.1 Motivation

Ever since Michael Rabin's seminal paper on Probabilistic Algorithms [Rab76], it has been widely recognized that introducing randomization in algorithms has several advantages. Often these algorithms are simpler and more efficient - in terms of time, space and communication complexity - than their deterministic[0] counterparts [Rab76; Rab82b; Rab82a; Her89; BGS88]. With the advent of multi-processing and distributed computing, it has been realized that for certain problems, it is possible to construct a probabilistic algorithm where no deterministic one exists. This is true especially for problems involving the resolution of symmetry. In the last decade several such algorithms for synchronization, communication and coordination between parallel programs have appeared in the literature [FR80; IR81; LR81; CLP84; Her89].

However the gain in simplicity, efficiency and tractability is not without its price. An integral part of algorithm design is a proof of correctness and for probabilistic algorithms, one has to sacrifice the traditional notion of correctness for a quantitative notion - correctness with a non-zero probability.

In this paper, we address this problem of correctness for probabilistic parallel programs. We distinguish between two notions of correctness – *deterministic* and *probabilistic*. The former is defined to capture the traditional notion of absolute correctness, regardless of the use of probabilistic transitions. For the latter, rather than specify a quantitative measure, we reason about the more qualitative notion - correctness *with probability one*. In the sequel, properties of interest will be classified as *deterministic* or *holding with probability one* depending on the notion of correctness used.

Erring on the side of caution, we require *safety* to be a deterministic property. For *progress*, we are interested in proving a restricted class of properties, namely those that are attained with probability one. Proofs of even such a constrained class of properties can be quite tenuous and tricky[1] ; a suggestion for a proof principle for such properties first appeared in [HSP83]. The proof principle was shown to be sound and complete for a class of finite state programs - that is programs with a fixed number of processes and variables ranging over finite domains. It was also

[0] We use the term *deterministic* to mean *non-probabilistic*.

[1] For a compelling example of how unintuitive probabilistic reasoning can be, consider the following example: two coins are independently tossed ad infinitum. Is it the case that with probability one, the system reaches a state in which both coins show heads ? For details, see Example 0 in Section 10.

shown that the proofs of such progress properties did not depend on the actual values of the probabilities used by the probabilistic transition; the properties of interest hold for a wide family of probability distributions. The proof principle provided the basis for a decision procedure for mechanically determining whether a program satisfied a progress property with probability one.

This proof principle has been the essence of several proof systems proposed since then. In [LS82], the authors generalized the temporal propositional logic of linear time to handle random events as well. In [HS84], two propositional probabilistic temporal logics based on temporal logics of branching time are presented. However, they do not address the methodological question of designing probabilistic programs.

In [Pnu83], Pnueli introduced the concept of *extreme fairness.* One of the results of the paper was that a property that held for all extremely fair computations would hold with probability one for all probabilistic computations. Thus, extreme fairness is a sufficient condition to be satisfied by a scheduler in executing probabilistic programs. A proof rule based on extreme fairness and linear time temporal logic was presented. This proof rule has been applied to prove some difficult algorithms in [Pnu83; PZ84; PZ86]. The proofs illustrating the use of this rule are extremely complicated. Some of the complexity stems from the interplay of randomization and parallelism but there is very little support from the proof-system to alleviate it.

In [CM88], Chandy and Misra introduce UNITY - a formalism to aid in the specification and derivation of parallel algorithms. Their thesis is that a small theory - a computation model and its associated proof system - is adequate for clearly stating and reasoning about specifications and developing programs for a variety of application areas. Our thesis is that probabilistic parallel algorithms can be specified, refined and derived with the same rigor and elegance that applies to parallel algorithms. By synthesizing ideas from [HSP83; Pnu83], the theory of predicate transformers [DS89] and UNITY [CM88] we construct a theory to reason about probability and parallelism.

In [HS85], the proof principle of [HSP83] is generalized to develop conditions for the qualitative analysis of *infinite state* probabilistic programs. To the author's knowledge, no proof system incorporating these conditions has yet been proposed.

0.2 Contributions

We begin by presenting a computational model in which the basic elements of synchrony (modelled as a multiple assignment), asynchrony (modelled as nondeterministic choice) and probabilistic choice are chosen as primitives. Unlike state-based computational models, we do not reason about execution sequences: we choose to reason about properties of programs. In our model, a program is a set of probabilistic and deterministic statements and an execution of a program proceeds by repeatedly picking a statement from the set and executing it, with the caveat that in an infinite execution sequence every statement is executed infinitely often. While *unconditional fairness* is required in the selection of a statement to be executed, we require *extreme fairness* in the selection of an alternative of a probabilistic statement.

By defining the *weakest precondition* for a probabilistic statement appropriately, we show that the UNITY relation **unless** and its associated theory can be used to reason about the safety properties of probabilistic programs as well. This is in keeping with our decision to treat all safety properties deterministically. In a like manner, we show how the theory of the UNITY relations **ensures** and $\mapsto$ (read, *leads-to*) can be extended to reason about progress properties that hold *deterministically.*

To specify and verify progress properties that *hold with probability one*[2], the **wp**-semantics are not adequate. It is necessary to define a new predicate transformer **wpp** (read, *weakest probabilistic precondition*) to capture the inherent non-determinacy of the probabilistic construct. The **wpp** is the dual of **wp** and this is reflected in its properties. It turns out that the predicate transformer **wpp** alongwith the notion of *extreme fairness* provides the right generalization of **wp** (or Hoare triples, for that matter). The predicate **wpp** $.s.X$ characterizes all possible states such that if s is executed infinitely often from such a state, then the infinitely often the execution of s terminates in a state satisfying X. Furthermore, it provides the basis for generalizing the relations **unless**, **ensures** and $\mapsto$ to new relations **upto**, **entails** and $\models\!\Rightarrow$ (read, *probabilistic leads-to*) respectively. Using this small set of operators, we construct a powerful theory in which specifications can be clearly stated and refined using a set of inference rules; furthermore the choice of operators is such that they provide heuristic guidance in extracting the program text from the final specification. We have investigated the properties of the operators in detail. A list of these, along with their proofs can be found in the appendix.

Although our proof theory is not compositional with respect to general progress properties, we have results on deriving basic progress properties of a com-

[2] To see the necessity of a new operator, see Example 1 in Section 10

posite probabilistic program from those of its components. Specifically, **unless**, **upto**, **ensures** and **entails** properties compose in our model. Finally, we show that our proof system is sound and complete for proving properties of *finite state* probabilistic programs.

Our proof system is novel in that it shows that probabilistic programs are amenable to the same process of specification, refinement and verification as sequential and parallel programs. We illustrate our proof system by examples from random walk and (two process) mutual exclusion problems. Furthermore, our proof system allows both probabilistic and deterministic properties to be manipulated within a unified framework. This allows one to reuse proofs and reason in a compositional manner. The most complicated example that we have proved in our system is the paradigm of *eventual determinism* [Rao90]. Till now, no proof system has aided the design and development of probabilistic programs. Proof rules are baroque and proofs are conveyed by pictures. Though intuitively appealing, this error-prone procedure has been known to fail. Furthermore, pictures can neither be manipulated nor generalized. For example, it is trivial to generalize the proof of our 2-process mutual exclusion algorithm to an N-process algorithm.

0.3 Plan of Paper

After a short introduction to our notation and preliminary theorems in Section 1, we present the format of a deterministic and a probabilistic statement and their **wp**-semantics in Section 2. Several properties of the **wp** of these statements are derived. In Section 3, we use these properties of **wp** to extend the UNITY operator **unless** to reason about safety properties of probabilistic programs. Section 4 summarizes the progress operators of UNITY and shows how the same operators can be used to reason about *deterministic* progress properties of probabilistic programs. Section 5 introduces the predicate transformer **wpp**. We present several theorems relating the **wp** and **wpp** in Section 6. In Section 7, we define operators to reason about progress properties which hold with probability one. Section 8 contains an exposition on program composition. In Section 9, we address soundness and completeness issues for our logic. We illustrate the application of our theory by classic examples from random walks and two process mutual exclusion in Section 10.

In this paper, all theorems have been proved in the finest detail. In the interests of brevity, all proofs have been collected together in an appendix.

1 Notation and Terminology

We will use the following notational conventions: the expression

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

where $\underline{Q} \in \{\forall, \exists\}$, denotes quantification over all $t.x$ for which x satisfies $r.x$. We call x the **dummy**, $r.x$ the **range** and $t.x$ the **term** of the quantification. We adopt the convention that all formulae are quantified over all free variables occurring in them (these are variables that are neither dummies nor program variables).

Universal quantification over all program variables is denoted by surrounding a predicate by square brackets ([], read: *everywhere*). This unary operator has all the properties of universal quantification over a non-empty range. For a detailed discussion of this notation the reader is referred to [Dija].

For an assignment statement of the form $\overline{x} := \overline{e}$, we denote the predicate **wp**.“$\overline{x} := \overline{e}$”.$X$ by $\{\overline{x} := \overline{e}\}X$.

Next we define a number of junctivity properties for our predicate transformers. The following definitions and theorems have been taken from [DS89].

Definition 0 *A predicate transformer f is said to be* conjunctive *over a bag of predicates V if and only if*

$$[f.\langle \forall X \,:\, X \in V \,:\, X\rangle \equiv \langle \forall X \,:\, X \in V \,:\, f.X\rangle]$$

Definition 1 *A predicate transformer f is said to be* disjunctive *over a bag of predicates V if and only if*

$$[f.\langle \exists X \,:\, X \in V \,:\, X\rangle \equiv \langle \exists X \,:\, X \in V \,:\, f.X\rangle]$$

In other words, the conjunctivity of f describes the extent to which f distributes over universal quantification and its disjunctivity describes how it distributes over existential quantification. The less restricted the V, the stronger the type of junctivity[3]. Accordingly, we can distinguish the following types of junctivity:

- *universally junctive* : junctive over all V.
- *positively junctive* : junctive over all non-empty V.
- *denumerably junctive* : junctive over all non-empty V with denumerably many distinct predicates.
- *finitely junctive* : junctive over all non-empty V with a finite number of distinct predicates.

[3] We use the term *junctive* and its noun form to stand for either *conjunctive* or *disjunctive*.

- *and-continuous* : conjunctive over all non-empty V, the distinct predicates of which can be ordered as a monotonic sequence.
- *or-continuous* : disjunctive over all non-empty V, the distinct predicates of which can be ordered as a monotonic sequence.
- *monotonic* : junctive over all non-empty V, the distinct predicates of which can be ordered as a monotonic sequence of finite length.

The various types of junctivity are related by the following theorem.

Theorem 0 *Relating Junctivity Properties :*

- *(universally junctivity $\Rightarrow$ positive junctivity)*
- *(positive junctivity $\Rightarrow$ denumerable junctivity)*
- *(denumerable conjunctivity $\Rightarrow$ finite conjunctivity and and-continuity)*
- *(denumerable disjunctivity $\Rightarrow$ finite disjunctivity and or-continuity)*
- *Both finite conjunctivity and and-continuous $\Rightarrow$ monotonicity*
- *Both finite disjunctivity and or-continuity $\Rightarrow$ monotonicity*

Theorem 1 *The weakest precondition of a multiple assignment statement is universally conjunctive and universally disjunctive.*

2 The Computational Model

Our computational model is the same as that of UNITY. Our programs consist of three parts: a collection of variable declarations, a set of initial conditions and a finite set of statements. As in UNITY, we call these sections **declare**, **initially** and **assign** respectively. In addition to the *conditional multiple assignment* statements that UNITY allows in the **assign** section, we allow *conditional probabilistic assignment* statements as well.

From an operational point of view, an execution of our program starts from any state that satisfies the initial conditions and proceeds by repeatedly selecting any statement from the **assign** set and executing it, with the constraint that in an infinite execution, each statement is picked infinitely often. This is the only notion of fairness - *unconditional fairness* - that is required in the selection of statements. We do not distinguish between deterministic and probabilistic statements at this level.

We now describe the format of the statements and their *weakest precondition* (**wp**) semantics.

2.1 Deterministic Statements

The only deterministic statement that we allow is the *conditional multiple assignment(CMA)*. This can be informally presented as:

$$\begin{array}{lllll} CMA & :: & \overline{x} & := & \overline{e.0} \quad \textbf{if} \quad b.0 \sim \\ & & & & \overline{e.1} \quad \textbf{if} \quad b.1 \sim \\ & & & & \vdots \\ & & & & \overline{e.(k-1)} \quad \textbf{if} \quad b.(k-1) \end{array}$$

A conditional multiple assignment, causes assignment of values from any list $\overline{e.i}$ whose associated boolean expression $b.i$ is true (The integer variable i can range over the closed interval 0 to $k-1$). If none of the boolean expressions is true, then the corresponding variables are left unchanged. More importantly, if more than one boolean expression is true, then all the corresponding expression lists *must have the same value*. This is required to guarantee that every assignment statement is *deterministic*. The assignment succeeds only if the numbers and types of variables match the corresponding expressions.

Formally, the *weakest precondition* (**wp**) semantics of a conditional multiple assignment is defined as follows. The range of i is from 0 to $k-1$, inclusive.

$$[\textbf{wp}\,.CMA.X \equiv$$
$$\langle \forall i :: b.i \Rightarrow \{\overline{x} := \overline{e.i}\}X\rangle \wedge (\langle \forall i :: \neg b.i\rangle \Rightarrow X)]$$

Theorem 2 *The predicate transformer* **wp** $.CMA$ *is universally conjunctive.*

To prove the disjunctivity of **wp** $.CMA$ requires some more groundwork. We have to make use of the fact that the statement is *deterministic*, that is for all boolean expressions that evaluate to true, the expressions have the same value. Using this notion of determinacy, we can reformulate the definition of **wp** $.CMA$ to show that,

Theorem 3 *The predicate transformer* **wp** $.CMA$ *is universally disjunctive.*

The proof of Theorem 3 is motivated by a similiar proof for the weakest precondition of an **if-fi** statement with disjoint guards, given in [DS89].

2.2 Probabilistic Statements

The only probabilistic statement that we allow is the *conditional probabilistic assignment statement (CPA)*. This can be informally presented as:

$$CPA \;::\; \overline{x} := \overline{e.0} \mid \overline{e.1} \mid \cdots \mid \overline{e.(k-1)} \quad \textbf{if} \quad b$$

A conditional probabilistic assignment is executed as follows. The boolean condition b is evaluated and if it is true, a *k-sided coin* is tossed. The outcome of the coin toss determines the list of expressions $\overline{e.i}$, to be assigned to the list of variables $\overline{x}$. Thus a CPA can give rise to one of k different assignments. Each of these possible assignments will be called a *mode* of the CPA. Notice that we do not attach a probability to each mode; we only require that each mode have a non-zero probability of occurrence and that the sum of probabilities over all the modes equal one.

We now formalize the notion of fairness required in selecting the mode to be executed (or equivalently, the fairness required in tossing the coin). Let X be an arbitrary predicate over program variables. An execution, σ, is *extremely-fair with respect to* X if for all probabilistic statements CPA, if CPA is executed infinitely often from states of σ satisfying X, then every mode of the CPA is executed infinitely often from states of σ satisfying X.

An execution is *extremely fair* if it is extremely fair with respect to all first order expressible predicates X.

In [Pnu83], Pnueli established that to prove that a property holds with probability one over all executions, it is sufficient to show that it holds over all extremely-fair executions. Thus by assuming that the execution of the probabilistic statements is extremely fair in our computational model, we are assured by Pnueli's result, that all properties hold with probability one.

To recapitulate, our computational model requires two notions of fairness – *unconditional fairness* in the selection of statements to be executed and *extreme fairness* in the execution of probabilistic statements.

Formally, the *weakest precondition* (**wp**) semantics of a conditional probabilistic assignment is defined as:

$$[\mathbf{wp}.CPA.X \equiv$$
$$(b \Rightarrow \langle \forall i :: \{\overline{x} := \overline{e.i}\} X \rangle) \wedge (\neg b \Rightarrow X)]$$

We now investigate the junctivity properties of this predicate transformer.

Theorem 4 *The predicate transformer* **wp** $.CPA$ *is universally conjunctive.*

Theorem 5 *The predicate transformer* **wp** $.CPA$ *is or-continuous.*

Theorem 6 *The predicate transformer* **wp** $.CPA$ *is* not *finitely disjunctive.*

2.3 Properties of wp

Based on the results of the previous two sections, we have

Theorem 7 *For all statements s,* **wp** $.s$ *is universally conjunctive and or-continuous. However,* **wp** $.s$ *is not finitely disjunctive.*

Corollary 0 *For all statements s,*

$$[\mathbf{wp}.s.true \equiv true]$$

Corollary 1 *(Law of the Excluded Miracle) For all statements s,*

$$[\mathbf{wp}.s.false \equiv false]$$

3 Reasoning about Safety

In this section, our aim is to define and develop a theory to reason about the safety properties of a probabilistic program. As emphasized in the introduction, we require safety properties to hold deterministically. Since a UNITY program is a special case of a probabilistic program, we would like the relation to be a generalization of UNITY relation for safety, namely, the **unless**. By doing so, we hope to draw on the extensive repertoire of theorems of **unless** that have already been discovered.

In UNITY, the **unless** relation is defined as follows :

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

For this definition of **unless** to satisfy the theory of **unless** as developed in UNITY, it is sufficient for the predicate transformer **wp** $.s$ to meet the following conditions.

1. Truth-preserving

$$[\mathbf{wp}.s.true \equiv true]$$

2. Monotonicity

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

3. Semi-finite conjunctivity

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

4. Semi-universal conjunctivity

$$[\langle \forall i :: \mathbf{wp}.s.(X.i)\rangle \Rightarrow \mathbf{wp}.s.\langle \forall i :: X.i\rangle]$$

From Corollary 0, the predicate transformer **wp** $.s$ is truth-preserving for all the statements that we allow in our probabilistic programs. By Theorem 7, it is universally conjunctive and hence satisfies monotonicity (by Theorem 0), semi-finite conjunctivity and semi-universal conjunctivity as well. Thus we can use the **unless** relation and its theory, as developed in UNITY to reason about the safety properties of probabilistic programs as well.

Remark: It is interesting to note that given that **wp** $.s$ takes a post-condition as an argument to produce a pre-condition and that in a theory of **unless**, we are interested in combining a set of **unless** properties to produce a new one, the direction of the implication sign in the semi-junctivity property of interest is already determined. (End of Remark)

4 UNITY and Progress: ensures and $\mapsto$

At times, it is necessary to prove deterministic progress properties of probabilistic programs. In this section, we extend the machinery of UNITY to handle this.

Basic progress properties in UNITY are specified using the **ensures** relation. This is defined as

$$(X \text{ } \mathbf{ensures} \text{ } Y) \equiv$$
$$(X \text{ } \mathbf{unless} \text{ } Y) \wedge \langle \exists s :: [X \wedge \neg Y \Rightarrow \mathbf{wp}.s.Y]\rangle.$$

For this definition of **ensures** to satisfy the theory of **ensures** as developed in UNITY, it is sufficient for the predicate transformer **wp** $.s$ to meet the following conditions.

1. Law of the Excluded Miracle

$$[\mathbf{wp}.s.\mathit{false} \equiv \mathit{false}]$$

2. Monotonicity

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

3. Semi-finite conjunctivity

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

By Corollary 1, the predicate transformer **wp** $.s$ satisfies the Law of the Excluded Miracle for all the statements that we allow in our probabilistic programs. By Theorem 7, it is universally conjunctive and hence satisfies the conditions of monotonicity (by Theorem 0) and semi-finite conjunctivity as well. Thus we can use the **ensures** relation and its theory as developed in UNITY to reason about **ensures** properties of our programs.

General progress properties in UNITY are defined using the $\mapsto$ (read, *leads to*). The $\mapsto$ relation is defined to be the strongest relation satisfying the following three conditions.

- $(X \text{ } \mathbf{ensures} \text{ } Y) \Rightarrow (X \mapsto Y)$
- $(X \mapsto Y) \wedge (Y \mapsto Z) \Rightarrow (X \mapsto Z)$
- $\langle \forall X : X \in W : X \mapsto Y\rangle$
  $\Rightarrow (\langle \exists X : X \in W : X\rangle \mapsto Y)$

The theorems about $\mapsto$ in [CM88] depend on the properties of the **unless**, **ensures** and the definition of $\mapsto$. We have shown that the properties of **unless** and **ensures** continue to hold even with the addition of probabilistic statements to our computational model. Thus retaining the UNITY definition of $\mapsto$, we can use the theory developed in [CM88] to reason about the *deterministic* progress properties of our probabilistic programs.

The theory of **unless** relation is sufficient to reason about safety properties of probabilistic programs. However the notions of **ensures** and $\mapsto$ are inadequate to reason about progress properties that hold with probability one. They can be used to prove progress properties that have nothing to do with probabilities.That is there exist programs for which $X \not\mapsto Y$ but X leads-to Y with probability one. This is illustrated by the first example in Section 10.

In the next three sections, we show how each of the **unless**, **ensures** and $\mapsto$ can be generalized to reason effectively about properties that hold with probability one.

5 The weakest probabilistic precondition : wpp

The predicate transformer **wp** allowed us to define safety properties of probabilistic programs. For defining progress properties, it turns out that **wp** $.s$ is too restrictive. Intuitively, **wp** $.CPA$ requires *all* modes of the CPA to behave in the same manner, whereas for progress, it is enough if there *exists* a single helpful mode that establishes a desired property. This weaker notion is nicely captured by the predicate transformer

wpp (read, the *weakest probabilistic precondition*).

The predicate transformer **wpp** is defined as follows

$$[\mathbf{wpp}.CMA.X \equiv \mathbf{wp}.CMA.X]$$

$$[\mathbf{wpp}.CPA.X \equiv (b \Rightarrow \langle \exists i :: \{\overline{x} := \overline{e.i}\}X\rangle) \wedge (\neg b \Rightarrow X)]$$

For deterministic statements, **wpp** is the same as **wp** and thus enjoys the same properties. The only difference between **wp**$.CPA$ and **wpp**$.CPA$ is in the presence of an existential quantifier in place of a universal one. In this sense, **wpp**$.CPA$ is the dual of **wp**$.CPA$ and this is reflected in its properties.

Theorem 8 *The predicate transformer* **wpp**$.CPA$ *is and-continuous.*

Theorem 9 *The predicate transformer* **wpp**$.CPA$ *is* not *finitely conjunctive.*

Theorem 10 *The predicate transformer* **wpp**$.CPA$ *is universally disjunctive.*

Remark: In introducing the notion of the *weakest precondition*, Dijkstra defines **wp**$.s.X$ as characterizing all possible states, such that if s is executed from a state satisfying **wp**$.s.X$, then the execution of s terminates in a state in which X is true.

The predicate transformer **wpp**$.s.X$ considered alongwith the notion of extreme fairness generalizes this idea. It characterizes all possible states such that if s is executed infinitely often from a state satisfying **wpp**$.s.X$, then infinitely often the execution of s terminates in a state in which X is true. (End of Remark)

From the definition of **wpp**$.s$, Theorem 2 and 3 and the above it follows that

Theorem 11 *For all statements s,* **wpp**$.s$ *is universally disjunctive and and-continuous. However it is not finitely conjunctive.*

Corollary 2 *For all statements s,*

$$[\mathbf{wpp}.s.true \equiv true]$$

Corollary 3 *For all statements s,*

$$[\mathbf{wpp}.s.false \equiv false]$$

6 Relating wp and wpp

In this section, we present theorems relating the predicate transformers **wp** and **wpp**.

Theorem 12 *For all statements s,*

$$[\mathbf{wp}.s.X \Rightarrow \mathbf{wpp}.s.X]$$

Theorem 13 *For all statements s,*

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

Theorem 14 *For all statements s,*

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

7 Reasoning about Progress

In this section, we develop a relation to reason about progress with probability one. The predicate transformer, **wpp** allows us to generalize the UNITY relations **unless** to **upto** and **ensures** to **entails**. We then introduce the $\rightsquigarrow$ as the reflexive, transitive closure of **entails**. These relations provide the basis for define $\models\!\Rightarrow$ – the probabilistic analog of the $\mapsto$.

7.1 upto

We begin by generalizing the relation **unless**. Consider the definition of **unless**.

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

We use Theorem 14 to weaken this definition to obtain the definition of **upto**.

$$(X\ \mathbf{upto}\ Y) \equiv \langle \forall s :: [X \wedge \neg Y \Rightarrow \mathbf{wp}.s.X \vee \mathbf{wpp}.s.Y]\rangle$$

Intuitively, X **upto** Y captures the following idea : If X holds at any point during the execution of a program, then either

1. Y never holds and X continues to hold forever, or
2. Y holds eventually (it may hold initially when X holds) and X continues to hold until Y holds, or
3. X continues to hold until $\neg X$ holds eventually; the transition from X to $\neg X$ being made by a statement that *could have* taken it to a state satisfying Y.

The interesting (third) case arises when a CPA is executed in a state satisfying $X \wedge \neg Y$. Suppose not all modes of the CPA when executed lead to a state satisfying X and furthermore there exists a mode which will take it to a state satisfying Y. Since there are no guarantees on which mode will be executed, execution of the CPA can lead to a state satisfying $\neg X$, even though there exists a mode that can take it to Y.

One of the consequences of this definition is that in general **upto** includes **unless** and if all statements are deterministic (i.e. conditional multiple assignments) the definition of **upto** reduces to **unless**.

Theorem 15 *The* **upto** *is a generalization of* **unless**.

$$(X \textbf{ unless } Y) \Rightarrow (X \textbf{ upto } Y)$$

Furthermore for a program consisting of only deterministic statements,

$$(X \textbf{ unless } Y) \equiv (X \textbf{ upto } Y)$$

The relation **upto** is weaker than **unless** and accordingly it enjoys a smaller set of properties. Many of the properties of **unless** are not inherited by **upto**. This is not a problem as **upto** is almost never used for specifications; its utility lies in defining operators for progress. There will be few manipulations involving **upto**. A number of properties of **upto** have been investigated and appear in the appendix.

7.2 entails

We propose a new relation **entails** to generalize **ensures**. Consider the definition of **ensures**.

$$(X \textbf{ ensures } Y) \equiv$$
$$(X \textbf{ unless } Y) \wedge \langle \exists s :: [X \wedge \neg Y \Rightarrow \textbf{wp}.s.Y] \rangle.$$

We weaken this definition to obtain the definition of **entails**.

$$(X \textbf{ entails } Y) \equiv$$
$$(X \textbf{ upto } Y) \wedge \langle \exists s :: [X \wedge \neg Y \Rightarrow \textbf{wpp}.s.Y] \rangle.$$

The intuitive meaning of $(X$ **entails** $Y)$ is that if X is infinitely often true in a computation, then Y is infinitely often true. The claim that Y is infinitely often true is justified as follows. Let a X-state be a state satisfying predicate X. Suppose $X \wedge \neg Y$ holds at some point in the execution of the program. By the first conjunct the only way a program can reach a $\neg X$-state, is to execute a statement that *may lead to* a Y-state. Note that the second conjunct assures us of the existence of a statement s, whose execution in a $(X \wedge \neg Y)$-state, *may lead to* a Y-state. By *unconditional fairness*, s must be executed, causing the program to transit to a $\neg X$-state. If X is infinitely often true then each time the transition from a X-state to a $\neg X$-state is made by a statement whose execution may lead to a Y-state. From the finiteness of the set of statements, some statement t whose execution may lead to Y-state is executed infinitely often from X-states. By extreme fairness, every mode of t is executed infinitely often from X-states. In particular, the mode leading to a Y-state is executed infinitely often. It follows that Y is infinitely often true.

The ideas introduced in our computational model - unconditional fairness and extreme fairness - were all intended to justify this definition of the **entails** relation. The relation **entails** plays an important role in the design of probabilistic programs. Besides being the keystone of the proof theory of progress properties, it has a methodological significance as well. In extracting a program from a specification, each **entails** property can usually be translated to a single probabilistic statement. This will be illustrated by examples in a later section.

Theorem 16 *The* **entails** *generalizes* **ensures**

$$(X \textbf{ ensures } Y) \Rightarrow (X \textbf{ entails } Y)$$

Furthermore for a program consisting only of deterministic statements,

$$(X \textbf{ ensures } Y) \equiv (X \textbf{ entails } Y)$$

Several properties of **entails** have been investigated and appear in the appendix.

7.3 The relation $\leadsto$

The relation **entails** is tied closely to the program. We abstract from this by defining the relation $\leadsto$ to be the reflexive, transitive closure of **entails**.

- $(X \textbf{ entails } Y) \Rightarrow (X \leadsto Y)$
- $(X \leadsto Y) \wedge (Y \leadsto Z) \Rightarrow (X \leadsto Z)$

Properties of the $\leadsto$ relation have been investigated and appear in the appendix.

7.4 Probabilistic Leadsto: $\models\!\!\Rightarrow$

In this paper, we shall express all probabilistic progress properties using the $\models\!\!\Rightarrow$ (read, *probabilistic leads-to*). A program has the property $X \models\!\!\Rightarrow Y$ if once X becomes true, Y will become true with probability one. The $\models\!\!\Rightarrow$ is defined to be the strongest relation satisfying the following three axioms.

- $(X \textbf{ unless } Y) \wedge (X \leadsto Y) \Rightarrow (X \models\!\!\Rightarrow Y)$
- $(X \models\!\!\Rightarrow Y) \wedge (Y \models\!\!\Rightarrow Z) \Rightarrow (X \models\!\!\Rightarrow Z)$
- $\langle \forall X :: X \models\!\!\Rightarrow Z \rangle \Rightarrow (\langle \exists X :: X \rangle \models\!\!\Rightarrow Z)$

According to the first axiom, if X is true at any point in the execution of a program, by X **unless** Y it remains true indefinitely or until Y becomes true. In the former case, X is infinitely often true and by $X \rightsquigarrow Y$, Y is infinitely often true. In either case, Y becomes true. The second axiom ensures that $\models\Rightarrow$ is transitively closed and the third axiom ensures that $\models\Rightarrow$ is disjunctively closed.

Probabilistic leads-to is a generalization of the UNITY leads-to. That is,

Theorem 17 $(X \mapsto Y) \Rightarrow (X \models\Rightarrow Y)$

The probabilistic leads-to ($\models\Rightarrow$) enjoys all the properties of $\mapsto$. Proofs of these properties are given in the appendix.

It is possible to formulate an induction principle over well-founded sets for the $\models\Rightarrow$. It is very similiar to the induction principle for $\mapsto$ and will appear in the paper.

8 On program composition

We use the same notions of program composition as UNITY, namely, *union* and *superposition*.

8.1 Composition by union

The *union* of two programs is the union of the sets of statements in the **assign** sections of the two programs. The union of programs F and G is written as $F \; [\!] \; G$. Like set union, it is a symmetric and associative operator. We assume that there are no inconsistencies in the declarations and initializations of the variables in the two programs.
The study of program composition by union is facilitated by the the union theorem.

Theorem 18 Union Theorem:

- (X **unless** Y **in** F $\wedge$ X **unless** Y **in** G) $\equiv$ (X **unless** Y **in** $F \; [\!] \; G$)
- (X **ensures** Y **in** F $\wedge$ X **unless** Y **in** G) $\vee$ (X **unless** Y **in** F $\wedge$ X **ensures** Y **in** G) $\equiv$ (X **ensures** Y **in** $F \; [\!] \; G$)
- (X **upto** Y **in** F $\wedge$ X **upto** Y **in** G) $\equiv$ (X **upto** Y **in** $F \; [\!] \; G$)
- (X **entails** Y **in** F $\wedge$ X **upto** Y **in** G) $\vee$ (X **upto** Y **in** F $\wedge$ X **entails** Y **in** G) $\equiv$ (X **entails** Y **in** $F \; [\!] \; G$)
- (X **entails** Y **in** F $\wedge$ X **unless** Y **in** G) $\vee$ (X **unless** Y **in** F $\wedge$ X **entails** Y **in** G) $\Rightarrow$ (X **entails** Y **in** $F \; [\!] \; G$)

Several corollaries of these theorems are presented in the appendix.

8.2 Conditional Properties

The union theorem illustrates that basic progress properties compose, that is, the property holds of the composite program if its holds of the components. This is not the case with $\rightsquigarrow$, $\mapsto$ and $\models\Rightarrow$.

To address this shortcoming, we resort to *conditional* properties as in UNITY. All program properties seen thusfar have been expressed using one or more relations – **unless**, **ensures**, **upto**, **entails**; these properties are called *unconditional* properties. A conditional property has two parts – *a hypothesis* and *a conclusion*, each of which is a set of unconditional properties. Both the hypothesis and the conclusion can be properties of the F, G or $F \; [\!] \; G$, where G is a generic program. The meaning of a conditional property is as follows : Given the hypothesis as a permise, the conclusion can be proven from the text or specificatin of F. Thus in proving properties, a conditional property is used as an inference rule. The interested reader is referred to [CM88] for further elucidation.

8.3 Superposition

The second structuring operator that we employ in our proofs is the *superposition* operator. This is exactly the same operator as in UNITY. We recapitulate the salient details.

Unlike program union, program superposition is an *asymmetric* operator. Given an *underlying program* (whose variables will be called *underlying variables*), superposition allows it to be transformed by the application of the following two rules.

1. Augmentation Rule. A statement s in the underlying program may be transformed to the statement $s \| r$ where r is a statement that does not assign to the underlying variables and is executed *in synchrony* with s.
2. Restricted Union Rule. A statement r may be added to the underlying program provided that r does not assign to the underlying variables.

By adhering to the discipline of superposition, it is ensured that every property of the underlying program is a property of the transformed program. This is also called the superposition theorem.

9 Comments on Soundness and Completeness

Informal arguments for the soundness of our logic have been presented alongwith the introduction of each operator. In this section, we informally argue that our logic is complete for finite state programs. This is based on the results introduced in [HSP83].

Let Σ be the set of states of the program. Let s be an initial state and let $X(X \subset \Sigma)$ be the set of final states of the program, with $s \notin X$. Define $\hat{I}$ as the set of all states that can be reached (with a non-zero probability) from s before a state in X is reached, using any finite sequence of processes. $\hat{I}$ includes s and is disjoint from X. Furthermore, let K be the set of processes of the program and let $P^k_{i,J}$ be the probability of process k taking the system from state i to any state in set J. One of the main results of [HSP83] is that assuming s, X, Σ and $\hat{I}$ as above and assuming $\hat{I}$ is *finite*, the following two conditions are equivalent.

- $\models s \Longmapsto X$
- There exists a decomposition of $\hat{I}$ into disjoint sets $I_1, I_2, \ldots, I_m$ such that, if we put $J_m = \cup_{r=0}^{m} I_r$, $m = 0, 1, \ldots, n$, with $I_0 = X$, then for each $m = 1, 2, \ldots, n$ we have the following:
  - For each $i \in I_m$, $k \in K$, if $P^k_{i,J_{m-1}} = 0$, then $P^k_{i,I_m} = 1$.
  - There exists $k \equiv k(m) \in K$ such that, for each $i \in I_m$, $P^k_{i,J_{m-1}} > 0$

The first condition says that if process k can transfer the system from a state in I_m to a state outside I_m, then some k-transitions (with non-zero probability) move the system "down" the chain $\{I_r\}$, towards the goal I_0; the second condition ensures the existence of at least one process that would do this for all states in I_m.

Thus given that some progress property holds in a model with probability one, we are guaranteed that the chain $\{I_r\}$ exists. Clearly, $\hat{I}$ **unless** I_0 holds by the definition of $\hat{I}$ and **unless**. For each element of the chain, we can show I_r **entails** J_{r-1}. By using transitivity of $\leadsto$ we can show, $I_r \leadsto I_0$. Using finite disjunction property of $\leadsto$, one can conclude that $\hat{I} \leadsto I_0$. The proof follows from the **unless** property, the $\leadsto$ property and the definition of $\Longmapsto$.

10 Examples

Example 0: (An Unintuitive Example)

To show how unintuitive, reasoning about probabilistic algorithms can be, consider the following program.

$$\begin{array}{lll} \textbf{declare} & & x, y : (heads, tails) \\ \textbf{assign} & & x := heads \mid tails \\ & [] & y := heads \mid tails \\ \textbf{end} & & \end{array}$$

It can be shown that

$$true \Longmapsto (x = heads) \wedge (y = heads)$$

This is because it is possible for the execution of the program to be unconditionally fair with respect to the selection of the coin to be tossed and extremely fair in the tossing of the coins, without reaching a state in which both coins turn up *heads*. Abbreviating *heads* by H and *tails* by T, consider the following segment σ of state transformations: (the state is denoted by the ordered pair giving the values of x and y).

$$(H,T) \xrightarrow{x:=heads} (H,T) \xrightarrow{x:=tails} (T,T)$$

$$\xrightarrow{y:=tails} (T,T) \xrightarrow{y:=heads} (T,H) \xrightarrow{y:=heads}$$

$$(T,H) \xrightarrow{y:=tails} (T,T) \xrightarrow{x:=tails} (T,T) \xrightarrow{x:=heads} (H,T)$$

The sequence σ iterated indefinitely gives an execution sequence which is *unconditionally fair* and *extremely fair*. One way of ensuring that a state satisfying $[(x = heads) \wedge (y = heads)]$ is reached is to use extreme fairness in the scheduling of the statements, rather than unconditional fairness, as illustrated by the program below. This also illustrates the power of extreme fairness over unconditional fairness.

$$\begin{array}{ll} \textbf{declare} & x, y : (H, T) \\ \textbf{assign} & x, y := H, H \mid H, T \mid T, H \mid T, T \\ \textbf{end} & \end{array}$$

(End of Example)

Example 1: (From [Pnu83])

Consider the UNITY program :

$$\begin{array}{lll} \textbf{declare} & & b : integer \\ \textbf{initially} & & b = 0 \\ \textbf{assign} & & b := b + 1 \ \textbf{ if } \ (b \bmod 3) \leq 1 \\ & [] & b := b + 2 \ \textbf{ if } \ (b \bmod 3) \leq 1 \\ \textbf{end} & & \end{array}$$

For this program, it is not the case that

$$true \mapsto (b \bmod 3 = 2)$$

Consider the execution sequence in which the two statements are alternately executed, leading to the following sequence of values for b:

$$0, 1, 3, 4, 6, 7, \ldots$$

This execution sequence is *unconditionally fair* with respect to the two statements but no state of the execution satisfies $(b \, mod \, 3 \; = \; 2)$. Thus the program does not satisfy the progress property *deterministically*.

Now consider the probabilistic program :

declare	$b : integer$
initially	$b = 0$
assign	$b := b + 1 \mid b + 2$ **if** $(b \, mod \, 3) \leq 1$
end	

We show that the required property is achieved with probability one, that is

$$true \models\!\!\!\Longrightarrow (b \, mod \, 3 \; = \; 2)$$

By applying the definition of **wpp** $.s$ it can be shown that **wpp** $.s.((b \, mod \, 3) = 2)]$ evaluates to true. Thus

$\langle \exists s :: [true \wedge \neg(b \, mod \, 3 = 2) \Rightarrow \textbf{wpp} \, .s.((b \, mod \, 3) = 2)] \rangle$

= {predicate calculus}

$[\neg(b \, mod \, 3 = 2) \Rightarrow true]$

= {predicate calculus}

$true$

0. $\langle \exists s :: [true \wedge \neg(b \, mod \, 3 = 2) \Rightarrow \textbf{wpp} \, .s.(b \, mod \, 3 = 2)] \rangle$
   ,From above
1. $true$ **upto** $(b \, mod \, 3 = 2)$
   ,Tautology for **upto**
2. $true$ **entails** $(b \, mod \, 3 = 2)$
   ,From 0, 1 and the definition of **entails**
3. $true$ **unless** $(b \, mod \, 3 = 2)$
   ,Tautology for **unless**
4. $true \models\!\!\!\Longrightarrow (b \, mod \, 3 = 2)$
   ,From 2,3 and the definition of $\models\!\!\!\Longrightarrow$

(End of Example)

Example 2: (Random walk[4] problems)

At any instant of time a particle inhabits one of the integer points of the real line. At time 0, it starts at the specified point and at each subsequent "clock-tick", it moves from its current position to the new position according to the following rule: with probability p it moves one step to the right and with probability $q = 1 - p$, it moves one step to the left; the moves are independent of each other.

[4] In general, random walks can be in many dimensions and the step size can be arbitrary. For ease of exposition we restrict ourselves to one dimension and a step size of 1.

For the random walk problem with no barriers on the real line, it is possible to show that the particle returns to 0 with probability one only if $p = q$. This is also called the *symmetric random walk problem*. Although this property holds with probability one, it is not possible to prove it in our proof system. This is because the property *depends* on the values of the probabilities of the transition, i.e. $p = q$.

There are a class of random walk problems whose progress properties are independent of the values of the probabilities of the transition. As our first example, we consider the same problem alongwith two *absorbing* barriers at 0 and M. This means that that the instant, the particle reaches a barrier it is trapped. The movement of the particle is modelled by the following program.

declare	$x : [0 \ldots M]$
assign	$x := x - 1 \mid x + 1$ **if** $(0 < x \wedge x < M)$
end	

For this program we prove that

$$true \models\!\!\!\Longrightarrow (x = 0) \vee (x = M)$$

We assume, without proof, that

$$\textbf{invariant} \;\; (0 \leq x) \wedge (x \leq M)$$

Assume that the range of k is given by $0 < k \wedge k < M$.

0. $\langle \forall k :: (x = k)$ **entails** $(x = k - 1) \rangle$
   ,From the program text
1. $\langle \forall k :: (x = k) \leadsto (x = 0) \rangle$
   ,Transitivity of $\leadsto$
2. $\langle \exists k :: (x = k) \rangle \leadsto (x = 0)$
   ,Finite disjunction for $\leadsto$
3. $\langle \exists k :: (x = k) \rangle \leadsto (x = M)$
   ,Proof similiar to 2
4. $\langle \exists k :: (x = k) \rangle \leadsto (x = 0) \vee (x = M)$
   ,Finite Disjunction using 2 and 3
5. $(x = 0) \vee (x = M) \leadsto (x = 0) \vee (x = M)$
   ,Implication for $\leadsto$
6. $\langle \exists k :: (x = k) \rangle \vee (x = 0) \vee (x = M) \leadsto (x = 0) \vee (x = M)$
   ,Disjunction of 4 and 5
7. $true \leadsto (x = 0) \vee (x = M)$
   ,predicate calculus and substitution axiom
   ,using invariant above
8. $true$ **unless** $(x = 0) \vee (x = M)$
   ,Tautology for **unless**
9. $true \models\!\!\!\Longrightarrow (x = 0) \vee (x = M)$
   ,From 7, 8 and the definition of $\models\!\!\!\Longrightarrow$

As our second example illustrating random walk, consider two *reflecting* barriers to be placed at 0 and M. This means that when the particle reaches the barrier at 0 (or M) it bounces back to 1 (or $M - 1$)

with probability one. The movement of the particle is modelled by the following program.

declare $x : [0 \ldots M]$
assign $x := x - 1 \mid x + 1$ **if** $(0 < x \wedge x < M)$
$[]$ $x := 1$ **if** $(x = 0)$
$[]$ $x := M - 1$ **if** $(x = M)$
end

For this program, it is easy to show that

$$\textbf{invariant } (0 \leq x) \wedge (x \leq M)$$

The range of k is assumed to be $0 \leq k \wedge k \leq M$. In a manner similiar to the first, we show that

$$true \models\Rightarrow (x = 0)$$

As our third example we consider the case of an *absorbing* barrier at 0 and a *reflecting* barrier at M. The movement of the particle would be modelled by the following program.

declare $x : [0 \ldots M]$
assign $x := x - 1 \mid x + 1$ **if** $(0 < x \wedge x < M)$
$[]$ $x := M - 1$ **if** $(x = M)$
end

For this program, we assume, without proof, that

$$\textbf{invariant } (0 \leq x) \wedge (x \leq M)$$

The range of k is assumed to be $0 < k \wedge k \leq M$. In a manner similiar to the first, we show that

$$true \models\Rightarrow (x = 0)$$

(End of Example)

Example 3: (Two process mutual exclusion)

In this example, we give a brief overview of specification refinement and program composition. Due to constraints of space, we only illustrate the first refinement and indicate what the final program looks like. The example is designed to give a flavor of proof machinery at work.

Specifically, we consider the problem of mutual exclusion between two processes – u, v . Each process u has a variable $u.dine$, which can take one of three values t, h or e, corresponding to *thinking*, *hungry* or *eating*. We abbreviate by $u.t$, $u.h$ and $u.e$, the expressions $u.dine = t$, $u.dine = h$ and $u.dine = e$. We assume that every thinking process eventually becomes hungry. A hungry process remains hungry till it eats. An eating process eats for a finite time and then transits to thinking.

It is required to transform the program *user* to a program *mutex* where $mutex = user' \; [] \; G$. Program $user'$ is obtained from *user* by superposition alone.

The following properties constitute a first refinement. They can be refined further and the final specification can be proven from the program text using the superposition and union theorems.

- **invariant** $u.e \Rightarrow (coin = u)$
- $(coin = v) \models\Rightarrow (coin = u)$
- $u.h \wedge (coin = u) \models\Rightarrow u.e$

The first property guarantees mutual exclusion. The second and third property guarantee starvation freedom.

0. $(coin = v) \models\Rightarrow (coin = u)$
   ,From spec
1. $u.h$ **unless** $u.e$
   ,Property of *mutex*
2. $(u.h \wedge coin = v) \models\Rightarrow (u.h \wedge coin = u) \vee u.e$
   ,PSP Theorem of $\models\Rightarrow$ on 0 and 1
3. $(u.h \wedge coin = v) \models\Rightarrow u.e$
   ,Cancellation on 2 and second property
4. $u.h \models\Rightarrow u.e$
   ,Disjunction on 3 and second property

The program follows. **declare** $coin : (u, v)$
initially $u.dine, v.dine := t, t$
transform $\{to\ user'\}$
all statements of *user* so that the whenever $u.dine$ is set to t,

$$coin := u \mid v$$

add
$\langle [] \; u :: u.dine := e \;\; \textbf{if} \;\; u.h \wedge (coin = u) \rangle$
end
(End of Example)

11 Acknowledgement

I am grateful to Professor Jayadev Misra for advising and supporting me. Thanks are also due to Charanjit Jutla and Sumit Ganguly for suggestions and criticisms.

References

[BGS88] Shaji Bhaskar, Rajive Gupta, and Scott Smolka. Probabilistic algorithms: A survey. Private Communication, 1988.

[CLP84] S. Cohen, Daniel Lehmann, and Amir Pnueli. Symmetric and economic solution to the mutual exclusion problem in distributed systems. *Theoretical Computer Science*, 34:215–226, 1984.

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

[Dija] Edsger W. Dijkstra. On structures. EWD 928.

[Dijb] Edsger W. Dijkstra. On the properties of our predicate transformers. EWD 1001.

[DS89] Edsger W. Dijkstra and Carel S. Scholten. *Predicate Calculus and Program Semantics*. Springer-Verlag, 1989.

[FR80] Nissim Francez and M. Rodeh. A distributed data type implemented by a probabilistic communication scheme. In *Proceedings of the 21st Symposiun on the Foundations of Computer Science*, pages 373–379, 1980.

[Her89] Ted Herman. Probabilistic self-stabilization. Private Communication, 1989.

[HS84] Sergiu Hart and Micha Sharir. Probabilistic propositional temporal logics. In *Proceedings of the 16th Symposium on Theory of Computing*, pages 1–13, 1984.

[HS85] Sergiu Hart and Micha Sharir. Concurrent probabilistic programs, or: How to schedule if you must. *Siam Journal of Computing*, 14:991–1012, 1985.

[HSP83] Sergiu Hart, Micha Sharir, and Amir Pnueli. Termination of probabilistic concurrent programs. *ACM Transactions on Programming Languages and Systems*, 5:356–380, 1983.

[IR81] A. Itai and M. Rodeh. The lord of the ring or probabilistic methods for breaking symmetry in distributive networks. Technical Report RJ 3110, IBM, San Jose, 1981.

[LR81] Daniel Lehmann and Michael O. Rabin. On the advantages of free choice: A symmetric and fully distributed solution to the dining philosophers problem (extended abstract). In *Conference Record of the 8th Annual ACM Symposium on Principles of Programming Languages*, pages 133–138, Williamsburg, VA, 1981.

[LS82] Daniel Lehmann and S. Shelah. Reasoning with time and chance. *Information and Control*, 53:165–190, 1982.

[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.

[PZ84] Amir Pnueli and Lenore Zuck. Verification of multiprocess probabilistic protocols. In *Proceedings of the 3rd Annual ACM Symposium on Principles of Distributed Computing*, pages 12–27, 1984.

[PZ86] Amir Pnueli and Lenore Zuck. Verification of multiprocess protocols. *Distributed Computing*, 1:53–72, 1986.

[Rab76] Michael O. Rabin. *Algorithms and Complexity*, chapter Probabilistic Algorithms, pages 21–40. Academic Press, New York, 1976.

[Rab82a] Michael O. Rabin. The choice coordination problem. *Acta Informatica*, 17:121–134, 1982.

[Rab82b] Michael O. Rabin. N process synchronization with a 4 log_2 n-valued shared variable. *J. Comp. Syst. Sciences*, 25:66–75, 1982.

[Rao90] Josyula R. Rao. Eventual determinism: Using probabilistic means to achieve deterministic ends. Submitted to the 9th Annual ACM Symposium on Principles of Distributed Computing, 1990.

12 Appendix

12.1 Proofs of Theorems: Section 3-7

Theorem 2 *The predicate transformer* **wp** $.CMA$ *is universally conjunctive.*

Proof (of 2):

$$
\begin{aligned}
& \mathbf{wp}.CMA.\langle\forall X : X \in W : X\rangle \\
= & \quad \{\text{Definition of } \mathbf{wp}.CMA\text{; omitting ranges}\} \\
& \langle\forall i :: b.i \Rightarrow \{\overline{x} := \overline{e.i}\}\langle\forall X :: X\rangle\rangle \wedge (\langle\forall i :: \neg b.i\rangle \Rightarrow \langle\forall X :: X\rangle) \\
= & \quad \{\text{universal conjunctivity of multiple assignment}\} \\
& \langle\forall i :: b.i \Rightarrow \langle\forall X :: \{\overline{x} := \overline{e.i}\}X\rangle \wedge (\langle\forall i :: \neg b.i\rangle \Rightarrow \langle\forall X :: X\rangle) \\
= & \quad \{\Rightarrow \text{ over } \forall \text{ twice}\} \\
& \langle\forall i :: \langle\forall X :: b.i \Rightarrow \{\overline{x} := \overline{e.i}\}X\rangle\rangle \wedge (\langle\forall X :: \langle\forall i :: \neg b.i\rangle \Rightarrow X\rangle) \\
= & \quad \{\text{Interchange quantification}\} \\
& \langle\forall X :: \langle\forall i :: b.i \Rightarrow \{\overline{x} := \overline{e.i}\}X\rangle\rangle \wedge (\langle\forall X :: \langle\forall i :: \neg b.i\rangle \Rightarrow X\rangle) \\
= & \quad \{\forall \text{ distributes over } \wedge\} \\
& \langle\forall X :: \langle\forall i :: b.i \Rightarrow \{\overline{x} := \overline{e.i}\}X\rangle \wedge (\langle\forall i :: \neg b.i\rangle \Rightarrow X)\rangle \\
= & \quad \{\text{Definition of } \mathbf{wp}.CMA\} \\
& \langle\forall X :: \mathbf{wp}.CMA.X\rangle
\end{aligned}
$$

(End of Proof)

The following Lemmas and their proofs are from [DS89].

Lemma 0 *We have for any b and R,*

$$[\langle\exists i :: b.i\rangle \equiv \langle\forall i : b.i : R.i\rangle \Rightarrow \langle\exists i : b.i : R.i\rangle]$$

Proof (of 0): We observe for any b and R,

$$
\begin{aligned}
& \langle\forall i : b.i : R.i\rangle \Rightarrow \langle\exists i : b.i : R.i\rangle \\
= & \quad \{\text{Predicate Calculus and de Morgan}\} \\
& \langle\exists i : b.i : \neg R.i\rangle \vee \langle\exists i : b.i : R.i\rangle \\
= & \quad \{\text{Combine the terms}\} \\
& \langle\exists i : b.i : \neg R.i \vee R.i\rangle \\
= & \quad \{\text{Excluded Middle and Trading}\} \\
& \langle\exists i :: b.i\rangle
\end{aligned}
$$

(End of Proof)

Lemma 1 *We have for any b and R,*

$$[\langle\forall i : b.i \wedge b.j : R.i \equiv R.j\rangle \Rightarrow (\langle\exists i : b.i : R.i\rangle \Rightarrow \langle\forall j : b.j : R.j\rangle)]$$

Proof (of 1): We observe for any b and R,

$$
\begin{aligned}
& \langle\exists i : b.i : R.i\rangle \Rightarrow \langle\forall j : b.j : R.j\rangle \\
= & \quad \{\text{Predicate Calculus}\} \\
& \langle\forall i : b.i : \neg R.i\rangle \vee \langle\forall j : b.j : R.j\rangle \\
= & \quad \{\vee \text{ distributes over } \forall\text{; unnesting}\} \\
& \langle\forall i, j : b.i \wedge b.j : \neg R.i \vee R.j\rangle \\
\Leftarrow & \quad \{\text{Excluded Middle}\} \\
& \langle\forall i, j : b.i \wedge b.j : R.i \equiv R.j\rangle
\end{aligned}
$$

(End of Proof)

Lemma 2 *We have for any b and R,*

$$[\langle\forall i : b.i \wedge b.j : R.i \equiv R.j\rangle] \quad \Rightarrow \quad [\langle\exists i :: b.i\rangle \wedge \langle\forall i : b.i : R.i\rangle \equiv \langle\exists i : b.i : R.i\rangle] \ \wedge$$
$$[\neg\langle\exists i :: b.i\rangle \vee \langle\exists i : b.i : R.i\rangle \equiv \langle\forall i : b.i : R.i\rangle]$$

Proof (of 2): We observe for any b and R,

$[\langle\exists i :: b.i\rangle \wedge \langle\forall i : b.i : R.i\rangle \equiv \langle\exists i : b.i : R.i\rangle]$
$=$ {Lemma 0 and predicate calculus}
$[\langle\exists i : b.i : R.i\rangle \wedge \langle\forall i : b.i : R.i\rangle \equiv \langle\exists i : b.i : R.i\rangle]$
$\Leftarrow$ {Lemma 1 and predicate calculus}
$[\langle\forall i : b.i \wedge b.j : R.i \equiv R.j\rangle]$

This gives us the first conjunct on the right. Substitution of $\neg R$ for R in the first conjunct and negating both sides yields the second conjunct. (End of Proof)

Lemma 3

$$[\mathbf{wp}.CMA.X \equiv \langle\exists i : b.i : \{\overline{x} := \overline{e.i}\}X\rangle \vee (\langle\forall i :: \neg b.i\rangle \wedge X)]$$

Proof (of 3):

$\mathbf{wp}.CMA.X$
$=$ {Definition of $\mathbf{wp}.CMA$}
$\langle\forall i : b.i : \{\overline{x} := \overline{e.i}\}X\rangle \wedge (\langle\forall i :: \neg b.i\rangle \Rightarrow X)$
$=$ {predicate calculus}
$(\langle\forall i : b.i : \{\overline{x} := \overline{e.i}\}X\rangle \wedge \langle\exists i :: b.i\rangle) \vee (\langle\forall i : b.i : \{\overline{x} := \overline{e.i}\}X\rangle \wedge X)$
$=$ {First Conjunct of Lemma 2 with $R.i := \{\overline{x} := \overline{e.i}\}X$}
$\langle\exists i : b.i : \{\overline{x} := \overline{e.i}\}X\rangle \vee (\langle\forall i : b.i : \{\overline{x} := \overline{e.i}\}X\rangle \wedge X)$
$=$ {Second Conjunct of Lemma 2 with $R.i := \{\overline{x} := \overline{e.i}\}X$ and predicate calculus}
$\langle\exists i : b.i : \{\overline{x} := \overline{e.i}\}X\rangle \vee (\langle\forall i :: \neg b.i\rangle \wedge X)$

(End of Proof)

Theorem 3 *The predicate transformer* $\mathbf{wp}.CMA$ *is universally disjunctive.*

Proof (of 3):

$\mathbf{wp}.CMA.\langle\exists X : X \in W : X\rangle$
$=$ {Lemma 3 with $X := \langle\exists X :: X\rangle$; omitting ranges}
$\langle\exists i : b.i : \{\overline{x} := \overline{e.i}\}\langle\exists X :: X\rangle\rangle \vee (\langle\forall i :: \neg b.i\rangle \wedge \langle\exists X :: X\rangle)$
$=$ {Universal disjunctivity of multiple assignment; $\wedge$ over $\exists$}
$\langle\exists i : b.i : \langle\exists X :: \{\overline{x} := \overline{e.i}\}X\rangle\rangle \vee (\langle\exists X :: \langle\forall i :: \neg b.i\rangle \wedge X\rangle)$
$=$ {Interchange of quantification}
$\langle\exists X :: \langle\exists i : b.i : \{\overline{x} := \overline{e.i}\}X\rangle\rangle \vee (\langle\exists X :: \langle\forall i :: \neg b.i\rangle \wedge X\rangle)$
$=$ {$\exists$ distributes over $\vee$}
$\langle\exists X :: \langle\exists i : b.i : \{\overline{x} := \overline{e.i}\}X\rangle \vee (\langle\forall i :: \neg b.i\rangle \wedge X)\rangle$
$=$ {Definition $\mathbf{wp}.CMA$}
$\langle\exists X :: \mathbf{wp}.CMA.X\rangle$

(End of Proof)

Theorem 4 *The predicate transformer* $\mathbf{wp}.CPA$ *is universally conjunctive.*

Proof (of 4):

$$
\begin{aligned}
& \mathbf{wp}.CPA.\langle\forall X :: X\rangle \\
= \quad & \{\text{Definition of } \mathbf{wp}.CPA\text{; Omitting ranges}\} \\
& (b \Rightarrow \langle\forall i :: \{\overline{x} := \overline{e.i}\}\langle\forall X :: X\rangle\rangle) \wedge (\neg b \Rightarrow \langle\forall X :: X\rangle) \\
= \quad & \{\text{Universal conjunctivity of multiple assignment}\} \\
& (b \Rightarrow \langle\forall i :: \langle\forall X :: \{\overline{x} := \overline{e.i}\}X\rangle\rangle) \wedge (\neg b \Rightarrow \langle\forall X :: X\rangle) \\
= \quad & \{\text{Interchange of universal quantification}\} \\
& (b \Rightarrow \langle\forall X :: \langle\forall i :: \{\overline{x} := \overline{e.i}\}X\rangle\rangle) \wedge (\neg b \Rightarrow \langle\forall X :: X\rangle) \\
= \quad & \{\Rightarrow \text{ over } \forall\text{, twice}\} \\
& \langle\forall X :: \langle\forall i :: b \Rightarrow \{\overline{x} := \overline{e.i}\}X\rangle\rangle \wedge \langle\forall X :: \neg b \Rightarrow X\rangle \\
= \quad & \{\forall \text{ distributes over } \wedge\} \\
& \langle\langle\forall X :: \langle\forall i :: b \Rightarrow \{\overline{x} := \overline{e.i}\}X\rangle \wedge (\neg b \Rightarrow X)\rangle \\
= \quad & \{\text{Definition of } \mathbf{wp}.CPA\} \\
& \langle\forall X :: \mathbf{wp}.CPA.X\rangle
\end{aligned}
$$

(End of Proof)

Theorem 5 *The predicate transformer* **wp** $.CPA$ *is or-continuous.*

Proof (of 5): Let **wp** $.CPA$ be expressed as

$$[\mathbf{wp}.CPA.X \equiv g.X \wedge h.X]$$

where

$$[g.X \equiv (b \Rightarrow \langle\forall i :: \{\overline{x} := \overline{e.i}\}X\rangle)]$$
$$[h.X \equiv (\neg b \Rightarrow X)]$$

- *g is or-continuous* : From the universal disjunctivity of multiple assignment, the finite range of i and Lemma 3.25, [Dijb] it follows that $\langle\forall i :: \{\overline{x} := \overline{e.i}\}X\rangle$ is or-continuous. It follows that g is or-continuous.
- *h is or-continuous* : It can be easily shown that h is positively disjunctive, which by Theorem 0 implies or-continuity.

Since the conjunction of two or-continuous predicate transformers is or-continuous (Lemma 3.24, [Dijb]), it follows that **wp** $.CPA$ is or-continuous. (End of Proof)

Theorem 6 *The predicate transformer* **wp** $.CPA$ *is* not *finitely disjunctive.*

Proof (of 6): Consider the statement

$$S \;::\; x := heads \mid tails$$

and the assertions

$$[X \equiv (x = heads)] \wedge [Y \equiv (x = tails)]$$

Then

$$
\begin{aligned}
& \mathbf{wp}.S.(X \vee Y) \\
= \quad & \{\text{Definition of } \mathbf{wp}.CPA\} \\
& \{x := heads\}(X \vee Y) \wedge \{x := tails\}(X \vee Y) \\
= \quad & \{\text{Axiom of Assignment}\} \\
& (heads = heads \vee heads = tails) \wedge (tails = heads \vee tails = tails) \\
= \quad & \{\text{predicate calculus}\} \\
& true
\end{aligned}
$$

whereas,

$$
\begin{array}{ll}
 & \mathbf{wp}.S.X \vee \mathbf{wp}.S.Y \\
= & \{\text{Definition of } \mathbf{wp}.S\} \\
 & (\{x := heads\}X \wedge \{x := tails\}X) \vee (\{x := heads\}Y \wedge \{x := tails\}Y) \\
= & \{\text{Axiom of Assignment}\} \\
 & (heads = heads \wedge tails = heads) \vee (heads = tails \wedge tails = tails) \\
= & \{\text{predicate calculus}\} \\
 & false
\end{array}
$$

(End of Proof)

Corollary 0 *For all statements s,*

$$[\mathbf{wp}.s.true \equiv true]$$

Proof (of 0): Follows from the universal conjunctivity of $\mathbf{wp}.s$ and the fact that universal quantification over an empty set is true. (End of Proof)

Corollary 1 *(Law of the Excluded Miracle) For all statements s,*

$$[\mathbf{wp}.s.false \equiv false]$$

Proof (of 1): The theorem holds for CMA, as $\mathbf{wp}.CMA$ is universally disjunctive and existential quantification over an empty set is false. For a probabilistic statement CPA, we proceed as follows.

$$
\begin{array}{ll}
 & \mathbf{wp}.CPA.false \\
= & \{\text{Definition of } \mathbf{wp}.CPA\} \\
 & (b \Rightarrow \langle \forall i :: \{\overline{x} := \overline{e.i}\}false\rangle) \wedge (\neg b \Rightarrow false) \\
= & \{\text{Law of the Excluded Miracle for multiple assignment}\} \\
 & (b \Rightarrow \langle \forall i :: false\rangle) \wedge (\neg b \Rightarrow false) \\
= & \{\text{predicate calculus}\} \\
 & (b \Rightarrow false) \wedge (\neg b \Rightarrow false) \\
= & \{\text{predicate calculus}\} \\
 & false
\end{array}
$$

(End of Proof)

Theorem 8 *The predicate transformer* $\mathbf{wpp}.CPA$ *is and-continuous.*

Proof (of 8): Let $\mathbf{wpp}.CPA$ be expressed as

$$[\mathbf{wpp}.CPA.X \equiv g.X \wedge h.X]$$

where

$$[g.X \equiv (b \Rightarrow \langle \exists i :: \{\overline{x} := \overline{e.i}\}X\rangle)]$$
$$[h.X \equiv (\neg b \Rightarrow X)]$$

- *g is and-continuous* : From the universal conjunctivity of multiple assignment, the finite range of i and the dual of Lemma 3.25, [Dijb] it follows that $\langle \exists i :: \{\overline{x} := \overline{e.i}\}X\rangle$ is and-continuous. It follows that g is and-continuous.
- *h is and-continuous* : It can be easily shown that h is universally conjunctive which by Theorem 0 implies and-continuity.

Since the disjunction of two and-continuous predicate transformers is and-continuous (dual of Lemma 3.24, [Dijb]), it follows that $\mathbf{wpp}.CPA$ is and-continuous. (End of Proof)

Theorem 9 *The predicate transformer* **wpp** $.CPA$ *is* not *finitely conjunctive.*

Proof (of 9): We use the same example as for Theorem 6. Consider the statement

$$S \ :: \ x := heads \mid tails$$

and the assertions

$$[X \equiv (x = heads)] \wedge [Y \equiv (x = tails)]$$

Then

$$\begin{array}{ll} & \mathbf{wpp}.S.(X \wedge Y) \\ = & \{\text{predicate calculus}\} \\ & \mathbf{wpp}.S.false \\ = & \{\text{Definition of } \mathbf{wpp}.S, \text{ Law of Excluded Miracle}\} \\ & false \end{array}$$

whereas,

$$\begin{array}{ll} & \mathbf{wpp}.S.X \vee \mathbf{wpp}.S.Y \\ = & \{\text{Definition of } \mathbf{wpp}.S\} \\ & (\{x := heads\}X \vee \{x := tails\}X) \wedge (\{x := heads\}Y \vee \{x := tails\}Y) \\ = & \{\text{Axiom of Assignment}\} \\ & (heads = heads \vee tails = heads) \wedge (heads = tails \vee tails = tails) \\ = & \{\text{predicate calculus}\} \\ & true \end{array}$$

(End of Proof)

Theorem 10 *The predicate transformer* **wpp** $.CPA$ *is universally disjunctive.*

Proof (of 10):

$$\begin{array}{ll} & \mathbf{wpp}.CPA.\langle \exists X :: X \rangle \\ = & \{\text{Definition of } \mathbf{wpp}.CPA\} \\ & (b \Rightarrow \langle \exists i :: \{\overline{x} := \overline{e.i}\}\langle \exists X :: X \rangle \rangle) \wedge (\neg b \Rightarrow \langle \exists X :: X \rangle) \\ = & \{\text{predicate calculus}\} \\ & (b \wedge \langle \exists i :: \{\overline{x} := \overline{e.i}\}\langle \exists X :: X \rangle \rangle) \vee (\neg b \wedge \langle \exists X :: X \rangle) \\ = & \{\text{Universal disjunctivity of multiple assignment}\} \\ & (b \wedge \langle \exists i :: \langle \exists X :: \{\overline{x} := \overline{e.i}\}X \rangle \rangle) \vee (\neg b \wedge \langle \exists X :: X \rangle) \\ = & \{\text{Interchange existential quantification}\} \\ & (b \wedge \langle \exists X :: \langle \exists i :: \{\overline{x} := \overline{e.i}\}X \rangle \rangle) \vee (\neg b \wedge \langle \exists X :: X \rangle) \\ = & \{\wedge \text{ over } \exists\} \\ & \langle \exists X :: b \wedge \langle \exists i :: \{\overline{x} := \overline{e.i}\}X \rangle \rangle \vee \langle \exists X :: \neg b \wedge X \rangle \\ = & \{\exists \text{ distributes over } \vee\} \\ & \langle \exists X :: (b \wedge \langle \exists i :: \{\overline{x} := \overline{e.i}\}X \rangle) \vee (\neg b \wedge X) \rangle \\ = & \{\text{predicate calculus}\} \\ & \langle \exists X :: (b \Rightarrow \langle \exists i :: \{\overline{x} := \overline{e.i}\}X \rangle) \wedge (\neg b \Rightarrow X) \rangle \\ = & \{\text{Definition of } \mathbf{wp}.CPA\} \\ & \langle \exists X :: \mathbf{wpp}.CPA.X \rangle \end{array}$$

(End of Proof)

Corollary 2 *For all statements s,*

$$[\mathbf{wpp}.s.true \equiv true]$$

Proof (of 2): The theorem holds for CMA, as **wpp** $.CMA$ is universally conjunctive and universal quantification over an empty set is true. For a probabilistic statement CPA, we proceed as follows.

$$
\begin{aligned}
& \mathbf{wpp}\,.CPA.true \\
= \quad & \{\text{Definition of } \mathbf{wpp}\,.CPA\} \\
& (b \Rightarrow \langle \exists i :: \{\overline{x} := \overline{e.i}\}true\rangle) \wedge (\neg b \Rightarrow true) \\
= \quad & \{\{\overline{x} := \overline{e.i}\}true \equiv true\} \\
& (b \Rightarrow \langle \exists i :: true\rangle) \wedge (\neg b \Rightarrow true) \\
= \quad & \{\text{predicate calculus}\} \\
& (b \Rightarrow true) \wedge (\neg b \Rightarrow true) \\
= \quad & \{\text{predicate calculus}\} \\
& true
\end{aligned}
$$

(End of Proof)

Corollary 3 *For all statements s,*

$$[\mathbf{wpp}\,.s.false \equiv false]$$

Proof (of 3): Follows from the universal disjunctivity of **wpp** $.s$ and the fact that existential quantification over an empty set is false. (End of Proof)

Theorem 11 *For all statements S,*

$$[\mathbf{wp}\,.S.X \Rightarrow \mathbf{wpp}\,.s.X]$$

Proof (of 11): The theorem holds for CMA as **wp** $.CMA$ is defined to be the same as **wpp** $.CMA$. For a probabilistic statement,

$$
\begin{aligned}
& \mathbf{wp}\,.CPA.X \\
= \quad & \{\text{ Definition of } \mathbf{wp}\,.CPA\} \\
& (b \Rightarrow \langle \forall i :: \{\overline{x} := \overline{e.i}\}X\rangle) \wedge (\neg b \Rightarrow X) \\
\Rightarrow \quad & \{\text{Predicate Calculus}\} \\
& (b \Rightarrow \langle \exists i :: \{\overline{x} := \overline{e.i}\}X\rangle) \wedge (\neg b \Rightarrow X) \\
= \quad & \{\text{ Definition of } \mathbf{wpp}\,.CPA\} \\
& \mathbf{wpp}\,.CPA.X
\end{aligned}
$$

(End of Proof)

Theorem 13 *For all statements s,*

$$[\mathbf{wp}\,.s.X \wedge \mathbf{wpp}\,.s.Y \Rightarrow \mathbf{wpp}\,.s.(X \wedge Y)]$$

Proof (of 13): The theorem holds for CMA as **wpp** $.CMA$ is the same as **wp** $.CMA$ and **wp** $.CMA$ is universally conjunctive. For probabilistic statements,

$$
\begin{aligned}
& \mathbf{wp}\,.CPA.X \wedge \mathbf{wpp}\,.CPA.Y \\
= \quad & \{\text{Definition of } \mathbf{wp}\,.CPA \text{ and } \mathbf{wpp}\,.CPA\} \\
& ((b \Rightarrow \langle \forall i :: \{\overline{x} := \overline{e.i}\}X\rangle) \wedge (\neg b \Rightarrow X)) \wedge ((b \Rightarrow \langle \exists i :: \{\overline{x} := \overline{e.i}\}Y\rangle) \wedge (\neg b \Rightarrow Y)) \\
= \quad & \{\text{predicate calculus}\} \\
& (b \Rightarrow \langle \forall i :: \{\overline{x} := \overline{e.i}\}X\rangle \wedge \langle \exists i :: \{\overline{x} := \overline{e.i}\}Y\rangle) \wedge (\neg b \Rightarrow X \wedge Y) \\
= \quad & \{\wedge \text{ over } \exists\} \\
& (b \Rightarrow \langle \exists i :: \langle \forall i :: \{\overline{x} := \overline{e.i}\}X\rangle \wedge \{\overline{x} := \overline{e.i}\}Y\rangle) \wedge (\neg b \Rightarrow X \wedge Y) \\
\Rightarrow \quad & \{\text{Instantiation}\} \\
& (b \Rightarrow \langle \exists i :: \{\overline{x} := \overline{e.i}\}X \wedge \{\overline{x} := \overline{e.i}\}Y\rangle) \wedge (\neg b \Rightarrow X \wedge Y) \\
= \quad & \{\text{Universal conjunctivity of multiple assignment}\} \\
& (b \Rightarrow \langle \exists i :: \{\overline{x} := \overline{e.i}\}(X \wedge Y)\rangle) \wedge (\neg b \Rightarrow X \wedge Y) \\
= \quad & \{\text{Definition of } \mathbf{wpp}\,.CPA\} \\
& \mathbf{wpp}\,.CPA.(X \wedge Y)
\end{aligned}
$$

(End of Proof)

Theorem 14 *For all statements s,*

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

Proof (of 14): The theorem holds for CMA as $\mathbf{wpp}.CMA$ is the same as $\mathbf{wp}.CMA$ and $\mathbf{wp}.CMA$ is universally disjunctive. For probabilistic statements,

$$\begin{array}{cl}
& \mathbf{wp}.CPA.(X \vee Y) \\
= & \{\text{Definition of } \mathbf{wp}.CPA\} \\
& (b \Rightarrow \langle \forall i :: \{\overline{x} := \overline{e.i}\}(X \vee Y)\rangle) \wedge (\neg b \Rightarrow X \vee Y) \\
= & \{\text{Universal disjunctivity of multiple assignment}\} \\
& (b \Rightarrow \langle \forall i :: \{\overline{x} := \overline{e.i}\}X \vee \{\overline{x} := \overline{e.i}\}Y\rangle) \wedge (\neg b \Rightarrow X \vee Y) \\
\Rightarrow & \{\text{predicate calculus}\} \\
& (b \Rightarrow \langle \forall i :: \{\overline{x} := \overline{e.i}\}X\rangle \vee \langle \exists i :: \{\overline{x} := \overline{e.i}\}Y\rangle) \wedge (\neg b \Rightarrow X \vee Y) \\
= & \{\text{predicate calculus}\} \\
& ((b \Rightarrow \langle \forall i :: \{\overline{x} := \overline{e.i}\}X\rangle) \wedge (\neg b \Rightarrow X)) \vee ((b \Rightarrow \langle \exists i :: \{\overline{x} := \overline{e.i}\}Y\rangle) \wedge (\neg b \Rightarrow Y)) \\
= & \{\text{Definition of } \mathbf{wp}.CPA \text{ and } \mathbf{wpp}.CPA\} \\
& \mathbf{wp}.CPA.X \vee \mathbf{wpp}.CPA.Y
\end{array}$$

(End of Proof)

12.2 Properties of upto

Theorem 15 *The* **upto** *is a generalization of* **unless**.

$$(X \textbf{ unless } Y) \Rightarrow (X \textbf{ upto } Y)$$

Furthermore for a program consisting of only deterministic statements,

$$(X \textbf{ unless } Y) \equiv (X \textbf{ upto } Y)$$

Proof (of 15):

$$\begin{array}{cl}
& (X \textbf{ upto } Y) \\
= & \{\text{Definition of } \textbf{upto}\} \\
& \langle \forall s :: [X \wedge \neg Y \Rightarrow \mathbf{wp}.s.X \vee \mathbf{wpp}.s.Y]\rangle \\
\Leftarrow & \{\text{Theorem 14}\} \\
& \langle \forall s :: [X \wedge \neg Y \Rightarrow \mathbf{wp}.s.(X \vee Y)]\rangle \\
= & \{\text{Definition of } \textbf{unless}\} \\
& (X \textbf{ unless } Y)
\end{array}$$

For a program consisting only of deterministic statements,

$$\begin{array}{cl}
& (X \textbf{ upto } Y) \\
= & \{\text{Definition } \textbf{upto}\} \\
& \langle \forall s :: [X \wedge \neg Y \Rightarrow \mathbf{wp}.s.X \vee \mathbf{wpp}.s.Y]\rangle \\
= & \{\text{Definition of } \mathbf{wpp} \text{ for deterministic statements}\} \\
& \langle \forall s :: [X \wedge \neg Y \Rightarrow \mathbf{wp}.s.X \vee \mathbf{wp}.s.Y]\rangle \\
= & \{\mathbf{wp}.CMA \text{ is universally disjunctive}\} \\
& \langle \forall s :: [X \wedge \neg Y \Rightarrow \mathbf{wp}.s.(X \vee Y)]\rangle \\
= & \{\text{Definition of } \textbf{unless}\} \\
& (X \textbf{ unless } Y)
\end{array}$$

(End of Proof)

Theorems about upto

The properties of **unless** follow from its definition.

1. Reflexivity and Anti-Reflexivity

$$X \ \mathbf{upto}\ X$$

$$X \ \mathbf{upto}\ \neg X$$

$$\begin{array}{ll} & (X \ \mathbf{upto}\ X) \\ = & \{\text{Definition of } \mathbf{upto}\} \\ & \langle \forall s :: [X \wedge \neg X \Rightarrow \mathbf{wp}.s.X \vee \mathbf{wpp}.s.X] \rangle \\ = & \{\text{predicate calculus}\} \\ & \langle \forall s :: [\mathit{false} \Rightarrow \mathbf{wp}.s.X \vee \mathbf{wpp}.s.X] \rangle \\ = & \{\text{predicate calculus}\} \\ & \mathit{true} \end{array}$$

$$\begin{array}{ll} & (X \ \mathbf{upto}\ \neg X) \\ = & \{\text{Definition of } \mathbf{upto}\} \\ & \langle \forall s :: [X \wedge X \Rightarrow \mathbf{wp}.s.X \vee \mathbf{wpp}.s.(\neg X)] \rangle \\ = & \{\text{predicate calculus and Theorem 11}\} \\ & \langle \forall s :: [X \Rightarrow \mathbf{wpp}.s.X \vee \mathbf{wpp}.s.(\neg X)] \rangle \\ = & \{\text{By Theorem 10, } \mathbf{wpp}.s \text{ is universally disjunctive}\} \\ & \langle \forall s :: [X \Rightarrow \mathbf{wpp}.s.(X \vee \neg X)] \rangle \\ = & \{\text{predicate calculus and Corollary 2}\} \\ & \langle \forall s :: [X \Rightarrow \mathit{true})] \rangle \\ = & \{\text{predicate calculus}\} \\ & \mathit{true} \end{array}$$

2. Consequence Weakening

$$\frac{X \ \mathbf{upto}\ Y, Y \Rightarrow Z}{X \ \mathbf{upto}\ Z}$$

$$\begin{array}{ll} & (X \ \mathbf{upto}\ Y) \wedge (Y \Rightarrow Z) \\ = & \{\text{Definition of } \mathbf{upto}\} \\ & \langle \forall s :: [X \wedge \neg Y \Rightarrow \mathbf{wp}.s.X \vee \mathbf{wpp}.s.Y] \rangle \wedge (Y \Rightarrow Z) \\ \Rightarrow & \{\text{By Theorem 10 and Theorem 0, } \mathbf{wpp}.s \text{ is monotonic}\} \\ & \langle \forall s :: [X \wedge \neg Y \Rightarrow \mathbf{wp}.s.X \vee \mathbf{wpp}.s.Y] \rangle \wedge (\mathbf{wpp}.s.Y \Rightarrow \mathbf{wpp}.s.Z) \wedge (\neg Z \Rightarrow \neg Y) \\ \Rightarrow & \{\text{transitivity of } \Rightarrow\} \\ & \langle \forall s :: [X \wedge \neg Z \Rightarrow \mathbf{wp}.s.X \vee \mathbf{wpp}.s.Z] \rangle \\ = & \{\text{Definition of } \mathbf{upto}\} \\ & X \ \mathbf{upto}\ Z \end{array}$$

3. Partial Conjunction

$$\frac{\begin{array}{c} X \ \mathbf{upto}\ Y \\ X' \ \mathbf{upto}\ Y' \end{array}}{(X \wedge X') \ \mathbf{upto}\ ((X' \wedge Y) \vee Y')} \quad \text{, conjunction}$$

$$
\begin{array}{cl}
& (X \textbf{ upto } Y) \wedge (X' \textbf{ upto } Y') \\
= & \{\text{Definition of } \textbf{upto}\} \\
& \langle \forall s :: [X \wedge \neg Y \Rightarrow \textbf{wp}.s.X \vee \textbf{wpp}.s.Y] \rangle \wedge \langle \forall s :: [X' \wedge \neg Y' \Rightarrow \textbf{wp}.s.X' \vee \textbf{wpp}.s.Y'] \rangle \\
\Rightarrow & \{\text{predicate calculus}\} \\
& \langle \forall s :: [X \wedge X' \wedge \neg(Y \vee Y') \Rightarrow (\textbf{wp}.s.X \wedge \textbf{wp}.s.X') \vee (\textbf{wp}.s.X \wedge \textbf{wpp}.s.Y') \vee \\
& \qquad (\textbf{wp}.s.X' \wedge \textbf{wpp}.s.Y) \vee (\textbf{wpp}.s.Y \wedge \textbf{wpp}.s.Y')] \rangle \\
\Rightarrow & \{\text{Theorem 7 and Theorem 12 and weakening}\} \\
& \langle \forall s :: [X \wedge X' \wedge \neg(Y \vee Y') \Rightarrow \textbf{wp}.s.(X \wedge X') \vee \textbf{wpp}.s.(X' \wedge Y) \vee \textbf{wpp}.s.Y'] \rangle \\
= & \{\text{Theorem 11}\} \\
& \langle \forall s :: [X \wedge X' \wedge \neg(Y \vee Y') \Rightarrow \textbf{wp}.s.(X \wedge X') \vee \textbf{wpp}.s.((X' \wedge Y) \vee Y')] \rangle \\
= & \{\text{predicate calculus}\} \\
& \langle \forall s :: [X \wedge X' \wedge \neg((X' \wedge Y) \vee Y') \Rightarrow \textbf{wp}.s.(X \wedge X') \vee \textbf{wpp}.s.((X' \wedge Y) \vee Y')] \rangle \\
= & \{\text{Definition of } \textbf{upto}\} \\
& (X \wedge X') \textbf{ upto } ((X' \wedge Y) \vee Y')
\end{array}
$$

4. Simple Conjunction and Simple Disjunction

$$
\frac{\begin{array}{c} X \textbf{ upto } Y \\ X' \textbf{ upto } Y' \end{array}}{\begin{array}{ll} (X \wedge X') \textbf{ upto } (Y \vee Y') & \text{, simple conjunction} \\ (X \vee X') \textbf{ upto } (Y \vee Y') & \text{, simple disjunction} \end{array}}
$$

$$
\begin{array}{cl}
& (X \textbf{ upto } Y) \wedge (X' \textbf{ upto } Y') \\
\Rightarrow & \{\text{Conjunction}\} \\
& (X \wedge X') \textbf{ upto } ((X' \wedge Y) \vee Y') \\
\Rightarrow & \{\text{Consequence Weakening}\} \\
& (X \wedge X') \textbf{ upto } (Y \vee Y')
\end{array}
$$

$$
\begin{array}{cl}
& (X \textbf{ upto } Y) \wedge (X' \textbf{ upto } Y') \\
= & \{\text{Definition of } \textbf{upto}\} \\
& \langle \forall s :: [X \wedge \neg Y \Rightarrow \textbf{wp}.s.X \vee \textbf{wpp}.s.Y] \rangle \wedge \langle \forall s :: [X' \wedge \neg Y' \Rightarrow \textbf{wp}.s.X' \vee \textbf{wpp}.s.Y'] \rangle \\
\Rightarrow & \{\text{By Theorem 7 and Theorem 0 } \textbf{wp}.s \text{ is monotonic; By Theorem 11 and Theorem 0 } \textbf{wpp}.s \text{ is monotonic}\} \\
& \langle \forall s :: [(X \wedge \neg Y) \vee (X' \wedge \neg Y') \Rightarrow \textbf{wp}.s.(X \vee X') \vee \textbf{wpp}.s.(Y \vee Y')] \rangle \\
\Rightarrow & \{\text{predicate calculus}\} \\
& \langle \forall s :: [(X \vee X') \wedge \neg(Y \vee Y') \Rightarrow \textbf{wp}.s.(X \vee X') \vee \textbf{wpp}.s.(Y \vee Y')] \rangle \\
= & \{\text{Definition of } \textbf{upto}\} \\
& (X \vee X') \textbf{ upto } (Y \vee Y')
\end{array}
$$

5. Conjunction with **unless**

$$
\frac{\begin{array}{c} X \textbf{ unless } Y \\ X' \textbf{ upto } Y' \end{array}}{(X \wedge X') \textbf{ upto } (X \wedge Y') \vee (X' \wedge Y) \vee (Y \wedge Y')}
$$

$$
\begin{array}{cl}
& (X \textbf{ unless } Y) \wedge (X' \textbf{ upto } Y') \\
= & \{\text{Definition of } \textbf{unless} \text{ and } \textbf{upto}\} \\
& \langle \forall s :: [X \wedge \neg Y \Rightarrow \textbf{wp}.s.(X \vee Y)] \rangle \wedge \langle \forall s :: [X' \wedge \neg Y' \Rightarrow \textbf{wp}.s.X' \vee \textbf{wpp}.s.Y')] \rangle \\
\Rightarrow & \{\text{predicate calculus}\} \\
& \langle \forall s :: [(X \wedge X' \wedge \neg Y \wedge \neg Y') \Rightarrow \textbf{wp}.s.(X \vee Y) \wedge (\textbf{wp}.s.X' \vee \textbf{wpp}.s.Y')] \rangle \\
\Rightarrow & \{\text{predicate calculus and Theorem 13}\} \\
& \langle \forall s :: [(X \wedge X' \wedge \neg Y \wedge \neg Y') \Rightarrow \textbf{wp}.s.(X \wedge X') \vee \textbf{wpp}.s.((X' \wedge Y) \vee (X \wedge Y') \vee (Y \wedge Y'))] \rangle \\
= & \{\text{predicate calculus and Definition of } \textbf{upto}\} \\
& (X \wedge X') \textbf{ upto } (X \wedge Y') \vee (X' \wedge Y) \vee (Y \wedge Y')
\end{array}
$$

Corollaries

The following corollaries hold for **upto**. These are all proved using the basic properties proved above. They do not depend on the definition of **upto**.

1. Implication

$$\frac{X \Rightarrow Y}{X \textbf{ upto } Y}$$

0. $X \textbf{ upto } X$
   ,Reflexivity
1. $X \Rightarrow Y$
   ,Given
2. $X \textbf{ upto } Y$
   ,Consequence Weakening using 0 and 1

2.

$$\frac{\neg X \Rightarrow Y}{X \textbf{ upto } Y}$$

0. $X \textbf{ upto } \neg X$
   ,Anti-Reflexivity
1. $\neg X \Rightarrow Y$
   ,Given
2. $X \textbf{ upto } Y$
   ,Consequence Weakening using 0 and 1

3.

$$\frac{X \textbf{ upto } (Y \vee Z)}{(X \wedge \neg Y) \textbf{ upto } (Y \vee Z)}$$

0. $X \textbf{ upto } (Y \vee Z)$
   ,Given
1. $\neg Y \textbf{ upto } Y$
   ,Anti-Reflexivity
2. $(X \wedge \neg Y) \textbf{ upto } (Y \vee Z)$
   ,Simple Conjunction on 0 and 1

4.

$$\frac{(X \wedge \neg Y) \textbf{ upto } (Y \vee Z)}{X \textbf{ upto } (Y \vee Z)}$$

0. $(X \wedge Y) \textbf{ upto } Y$
   ,Corollary 1 given above
1. $(X \wedge \neg Y) \textbf{ upto } (Y \vee Z)$
   ,Given
2. $X \textbf{ upto } (Y \vee Z)$
   ,Simple Disjunction on 0 and 1

5.

$$\frac{(X \vee Y) \textbf{ upto } Z}{X \textbf{ upto } (Y \vee Z)}$$

0. $(X \vee Y)$ **upto** Z
   ,Given
1. $\neg Y$ **upto** Y
   ,Anti-Reflexivity
2. $(X \wedge \neg Y)$ **upto** $(Y \vee Z)$
   ,Simple Conjunction on 0 and 1
3. X **upto** $(Y \vee Z)$
   ,Corollary 4 given above

6.

$$X \textbf{ upto } true$$

0. X **upto** X
   ,Reflexivity
1. $X \Rightarrow true$
   ,predicate calculus
2. X **upto** $true$
   ,Consequence weakening using 0 and 1

7.

$$true \textbf{ upto } X$$

0. X **upto** X
   ,Reflexivity
1. $\neg X$ **upto** X
   ,Anti-Reflexivity
2. $true$ **upto** X
   ,Simple Disjunction on 0 and 1

8.

$$false \textbf{ upto } X$$

0. X **upto** X
   ,Reflexivity
1. $\neg X$ **upto** X
   ,Anti-Reflexivity
2. $false$ **upto** X
   ,Simple Conjunction on 0 and 1

12.3 Properties of entails

Theorem 16 *The* **entails** *is a generalization of* **ensures**

$$(X \textbf{ ensures } Y) \Rightarrow (X \textbf{ entails } Y)$$

Furthermore for a program consisting only of deterministic statements,

$$(X \textbf{ ensures } Y) \equiv (X \textbf{ entails } Y)$$

Proof (of 16):

$(X \textbf{ entails } Y)$
$=$ {Definition of **entails**}
$(X \textbf{ upto } Y) \wedge \langle \exists s :: [X \wedge \neg Y \Rightarrow \textbf{wpp}.s.Y] \rangle$
$\Leftarrow$ {**upto** include **unless**; Theorem 12}
$(X \textbf{ unless } Y) \wedge \langle \exists s :: [X \wedge \neg Y \Rightarrow \textbf{wp}.s.Y] \rangle$
$=$ {Definition of **ensures**}
$(X \textbf{ ensures } Y)$

For a program consisting only of deterministic statements,

$$
\begin{array}{cl}
 & (X \textbf{ entails } Y) \\
= & \{\text{Definition of } \textbf{entails}\} \\
 & (X \textbf{ upto } Y) \wedge \langle \exists s :: [X \wedge \neg Y \Rightarrow \textbf{wpp}.s.Y] \rangle \\
= & \{\text{Definition of } \textbf{upto} \text{ and } \textbf{wp} \text{ for deterministic statements}\} \\
 & (X \textbf{ unless } Y) \wedge \langle \exists s :: [X \wedge \neg Y \Rightarrow \textbf{wp}.s.Y] \rangle \\
= & \{\text{Definition of } \textbf{ensures}\} \\
 & (X \textbf{ ensures } Y)
\end{array}
$$

(End of Proof)

Theorems of entails

The properties of **entails** follow from its definition.

1. Reflexivity

$$X \textbf{ entails } X$$

$$
\begin{array}{cl}
 & X \textbf{ entails } X \\
= & \{\text{Definition of } \textbf{entails}\} \\
 & (X \textbf{ upto } X) \wedge \langle \exists s :: [X \wedge \neg X \Rightarrow \textbf{wpp}.s.X] \rangle \\
\Leftarrow & \{\text{Reflexivity of } \textbf{upto} \text{ and predicate calculus}\} \\
 & \langle \exists s :: [false \Rightarrow \textbf{wpp}.s.X] \rangle \\
= & \{\text{predicate calculus}\} \\
 & true
\end{array}
$$

2. Consequence Weakening

$$\frac{X \textbf{ entails } Y, Y \Rightarrow Z}{X \textbf{ entails } Z}$$

$$
\begin{array}{cl}
 & (X \textbf{ entails } Y) \wedge (Y \Rightarrow Z) \\
= & \{\text{Definition of } \textbf{entails}\} \\
 & (X \textbf{ upto } Y) \wedge \langle \exists s :: [X \wedge \neg Y \Rightarrow \textbf{wpp}.s.Y] \rangle \wedge (Y \Rightarrow Z) \\
\Rightarrow & \{\text{Consequence Weakening for } \textbf{upto}\text{; predicate calculus; Monotonicity of } \textbf{wpp}.s\} \\
 & (X \textbf{ upto } Z) \wedge \langle \exists s :: [X \wedge \neg Z \Rightarrow \textbf{wpp}.s.Y] \rangle \wedge (\textbf{wpp}.s.Y \Rightarrow \textbf{wpp}.s.Z) \\
\Rightarrow & \{\text{transitivity of implication}\} \\
 & (X \textbf{ upto } Z) \wedge \langle \exists s :: [X \wedge \neg Z \Rightarrow \textbf{wpp}.s.Z] \rangle \\
= & \{\text{Definition of } \textbf{entails}\} \\
 & (X \textbf{ entails } Z)
\end{array}
$$

3. Impossibility

$$\frac{X \textbf{ entails } false}{\neg X}$$

$$
\begin{array}{cl}
 & (X \textbf{ entails } false) \\
= & \{\text{Definition of } \textbf{entails}\} \\
 & (X \textbf{ upto } false) \wedge \langle \exists s :: [X \wedge \neg false \Rightarrow \textbf{wpp}.s.false] \rangle \\
\Rightarrow & \{\text{Corollary 3}\} \\
 & \langle \exists s :: [X \Rightarrow false] \rangle \\
= & \{\text{predicate calculus}\} \\
 & \neg X
\end{array}
$$

4. Conjunction with **unless**

$$\frac{\begin{array}{c} X \textbf{ entails } Y \\ X' \textbf{ unless } Y' \end{array}}{(X \wedge X') \textbf{ entails } (X \wedge Y') \vee (X' \wedge Y) \vee (Y \wedge Y')}$$

$$\begin{array}{cl}
 & (X \textbf{ entails } Y) \wedge (X' \textbf{ unless } Y') \\
= & \{\text{Definition of } \textbf{entails}\} \\
 & (X \textbf{ upto } Y) \wedge \langle \exists s :: [X \wedge \neg Y \Rightarrow \textbf{wpp}.s.Y] \rangle \wedge (X' \textbf{ unless } Y') \\
= & \{\text{Conjunction with } \textbf{unless} \text{ for } \textbf{upto}\text{; Definition of } \textbf{upto}\} \\
 & ((X \wedge X') \textbf{ upto } (X \wedge Y') \vee (X' \wedge Y) \vee (Y \wedge Y')) \wedge \\
 & \langle \exists s :: [(X \wedge X' \wedge \neg Y \wedge \neg Y') \Rightarrow (\textbf{wpp}.s.Y \wedge \textbf{wp}.s.(X' \vee Y'))] \rangle \\
= & \{\text{Theorem 13}\} \\
 & ((X \wedge X') \textbf{ upto } (X \wedge Y') \vee (X' \wedge Y) \vee (Y \wedge Y')) \wedge \\
 & \langle \exists s :: [(X \wedge X' \wedge \neg Y \wedge \neg Y') \Rightarrow \textbf{wpp}.s.((X' \wedge Y) \vee (Y' \wedge Y))] \rangle \\
= & \{\text{Weakening the consequence of the second conjunct}\} \\
 & ((X \wedge X') \textbf{ upto } (X \wedge Y') \vee (X' \wedge Y) \vee (Y \wedge Y')) \wedge \\
 & \langle \exists s :: [(X \wedge X' \wedge \neg Y \wedge \neg Y') \Rightarrow \textbf{wpp}.s.((X' \wedge Y) \vee (Y' \wedge Y) \vee (X \wedge Y'))] \rangle \\
= & \{\text{Definition of } \textbf{entails}\} \\
 & ((X \wedge X') \textbf{ entails } (X \wedge Y') \vee (X' \wedge Y) \vee (Y \wedge Y'))
\end{array}$$

5. Conjunction with **upto**

$$\frac{\begin{array}{c} X \textbf{ entails } Y \\ X' \textbf{ upto } Y' \end{array}}{(X \wedge X') \textbf{ entails } (X' \wedge Y) \vee Y'}$$

$$\begin{array}{cl}
 & (X \textbf{ entails } Y) \wedge (X' \textbf{ upto } Y') \\
= & \{\text{Definition of } \textbf{entails} \text{ and } \textbf{upto}\} \\
 & (X \textbf{ upto } Y) \wedge (X' \textbf{ upto } Y') \wedge \langle \exists s :: [X \wedge \neg Y \Rightarrow \textbf{wpp}.s.Y] \rangle \\
= & \{\text{Conjunction for } \textbf{upto}\text{; Definition of } \textbf{upto}\} \\
 & ((X \wedge X') \textbf{ upto } ((X' \wedge Y) \vee Y)) \wedge \\
 & \langle \exists s :: [X \wedge X' \wedge \neg Y \wedge \neg Y' \Rightarrow (\textbf{wp}.s.X' \vee \textbf{wpp}.s.Y') \wedge \textbf{wpp}.s.Y] \rangle \\
= & \{\text{predicate calculus}\} \\
 & ((X \wedge X') \textbf{ upto } ((X' \wedge Y) \vee Y)) \wedge \\
 & \langle \exists s :: [X \wedge X' \wedge \neg Y \wedge \neg Y' \Rightarrow (\textbf{wp}.s.X' \wedge \textbf{wpp}.s.Y) \vee (\textbf{wpp}.s.Y' \wedge \textbf{wpp}.s.Y)] \rangle \\
\Rightarrow & \{\text{Weaken consequence of second conjunct using Theorem 13}\} \\
 & ((X \wedge X') \textbf{ upto } ((X' \wedge Y) \vee Y)) \wedge \\
 & \langle \exists s :: [X \wedge X' \wedge \neg Y \wedge \neg Y' \Rightarrow (\textbf{wpp}.s.(X' \wedge Y) \vee \textbf{wpp}.s.Y')] \rangle \\
= & \{\text{Universal disjunctivity of } \textbf{wpp}.s\} \\
 & ((X \wedge X') \textbf{ upto } ((X' \wedge Y) \vee Y)) \wedge \\
 & \langle \exists s :: [X \wedge X' \wedge \neg Y \wedge \neg Y' \Rightarrow \textbf{wpp}.s.((X' \wedge Y) \vee Y')] \rangle \\
= & \{\text{Definition of } \textbf{entails}\} \\
 & ((X \wedge X') \textbf{ upto } ((X' \wedge Y) \vee Y))
\end{array}$$

6. Disjunction

$$\frac{X \textbf{ entails } Y}{(X \vee Z) \textbf{ entails } (Y \vee Z)}$$

$$
\begin{array}{ll}
& X \textbf{ entails } Y \\
= & \{\text{Definition of } \textbf{entails}\} \\
& (X \textbf{ upto } Y) \wedge \langle \exists s :: [X \wedge \neg Y \Rightarrow \textbf{wpp}.s.Y] \rangle \\
= & \{\text{Reflexivity of } \textbf{upto}\} \\
& (X \textbf{ upto } Y) \wedge (Z \textbf{ upto } Z) \wedge \langle \exists s :: [X \wedge \neg Y \Rightarrow \textbf{wpp}.s.Y] \rangle \\
\Rightarrow & \{\text{Simple Disjunction for } \textbf{upto}\} \\
& (X \vee Z) \textbf{ upto } (Y \vee Z) \wedge \langle \exists s :: [X \wedge \neg Y \Rightarrow \textbf{wpp}.s.Y] \rangle \\
\Rightarrow & \{\text{predicate calculus; Monotonicity of } \textbf{wpp}.s\} \\
& ((X \vee Z) \textbf{ upto } (Y \vee Z)) \wedge \langle \exists s :: [(X \vee Z) \wedge \neg (Y \vee Z) \Rightarrow \textbf{wpp}.s.(Y \vee Z)] \rangle \\
= & \{\text{Definition of } \textbf{entails}\} \\
& (X \vee Z) \textbf{ entails } (Y \vee Z)
\end{array}
$$

Corollaries

The following corollaries hold for **entails**. They follow from the basic properties defined above.

1. Implication

$$\frac{X \Rightarrow Y}{X \textbf{ entails } Y}$$

0. X **entails** X
   ,Reflexivity of **entails**
1. $X \Rightarrow Y$
   ,Given
2. X **entails** Y
   ,Consequence Weakening on 0 and 1

2.

$$\frac{X \textbf{ entails } (Y \vee Z)}{(X \wedge \neg Y) \textbf{ entails } (Y \vee Z)}$$

0. X **entails** $(Y \vee Z)$
   ,Given
1. $\neg Y$ **upto** Y
   ,Anti-Reflexivity of **upto**
2. $(X \wedge \neg Y)$ **entails** $((\neg Y \wedge Z) \vee Y)$
   ,Conjunction with **upto** for **entails** using 0 and 1
3. $(X \wedge \neg Y)$ **entails** $(Y \vee Z)$
   ,predicate calculus on 2

3.

$$\frac{(X \vee Y) \textbf{ entails } Z}{X \textbf{ entails } (Y \vee Z)}$$

0. $(X \vee Y)$ **entails** Z
   ,Given
1. $(X \wedge \neg Y)$ **entails** $(Y \vee Z)$
   ,Corollary 2 given above
2. X **entails** $(Y \vee Z)$
   ,Disjunction on 3 using $Z := X \wedge Y$ and predicate calculus

12.4 Properties of $\leadsto$

1. Implication

$$\frac{X \Rightarrow Y}{X \leadsto Y}$$

0. $X \Rightarrow Y$
   ,Given
1. X **entails** Y
   ,Corollary 1 of **entails**
2. $X \leadsto Y$
   ,Definition of $\leadsto$

2. Impossibility

$$\frac{X \leadsto false}{\neg X}$$

The proof is by induction on the definition of $\leadsto$. The base case is proved by the impossibility property of **entails**. The induction step is proved by

0. $(X \leadsto Y) \wedge (Y \leadsto false)$
   ,Given
1. $(X \leadsto Y) \wedge \neg Y$
   ,Induction hypothesis
2. $(X \leadsto false)$
   ,From 1
3. $\neg X$
   ,Induction hypothesis

3. Disjunction

$$(X \leadsto Y) \Rightarrow (X \vee Z \leadsto Y \vee Z)$$

The proof is by induction on $\leadsto$. The base case follows from the disjunction property of **entails**. The induction step is as follows.

0. $(X \leadsto U) \wedge (U \leadsto Y)$
   ,Given
1. $(X \vee W \leadsto U \vee W) \wedge (U \vee W \leadsto Y \vee W)$
   ,Induction hypothesis, twice
2. $(X \vee W \leadsto Y \vee W)$
   ,Transitivity of $\leadsto$

4. Finite Disjunction

$$\frac{(X \leadsto Z), (Y \leadsto Z)}{((X \vee Y) \leadsto Z)}$$

0. $(X \leadsto Z) \wedge (Y \leadsto Z)$
   ,Given
1. $(X \vee Y \leadsto Z \vee Y) \wedge (Y \vee Z \leadsto Z \vee Z)$
   ,Disjunction, twice
2. $(X \vee Y \leadsto Z)$
   ,Transitivity of $\leadsto$

5. Cancellation

$$\frac{U \leadsto V \vee W,\ W \leadsto X}{U \leadsto V \vee X}$$

0. $(U \leadsto V \vee W) \wedge (W \leadsto X)$
,Given
1. $(U \leadsto V \vee W) \wedge (V \vee W \leadsto V \vee X)$
,Disjunction on second property of 0
2. $(U \leadsto V \vee X)$
,Transitivity of $\leadsto$

6. PSP (Progress-Safety-Progress)

$$\frac{X \leadsto Y,\ U \textbf{ unless } V}{(X \wedge U) \leadsto (Y \wedge U) \vee V}$$

The proof is by induction on the definition of $\leadsto$. The base case follows from the conjunction with **unless** rule of **entails** and consequence weakening. The induction step is as follows.

0. $(X \leadsto Z) \wedge (Z \leadsto Y) \wedge (U \textbf{ unless } V)$
,Given
1. $((X \wedge U) \leadsto (U \wedge Z) \vee V) \wedge$
$((Z \wedge U) \leadsto (U \wedge Y) \vee V)$
,Applying induction hypothesis twice
2. $(X \wedge U) \leadsto (Y \wedge U) \vee V$
,Cancellation on 1

7. Completion Theorem (Proof Omitted)

12.5 Properties of $\models\!\!\Rightarrow$

Theorem 17 $(X \mapsto Y) \Rightarrow (X \models\!\!\Rightarrow Y)$

Proof (of 17): The proof is by an induction on the definition of $\mapsto$.
Base Case :

$(X \textbf{ ensures } Y)$
$\Rightarrow$ {Definition of **ensures**}
$(X \textbf{ unless } Y) \wedge (X \textbf{ entails } Y)$
$\Rightarrow$ {Definition of $\models\!\!\Rightarrow$}
$(X \models\!\!\Rightarrow Y)$

Induction Step (transitivity) :

$(X \mapsto Y) \wedge (Y \mapsto Z)$
$\Rightarrow$ {Induction hypothesis, twice}
$(X \models\!\!\Rightarrow Y) \wedge (Y \models\!\!\Rightarrow Z)$
$\Rightarrow$ {Transitivity of $\models\!\!\Rightarrow$}
$(X \models\!\!\Rightarrow Z)$

Induction Step (disjunction) :

$\langle \forall X :: X \mapsto Y \rangle$
$\Rightarrow$ {Induction hypothesis}
$\langle \forall X :: X \models\!\!\Rightarrow Y \rangle$
$\Rightarrow$ {Disjunctivity of $\models\!\!\Rightarrow$}
$(\langle \exists X :: X \rangle \models\!\!\Rightarrow Z)$

(End of Proof)

The probabilistic leads-to ($\models\!\!\Rightarrow$) enjoys all the properties of $\mapsto$.

1. Implication

$$\frac{X \Rightarrow Y}{X \models\Rightarrow Y}$$

0. $X \Rightarrow Y$
   ,Given
1. $X \mapsto Y$
   ,Property of $\mapsto$
2. $X \models\Rightarrow Y$
   ,1 and Theorem 17

2. Impossibility

$$\frac{X \models\Rightarrow \mathit{false}}{\neg X}$$

The proof is by induction on the definition of $\models\Rightarrow$.
Base Case :

0. $(X$ **unless** $\mathit{false}) \wedge (X$ **entails** $\mathit{false})$
   ,Given
1. $\neg X$
   ,Impossibility property of **entails**

Induction Step (transitivity) :

0. $(X \models\Rightarrow Y) \wedge (Y \models\Rightarrow \mathit{false})$
   ,Given
1. $(X \models\Rightarrow Y) \wedge \neg Y$
   ,Induction hypothesis
2. $(X \models\Rightarrow \mathit{false})$
   ,From 1
3. $\neg X$
   ,Induction hypothesis

Induction Step (disjunctivity) :

0. $\langle \forall X :: X \models\Rightarrow \mathit{false}\rangle$
   ,Given
1. $\langle \forall X :: \neg X\rangle$
   ,Induction hypothesis
2. $\neg\langle \exists X :: X\rangle$
   ,predicate calculus

3. General Disjunction

$$\frac{\langle \forall m : m \in W : X.m \models\Rightarrow Y.m\rangle}{\langle \exists m : m \in W : X.m\rangle \models\Rightarrow \langle \exists m : m \in W : Y.m\rangle}$$

0. $Y.m \Rightarrow \langle \exists m :: Y.m\rangle$
   ,predicate calculus
1. $\langle \forall m :: X.m \models\Rightarrow Y.m\rangle$
   ,Given
2. $\langle \forall m :: X.m \models\Rightarrow \langle \exists m :: Y.m\rangle\rangle$
   ,transitivity of $\models\Rightarrow$
3. $\langle \exists m :: X.m\rangle \models\Rightarrow \langle \exists m :: Y.m\rangle\rangle$
   ,disjunctivity of $\models\Rightarrow$

4. Cancellation

$$\frac{U \models\Rightarrow V \vee W,\ W \models\Rightarrow X}{U \models\Rightarrow V \vee X}$$

0. $(U \models\Rightarrow V \vee W) \wedge (W \models\Rightarrow X)$
   ,Given
1. $(U \models\Rightarrow V \vee W) \wedge (V \vee W \models\Rightarrow V \vee X)$
   ,Disjunction on second conjunct of 0
2. $(U \models\Rightarrow V \vee X)$
   ,Transitivity of $\models\Rightarrow$

5. PSP (Progress-Safety-Progress)

$$\frac{X \models\Rightarrow Y,\ U \textbf{ unless } V}{(X \wedge U) \models\Rightarrow (Y \wedge U) \vee V}$$

The proof is by induction on the definition of $\models\Rightarrow$.
Base case :

0. $(X \textbf{ unless } Y) \wedge (X \textbf{ entails } Y) \wedge (U \textbf{ unless } V)$
   ,Given
1. $(X \wedge U \textbf{ unless } (Y \wedge U) \vee V) \wedge ((X \wedge U) \textbf{ entails } (Y \wedge U) \vee V)$
   ,Conjunction of the two **unless** properties; Conjunction with **unless** for **entails**
2. $(X \wedge U \models\Rightarrow (Y \wedge U) \vee V)$
   ,Definition of $\models\Rightarrow$

Induction Step (transitivity) :

0. $(X \models\Rightarrow Y) \wedge (Y \models\Rightarrow Z) \wedge (U \textbf{ unless } V)$
   ,Given
1. $((X \wedge U) \models\Rightarrow (Y \wedge U) \vee V) \wedge ((Y \wedge U) \models\Rightarrow (Z \wedge U) \vee V)$
   ,Induction hypothesis, twice
2. $((X \wedge U) \models\Rightarrow (Z \wedge U) \vee V)$
   ,Cancellation

Induction Step (disjunctivity) :

0. $\langle \forall X :: X \models\Rightarrow Y \rangle \wedge (U \textbf{ unless } V)$
   ,Given
1. $\langle \forall X :: (X \wedge U) \models\Rightarrow (Y \wedge U) \vee V \rangle$
   ,Induction hypothesis
2. $\langle \exists X :: X \wedge U \rangle \models\Rightarrow (Y \wedge U) \vee V$
   ,Disjunctivity of $\models\Rightarrow$
3. $\langle \exists X :: X \rangle \wedge U \models\Rightarrow (Y \wedge U) \vee V$
   ,predicate calculus

6. Completion (Proof omitted)

12.6 On program composition

Theorem 18 Union Theorem

- $(X \textbf{ unless } Y \textbf{ in } F \wedge X \textbf{ unless } Y \textbf{ in } G) \equiv (X \textbf{ unless } Y \textbf{ in } F \,[\!]\, G)$ The proof is exactly as in [CM88].
- $(X \textbf{ ensures } Y \textbf{ in } F \wedge X \textbf{ unless } Y \textbf{ in } G) \vee (X \textbf{ unless } Y \textbf{ in } F \wedge X \textbf{ ensures } Y \textbf{ in } G) \equiv (X \textbf{ ensures } Y \textbf{ in } F \,[\!]\, G)$ The proof is exactly as in [CM88].

- $(X \textbf{ upto } Y \textbf{ in } F \wedge X \textbf{ upto } Y \textbf{ in } G) \equiv (X \textbf{ upto } Y \textbf{ in } F \ [] \ G)$

$$
\begin{array}{ll}
 & (X \textbf{ upto } Y \textbf{ in } F \wedge X \textbf{ upto } Y \textbf{ in } G) \\
= & \{\text{Definition of } \textbf{upto} \text{ twice}\} \\
 & \langle \forall s : s \textbf{ in } F : [X \wedge \neg Y \Rightarrow \textbf{wp}.s.X \vee \textbf{wpp}.s.Y] \rangle \wedge \\
 & \langle \forall s : s \textbf{ in } G : [X \wedge \neg Y \Rightarrow \textbf{wp}.s.X \vee \textbf{wpp}.s.Y] \rangle \\
= & \{\text{predicate calculus}\} \\
 & \langle \forall s : s \textbf{ in } F \vee s \textbf{ in } G : [X \wedge \neg Y \Rightarrow \textbf{wp}.s.X \vee \textbf{wpp}.s.Y] \rangle \\
= & \{\text{Definition of } []\} \\
 & \langle \forall s : s \textbf{ in } F \ [] \ G : [X \wedge \neg Y \Rightarrow \textbf{wp}.s.X \vee \textbf{wpp}.s.Y] \rangle \\
= & \{\text{Definition of } \textbf{upto}\} \\
 & X \textbf{ upto } Y \textbf{ in } F \ [] \ G
\end{array}
$$

- $(X \textbf{ entails } Y \textbf{ in } F \wedge X \textbf{ upto } Y \textbf{ in } G) \vee (X \textbf{ upto } Y \textbf{ in } F \wedge X \textbf{ entails } Y \textbf{ in } G) \equiv (X \textbf{ entails } Y \textbf{ in } F \ [] \ G)$

$$
\begin{array}{ll}
 & X \textbf{ entails } Y \textbf{ in } F \ [] \ G \\
= & \{\text{Definition of } \textbf{entails}\} \\
 & (X \textbf{ upto } Y \textbf{ in } F \ [] \ G) \wedge \langle \exists s : s \textbf{ in } F \ [] \ G : [X \wedge \neg Y \Rightarrow \textbf{wpp}.s.Y] \rangle \\
= & \{\text{Union theorem for } \textbf{upto}\} \\
 & (X \textbf{ upto } Y \textbf{ in } F) \wedge (X \textbf{ upto } Y \textbf{ in } G) \wedge \langle \exists s : s \textbf{ in } F \ [] \ G : [X \wedge \neg Y \Rightarrow \textbf{wpp}.s.Y] \rangle \\
= & \{\text{predicate calculus}\} \\
 & (X \textbf{ upto } Y \textbf{ in } F) \wedge (X \textbf{ upto } Y \textbf{ in } G) \wedge \\
 & (\langle \exists s : s \textbf{ in } F : [X \wedge \neg Y \Rightarrow \textbf{wpp}.s.Y] \rangle \vee \langle \exists s : s \textbf{ in } G : [X \wedge \neg Y \Rightarrow \textbf{wpp}.s.Y] \rangle) \\
= & \{\text{predicate calculus and definition of } \textbf{entails}\} \\
 & (X \textbf{ entails } Y \textbf{ in } F \wedge X \textbf{ upto } Y \textbf{ in } G) \vee (X \textbf{ upto } Y \textbf{ in } F \wedge X \textbf{ entails } Y \textbf{ in } G)
\end{array}
$$

- $(X \textbf{ entails } Y \textbf{ in } F \wedge X \textbf{ unless } Y \textbf{ in } G) \vee (X \textbf{ unless } Y \textbf{ in } F \wedge X \textbf{ entails } Y \textbf{ in } G) \Rightarrow (X \textbf{ entails } Y \textbf{ in } F \ [] \ G)$ The proof follows from the union theorem for **entails** and **upto** by strengthening the right side using Theorem 15.

Corollaries

1. $X \textbf{ stable in } F \ [] \ G \equiv (X \textbf{ stable in } F \wedge X \textbf{ stable in } G)$
2. $$\frac{X \textbf{ unless } Y \textbf{ in } F,\ X \textbf{ stable in } G}{X \textbf{ unless } Y \textbf{ in } F \ [] \ G}$$
3. $$\frac{X \textbf{ invariant in } F,\ X \textbf{ stable in } G}{X \textbf{ invariant in } F \ [] \ G}$$
4. $$\frac{X \textbf{ ensures } Y \textbf{ in } F,\ X \textbf{ stable in } G}{X \textbf{ ensures } Y \textbf{ in } F \ [] \ G}$$
5. $$\frac{X \textbf{ upto } Y \textbf{ in } F,\ X \textbf{ stable in } G}{X \textbf{ upto } Y \textbf{ in } F \ [] \ G}$$
6. $$\frac{X \textbf{ entails } Y \textbf{ in } F,\ X \textbf{ stable in } G}{X \textbf{ entails } Y \textbf{ in } F \ [] \ G}$$