Reasoning About Conditional Progress Properties

Ken Calvert
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332–0280

GIT–CC–94/03

March 1994

Abstract

In some otherwise attractive formalisms, it can be difficult or even impossible to specify progress in such a way that a component of a distributed system can be proved correct independent of its environment. This problem arises because the nested dependencies between the component and its environment cannot be conveniently expressed in the formalism. A typical example is a communication protocol, which is supposed to provide reliable data transfer even over channels that are unboundedly lossy: the channels only deliver messages if the protocol transmits them often enough, while the protocol only guarantees reliable service if the channels deliver sufficiently many messages. This paper investigates the extent to which such progress specifications can be dealt with using predicate calculus and a single temporal operator (*leads-to*) having a simple proof theory. It turns out that under the proper semantic interpretation, many progress specifications expressing complex dependences can be represented using certain boolean combinations of leads-to properties. By adding two simple inference rules to an existing proof theory, we obtain a (relatively) complete theory for a large class of conditional progress properties, without the complexity of the full temporal logic; such a theory can be used with various compositional specification formalisms. Based on the results, an approach to specification of protocol progress is outlined and illustrated with an example.

College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332–0280

1 Introduction

Temporal logic is a widely-studied tool for reasoning about concurrent and reactive systems. Powerful theories have been developed, which allow very general properties of such systems to be specified and verified [14]. As a practical matter, however, the more expressive the specification language, the more complex is the logical machinery required to prove that a program satisfies a specification. Theories like UNITY [7], which feature a small, elegant proof system, have proven quite useful for reasoning about all kinds of distributed algorithms and programs [11, 12, 13, 18]. However, when it comes to specifying certain kinds of progress properties, these systems are sometimes inadequate because they trade expressive power for a simpler proof mechanism.

The *progress* specification of a distributed program component defines what it may be relied upon to do. For example, consider a protocol implementing a reliable message-stream service over unreliable channels. The typical progress specification for such a protocol requires that every message sent be eventually received. However, that specification cannot be satisfied by any protocol alone: the channels must not be "broken," i.e. they must not continually lose data. Thus, the requirement that every message sent be received constrains *both* the protocol and the underlying channels.

It is desirable to separate the protocol's progress specification from that of the channels, so that the protocol can be designed and implemented separately. One way to accomplish this is to *condition* the protocol's progress on channel progress: if the channels satisfy X, the protocol satisfies Y, where X is the underlying channels' progress specification, and Y says that every message sent is delivered. However, in many cases X will itself be in the form of a conditional. For example, a common specification of an unboundedly lossy channel says "any packet sent infinitely often will be received infinitely often." In this case, X in the above specification would have the form "If the user (protocol) satisfies P, the channel will satisfy Q," where P says the packet is transmitted infinitely often, and Q says it is received infinitely often. Thus the natural form of the protocol's progress specification is a *nested* conditional $(P \Rightarrow Q) \Rightarrow Y$. We would like to have a formalism that allows the protocol to be verified with respect to such specifications, completely independently of the channels. Ideally, it should also be *compositional*, allowing the immediate conclusion that the composite of protocol and channels satisfies Y if $P \Rightarrow Q$ has been proved of the channels.

While UNITY admits a form of conditional property, *nested* dependencies like $(P \Rightarrow Q) \Rightarrow Y$ are not admitted in the specification languages of many of the "streamlined" formalisms, including UNITY. Thus it becomes necessary to employ various tricks when specifying protocol progress using such formalisms. A common approach is to introduce a special proof rule dealing with messages and channels, and then use the stronger Y as the protocol specification [7, 11, 19]. Such ad hoc approaches work, but showing that the extra rule is valid for a *particular* channel must be done outside the system. Moreover, the approach may not generalize to other instances of progress dependency.

Another approach is to introduce the full generality of temporal logic, admitting arbitrary combinations of whatever temporal operators are used [14]. This can yield a very powerful specification language, but with a corresponding increase in the size and complexity of the proof system. For practical applications, we would like to restrict the size of the formal mechanism to be comparable to, say, a medium-sized programming language.

conditional rules	$(x \leadsto y) \Rightarrow (p \leadsto q)$
closure rules	$p \leadsto q$
program logic	p *ensures* q, p *unless* q
predicate calculus	$[p]$

Figure 1: Layers of Logic

This paper considers progress specifications constructed using a single temporal operator, *leads-to*. We show that a large class of progress specifications, including those with nested dependencies as above, can be represented using sets of leads-to properties of a certain simple form. The advantage of this approach is that this additional power comes almost for free: the proof theory, which is sound and (under certain assumptions) complete, is essentially the UNITY proof theory for leads-to with a different semantic interpretation of conditional properties, plus the predicate calculus. These results support an approach in which a progress specification is first written down in a natural form as a boolean combination of leads-to properties, and then transformed into an equivalent specification in a form that is provable. The approach is intended for use with a particular compositional theory of module specifications [3, 5], but it can be applied with other, similar theories, as well.

The rest of the paper is organized as follows. In the next section we describe the semantic model and the basic proof theory for leads-to properties. In Section 3 we introduce two additional inference rules, and show that the resulting extended theory is sound and, under certain assumptions, complete, with respect to conditional progress properties. Section 4 introduces an example showing how such properties can be used to specify progress for a communication protocol that uses lossy channels. Section 5 shows how a large class of properties can be transformed, using predicate calculus, into a provable form. Section 6 applies such transformations to the example of Section 4, and shows how the top-level progress property can be derived from the specifications of the protocol and the underlying channels. Section 7 concludes the paper with some discussion, including an outline of related work.

2 Foundations

The semantics used here is pretty standard for (individual) reactive systems: a *program*[1] is viewed as a generator of *behaviors*, which are represented as sequences of states. An *abstract* specification defines a temporal predicate, i.e., a boolean function on such sequences of states. A given program satisfies a given abstract specification if and only if each of the program's possible behaviors satisfies the temporal predicate.

We assume a logical mechanism for proving that a program satisfies an abstract specification. It is convenient to view this mechanism as having a "layered" structure. Figure 1 shows this structure, and an example of the kind of conclusion that can be proved using each layer. The foundation is the *predicate calculus*, by which we reason about boolean functions on the state space; *predicates* are the basic building blocks of specifications. The next layer forms the interface between the program and the abstract specification, by allowing certain "basis"

[1]The term "program" should be understood throughout to mean "program or concrete specfication", i.e. the object to be verified with respect to some abstract specification.

Symbol or Expression	Meaning
$\mathbf{g}, \mathbf{h}, \mathbf{k}$	program states
p, b, x_m, p' etc.	state predicates
$p.\mathbf{g}$	value of p at state $\mathbf{g}$
$[p]$	$p.\mathbf{g} = \mathit{true}$ for every $\mathbf{g}$
$\overline{w}$	sequence of states (behavior)
P, Q, R	simple leads-to properties, like $p \leadsto q$
X, Y	composite leads-to properties, like $((p \leadsto q) \vee (x \leadsto y)) \wedge (r \leadsto b)$
$[\![X]\!]$	every behavior satisfies X
$\Theta \vdash P$	P is derivable from hypotheses Θ
$\vdash P$	P is provable without hypotheses
α, β	ordinal numbers
m, n	predicate indices (natural numbers)

Table 1: Notational Conventions

properties (*ensures* and *unless*) to be proved from the program itself. On top of this *program logic* is a layer of "closure rules" allowing *leads-to* properties to be derived from basis properties and other leads-to properties. The idea is that leads-to properties are more abstract than basis properties, and are used in formulating abstract specifications. The main result of this paper shows that under certain conditions it is possible to add a simple top layer —consisting of two additional rules— to deal with *conditional* progress properties, and thereby enable a large class of conditional progress properties to be verified.

In the sequel, we follow some typographical conventions intended to aid the reader in parsing the rather complex expressions that arise. Different letters and typefaces are used to represent different types, as summarized in Table 1.

2.1 States, Transitions and Fairness

In our model, a program defines: a *state space*, which is the cartesian product of the ranges of all of its state variables; a *transition relation* (set of pairs of states) representing the possible state changes that can occur during execution of the program; and an *initial condition*. Throughout this paper the initial condition is assumed to be *true*; this assumption simplifies the presentation, but can be relaxed by standard techniques [17]. The transition relation is not necessarily irreflexive, i.e. transitions from a state to itself are permitted. In what follows, we consider a fixed program, and variables $\mathbf{g}$, $\mathbf{h}$ and $\mathbf{k}$ range over its individual states.

A *segment* is a finite path through the state space as defined by the transition relation. More precisely, the state space, initial condition, and transition relation define a set of segments as follows:

- Every state is a segment.
- If there is a transition from the last state of one segment to the first state of another, the concatenation of the two segments is a segment, defined in the obvious way.

The existence of a segment whose first state is **g** and whose last state is **h** represents the possibility of a (possibly empty) sequence of state transitions taking the system from state **g** to state **h**. We say **h** *is reachable from* **g** in this case.

A nonempty subset of the set of all segments are designated as (locally) *fair*. The precise connection between the program and the set of fair segments is irrelevant here and is left unspecified.[2] If there exists a fair segment whose first state is **g** and whose last state is **h**, we say **h** is *fairly reachable* from **g**.

A *behavior* is any infinite sequence obtainable by concatenating infinitely many fair segments with a segment beginning with an initial state. Behaviors are denoted by roman letters from the end of the alphabet, e.g. w.

The following statements summarize the characteristics of the semantics that are required for later results (in particular for Lemma 2).

A0 The concatenation of a fair segment and a segment is a fair segment.

A1 The set of behaviors of the program contains exactly the infinite sequences that can be constructed inductively by beginning with an initial state and iteratively concatenating a fair segment with it.

A2 The set of states reachable from any state is countable.

2.2 Predicates

For reasoning about state predicates, we use the predicate calculus of Dijkstra and Scholl-ten [8]. Function (or predicate) application is represented by an infix period, which has the highest binding power. Universal and existential quantification are rendered as $\langle \forall\, m : r.m : p.m \rangle$ and $\langle \exists\, m : r.m : q.m \rangle$. These are equivalent to $\langle \forall\, m :: r.m \Rightarrow p.m \rangle$ and $\langle \exists\, m :: r.m \wedge p.m \rangle$ respectively. An omitted range is understood to be *true*.

The syntactic precedence of the boolean operators used in this paper is as follows (most binding at left; symbols grouped together in parentheses have the same precedence):

$$\neg, (\wedge, \vee), \Rightarrow, \equiv$$

In later sections the predicate calculus will be used with *temporal predicates* as well as state predicates. The boolean operator symbols will be overloaded and used with both temporal predicates and state predicates. Where the operators appear more than once, with operands of different types, liberal use of brackets will be made to avoid parsing ambiguity. However, to avoid too many parentheses, we adopt the convention that boolean operators by default bind more tightly than $\rightsquigarrow$; thus $p \rightsquigarrow q \vee r$ is parsed as $p \rightsquigarrow (q \vee r)$.

2.3 Temporal Predicates

Both safety and progress properties can be specified using *temporal predicates*, i.e. boolean functions on (infinite) sequences of states. A program is said to satisfy the property represented by a temporal predicate if and only if the predicate is true for each of its behaviors. As a shorthand for universal quantification over the behavior set of the program, we introduce

[2] Intuitively, a segment would be designated fair if it corresponds to a sequence of transitions that includes a transition attributable to certain program components or statements.

the double square brackets: $[\![X]\!]$ means that the property X is satisfied by every behavior of the (fixed) program.

Temporal predicates are constructed from state predicates using temporal operators; the only temporal operators actually used in this paper are *unless* and $\leadsto$. For state predicates p and q, a sequence satisfies p *unless* q iff[3] for every pair of adjacent states $\mathbf{g}, \mathbf{h}$ in the sequence,

$$(p \wedge \neg q).\mathbf{g} \Rightarrow (p \vee q).\mathbf{h}$$

Because every pair of adjacent states in a behavior is a transition, it is not difficult to see that $[\![p \textit{ unless } q]\!]$ holds if and only if every state transition satisfies $(\mathbf{g}, \mathbf{h})$ the above condition.

A sequence satisfies the leads-to property $p \leadsto q$ iff every $(p \wedge \neg q)$-state in the sequence is followed by a q-state. For a given program, we say a leads-to property $p \leadsto q$ is *persistent* iff $[\![p \textit{ unless } q]\!]$. The importance of this property is that a given behavior satisfies a persistent leads-to property $p \leadsto q$ if and only if it has an infinite suffix containing only $(p \wedge \neg q)$-states.

The semantics of boolean combinations of leads-to properties are defined in the usual way: a sequence satisfies $(p \leadsto q) \wedge (x \leadsto y)$ iff it satisfies both $p \leadsto q$ and $x \leadsto y$; $(p \leadsto q) \vee (x \leadsto y)$ is satisfied iff either $p \leadsto q$ or $x \leadsto y$ is satisfied; $\neg(p \leadsto q)$ is satisfied iff $p \leadsto q$ is not; and a sequence satisfies $p \leadsto q \Rightarrow x \leadsto y$ iff either $p \leadsto q$ is not satisfied or $x \leadsto y$ is satisfied.

The following properties follow from the foregoing definitions:

$$[\![(p \leadsto r) \wedge (r \leadsto q) \Rightarrow (p \leadsto q)]\!] \tag{1}$$

$$[\![\langle \forall\, m :: p_m \leadsto q \rangle \Rightarrow (\langle \exists\, m :: p.m \rangle \leadsto q)]\!] \tag{2}$$

$$[q \Rightarrow r] \Rightarrow [\![(p \leadsto q) \Rightarrow (p \leadsto r)]\!] \tag{3}$$

$$[\![(p \leadsto b \vee r) \wedge (b \leadsto q) \Rightarrow (p \leadsto q \vee r)]\!] \tag{4}$$

$$[\![r \textit{ unless } b]\!] \Rightarrow [\![(p \leadsto q) \Rightarrow (p \wedge r \leadsto (q \wedge r) \vee b)]\!] \tag{5}$$

For any state function M ranging over a set that is well-founded under an ordering $<$, we have also the following:

$$[\![\langle \forall\, k :: p \wedge M{=}k \;\leadsto\; (p \wedge M{<}k) \vee q \rangle \Rightarrow (p \leadsto q)]\!] \tag{6}$$

2.4 Program Logic

The next level of the logic enables the derivation of properties from the program itself. We assume that a mechanism exists through which two types of properties can be proved, *unless* and *ensures*. Well-known examples of such logics exist [7, 11]. We write $\vdash p$ *unless* q and $\vdash p$ *ensures* q to mean that these properties are provable for the given program and particular p and q. We do not define *ensures* here; we require only that for any predicates p and q,

$$\vdash p \textit{ ensures } q \Rightarrow [\![p \leadsto q]\!].$$

2.5 Leads-to Logic

Leads-to properties are proved in the classical deductive manner, by applying *inference rules* to other properties which have already been proved, postulated, or assumed. For a given (fixed) program, we write $\Theta \vdash p \leadsto q$ to indicate that $p \leadsto q$ is derivable from the properties

[3] Throughout this paper, "iff" is used with the meaning "defined to be equivalent to".

$$\frac{\Theta \vdash p \textit{ ensures } q}{\Theta \vdash p \leadsto q} \text{ (Promotion)} \qquad \frac{\Theta \vdash p \leadsto r, \quad \Theta \vdash r \leadsto q}{\Theta \vdash p \leadsto q} \text{ (Transitivity)}$$

$$\frac{\langle \forall\, m :: \Theta \vdash p_m \leadsto q \rangle}{\Theta \vdash \langle \exists\, m :: p_m \rangle \leadsto q} \text{ (Disjunction)} \qquad \frac{\Theta \vdash p \leadsto q, \quad \vdash r \textit{ unless } b}{\Theta \vdash p \wedge r \leadsto (q \wedge r) \vee b} \text{ (PSP)}$$

Figure 2: Proof rules for leads-to

that can be proved of the program using the program logic plus the set Θ of *hypothesis* leads-to properties. (This "sequent calculus" form of assertion simplifies the bookkeeping regarding which assumptions have been discharged. This will be important in the extended logic; at this level, however, Θ is always empty.)

An inference rule such as

$$\frac{\Theta \vdash p \leadsto p', \ \Theta \vdash q \leadsto q'}{\Theta \vdash p \wedge q \leadsto p' \vee q'}$$

has a set of *premises* (in this case $\Theta \vdash p \leadsto p'$ and $\Theta \vdash q \leadsto q'$), and a *conclusion* (in this case $\Theta \vdash p \wedge q \leadsto p' \vee q'$). The rule means "if $p \leadsto p'$ is and $q \leadsto q'$ are both derivable from properties provable in the program logic or in the set Θ, then $p \wedge q \leadsto p' \vee q'$ is also derivable". A *derivation* is a finite sequence of lines, each containing an assertion of the form $\Theta \vdash P$, along with the inference rule justifying its introduction, including an indication of the already-derived assertions that match the premises of the rule. We define a *proof* to be a derivation with no undischarged hypotheses (i.e. where Θ is empty); since the logic so far does not admit the introduction of hypothesis properties, the only derivations we can construct using this part of the logic are proofs.

The results that follow do not depend upon the *particular* inference rules used in the logic; rather, they depend upon certain properties of the rules. However, for concreteness we present a definite set of rules, similar to those used in UNITY [7]. The set comprises two kinds of rules:

- *Basis* rules, which have leads-to properties in the conclusion but not the premises, and
- *Closure* rules, which have one or more leads-to properties in the premises and a single leads-to property in the conclusion.

The rules used in this paper are shown in Figure 2.

Additional closure rules may be proved as metatheorems from the rules in Figure 2 by induction on the length of the derivation [7]. For example, inference rules corresponding to properties (3)–(6) can be proved as metatheorems in this way; they are shown in Figure 3.

The main characteristic of the inference rules that we require later is the following:

A3 All closure rules are valid for *individual behaviors*. That is, any behavior that satisfies each premise of a rule also satisfies the conclusion.

$$\frac{\Theta \vdash p \leadsto q, \quad [q \Rightarrow r]}{\Theta \vdash p \leadsto r} \text{ (Cons. Weakening)} \qquad \frac{\Theta \vdash p \leadsto q \vee b, \quad \Theta \vdash q \leadsto r}{\Theta \vdash p \leadsto r \vee b} \text{ (Cancellation)}$$

$$\frac{[p \Rightarrow q]}{\Theta \vdash p \leadsto q} \text{ (Implication)} \qquad \frac{\langle \forall\, k :: \Theta \vdash p \wedge M{=}k \;\leadsto\; (p \wedge M{<}k) \vee q \rangle}{\Theta \vdash p \leadsto q} \text{ (Induction)}$$

Figure 3: Derived Inference Rules

3 Proving Conditional Properties

We are now in a position to show how to add another "layer" on top of the logical mechanism already described, to support proof of implications of the form $P \Rightarrow Q$, or more generally $\langle \forall\, m :: P_m \rangle \Rightarrow Q$. The additional mechanism consists of two additional rules for the introduction and elimination of sets of leads-to properties as hypotheses in proofs. Throughout this section, we refer to the proof system for proving simple leads-to properties as the *underlying* proof system.

We want to consider sets of *hypothesis* leads-to properties. The notations $\{P_m\}$ and $\{x_m \leadsto y_m\}$ hereafter denote *countable* sets of persistent leads-to properties. Without loss of generality, we assume the hypothesis properties are ordered in some fashion, and the range of m in both notations is understood to be the natural numbers less than the cardinality of the set. The notation $\Theta, P \vdash \ldots$ is an abbreviation for $\Theta \cup P \vdash \ldots$. Also, when the set of hypothesis properties is small, we may write the antecedent of an implication as a conjunction instead of defining a set explicitly (e.g. $P \wedge Q \Rightarrow R$).

3.1 Inference Rules

We present two rules. One is a basis rule, allowing the introduction of any persistent leads-to property as a hypothesis (i.e. on the left of the turnstile); the other is a rule allowing hypotheses to be discharged (by moving them to the right of the turnstile). The rules are shown in Figure 4. Note that in the Assumption rule, the property $x_m \leadsto y_m$ on the right side of the turnstile is implicitly universally quantified over the index m. Leaving such universal quantifications implicit (or equivalently, introducing universal quantification implicitly by generalization) is traditional in the UNITY proof theory; the idea is that the index m is a free variable about which nothing is known —or used in the proof— besides what is implied by its range.

The addition of a rule for introducing hypotheses renders nontrivial the distinction between a *derivation* and a *proof*. We reserve the latter term for a derivation in which no hypothesis is undischarged, i.e. in which the set of properties on the left of the turnstile in the last line is empty. As before, we write $\vdash P$ to indicate that P is provable.

Because we have added a new "basis" rule to the logic, any derived inference rules proved by induction must be re-proved, taking into account that there are now *two* base cases. (This is one reason for restricting hypotheses to *persistent* leads-to properties: the other basis rule (Promotion) establishes a persistent leads-to property, and the corresponding unless property is sometimes used in inductive proofs.) Verification that the proofs go through with the

$$\frac{\langle \forall\, m :: \vdash x_m \ \mathit{unless}\ y_m \rangle}{\Theta, \{x_m \leadsto y_m\} \vdash x_m \leadsto y_m}\ \text{(Assumption)} \qquad \frac{\Theta, \{x_m \leadsto y_m\} \vdash p \leadsto q}{\Theta \vdash \langle \forall\, m :: x_m \leadsto y_m \rangle \Rightarrow (p \leadsto q)}\ \text{(Discharge)}$$

Figure 4: Inference Rules for Conditional Properties

additional basis rule is left as an excercise for the reader.

3.2 Soundness and Completeness

Next we show that the extended theory is sound and, under certain assumptions, complete.

Lemma 0. Let $\{x_m \leadsto y_m\}$ be a countable set of leads-to properties. If $\{x_m \leadsto y_m\} \vdash p \leadsto q$, then every behavior satisfying $x_m \leadsto y_m$ for each m also satisfies $p \leadsto q$. □

Proof. The proof is a straightforward induction on the length of the derivation of $p \leadsto q$ from $\{x_m \leadsto y_m\}$. The closure rules are valid for individual behaviors (**A3**) so every behavior that satisfies each leads-to property $x_m \leadsto y_m$ also satisfies every intermediate property appearing in the proof, including $p \leadsto q$. □

Theorem 1. If there exists a proof of $\langle \forall\, m :: x_m \leadsto y_m \rangle \Rightarrow (p \leadsto q)$, then

$$[\![\langle \forall\, m :: x_m \leadsto y_m \rangle \Rightarrow (p \leadsto q)]\!]$$

□

Proof. The only way a property containing an implication can be derived is via the discharge rule. Thus to prove $\langle \forall\, m :: x_m \leadsto y_m \rangle \Rightarrow (p \leadsto q)$, there must be a derivation of $p \leadsto q$ using the properties $x_m \leadsto y_m$ (and no others) as hypotheses, i.e. $\{x_m \leadsto y_m\} \vdash p \leadsto q$. The theorem then follows by Lemma 0. □

We are able to prove completeness only certain assumptions, some of which have already been mentioned (**A0**–**A3**). The following key Lemma introduces the other assumptions.

Lemma 2. Let $\{x_m \leadsto y_m\}$ be a countable set of leads-to properties such that

$$\langle \forall\, m :: [\![x_m \ \mathit{unless}\ y_m]\!] \rangle \tag{7}$$

$$[\![\langle \forall\, m :: x_m \leadsto y_m \rangle \Rightarrow (p \leadsto q)]\!] \tag{8}$$

and assume further that

$$[\![p \ \mathit{unless}\ q]\!] \tag{9}$$

Then there exists a state function M mapping $(p \wedge \neg q)$-states to ordinal numbers, and for each m, and each ordinal number α there exists a predicate u_m^α, such that the following hold:

$$[y_m \wedge u_m^\alpha \ \Rightarrow\ M < \alpha \vee q] \tag{10}$$

$$[\![u_m^\alpha \ \mathit{unless}\ M < \alpha \vee q]\!] \tag{11}$$

$$[\![p \wedge M = \alpha \ \leadsto\ \langle \exists\, m :: x_m \wedge u_m^\alpha \rangle \vee M < \alpha \vee q]\!] \tag{12}$$

□

The proof of Lemma 2 is given in the appendix.

Theorem 3. Let $\{x_m \leadsto y_m\}$ be a countable set of leads-to properties, and let p and q be such that (7–8) hold. Assume further that the underlying logic is complete with respect to (9–12), i.e. proofs of those properties exist if they hold. Then a proof of $\langle \forall\, m :: x_m \leadsto y_m \rangle \Rightarrow (p \leadsto q)$ exists. □

Proof. A derivation of $\langle \forall\, m :: x_m \leadsto y_m \rangle \Rightarrow (p \leadsto q)$ is shown below. By Lemma 2, the hypotheses of the Theorem imply properties (10–12); by the assumption of completeness, proofs of (9–12) exist. In the derivation, the justification "completeness" is used for the introduction of these properties. Note that in $x_m \leadsto y_m$, m is implicitly universally quantified over the whole set of assumptions.

0.	$\vdash u_m^\alpha$ *unless* $M{<}\alpha \vee q$	{ (11), completeness (all m, α) }
1.	$\{x_m \leadsto y_m\} \vdash x_m \leadsto y_m$	{ Assumption (all m) }
2.	$\{x_m \leadsto y_m\} \vdash x_m \wedge u_m^\alpha \leadsto (y_m \wedge u_m^\alpha) \vee M{<}\alpha \vee q$	{ **0**,**1**, PSP }
3.	$\{x_m \leadsto y_m\} \vdash x_m \wedge u_m^\alpha \leadsto M{<}\alpha \vee q$	{ **2**, (10), Consequence Weakening. }
4.	$\{x_m \leadsto y_m\} \vdash \langle \exists\, m :: x_m \wedge u_m^\alpha \rangle \leadsto M{<}\alpha \vee q$	{ **3**, Disjunction }
5.	$\{x_m \leadsto y_m\} \vdash p \wedge M{=}\alpha \;\leadsto\; \langle \exists\, m :: x_m \wedge u_m^\alpha \rangle \vee M{<}\alpha \vee q$	{ (12), completeness }
6.	$\{x_m \leadsto y_m\} \vdash p \wedge M{=}\alpha \;\leadsto\; M{<}\alpha \vee q$	{ **4**,**5**, Cancellation }
7.	$\{x_m \leadsto y_m\} \vdash p$ *unless* q	{ (9), completeness }
8.	$\{x_m \leadsto y_m\} \vdash p \wedge M{=}\alpha \;\leadsto\; (p \wedge (M{<}\alpha \vee q)) \vee q$	{ **7**, **6**, PSP }
9.	$\{x_m \leadsto y_m\} \vdash p \wedge M{=}\alpha \;\leadsto\; (p \wedge M{<}\alpha) \vee q$	{ **8**, pred. calc. }
10.	$\{x_m \leadsto y_m\} \vdash p \leadsto q$	{ **9**, induction }
11.	$\vdash \langle \forall\, m :: x_m \leadsto y_m \rangle \Rightarrow (p \leadsto q)$	{ **10**, Discharge }

□

4 Conditional Progress Specifications

Let us consider the example of a data transport protocol that uses unreliable channels at the lower level. The protocol layer consists of two peers, a Sender and a Receiver. Its environment consists of a Sending User and Receiving User above, and two lossy channels below. We describe the interfaces in terms of *auxiliary variables*, which are state variables introduced to make the specification easier to formulate. They need not be present in the actual implementation; any information they contain must be recoverable from other (non-auxiliary) state variables.

The upper interface of the protocol is defined in terms of two auxiliary variables, *ins* and *outs*. Each is a sequence of messages, initially empty. At any time, *ins* contains the sequence of all bytes sent by the Sending User, while *outs* is the sequence of bytes delivered to the Receiving User. When the User sends a group of bytes, they are simultaneously appended to *ins*; when a group of bytes is delivered to the Receiving User, they are appended to *outs*. The number of bytes in a sequence s is denoted by $|s|$. At any state where $|ins| > |outs|$, a byte has been sent but not yet received, and the system is required to make progress. (Because

we are concerned with *progress* specifications, we will ignore most of the *safety* aspects of the specification. The latter would require, for example, that *outs* be a prefix of *ins*.)

The lower-level service used by the protocol is modeled by a lossy channel in each direction, from the Sender to the Receiver and vice versa. These channels are designated *sr* and *rs* respectively. For each channel $c \in \{sr, rs\}$, the set G_c contains the messages that may be sent over the channel; we also define two boolean state functions, *c-in* and *c-out*, which map messages in G_c to *true* or *false*. The function *c-in* is *true* for message $m \in G_c$ at a given state iff m is being "transmitted" on channel c at that state. When the protocol sends message m on channel *sr*, it causes some state change that makes *sr-in.m* true; after a time, the channel causes a state change that falsifies *sr-in.m*. There may be more than one message m such that *c-in.m* is true. In a similar way *c-out.m* indicates that m is being "received" on c.

4.1 Lossy Channel Progress

A lossy channel can lose a message any finite number of times, but if the message is repeatedly transmitted, it *will* eventually be delivered. In other words, for any message m, if m is transmitted infinitely often, m is received infinitely often. The leads-to property that corresponds to "p holds infinitely often" is $true \leadsto p$. We thus arrive at the following progress specifications for channels *sr* and *rs*:

$$\langle \forall\, m : m \in G_{sr} : (true \leadsto \textit{sr-in.m}) \Rightarrow (true \leadsto \textit{sr-out.m}) \rangle \tag{13}$$

$$\langle \forall\, m : m \in G_{rs} : (true \leadsto \textit{rs-in.m}) \Rightarrow (true \leadsto \textit{rs-out.m}) \rangle \tag{14}$$

It is important to note that each leads-to property constrains *only* the channel, and says that *it need not deliver any message that is transmitted only a finite number of times.* It might seem that this requirement is too weak to be useful, since at any point a protocol will have transmitted a message only a finite number of times. We shall see, however, that it is adequate.

4.2 Protocol Progress

The protocol must ensure that for each byte sent (appended to *ins*), a byte is delivered (appended to *outs*). This is expressed by the property

$$\langle \forall\, j :: |ins| \geq j \leadsto |outs| \geq j \rangle$$

This specification can only be implemented if the underlying channels are reliable. However, the protocol progress specification should be written in such a way that it can still be satisfied even if one or both of the underlying channels fails to satisfy *its* specification. Therefore the above property should be conditioned on the progress property that the protocol expects of the channels. Following the convention of leaving an outermost universal quantification implicit, the resulting specification is:

$$\begin{array}{rl} & \langle \forall\, m : m \in G_{sr} : (true \leadsto \textit{sr-in.m}) \Rightarrow (true \leadsto \textit{sr-out.m}) \rangle \\ \wedge & \langle \forall\, n : n \in G_{rs} : (true \leadsto \textit{rs-in.n}) \Rightarrow (true \leadsto \textit{rs-out.n}) \rangle \\ & \Rightarrow \quad |ins| \geq j \leadsto |outs| \geq j \end{array} \tag{15}$$

For conciseness, define the following abbreviations:

$$\begin{array}{rcl} R.j & \stackrel{\text{def}}{=} & |ins| \geq j \leadsto |outs| \geq j \\ P.m & \stackrel{\text{def}}{=} & true \leadsto sr\text{-}in.m \\ Q.m & \stackrel{\text{def}}{=} & true \leadsto sr\text{-}out.m \\ P'.m & \stackrel{\text{def}}{=} & true \leadsto rs\text{-}in.m \\ Q'.m & \stackrel{\text{def}}{=} & true \leadsto rs\text{-}out.m \end{array}$$

Using these abbreviations and omitting ranges, the specification becomes:

$$\langle \forall\, m :: P.m \Rightarrow Q.m \rangle \wedge \langle \forall\, n :: P'.n \Rightarrow Q'.n \rangle \;\Rightarrow\; R.k \tag{16}$$

5 Provable Progress Specifications

The specification (16) given above is not in a form that can be proved using the rules given in Section 3, because of the nested implications. However, it turns out that for a large class of progress properties, including the one above, it is possible to construct equivalent forms that *are* provable in the extended theory. The equivalent properties are derived from the originals by treating the properties as temporal predicates, and manipulating them using predicate calculus. For example, a specification of the form

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

is propositionally equivalent to

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

As in this case, such manipulations may introduce disjunctions of leads-to properties, for which we have no proof rule. It turns out, however, that any disjunction of *persistent* leads-to properties can be replaced by single leads-to property (which also happens to be persistent). It is also possible to similarly replace *finite* disjunctions of leads-to properties that are *not* persistent with a single leads-to property [6]. For simplicity, we focus here on the case in which all disjuncts are persistent.

5.1 Disjunctions of Leads-to Properties

The following theorem is the basis for the results that follow in this section.

Theorem 4. If $[\![p \textit{ unless } q]\!]$ and $[\![p' \textit{ unless } q']\!]$, then

$$[\![(p \leadsto q) \vee p' \leadsto q' \equiv (p \wedge p' \leadsto q \vee q')]\!]$$

□

Proof. For infinite sequence $\overline{w}$ and state predicate p, define $\overline{w} \nearrow p$ ("$\overline{w}$ converges to p") to mean that $\overline{w}$ has an infinite suffix in which every state is a p-state. We first observe, for any p and q:

(i) $\overline{w}\nearrow(p \wedge q) \equiv (\overline{w}\nearrow p) \wedge (\overline{w}\nearrow q)$

(ii) If $[\![p \ \textit{unless}\ q]\!]$, then any behavior $\overline{w}$ satisfies $p \leadsto q$ if and only if $\neg(\overline{w}\nearrow(p \wedge \neg q))$.

Next we observe that from $[\![p \ \textit{unless}\ q]\!]$ and $[\![p' \ \textit{unless}\ q']\!]$ follows for any $\overline{w}$, p and q:

(iii) $[\![p \wedge p' \ \textit{unless}\ q \vee q']\!]$

Now we calculate:

$$
\begin{array}{ll}
 & \overline{w} \text{ satisfies } (p \leadsto q) \ \vee \ (p' \leadsto q') \\
= & \quad \{ \text{ definition of } \vee \ \} \\
 & (\overline{w} \text{ satisfies } p \leadsto q) \ \vee \ (\overline{w} \text{ satisfies } p' \leadsto q') \\
= & \quad \{ \text{ Observation (ii) } \} \\
 & \neg(\overline{w}\nearrow(p \wedge \neg q)) \vee \neg(\overline{w}\nearrow(p' \wedge \neg q')) \\
= & \quad \{ \text{ DeMorgan's Law } \} \\
 & \neg((\overline{w}\nearrow(p \wedge \neg q)) \ \wedge \ (\overline{w}\nearrow(p' \wedge \neg q'))) \\
= & \quad \{ \text{ Observation (i) } \} \\
 & \neg(\overline{w}\nearrow(p \wedge \neg q \wedge p' \wedge \neg q')) \\
= & \quad \{ \text{ predicate calculus and observations (ii) and (iii) } \} \\
 & \overline{w} \text{ satisfies } p \wedge p' \leadsto q \vee q'
\end{array}
$$

□

For any persistent leads-to properties $P = p \leadsto q$ and $Q = p' \leadsto q'$, we now *define* $P \vee Q \stackrel{\text{def}}{=} p \wedge p' \leadsto q \vee q'$. We need to show that this operation enjoys the usual properties of disjunction—e.g., it is idempotent, associative, and monotonic. These properties are necessary to make full use of implications in a compositional theory. For example, in the protocol specification, monotonicity is needed to derive the top-level progress specification $R.k$ from the conditional protocol specification and the channel specifications.

It is immediate from its definition that $\vee$ for persistent leads-to properties is idempotent and associative. Moreover, its unit is the leads-to equivalent of "*false*", namely $\textit{true} \leadsto \textit{false}$. For our purposes, the following (weak) form of monotonicity of $\vee$ suffices:

$$[\![P \Rightarrow Q]\!] \wedge [\![P \vee R]\!] \Rightarrow [\![Q \vee R]\!]$$

An inference rule that states this form of the monotonicity of $\vee$ (and also serves as a form of Modus Ponens) is the following:

$$\frac{\langle \forall\, m :: \Theta \vdash P_m \vee R\rangle, \qquad \Theta \vdash \langle \forall\, m :: P_m\rangle \Rightarrow Q}{\Theta \vdash Q \vee R} \text{(Monotonicity of } \vee\text{)}$$

Proof of Monotonicity. We first observe that $\Theta \vdash \langle \forall\, m :: P_m\rangle \Rightarrow Q$ holds only if $\Theta, \{P_m\} \vdash Q$. Therefore it is sufficient to prove

$$\frac{\langle \forall\, m :: \Theta \vdash P_m \vee R\rangle, \qquad \Theta, \{P_m\} \vdash Q}{\Theta \vdash Q \vee R}$$

Letting $P_m = p_m \leadsto p'_m$ and $Q = q \leadsto q'$, and rewriting the latter rule using the definition of $\vee$, we have:

$$\frac{\langle \forall\, m :: \Theta \vdash p_m \wedge r \leadsto p'_m \vee r' \rangle \qquad \Theta, \{p_m \leadsto p'_m\} \vdash q \leadsto q'}{\Theta \vdash q \wedge r \leadsto q' \vee r'}$$

For later use, we observe that $q \wedge r \leadsto q' \vee r'$ follows from $q \leadsto q'$ using Implication, Transitivity, and Consequence Weakening.

Now we induct on the length of the derivation of $q \leadsto q'$ from $\Theta \cup \{p_m \leadsto p'_m\}$. There are two base cases and three step cases, depending on the rule rule applied to obtain $q \leadsto q'$. Our obligation in each case is to establish $q \wedge r \leadsto q' \vee r'$ (i.e. $(q \leadsto q') \vee (r \leadsto r')$), using the other premise of the monotonicity rule, the premises of the rule applied in each step case, and the inductive hypothesis.

Base (Promotion) In this case $q \leadsto q'$ follows by Promotion, so we have $\vdash q$ *ensures* q'. Then $q \wedge r \leadsto q' \vee r'$ is provable, as we observed above.

Base (Assumption) In the second base case $q \leadsto q'$ is one of the assumptions, i.e. $q \leadsto q' \in \Theta$ or $q = p_m$ and $q' = p'_m$ for some m. In either case $q \wedge r \leadsto q' \vee r'$ again follows as observed above.

Step (Transitivity) In this step case $q \leadsto q'$ follows by transitivity from $q \leadsto b$ and $b \leadsto q'$.

0.	$\Theta \vdash p_m \wedge r \;\leadsto\; p'_m \vee r'$	{ premise of rule being proved (all m) }
1.	$\Theta, \{p_m \leadsto p'_m\} \vdash q \leadsto b$	{ premise of Transitivity }
2.	$\Theta, \{p_m \leadsto p'_m\} \vdash b \leadsto q'$	{ other premise of Transitivity }
3.	$\Theta \vdash q \wedge r \;\leadsto\; b \vee r'$	{ **0**,**1**, Inductive Hypothesis }
4.	$\Theta \vdash b \wedge r \;\leadsto\; q' \vee r'$	{ **0**,**2**, Ind. Hyp. }
5.	$\vdash r$ *unless* r'	{ $r \leadsto r'$ is persistent }
6.	$\Theta \vdash q \wedge r \;\leadsto\; (b \wedge r) \vee r'$	{ **3**,**5**, PSP and pred. calc. }
7.	$\Theta \vdash q \wedge r \leadsto q' \vee r'$	{ **6**,**4**, Cancellation }

Step (Disjunction) In this case $q = \langle \exists\, i :: q_i \rangle$, and $\Theta, \{p_m \leadsto p'_m\} \vdash \langle \exists\, i :: q_i \rangle \;\leadsto\; q'$ follows from the fact that, for each i, $\Theta, \{p_m \leadsto p'_m\} \vdash q_i \leadsto q'$. In the following derivation, i is implicitly universally quantified.

0.	$\Theta \vdash \langle \forall\, m :: p_m \wedge r \leadsto p'_m \vee r' \rangle$	{ premise of rule being proved }
1.	$\Theta, \{p_m \leadsto p'_m\} \vdash q_i \leadsto q'$	{ premise of Disjunction rule }
2.	$\Theta \vdash q_i \wedge r \leadsto q' \vee r'$	{ **0**,**1**, Inductive Hypothesis }
3.	$\Theta \vdash \langle \exists\, i :: q_i \wedge r \rangle \leadsto q' \vee r'$	{ **2**, Disjunction }
4.	$\Theta \vdash \langle \exists\, i :: q_i \rangle \wedge r \leadsto q' \vee r'$	{ **3**, pred. calc. }

Step (PSP) In this case we have $q = (b \wedge x)$, $q' = (b' \wedge x) \vee x'$, and $q \leadsto q'$ follows from $b \leadsto b'$, x *unless* x', by PSP.

0.	$\Theta \vdash p_m \wedge r \;\leadsto\; p'_m \vee r'$	{ premise of rule being proved (note: all m) }
1.	$\Theta, \{p_m \leadsto p'_m\} \vdash b \leadsto b'$	{ premise of PSP }

2.	$\vdash x \ \mathit{unless}\ x'$	{ other premise of PSP }
3.	$\Theta \vdash b \wedge r \rightsquigarrow b' \vee r'$	{ **0**,**1**, Ind. Hyp. }
4.	$\Theta \vdash b \wedge r \wedge x \rightsquigarrow ((b' \vee r') \wedge x) \vee x'$	{ **2**,**3**, PSP }
5.	$\Theta \vdash b \wedge x \wedge r \rightsquigarrow (b' \wedge x) \vee x' \vee r'$	{ **4**, pred. calc. }

This concludes the induction, and we have thus established the simpler inference rule, from which the Monotonicity rule follows. □

A simpler form of monotonicity is stated in the following inference rule:

$$\frac{\Theta \vdash P \vee R, \quad \Theta \vdash P \Rightarrow Q}{\Theta \vdash Q \vee R} \text{ (Simple Monotonicity)}$$

5.2 Putting Progress Properties in Provable Form

We say a property is in *provable form* iff it satisfies one of the following conditions:

- It is a simple leads-to property.
- It is of the form $\langle \forall\, m :: P_m \rangle \Rightarrow Q$, where Q and each P_m is a simple property.

Using the results of the previous section, we can define a class of progress properties such that for each member of the class there is an equivalent set of provable-form properties. More precisely, for any property X in this class, there exists a collection of properties $Q_0, \ldots Q_N$ such that each Q_i is in provable form, and

$$[\![X]\!] \equiv [\![Q_0 \wedge Q_1 \wedge \ldots \wedge Q_N]\!]$$

The right-hand side of the above is equivalent to

$$[\![Q_0]\!] \wedge [\![Q_1]\!] \wedge \ldots \wedge [\![Q_N]\!]$$

and thus the collection of properties constitutes a specification that is exactly equivalent to X with respect to the given program.

For a given program, define the class $\mathcal{P}$ of progress properties to be the smallest class satisfying the following conditions:

- For each leads-to property $p \rightsquigarrow q$ such that $[\![p \ \mathit{unless}\ q]\!]$, $p \rightsquigarrow q$ is in $\mathcal{P}$.
- If X and Y are in $\mathcal{P}$, then so are $X \vee Y$, $X \wedge Y$, and $X \Rightarrow Y$.

Note that P contains properties expressing arbitrarily-nested dependencies like $(\cdots(P \Rightarrow Q) \cdots \Rightarrow R) \Rightarrow X$. However, it can be shown that for every property in $\mathcal{P}$ there is an equivalent specification consisting entirely of provable-form properties.

Theorem 5. For every property $X \in \mathcal{P}$, there exists a finite set W_X of properties such that

(i) each property in W_X is in provable form, and

(ii) any behavior satisfies X if and only if it satisfies every property in W_X.

□

A proof of this result appears in an earlier report [6]. Translating to provable form involves finding a conjunctive normal form (CNF) equivalent for X; unfortunately, this may result in an exponential blowup of the "size" of the property (i.e. the number of occurrences of simple leads-to properties in the expression defining it).

Because universal quantification distributes over conjunction, we can actually enlarge the class of properties having provable-form equivalents to include some quantified properties.

Theorem 6. If $P_m \in \mathcal{P}$ for each m, then there exists a set of provable-form properties, the conjunction of which is equivalent to $\langle \forall\, m :: P_m \rangle$. □

Proof. The set is just the union of the sets of provable-form equivalents for the properties P_m. □

In the next section we apply these results to the example protocol specification introduced earlier.

6 Example Revisited

Let us consider how the results of the previous section can be applied to the specification given earlier.

Using the predicate calculus, it is straightforward to show that the specification given in (16) is equivalent to the following:

$$\begin{array}{lllcll} \langle \exists\, m, n :: & & (& & P.m \vee P'.n \vee R.k &) \\ & \wedge & (& Q.m & \Rightarrow \quad P'.n \vee R.k &) \\ & \wedge & (& Q'.n & \Rightarrow \quad P.m \vee R.k &) \\ & \wedge & (& Q.m \wedge Q'.n & \Rightarrow \quad R.k &) \ \rangle \end{array} \tag{17}$$

Because each leads-to property in the original specification is persistent, we can replace each disjunction in the above property by a single leads-to property as defined earlier. However, the resulting property is not in provable form because of the enclosing existential quantification. That is, the above specification has the general form

$$[\![\langle \exists\, m :: X(m,k) \rangle]\!]$$

i.e., for every k, and every behavior $\overline{w}$, there exists m such that $\overline{w}$ satisfies $X(m,k)$. Our proof theory gives no way to establish such an assertion. If the message sets G_{sr} and G_{rs} (i.e. the ranges of m and n) are finite, (17) is actually a finite disjunction. It is thus in the set $\mathcal{P}$ defined earlier, and has a provable-form equivalent. However, as noted earlier, the transformation to provable form can result in an exponential blowup in the size of the specification: in this case the equivalent set contains an implication of the form $\langle \forall\, m : m \in G' : Q.m \rangle \Rightarrow \cdots$ for every subset G' of $G_{sr} \cup G_{rs}$.

A more reasonable approach is Skolemization. The specification (17) is equivalent to one of the form

$$\langle \exists\, f :: \langle \forall\, \overline{w} :: X(f(\overline{w}, k), k) \rangle \rangle$$

where f is a function from behaviors and natural numbers to messages. Such a specification can sometimes be proved *constructively* by exhibiting a particular function of the appropriate type, i.e. defining f and then showing that for every k and every behavior $\overline{w}$, $\overline{w}$ satisfies $X(f(\overline{w}, k), k)$. However, we must take care in defining f so the specification is provable, because in general the value of f depends on the *entire* behavior, which is not visible at any one state. The key observation in the case of our example is that *the value of the Skolem function is irrelevant for any behavior in which $R.k$ holds.* Operationally speaking, we need only identify a *particular* lower-level message for a given behavior so that the protocol can "blame" the lower-level channels if the higher-level progress specification $R.k$ is not satisfied by that behavior. We therefore define a state function for each channel, whose value is constant in some infinite suffix of any behavior that does *not* satisfy $R.k$ for some k. Fortunately this is typically not difficult: there is usually *some* lower-level message that will be transmitted repeatedly if progress is not made.

For the example, let us assume that auxiliary variables nxt_{sr} and nxt_{rs} denote messages that are transmitted infinitely often if $R.k$ is not satisfied for some k. (That is, nxt_{sr} and nxt_{rs} act like Skolem functions, but they are functions only of the current state.) The above specification then becomes

$$\begin{array}{lrcl} & (& & P.nxt_{sr} \vee P'.nxt_{rs} \vee R.k \quad) \\ \wedge & (\quad Q.nxt_{sr} & \Rightarrow & P'.nxt_{rs} \vee R.k \quad) \\ \wedge & (\quad Q'.nxt_{rs} & \Rightarrow & P.nxt_{sr} \vee R \quad) \\ \wedge & (\quad Q.nxt_{sr} \wedge Q'.nxt_{rs} & \Rightarrow & R.k \quad) \end{array} \tag{18}$$

Now we need only replace the disjunctions with elementary leads-to properties in order to have a provable-form specification. Thanks to the fact that the corresponding *unless*-properties hold for P, P' and R, this is a valid transformation. The final specification then consists of the following four properties:

$$|ins| \geq k \rightsquigarrow \textit{sr-in}.nxt_{sr} \vee \textit{rs-in}.nxt_{rs} \vee |outs| \geq k \tag{19}$$

$$(true \rightsquigarrow \textit{sr-out}.nxt_{sr}) \Rightarrow (|ins| \geq k \rightsquigarrow \textit{rs-in}.nxt_{rs} \vee |outs| \geq k) \tag{20}$$

$$(true \rightsquigarrow \textit{rs-out}.nxt_{rs}) \Rightarrow (|ins| \geq k \rightsquigarrow \textit{sr-in}.nxt_{sr} \vee |outs| \geq k) \tag{21}$$

$$(true \rightsquigarrow \textit{sr-out}.nxt_{sr}) \wedge (true \rightsquigarrow \textit{rs-out}.nxt_{rs}) \Rightarrow (|ins| \geq k \rightsquigarrow |outs| \geq k) \tag{22}$$

It remains to show that the desired progress property $|ins| \geq k \rightsquigarrow |outs| \geq k$ can be proved from the above and the channel specifications (13) and (14). The proof follows. To aid in following the proof, the pattern match required to apply the inference rule is indicated in each justification.

0. $|ins| \geq k \rightsquigarrow \textit{sr-out}.nxt_{sr} \vee \textit{rs-in}.nxt_{rs} \vee |outs| \geq k$

$$\{\ (13), (19), \text{simple or-mono w/} \begin{pmatrix} p, p' := true, \textit{sr-in}.nxt_{sr} \\ q, q' := true, \textit{sr-out}.nxt_{sr} \\ r, r' := |ins| \geq k,\ \textit{rs-in}.nxt_{rs} \vee |outs| \geq k \end{pmatrix} \ \}$$

1. $|ins| \geq k \leadsto rs\text{-}out.nxt_{rs} \vee sr\text{-}in.nxt_{sr} \vee |outs| \geq k$

$$\{\ (14),\ (19),\ \text{simple or-mono w/} \begin{pmatrix} p, p' := true, rs\text{-}in.nxt_{rs} \\ q, q' := true, rs\text{-}out.nxt_{rs} \\ r, r' := |ins| \geq k,\ sr\text{-}in.nxt_{sr} \vee |outs| \geq k \end{pmatrix}\ \}$$

2. $|ins| \geq k \leadsto rs\text{-}in.nxt_{rs} \vee |outs| \geq k$

$$\{\ \mathbf{0},\ (20),\ \text{simple or-mono w/} \begin{pmatrix} p, p' := true, sr\text{-}out.nxt_{sr} \\ q, q' := |ins| \geq k,\ rs\text{-}in.nxt_{rs} \vee |outs| \geq k \\ r, r' := |ins| \geq k,\ rs\text{-}in.nxt_{rs} \vee |outs| \geq k \end{pmatrix}\ \}$$

3. $|ins| \geq k \leadsto sr\text{-}in.nxt_{sr} \vee |outs| \geq k$

$$\{\ \mathbf{1},\ (21),\ \text{simple or-mono w/} \begin{pmatrix} p, p' := true, rs\text{-}out.nxt_{rs} \\ q, q' := |ins| \geq k, sr\text{-}in.nxt_{sr} \vee |outs| \geq k \\ r, r' := |ins| \geq k,\ sr\text{-}in.nxt_{sr} \vee |outs| \geq k \end{pmatrix}\ \}$$

4. $|ins| \geq k \leadsto rs\text{-}out.nxt_{rs} \vee |outs| \geq k$

$$\{\ \mathbf{2},\ (14),\ \text{simple or-mono w/} \begin{pmatrix} p, p' := true, rs\text{-}in.nxt_{rs} \\ q, q' := true, rs\text{-}out.nxt_{rs} \\ r, r' := |ins| \geq k, |outs| \geq k \end{pmatrix}\ \}$$

5. $|ins| \geq k \leadsto sr\text{-}out.nxt_{sr} \vee |outs| \geq k$

$$\{\ \mathbf{3},\ (13),\ \text{simple or-mono w/} \begin{pmatrix} p, p' := true, sr\text{-}in.nxt_{sr} \\ q, q' := true, sr\text{-}out.nxt_{sr} \\ r, r' := |ins| \geq k, |outs| \geq k \end{pmatrix}\ \}$$

6. $|ins| \geq k \leadsto |outs| \geq k$

$$\{\ \mathbf{4},\ \mathbf{5},\ (22),\ \text{or-mono w/} \begin{pmatrix} p_0, p_0' := true, sr\text{-}out.nxt_{sr} \\ p_1, p_1' := true, rs\text{-}out.nxt_{rs} \\ q, q' := |ins| \geq k, |outs| \geq k \\ r, r' := |ins| \geq k, |outs| \geq k \end{pmatrix}\ \}$$

7 Discussion

7.1 Using Persistent Properties in Specifications

The foregoing results depend in several places on *persistent* properties being used in specifications, i.e. that p *unless* q holds for each leads-to property $p \leadsto q$ used. This seems, on the one hand, a reasonable condition—it suggests that each leads-to property in the specification is somehow as weak as possible. On the other hand, it can be difficult to satisfy if a progress requirement $p \leadsto q$ is "distributed"—in the sense that p can only become true at one location in the system and q can only become true at a different location. In such cases, it may be most convenient to have either p or q or both be stable (i.e. once true they remain true forever, as with $|ins| \geq k$ in the example above); this is because p *unless* q requires that p be falsified *only while* q *holds*. If neither p nor q is stable, satisfying p *unless* q may impose additional synchronization requirements in a distributed system in which $p \leadsto q$ is a progress requirement and p and q are controlled at different locations: once p becomes true, it is necessary to ensure that q becomes true. Once p and q are both true, q must remain true until p is falsified; otherwise the progress obligation is not discharged.

7.2 Relationship to UNITY Theory

A number of completeness proofs for the UNITY logic have been given in the literature [15, 16, 17, 9]. Althougth the underlying logic described in Section 2 differs in several ways from the UNITY logic, the differences do not affect completeness.

The most obvious difference is the extra inference rule, PSP (which stands for Progress-Safety-Progress). This rule is derived as a metatheorem in the standard theory [7]. Thus, adding it as one of the postulated inference rules does not change the set of leads-to properties that can be derived for a given program from a given set of assumptions. The reason the rule is *postulated* here is that its inductive proof does not go through when properties can be introduced by assumption: without *ensures* as a basis, there is no way to derive any kind of interaction between safety and progress properties.

It should be noted that the additional postulated rule means that inductive proofs of derived rules have to include an additional "step" case besides Disjunction and Transitivity. The derived rules shown in Figure 3 (including the rules used in the proof of Lemma 3) are all provable in this way. However, while the set of *derivations* of leads-to properties of the new theory has not changed (and therefore its soundness and completeness is not affected), it is not immediately clear whether all of the *metatheorems* that were provable for the old theory are provable for the new one, although it seems likely that they are.[4]

Another difference between our theory and UNITY is the explicit treatment of conditional properties and the semantic interpretation of $P \Rightarrow Q$. In UNITY, the assertion "$p \leadsto q$" means what is rendered here as $\vdash p \leadsto q$. Conditional properties are formulated as pairs of the form "Hypothesis: P, Conclusion: Q", where P and Q may be *unless* or *ensures* properties as well as $\mapsto$ properties. The meaning of a specification with hypothesis $x \leadsto y$ and conclusion $p \leadsto q$ is really (in our notation)[5]

$$[\![x \leadsto y]\!] \Rightarrow [\![p \leadsto q]\!]$$

This means that either there exists a behavior that does not satisfy $x \leadsto y$, or every behavior satisfies $p \leadsto q$. This is weaker than our interpretation, which is

$$[\![x \leadsto y \Rightarrow p \leadsto q]\!]$$

The latter specification says something about *every* behavior, namely that it either does not satisfy $x \leadsto y$ or it satisfies $p \leadsto q$.

7.3 Compositionality and Progress

A *compositional* theory is one allowing both programs and proofs to be composed. In such a system, properties of programs are preserved under composition with other programs; this permits proof structure to reflect the modular structure of a composite program. The primary motivation for introducing conditional properties is so that when a program satisfying, say, $P \Rightarrow Q$ is composed with one satisfying P, we can immediately conclude that Q holds in the composite program, without having to prove it from scratch. The $\vee$-monotonicity rule, proved in Section 5, justifies such a conclusion in a compositional theory.

Obviously the results presented here are of greatest utility in the context of a compositional theory. The extended theory was developed for use with such a theory of module specifications [3, 5]. Proving conditional progress is tricky in compositional theories, becuase of the

[4] At this writing, the Completion rule [7] has not been proved for the new theory.

[5] We assume here that all properties apply to the same program.

infinitary nature of progress properties: showing that a system does *not* satisfy a progress property requires exhibiting an infinite behavior. When two components interact, and one has a progress specification of the form $P \Rightarrow P' \wedge Q$, while the other has a specification of the form $P' \Rightarrow P \wedge R$, it is not always valid to conclude that the composite satisfies $Q \wedge R$. Abadi and Lamport [2] have given sufficient *semantic* conditions for such a conclusion to be valid. The results of this paper, together with the compositional theory of module specifications presented elsewhere [5, 4] constitute a theory that satisfies the semantic conditions given by Abadi and Lamport.

In the theory of Lam and Shankar, on the other hand, such "circular" dependencies are avoided altogether by requiring the composite to have a certain kind of well-founded structure. This ensures that *some* module satisfies its progress requirements unconditionally; progress can then be proved hierarchically, with each module only depending on those whose progress has already been proved correct. Using the theory presented here, such restrictions would be unecessary.

7.4 Related Work

Various examples of problems exhibiting progress dependencies have appeared in the literature [11, 7, 18]. In some of these, boolean combinations of leads-to properties are used and proved informally.

Tsay and Bagrodia [20] have shown equivalence between provability of UNITY conditional properties of the form "Hypothesis: $true \leadsto p$ Conclusion: $true \leadsto q$" and formulas of the form "$\Box\Diamond p \Rightarrow \Box\Diamond q$" in Manna and Pnueli's temporal logic [14]. They also give a relatively complete inference rule for properties of this particular, which may be used to specify strong fairness.

Lamport's Temporal Logic of Actions (TLA) [13] combines the single temporal operator "$\Box$" with two-place predicates (relations) on states to obtain a theory in which programs and abstract specifications are both defined using the same language, and satisfaction is simply logical implication. Like UNITY and the theory presented here, TLA is a restricted form of temporal logic. However, it admits arbitrary temporal formulas constructed using the "$\Box$" operator (and its dual, "$\Diamond$"), and is thus more powerful than the theory presented here. It also has a more complex proof system: an axiomatization of the propositional fragment of TLA by Abadi [1] has 14 axioms and three inference rules, in addition to the underlying propositional calculus. The comparable portion of the formalism presented here (assuming UNITY as the underlying program logic layer) has three axioms (the definitions of wp, *unless* and *ensures*) and seven inference rules, in addition to the predicate logic.

7.5 Conclusions

The results presented here suggest the following approach to specification of conditional progress properties of protocols:

1. Write down the specification in the form $X \Rightarrow Y$, where X includes all properties expected of the environment, and Y includes all properties required of the protocol.

2. Try to massage the specification into an equivalent conjunction of provable form leads-to properties and implications. Whether this is possible depends on the scopes of the quantifiers appearing in the specification.

3. If the result is in provable form except for enclosing existential quantifications, replace existentially-quantified variables with state functions to the extent possible.

This approach is most likely to be useful for specifying and verifying *algorithms*, as opposed *implementations*, e.g. of "real" protocols. The latter typically feature timeouts and fixed bounds on the number of attempts before giving up on delivering a message. In designing and verifying an algorithm, however, it is better to abstract from such details and *characterize the conditions needed for correctness as precisely and generally as possible.*

References

[1] Martin Abadi. An axiomatization of lamport's temporal logic of actions. Technical Report Research Report 65, DEC Systems Research Center, October 1990.

[2] Martín Abadi and Leslie Lamport. Composing specifications. In *Stepwise Refinement of Distributed Systems (LNCS 430)*. Springer-Verlag, 1990.

[3] Kenneth L. Calvert. Module composition and refinement: Extending the Lam-Shankar theory. Technical Report GIT–CC–91/58, College of Computing, Georgia Institute of Technology, 1991.

[4] Kenneth L. Calvert. *Protocol Conversion and Quotient Problems*. PhD thesis, University of Texas at Austin, May 1991.

[5] Kenneth L. Calvert. Module composition and refinement with applications to protocol conversion. In *Proceedings XII Symposium on Protocol Specification, Testing, and Verification, Orlando, Florida*. North-Holland, June 1992.

[6] Kenneth L. Calvert. Specifying progress properties with leads-to. Technical Report GIT–CC–92/59, College of Computing, Georgia Institute of Technology, December 1992. available via anonymous FTP from ftp.cc.gatech.edu.

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

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

[9] Rob Gerth and Amir Pnueli. Rooting UNITY. In *Proceedings of the Fifth International Workshop on Software Specification and Design, Pittsburgh*, May 1989.

[10] Geoffrey Hunter. *Metalogic*. University of California Press, 1971.

[11] Simon S. Lam and A. Udaya Shankar. Specifying modules to satisfy interfaces: A state transition system approach. Technical Report TR88-30, University of Texas at Austin, Department of Computer Sciences, August 1988. (revised September 1990).

[12] Simon S. Lam and A. Udaya Shankar. A relational notation for state transition systems. *IEEE Transactions on Software Engineering*, 16(7):755–775, July 1990.

[13] Leslie Lamport. A temporal logic of actions. Technical Report Research Report 57, DEC Systems Research Center, April 1990.

[14] Zohar Manna and Amir Pnueli. *Temporal Logic of Reactive and Concurrent Systems.* Springer-Verlag, 1992.

[15] Jan Pachl. A simple proof of a completeness result for *leads-to* in the UNITY logic. *Information Processing Letters*, 41:35–38, 1992.

[16] J. R. Rao. On a notion of completeness for the leads-to. Notes on UNITY: 24-90, July 1991.

[17] Beverly A. Sanders. Eliminating the substitution axiom from UNITY logic. Technical Report 128, Eidgenössische Technische Hochschule Zürich, Institut für Computersysteme, May 1990.

[18] Beverly A. Sanders. Stepwise refinement of mixed specifications of concurrent programs. In *Proceedings of IFIP TC2/WG2.3 Working Conference on Programming Concepts and Methods, Sea of Gallilee, Isreal, April 1990.* Elsevier Science Publishers B.V., 1990.

[19] A. Udaya Shankar and Simon S. Lam. Time-dependent distributed systems: Proving safety, liveness, and real-time properties. *Distributed Computing*, 1987(2):61–78, 1987.

[20] Yih-Kuen Tsay and Rajive L. Bagrodia. Deducing fairness properties for UNITY programs. unpublished manuscript, 1993.

Appendix: Proof of Lemma 2

Lemma 2 asserts that for any countable set of properties $x_m \leadsto y_m$, the assumptions (7)–(9) imply the existence of a metric function M (mapping $(p \wedge \neg q)$-states to ordinal numbers) and a family of predicates u_m^α (for each index m and ordinal α) satisfying properties (10)–(12). For convenience, the relevant assumptions about the underlying proof theory are restated here, along with the main hypotheses of the lemma.

A0 The concatenation of a fair segment and a segment is a fair segment.

A1 The set of behaviors of the program contains exactly the infinite sequences that can be constructed inductively by beginning with an initial state and iteratively extending the sequence by concatenating a fair segment to it.

A2 The set of states reachable from any state is countable.

The hypotheses of the Lemma are:

$$\langle \forall\, m :: [\![x_m \ \mathit{unless}\ y_m]\!] \rangle \tag{7}$$

$$[\![\langle \forall\, m :: x_m \leadsto y_m \rangle \Rightarrow p \leadsto q]\!] \tag{8}$$

$$[\![p \ \mathit{unless}\ q]\!] \tag{9}$$

where m ranges over some countable set. Our proof obligations are:

$$[y_m \wedge u_m^\alpha \;\Rightarrow\; M{<}\alpha \vee q] \tag{10}$$

$$[\![u_m^\alpha \;\mathit{unless}\; M{<}\alpha \vee q]\!] \tag{11}$$

$$[\![p \wedge M{=}\alpha \leadsto \langle \exists\, m :: x_m \wedge u_m^\alpha \rangle \vee M{<}\alpha \vee q]\!] \tag{12}$$

The definitions of M and u_m^α will follow from what these hypotheses imply about the structure of the state space itself; in this sense the derivation is "constructive". However, our earlier assumption that we are dealing with mathematical objects rather than textual representations becomes important here: M and u_m^α *may* not be representable as well-formed formulas in any particular language.

To provide some motivation and intuition for the definition of M, consider first the special case of a single hypothesis $x \leadsto y$. In general, if the hypotheses of the Lemma all hold, the program may have behaviors that begin at $(p \wedge \neg q)$-states and do not contain any q-states, and thus do not satisfy $p \leadsto q$. However, we can assert that *any such behavior contains only a finite number of y-states* (otherwise it would satisfy $x \leadsto y$, contradicting (8)). It follows from this that no $(p \wedge y)$-state is fairly reachable from itself except via a q-state, or in other words, *every fair segment that begins and ends with the same $(p \wedge y)$-state contains a q-state.* In fact, we can go further. For a particular $(p \wedge \neg q)$-state $\mathbf{g}$, consider any x-state $\mathbf{h}$ and y-state $\mathbf{k}$ such that (refer to Figure 5):

- $\mathbf{h}$ is fairly reachable from $\mathbf{g}$ via a fair segment containing no q-state
- $\mathbf{k}$ is reachable from $\mathbf{h}$ via a segment containing no q-state.

We can assert that $\mathbf{g}$ *is not reachable from* $\mathbf{k}$ *without passing through a q-state*—otherwise the loop from $\mathbf{g}$ to $\mathbf{k}$ via $\mathbf{h}$ could be concatenated with itself infinitely many times to construct a behavior satisfying $x \leadsto y$ (because y holds infinitely often) but not $p \leadsto q$, violating (8). The point is that, along a path like the one shown in Figure 5, which goes through an x-state and then a y-state, *some states become reachable only via segments containing q-states.* Moreover, from any such $\mathbf{g}$, *some* x-state such as $\mathbf{h}$ will eventually be reached if a q-state is not reached; otherwise the property $x \leadsto y$ will always be discharged, and the behavior will violate (8). In going from the x-state $\mathbf{h}$ to the y-state $\mathbf{k}$, something "decreases", namely the number of $(p \wedge \neg q)$-states reachable via segmenta that do not contain q-states.

Thus we arrive at the intuition behind the metric for the single-hypothesis case: for a given $(p \wedge \neg q)$-state, there exists a (possibly transfinite) "upper bound" on the number of fair segments ending in y-states that can be passed through without encountering a q-state.

For the multiple-hypothesis case, the reasoning is somewhat more complicated, but similar. Again we consider $\mathbf{g}$, and $\mathbf{h}$ fairly reachable from $\mathbf{g}$, except that $\mathbf{h}$ is an x_m-state for some m (and possibly more than one). The key idea is that there must be a *particular* index n such that there is *no* y_n-state $\mathbf{k}_n$, reachable from $\mathbf{h}$, such that $\mathbf{g}$ is reachable from $\mathbf{k}_n$. Otherwise, for *every* m, it is possible to go (fairly) back and forth between $\mathbf{g}$ and some y_m-state infinitely often without encountering a q-state; in this way we can construct a behavior that satisfies every hypothesis property but not $p \leadsto q$, again violating (8).

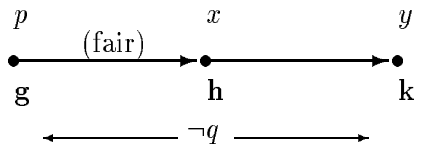

Figure 5: State reachability limitations

The definition of M begins with the introduction of two reachability relations on individual states. The relation $\longrightarrow$ (respectively $\stackrel{\circ}{\longrightarrow}$) denotes reachability (respectively fair reachability) via a segment (fair segment) containing no q-state:

$$\mathbf{g} \longrightarrow \mathbf{h} \quad \stackrel{\text{def}}{=} \quad \mathbf{h} \text{ is reachable from } \mathbf{g} \text{ via a segment containing no } q\text{-state}$$
$$\mathbf{g} \stackrel{\circ}{\longrightarrow} \mathbf{h} \quad \stackrel{\text{def}}{=} \quad \mathbf{h} \text{ is reachable from } \mathbf{g} \text{ via a } \textit{fair} \text{ segment containing no } q\text{-state}$$

Note that $\mathbf{g} \longrightarrow \mathbf{h}$ or $\mathbf{g} \stackrel{\circ}{\longrightarrow} \mathbf{h}$ hold only if $\neg q.\mathbf{g}$ and $\neg q.\mathbf{h}$. Also, because an individual state constitutes a segment, we have

$$\mathbf{g} \longrightarrow \mathbf{g} \quad \equiv \quad \neg q.\mathbf{g} \tag{23}$$

Next we define two other relations in terms of $\longrightarrow$ and $\stackrel{\circ}{\longrightarrow}$. The relation $\stackrel{\circ\star}{\longrightarrow}$ holds between two states when $\stackrel{\circ}{\longrightarrow}$ holds and the latter state satisfies x_m for some m:

$$\mathbf{g} \stackrel{\circ\star}{\longrightarrow} \mathbf{h} \quad \stackrel{\text{def}}{=} \quad \mathbf{g} \stackrel{\circ}{\longrightarrow} \mathbf{h} \wedge \langle \exists\, m :: x_m.\mathbf{h} \rangle$$

The other relation is actually a family of relations, one per index (i.e. one per value in the range of m): $\stackrel{m}{\longrightarrow}$ holds between two states related by $\longrightarrow$ when the latter state satisfies y_m:

$$\mathbf{g} \stackrel{m}{\longrightarrow} \mathbf{h} \quad \stackrel{\text{def}}{=} \quad \mathbf{g} \longrightarrow \mathbf{h} \wedge y_m.\mathbf{h}$$

Next we make the key observation that *states do not become reachable, only unreachable.*

Lemma 7. Let R be any of the relations $\longrightarrow$, $\stackrel{\circ}{\longrightarrow}$, $\stackrel{\circ\star}{\longrightarrow}$, and $\stackrel{m}{\longrightarrow}$ (for $m \in W$). Then for any states $\mathbf{g}$, $\mathbf{h}$, $\mathbf{k}$:

$$\mathbf{g} \longrightarrow \mathbf{h} \wedge \mathbf{h}\, R\, \mathbf{k} \quad \Rightarrow \quad \mathbf{g}\, R\, \mathbf{k}$$

□

Proof. If R is $\longrightarrow$ or $\stackrel{\circ}{\longrightarrow}$, the result is immediate from **A0** and the fact that none of the segments contain q-states. For $\stackrel{\circ\star}{\longrightarrow}$ we observe, for any state $\mathbf{k}$:

$$
\begin{array}{ll}
 & \mathbf{g} \longrightarrow \mathbf{h} \wedge \mathbf{h} \stackrel{\circ\star}{\longrightarrow} \mathbf{k} \\
= & \quad \{ \text{ definition } \} \\
 & \mathbf{g} \longrightarrow \mathbf{h} \wedge \mathbf{h} \stackrel{\circ}{\longrightarrow} \mathbf{k} \wedge \langle \exists\, m :: x_m.\mathbf{k} \rangle \\
\Rightarrow & \quad \{ \text{ result for } \stackrel{\circ}{\longrightarrow}, \text{ already proved } \} \\
 & \mathbf{g} \stackrel{\circ}{\longrightarrow} \mathbf{k} \wedge \langle \exists\, m :: x_m.\mathbf{k} \rangle \\
= & \quad \{ \text{ definition } \} \\
 & \mathbf{g} \stackrel{\circ\star}{\longrightarrow} \mathbf{k}
\end{array}
$$

The proof for $\stackrel{m}{\longrightarrow}$ is similar. □

Lemma 8. Let R be any of the relations $\longrightarrow$, $\stackrel{\circ}{\longrightarrow}$, $\stackrel{\circ\star}{\longrightarrow}$ and $\stackrel{m}{\longrightarrow}$, and let v be a state predicate such that

$$v.\mathbf{g} \equiv \langle \forall\, \mathbf{k} : \mathbf{g}\ R\ \mathbf{k} : \cdots \rangle$$

where "$\cdots$" stands for any term. Then $[\![v \ \textit{unless}\ q]\!]$. □

Proof. We have to show that every transition from a state where $v \wedge \neg q$ holds either preserves v or establishes q. Assume $\mathbf{g}$ and $\mathbf{h}$ are any states such that:

- $(\mathbf{g}, \mathbf{h})$ is a transition
- $v.\mathbf{g}$
- $\neg q.\mathbf{g}$
- $\neg q.\mathbf{h}$

We shall establish $v.\mathbf{h}$.

Because $(\mathbf{g}, \mathbf{h})$ is a transition, there exists a segment with $\mathbf{g}$ and $\mathbf{h}$ as its first and last states, respectively. Since by hypothesis $\neg q$ holds at both $\mathbf{g}$ and $\mathbf{h}$, we have also $\mathbf{g} \longrightarrow \mathbf{h}$. By Lemma 7, we have

$$\langle \forall\, \mathbf{k} :: \mathbf{h}\ R\ \mathbf{k} \Rightarrow \mathbf{g}\ R\ \mathbf{k} \rangle \tag{24}$$

Now we calculate:

$$
\begin{array}{ll}
 & v.\mathbf{g} \\
= & \quad \{ \text{ definition } \} \\
 & \langle \forall\, \mathbf{k} : \mathbf{g}\ R\ \mathbf{k} : \cdots \rangle \\
\Rightarrow & \quad \{ \text{ strengthen the range using (24) } \} \\
 & \langle \forall\, \mathbf{k} : \mathbf{h}\ R\ \mathbf{k} : \cdots \rangle \\
= & \quad \{ \text{ definition } \} \\
 & v.\mathbf{h}
\end{array}
$$

□

Now we can define M as a function from $(p \wedge \neg q)$-states to ordinal numbers. The definition is carefully constructed so that a contradiction can be derived from the assumption that M is *not* defined at some $(p \wedge \neg q)$-state.

$$M.\mathbf{g} \stackrel{\text{def}}{=} \langle \min \alpha : \langle \forall\, \mathbf{h} : \mathbf{g} \stackrel{\circ\star}{\longrightarrow} \mathbf{h} : \langle \exists\, n : x_n.\mathbf{h} : \langle \forall\, \mathbf{k} : \mathbf{h} \stackrel{n}{\longrightarrow} \mathbf{k} : M.\mathbf{k} < \alpha \rangle \rangle \rangle : \alpha \rangle$$

As discussed above, M can be viewed as an upper bound on the number of "applications" of hypothesis properties required to force progress to a q-state.

The following property is immediate from the definition of M:

$$M.\mathbf{g} \leq \alpha \equiv \langle \forall\, \mathbf{h} : \mathbf{g} \xrightarrow{\circ\star} \mathbf{h} : \langle \exists\, n : x_n.\mathbf{h} : \langle \forall\, \mathbf{k} : \mathbf{h} \xrightarrow{n} \mathbf{k} : M.\mathbf{k} < \alpha \rangle \rangle \rangle \tag{25}$$

Clearly, $M.\mathbf{g}$ is defined if the range of the outer quantification is empty, i.e. if there are no states $\mathbf{h}$ such that $\mathbf{g} \xrightarrow{\circ\star} \mathbf{h}$. At such states, it is not possible to reach a state where any hypothesis is undischarged without passing through a q-state, and the value of M is 0. Also, $M.\mathbf{g}$ is defined if, for every $\mathbf{h}$ such that $\mathbf{g} \xrightarrow{\circ\star} \mathbf{h}$, there is some n such that $x_n.\mathbf{h}$ is true and no y_n-state is reachable from $\mathbf{h}$ without passing through a q-state; at such states again M's value is 0. Lemma 9 now establishes the definedness of M at a certain set of states, which includes the above as "base cases"; Lemma 10 then shows that the set includes all $(p \wedge \neg q)$-states.

Lemma 9. M is defined at any $\mathbf{g}$ such that

$$\langle \forall\, \mathbf{h} : \mathbf{g} \xrightarrow{\circ\star} \mathbf{h} : \langle \exists\, n : x_n.\mathbf{h} : \langle \forall\, \mathbf{k} : \mathbf{h} \xrightarrow{n} \mathbf{k} : M.\mathbf{k} \text{ is defined} \rangle \rangle \rangle \tag{26}$$

□

Proof. Let $\mathbf{g}$ be a state satisfying (26). We shall show the existence of an ordinal β such that $M.\mathbf{g} \leq \beta$, i.e. such that the right-hand side of (25) holds with α replaced by β. The existence of an ordinal satisfying some property implies the existence of a *least* ordinal satisfying that property; thus there exists a least ordinal α such that the right-hand side of (25) holds. By definition the value of M is that α.

We begin by defining a function from states to indices (of hypotheses). For any state $\mathbf{h}$ we let $D.\mathbf{h}$ be the least index n such that

$$x_n.\mathbf{h} \wedge \langle \forall\, \mathbf{k} : \mathbf{h} \xrightarrow{n} \mathbf{k} \wedge y_n.\mathbf{k} : M.\mathbf{k} \text{ is defined} \rangle$$

That D is well-defined for each $\mathbf{h}$ such that $\mathbf{g} \xrightarrow{\circ\star} \mathbf{h}$ follows from (26). Now we define one more relation:

$$\mathbf{g} \hookrightarrow \mathbf{k} \stackrel{\text{def}}{=} \langle \exists\, \mathbf{h} : \mathbf{g} \xrightarrow{\circ\star} \mathbf{h} : \mathbf{h} \xrightarrow{D.\mathbf{h}} \mathbf{k} \rangle$$

By (26), M is defined at each state $\mathbf{k}$ such that $\mathbf{g} \hookrightarrow \mathbf{k}$. Also, because $\mathbf{g} \hookrightarrow \mathbf{k}$ implies that $\mathbf{k}$ is reachable from $\mathbf{g}$, and because the number of states reachable from $\mathbf{g}$ is countable (**A2**), the number of states $\mathbf{k}$ such that $\mathbf{g} \hookrightarrow \mathbf{k}$ holds is countable. We may therefore conclude that there exists an ordinal β greater than $M.\mathbf{k}$ at each state $\mathbf{k}$ such that $\mathbf{g} \hookrightarrow \mathbf{k}$. (The sum of any countable set of ordinals is defined and is an ordinal; the successor of that sum exceeds any ordinal in the sum [10].) That is, there exists an ordinal β such that:

$$\begin{array}{ll} & \langle \forall\, \mathbf{k} : \mathbf{g} \hookrightarrow \mathbf{k} : M.\mathbf{k} < \beta \rangle \\ = & \quad \{ \text{ definition } \hookrightarrow \} \\ & \langle \forall\, \mathbf{k} : \langle \exists\, \mathbf{h} : \mathbf{g} \xrightarrow{\circ\star} \mathbf{h} : \mathbf{h} \xrightarrow{D.\mathbf{h}} \mathbf{k} \rangle : M.\mathbf{k} < \beta \rangle \\ = & \quad \{ \text{ predicate calculus } \} \\ & \langle \forall\, \mathbf{h}, \mathbf{k} : \mathbf{g} \xrightarrow{\circ\star} \mathbf{h} \wedge \mathbf{h} \xrightarrow{D.\mathbf{h}} \mathbf{k} : M.\mathbf{k} < \beta \rangle \\ = & \quad \{ \text{ predicate calculus } \} \end{array}$$

$$
\begin{array}{ll}
 & \langle \forall\, \mathbf{h} : \mathbf{g} \xrightarrow{\circ\star} \mathbf{h} : \langle \forall\, \mathbf{k} : \mathbf{h} \xrightarrow{D.\mathbf{h}} \mathbf{k} : M.\mathbf{k} < \beta \rangle \rangle \\
= & \quad \{ \text{ by definition every } \mathbf{h} \text{ in the range satisfies } x_{D.\mathbf{h}} \ \} \\
 & \langle \forall\, \mathbf{h} : \mathbf{g} \xrightarrow{\circ\star} \mathbf{h} : (x_{D.\mathbf{h}}).\mathbf{h} \wedge \langle \forall\, \mathbf{k} : \mathbf{h} \xrightarrow{D.\mathbf{h}} \mathbf{k} : M.\mathbf{k} < \beta \rangle \rangle \\
\Rightarrow & \quad \{ \text{ existential generalization } \} \\
 & \langle \forall\, \mathbf{h} : \mathbf{g} \xrightarrow{\circ\star} \mathbf{h} : \langle \exists\, n :: x_n.\mathbf{h} \wedge \langle \forall\, \mathbf{k} : \mathbf{h} \xrightarrow{n} \mathbf{k} : M.\mathbf{k} < \beta \rangle \rangle \rangle \\
= & \quad \{ \text{ (25) } \} \\
 & M.\mathbf{g} \leq \beta
\end{array}
$$

We have shown that an ordinal α satisfying the right-hand side of (25) exists; by the properties of ordinal numbers, there exists a least such ordinal, which is the value of $M.\mathbf{g}$. □

The foregoing Lemma implies that if M is not defined at some $(p \wedge \neg q)$-state, then there exists a particular state $\mathbf{g}$ that is fairly reachable from that state without passing through a q-state, and from which, for *every* index m, either x_m does not hold at $\mathbf{g}$, or a y_m-state $\mathbf{h}$ is reachable from $\mathbf{g}$ without passing through a q-state such that M is not defined at $\mathbf{h}$. Using this fact, we can construct inductively an infinite behavior satisfying every hypothesis property but not $p \rightsquigarrow q$.

Lemma 10. Under the assumptions of Lemma 2, the value of the state function M as defined above is well-defined at every $(p \wedge \neg q)$-state. □

Proof. Assume $\mathbf{g}$ is a state satisfying $(p \wedge \neg q)$ at which M is not defined. By Lemma 9 and predicate calculus, $\mathbf{g}$ satisfies

$$\langle \exists\, \mathbf{h} : \mathbf{g} \xrightarrow{\circ\star} \mathbf{h} : \langle \forall\, m : x_m.\mathbf{h} : \langle \exists\, \mathbf{k} : \mathbf{h} \xrightarrow{m} \mathbf{k} : M.\mathbf{k} \text{ is undefined} \rangle \rangle \rangle \tag{27}$$

By the definition of $\xrightarrow{\circ\star}$, this implies the existence of a fair segment, containing no q-state, from $\mathbf{g}$ to another state $\mathbf{h}$ which satisfies x_n for some index n. Moreover, using the definition of $\xrightarrow{m}$, $\mathbf{h}$ has the property that, for *each* index m such that x_m holds at $\mathbf{h}$ (i.e., such that $x_m \rightsquigarrow y_m$ is undischarged at $\mathbf{h}$), there exists a segment containing no q-state leading from $\mathbf{h}$ to a y_m-state $\mathbf{k}$, such that $M.\mathbf{k}$ is also undefined.

We shall use this structure to construct (operationally) a fair behavior that contains no q-state, in which, for each m, a y_m-state is reached infinitely often. The behavior so constructed satisfies $x_m \rightsquigarrow y_m$ for each m. However, it does not satisfy $p \rightsquigarrow q$ (the first state is a p-state and it contains no q-states), and so contradicts (8).

The construction proceeds as follows. Let some enumeration of indices be given in which each index occurs infinitely often. (Because by assumption the set of hypotheses is countable, such an enumeration exists.) Let the initial segment consist of the $(p \wedge \neg q)$-state $\mathbf{g}$, at which M is (assumed to be) undefined, let the current position in the enumeration of indices be at the beginning, and perform the following sequence of steps infinitely many times:

1. Concatenate to the existing segment the fair segment whose existence is asserted by (27), leading from $\mathbf{g}$ to a state $\mathbf{h}$ and containing no q-state.

2. Let n be the first index to appear in the enumeration of indices after the current position, such that $x_n.\mathbf{h}$ holds. (Such an n exists, by (27) and the definition of $\xrightarrow{\circ\star}$.) The position of n becomes the new current position in the enumeration.

3. Concatenate to the existing segment the segment whose existence is asserted by (27) and the definition of $\xrightarrow{n}$, leading from $\mathbf{h}$ to a y_n-state $\mathbf{k}$, and containing no q-state. The property $x_n \rightsquigarrow y_n$ is discharged at $\mathbf{k}$.

4. M is undefined at $\mathbf{k}$; let $\mathbf{k}$ become the next $\mathbf{g}$. Note that (27) holds at the new $\mathbf{g}$; iterate.

The construction above yields an infinite concatenation of segments and fair segments which, according to **A0** and **A1**, is a behavior of the program. Also by construction, the behavior satisfies $x_m \rightsquigarrow y_m$ for each index m. (If $x_m \rightsquigarrow y_m$ were not satisfied for some m, the behavior would contain an infinite suffix of $x_m \wedge \neg y_m$ states. However, no predicate $x_m \wedge \neg y_m$ holds continuously in the constructed behavior, because m always comes up again in the enumeration, ensuring that a y_m-state is reached.) Also, every state in the constructed behavior is a $(p \wedge \neg q)$-state.

Thus we have shown the existence of a sequence satisfying $\langle \forall\, m :: x_m \rightsquigarrow y_m \rangle$ but not $p \rightsquigarrow q$, contradicting (8). Because the assumption that M is undefined at a $(p \wedge \neg q)$-state leads to a contradiction, we conclude that M is well-defined at each $(p \wedge \neg q)$-state. □

Having defined M, it remains to define the family of predicates u_m^α and establish the required properties. We define u_m^α for each index m and ordinal α as follows:

$$u_m^\alpha.\mathbf{g} \stackrel{\text{def}}{=} \langle \forall\, \mathbf{h} : \mathbf{g} \xrightarrow{m} \mathbf{h} : M.\mathbf{h} < \alpha \rangle$$

Operationally, u_m^α holds at $\mathbf{g}$ iff α exceeds the value of M at every y_m state reachable from $\mathbf{g}$. To prove the properties of the predicates u_m^α, we need the following Lemma:

Lemma 11. In any behavior, for every $(p \wedge \neg q)$-state $\mathbf{g}$ there exists a later state $\mathbf{h}$ such that either $q.\mathbf{h}$ or $\mathbf{g} \xrightarrow{\circ\star} \mathbf{h}$. □

Proof. Consider a behavior containing a $(p \wedge \neg q)$-state $\mathbf{g}$ that is not followed by any q-state. Because the behavior does not satisfy $p \rightsquigarrow q$, it does not satisfy $x_n \rightsquigarrow y_n$ for some index n, and therefore it contains an infinite suffix in which all states satisfy $\langle \exists\, m :: x_m \rangle$. Moreover, because the behavior can be partitioned into fair segments (**A1**), and any extension of a fair segment is a fair segment (**A0**), the behavior contains a fair segment beginning with $\mathbf{g}$ that ends in a $\langle \exists\, m :: x_m \rangle$-state $\mathbf{h}$. By definition, $\mathbf{g} \xrightarrow{\circ\star} \mathbf{h}$ holds. □

We can now established the required properties of u_m^α:

Proof of (10). We observe, for any state $\mathbf{g}$:

$$\begin{array}{ll} & y_m.\mathbf{g} \wedge u_m^\alpha.\mathbf{g} \\ = & \quad \{ \text{ definition } \} \\ & y_m.\mathbf{g} \wedge \langle \forall\, \mathbf{h} : \mathbf{g} \xrightarrow{m} \mathbf{h} : M.\mathbf{h} < \alpha \rangle \\ = & \quad \{ \text{ definition } \xrightarrow{m} \} \\ & y_m.\mathbf{g} \wedge \langle \forall\, \mathbf{h} : \mathbf{g} \longrightarrow \mathbf{h} \wedge y_m.\mathbf{h} : M.\mathbf{h} < \alpha \rangle \\ \Rightarrow & \quad \{ \text{ instantiate } \mathbf{h} := \mathbf{g} \} \\ & y_m.\mathbf{g} \wedge (\mathbf{g} \longrightarrow \mathbf{g} \wedge y_m.\mathbf{g} \Rightarrow M.\mathbf{g} < \alpha) \\ = & \quad \{ (23) \} \\ & y_m.\mathbf{g} \wedge (\neg q.\mathbf{g} \wedge y_m.\mathbf{g} \Rightarrow M.\mathbf{g} < \alpha) \end{array}$$

$$
\begin{array}{ll}
= & \quad \{ \text{ predicate calculus } \} \\
& y_m.\mathbf{g} \wedge (q.\mathbf{g} \vee \neg y_m.\mathbf{g} \vee M.\mathbf{g} < \alpha) \\
= & \quad \{ \text{ again } \} \\
& (y_m.\mathbf{g} \wedge q.\mathbf{g}) \vee (y_m.\mathbf{g} \wedge M.\mathbf{g} < \alpha) \\
\Rightarrow & \quad \{ \text{ again } \} \\
& q.\mathbf{g} \vee M.\mathbf{g} < \alpha
\end{array}
$$

□

Proof of (11). From the definition of u^α_m and Lemma 8, $[\![u^\alpha_m \ \textit{unless}\ q]\!]$ follows immediately. The required $[\![u^\alpha_m \ \textit{unless}\ M{<}\alpha \vee q]\!]$ then follows by properties of *unless* (consequence weakening). □

Proof of (12). Consider any behavior containing a state $\mathbf{g}$ satisfying $(p \wedge \neg q) \wedge M{=}\alpha$. We have to show that $\mathbf{g}$ is followed by a state satisfying $\langle \exists\, m :: x_m \wedge u^\alpha_m \rangle \vee M{<}\alpha \vee q$.

If no state following $\mathbf{g}$ in the behavior is a q-state, by Lemma 11 there exists a state $\mathbf{h}$ later in the behavior such that $\mathbf{g} \xrightarrow{\circ\star} \mathbf{h}$. Now, because $M.\mathbf{g} = \alpha$, the right-hand side of (25) holds. Instantiating the universal quantification for this $\mathbf{h}$, we observe:

$$
\begin{array}{ll}
& \langle \exists\, n : x_n.\mathbf{h} : \langle \forall\, \mathbf{k} : \mathbf{h} \xrightarrow{n} \mathbf{k} : M.\mathbf{k} < \alpha \rangle \rangle \\
= & \quad \{ \text{ definition of } u^\alpha_m \ \} \\
& \langle \exists\, n : x_n.\mathbf{h} : u^\alpha_m.\mathbf{h} \rangle \\
= & \quad \{ \text{ predicate calculus } \} \\
& \langle \exists\, n : x_m : u^\alpha_m \rangle.\mathbf{h}
\end{array}
$$

Thus we have shown that every state satisfying $p \wedge \neg q \wedge M = \alpha$ is followed by a state satisfying $q \vee \langle \exists\, n : x_m : u^\alpha_m \rangle$, from which the desired property (12) follows. □

This concludes the proof of the existence of M and u^α_m with the properties claimed in Lemma 2.