Sh:134

On the Temporal Analysis of Fairness

by

Dov Gabbay(*, Amir Pnueli(**, Saharon Shelah(***, Jonathan Stavi(*

(* Bar Ilan University, Ramat-Gan, Israel

(** Tel Aviv University, Israel

(*** The Hebrew University, Jerusalem, Israel.

Abstract

The use of the temporal logic formalism for program reasoning is reviewed. Several aspects of responsiveness and fairness are analyzed, leading to the need for an additional temporal operator: the 'until' operator -U. Some general questions involving the 'until' operator are then discussed. It is shown that with the addition of this operator the temporal language becomes expressively complete. Then, two deductive systems DX and DUX are proved to be complete for the languages without and with the new operator respectively.

1. Introduction

The formalism of temporal Logic has been suggested as a most appropriate tool, for reasoning about programs and their executions. Originally, temporal Logic has been designed in order to analyze and reason about time sequences in general. Formalizing the possible variations in time of a varying (dynamic) situation, we consider time sequences $s_0, s_1, \ldots$, where each s_i is a state giving a full description of the situation at instant $i = 0,1,\ldots$. Temporal Logic introduces a clear distinction between variability within a state, which is described using classic connectives and quantifiers, and the variability over time, moving from one state to another. For changes over time, special temporal operators are introduced. The Temporal operators that we will consider in this paper are given below with their interpretation for an arbitrary time sequence:

$\sigma = s_0, s_1, \ldots$.

The truth value of a classical (non-temporal) formula w at instant i is found by evaluating w on s_i.

F is the existential future operator. Fw is true at an instant i if there exists some future instant j, $j>i$, such that w is true at the instant j.

G is the universal future operator. Gw is true at an instant i if for all future instants j, $j>i$, w holds at j.

X is the next instant operator. Xw is true at i if w is true at $i+1$.

U is the until operator. w_1Uw_2 is true at i if there exists a future instant j, $j>i$ such that w_2 is true at j and for all instants k in between, i.e., $i<k<j$, w_1 is true at k. Thus w_1Uw_2 means that w_1 will be true until the next occurrence of w_2. By our definition w_1Uw_2 implies Fw_2.

To write the above truth definition formally we consider a language $L(F,X,U)$

with atomic sentences $q_0, q_1, q_2, \ldots$ the classical connectives $\sim, \wedge, \vee, \supset$ and the tense connectives F,X,U. We let $M = (M,<,Q_0,Q_1,Q_2,\ldots)$ be a model, where $(M,<)$ is a linearly ordered set and $Q_i \subseteq M$. For each atom q_i of the language, Q_i is considered the truth-set of q_i. We define the notion $\|w\|^M_m$, for $m \in M$ by induction on the formula w. ($\|w\|^M_m$ is the truth value of w at m in the model M). M is usually omitted and we write $\|w\|_m$

(1) For $w = q_i$ atomic, $\|w\|_m$ = true iff $m \in Q_i$.

(2) $\|\sim w\|_m$ = true iff $\|w\|_m$ = false.
$\|w_1 \supset w_2\|_m$ = true iff $\|w_1\|_m$ = false or $\|w_2\|_m$ = true

(3) $\|Xw\|_m$ = true in case m has an immediate successor $m' \in M$ and $\|w\|_{m'}$ = true

(4) $\|Fw\|_m$ = true iff $\exists m_1 < m$ s.t. $\|w\|_{m_1}$ = true

(5) $\|w_1 U w_2\|_m$ = true if $\exists m_2 > m$ such that $\|w_2\|_{m_2}$ = true and for all m_1, $m < m_1 < m_2 \Rightarrow \|w_1\|_{m_1}$ = true.

M is called an ω-model when its "time structure" $(M,<)$ consists of (set ω of) the natural numbers with their usual ordering. A formula w is said to be ω-valid when $\|w\|_n$ = true for all ω-models M and all $n \in \omega$.

We denote by L(U) the sublanguage obtained from L(X,F,U) by omitting X,F. This involves no loss of expressive power since

$$Fw \equiv (\underline{\text{true}}\ U\ w)$$
$$Xw \equiv (\underline{\text{false}}\ U\ w).$$

Gw is defined to be the formula $\sim F \sim W$.

The application of temporal Logic to Program Reasoning is based on the view of programs as generators of execution sequences. If we allow nondeterministic programs, then each input to a program generates a set of possible execution sequences. Each execution sequence is a sequence $\sigma = s_0, s_1, \ldots$ of program states, such that s_0 is the initial state containing the input and each s_{i+1} is derived from s_i by executing the appropriate program instruction. By including the program location as part of the state, the appropriate instruction (or set of possible instructions in the nondeterministic case) is uniquely identifiable. Obviously, by stating properties which hold for all the execution sequences of a program, we are stating properties of the generating program.

To illustrate this point let the program be labeled by labels $\ell_0, \ldots \ell_e$, ℓ_0 being the entry point and ℓ_e the exit point. Let us select a set of propositions $at\ell_0, \ldots at\ell_e$ such that $at\ell_i$ is true in a program state s iff when in state s the program is about to execute the instruction at location ℓ_i. The following temporal formula expresses then the property of <u>partial correctness</u> with respect to an input-output specification (φ,ψ):

$$(at\ell_0 \wedge \bar{y} = \xi \wedge \varphi(\xi)) \supset G(at\ell_e \supset \psi(\xi,\bar{y}))$$

Interpreting this formula in state s_0 of an execution sequence $s_0, s_1, \ldots$ it reads:

If at s_0 we are at ℓ_0 with the program variables $\bar{y}$ equal to an input ξ

satisfying $\varphi(\bar{\xi})$, then it is invariably true that whenever we reach the termination point ℓ_e, ψ will hold between the input values $\bar{\xi}$ and the current value of the program variables. Note that this guarantees correctness provided the program terminates, but does not guarantee termination itself.

Similarly, to express total correctness we write:

$$(at\ell_0 \wedge \bar{y}=\bar{\xi} \wedge \varphi(\bar{\xi})) \supset F(at\ell_e \wedge \psi(\bar{\xi},\bar{y})).$$

This guarantees that if we satisfy the proper initial conditions then we will eventually get to a state at which both $at\ell_e$ and ψ are true, hence to correct termination.

This general idea of expressing and proving properties of programs, by temporal reasoning on the set of their execution sequences was pursued in greater detail in several works, ([P1], [P2], [MP], [KR]). It has been shown there that the language L(X,F), consisting of the operators X and F (and hence also G) is a powerful tool for formalizing many important properties of programs. Its main strength and advantage lie in the treatment of properties which cannot be expressed when viewing programs as functions or relations. These properties are of course essential in discussing operating systems, concurrent programs and nonterminating (cyclic) programs in general. Classifying properties according to the form of the temporal formula (and hence the applicable proof method) expressing them, we arrive at the following simplest classes:

Invariances - Properties expressible by a formula of the form: $p \supset Gq$. The class of invariances contains the properties of Safety, Partial Correctness, Mutual Exclusion, Clean Performance, Deadlock Freedom.

Eventualities - Properties expressible by a formula of the form $p \supset Fq$. It includes among others the properties of: Liveness, Total Correctness, Accessibility, Livelock Freedom, etc.

The language was also shown to be adequate for the detailed description of the execution of a concurrent program model. This enables the definition of the semantics of a concurrent system and a derivation of proof principles for proving its properties. ([P2]).

However, investigating more complicated properties, in particular these involving fairness, led us to the realization that the L(X,F) language is not expressively complete. By this we mean that there exist some important properties which cannot be expressed in such a limited system. The simplest one is that of the 'until' condition itself, p U q, which states that q is bound to happen, and until it happens, p will hold. As a result, the language that we present here is an augmented language L(U) = L(X,F,U), which is based on the operators: F,X,U (and G). The question naturally arises, whether we will not eventually encounter some further properties which could not be expressed even in this extended system. We provide a negative answer to this question by showing that the L(U) language is expressively complete. We also provide an axiomatic system DUX which is shown to be deductively complete for L(U).

We will emphasize the importance of the operator 'U' by listing first some properties, involving responsiveness which can be expressed without the 'U', and then some properties which must use the 'until' operator for their expression. We will list first different shades of responsiveness according to the strength of their commitment:

a) Response to Insistence

The weakest form of responsiveness states that a permanent holding of a condition or a request p will eventually force a response q.

This can be written as: $Gp \supset Fq$, or if stated over all future behaviors $G(Gp \supset Fq)$.

Sometimes, the response q frees the requester from being frozen at the requesting state. In this case once q becomes true, p ceases to hold, apparently falsifying the hypothesis Gp. This difficulty is only interpretational and we can write instead the logically equivalent condition $\sim G(p \wedge \sim q)$, namely, it is impossible for p to be frozen indefinitely and q never to realize.

To illustrate the utilization of such a statement, consider a component of a process which is busy waiting for a condition (say, $x>0$) which presumably some other process is expected to generate.

m : <u>if</u> x = 0 <u>then</u> <u>go</u> <u>to</u> m

We use the proposition 'atm' which is true at each time instant in which the execution of our process is at m, namely about to execute m. Then, the only statement we can make about this situation and its resolution is:

$$\sim G(atm \wedge x>0)$$

i.e. it is impossible for the process to be stuck at m while x is continuously positive. It implies that the scheduler will eventually schedule this process which will find x positive and proceed beyond m.

This is also the weakest definition of a semaphore's behavior, as well as the minimal fairness requirement from any scheduler. It requires that any process will eventually be scheduled for execution. Note that for semaphores this allows infinite overtaking.

b) Response to Persistence

A stronger requirement and the one which most suits a semaphore's behavior is that the infinitely repeating occurrence of the condition p will eventually cause q. We do not require p to hold continuously, but only to be true infinitely often. The statement of this eventuality is:

$$GFp \supset Fq$$

or if required continuously: $G(GFp \supset Fq)$.

Similarly to the case above the first form is logically equivalent to $\sim G(Fp \wedge \sim q)$ which we will sometimes prefer.

Consider a semaphore instruction:

$$m : P(x)$$

The proper statement of its behavior is:

$$\sim G(F(x>0) \wedge at\,m)$$

This states that no process can be waiting indefinitely at m for a semaphore entry while the semaphore's variable turns true (positive) infinitely often. Note that to ensure this it is not sufficient to guarantee that this process will be scheduled infinitely often. Because by some highly improbable run of bad luck all these scheduling instances could exactly coincide with the instances in which $x=0$, and the process will never proceed. The same criterion should also apply to the fairness of conditional critical section instructions:

m : <u>with</u> r <u>when</u> B <u>do</u> S.

Here we should also require:

$$\sim G(F(r>0 \wedge B) \wedge at\,m)$$

i.e. it is impossible to remain stuck forever at m while states in which both B hold and the resource (r) is free, repeat infinitely often. This can be implemented by means of semaphores and a queue, or using an unbounded counter ([L1]). It cannot be done with semaphores alone (see [GR] and [KR] for a discussion).

A weaker interpretation of the conditional critical section is also possible:

$$\sim G(F(r>0) \wedge at\,m \wedge B)$$

This one guarantees admission only if B is permanently held true from a certain point on, while r becomes free (positive) infinitely often. The implementation of this weaker construct by semaphores is easy.

c) <u>Response to an Impulse</u>

The strongest type of responsiveness is the one in which a single occurrence of p guarantees q. This is written as: $p \supset Fq$ (more generally $G(P \supset Fq)$). It is a natural and useful expression for the discussion of total correctness, 'sometimes' reasonings, and other temporal causalities of <u>internal</u> events. It is too strong, on the other hand, for the expression of responses to external signals and conditions. This is so because it is seldom the case that a system is so attentive that it could always detect a single occurrence of an incoming signal which does not repeat.

Having formulated the different types of responsiveness, we can augment them with requirements for fairness. While managing quite well in cases a) - c) with just the F,G operators, we must introduce now the 'until' operator U.

Consider the following situations:

d) <u>Absence of Unsolicited Responses</u>

We may wish to complement the statement that p will eventually cause q ($p \supset Fq$), by saying that on the other hand, q will never happen unless preceded by a p. We may wish to state for example that a resource allocation system will not grant a resource to somebody who did not request it. Ignoring boundary effects (i.e. p and q happening simultaneously), this can be expressed as:

$$Fq \supset (\sim q \, U \, p)$$

i.e. that if q is going to happen at all, it cannot happen until p happened first (or concurrently).

e) <u>Strict Fairness</u>

Suppose there are two requests p_1 and p_2 and two corresponding responses q_1 and q_2. We may wish to impose strong FIFO discipline on the responding agent and state that if p_1 preceded p_2 then q_1 will precede q_2.

For this it is convenient to express the fact that starting from the present, the first occurrence of p_2 must be preceded by an occurrence of p_1. In fact this is exactly the expression used in d) above and we define generally:

$$Pr(p_1, p_2) \equiv Fp_2 \supset (\sim p_2 \, U \, p_1)$$

The strict responsiveness discipline (FIFO) can now be written as:

$$Pr(p_1, p_2) \supset Pr(q_1, q_2)$$

Consider for example two competing semaphore instructions:

```
m : P(x)      n : P(x)

m':           n':
```

A FIFO implementation of semaphores would require:

$$Pr(atm, atn) \supset Pr(atm', atn').$$

One could use this as a basic axiom for the behavior of semaphores under a given (FIFO) implementation and then deduce from it a global FIFO behavior of larger parts of the program. An example of statements and reasoning about strict fairness exists in [L1].

d) and e) are really only a first approximation to the properties we had in mind. For example Pr(p,q) strictly ensures only that the <u>first</u> occurrence of q is preceded by an occurrence of p. What exactly we would like to have is: "Every q is preceded by a p which happened after the last q, if any". Thus we cannot be satisfied with a single p succeeded by many q's, even though this situation formally satisfies the requirement: "every q is preceded by a p". The property described above, i.e. interleaving of at least one p between consecutive q's can be described as:

$$Pr(p,q) \wedge G(q \supset Pr(p,q))$$

It states that picking as a reference point either the initial state, or any instant in which q holds, ensures that the first instance of q <u>after</u> the reference point is preceded by an instance of p occurring between the reference point and the instance of q.

In view of the demonstrated importance of the operator 'U' we will now address ourselves to the more theoretical questions of its contribution to the expressive power of the language, and the appropriate deductive system for reasoning about L(U).

Sepcifically, we pose the following questions:

1) Is L(F,X,U) expressively stronger than L(F,X), or could we perhaps express U in terms of F,X?

2) Is $L(U)=L(F,X,U)$ expressively complete, or are there some other reasonable properties which cannot be expressed even in $L(U)$?

3) What is an appropriate deductive system for $L(F,X)$?

4) What is an appropriate deductive system for $L(F,X,U)$?

Question no. 1 was answered by Kamp ([KP1], Ch. IV Theorem 3) who showed that U cannot be expressed in terms of an even richer language than $L(F,X)$. Kamp was actually the first to deal with these questions, realizing that $L(F,X)$ is expressively incomplete, and to introduce the 'until' operator as well as its past counterpart, the 'since' operator. He also showed that the Language with both the 'since' and 'until' operators is expressively complete for expressing situations extending over the past, present and future. The main new point in our work, as compared to his is, in addition to a simpler proof, the isolation of the future fragment and showing its independent completeness. Since a dual proof applies to the past fragment, this provides an alternate derivation of Kamp's original result, and a separation property of the mixed Language $L(U,S)$, into $L(U)$ and $L(S)$. Also our methods have led to a solution of a problem left open by Kamp, that of obtaining a natural temporal language which is expressively complete for arbitrary linearly ordered time structures (see end of section 2.). The affirmative solution to question 2) is presented in section 2.

Deductive systems for both $L(X,F)$ and $L(X,F,U)$ have been suggested in [P2] and [MY] respectively. These system, DX for $L(X,F)$ and DUX for $L(X,F,U)$, have been claimed complete in these works but proofs were not presented. In section 3 we provide an outline of a proof which applies to both systems and establishes their completeness.

2. L(U) is expressively complete

We begin by defining our notion of expressive completeness. The standard of comparison will be the first-order language L_1 of linear order. L_1 is a first-order language with equality, containing the binary predicate $<$ and unary predicates $q_0,q_1,q_2,\ldots$, which we identify with the atomic sentences of the propositional temporal language $L(U)$ described before. Note that L_1 and $L(U)$ are interpreted in the same kind of models, namely in structures $M = \langle M,<,Q_0,Q_1,Q_2,\ldots\rangle$ where $<$ is a linear ordering of M and $Q_i \subseteq M$ for $i \in \omega$. The model M is said to be <u>complete</u> when every non-empty bounded subset of M has a least upper bound and a greatest lower bound (in the ordering $<$); <u>discrete</u> when every element $m \in M$ which is not maximal in $\langle M,<\rangle$ has an immediate successor, and every non-minimal element has an immediate predecessor; an <u>ω-model</u> when $\langle M,<\rangle$ is the set of natural numbers with its usual ordering. Every ω-model, as well as a model ordered like the integers, is both discrete and complete. The ordered set of real numbers is complete but not discrete, and the ordered set of rational numbers is incomplete.

There is a natural translation of $L(\bar{U})$-formulas θ into L_1-formulas $\theta^*(x)$ such that for every model $M = \langle M,<,\ldots\rangle$ and $m \in M$, θ^* is satisfied in M by the element m iff θ is true in M at (the instant) m. We express this by saying that

$\theta^* \equiv \theta$ in M at m for all models M and all $m \in M$. Expressive completeness means the existence of a translation in the other direction, and is given for ω-models in the following theorem.

Theorem 2.1. For every formula $\varphi(x)$ of L_1 there is an $L(U)$ - formula θ such that $\varphi \equiv \theta$ in every ω-model at the point 0. [The restriction to point 0 is necessary because the truth value of any $\theta \in L(U)$ in M at m depends only on what happens in M at and after m, not before m, whereas L_1-formulas may refer also to the past.]

This theorem is a special case of the following one, in which the restriction to $m = 0$ is removed but we assume that $\varphi(x)$ has a special form. By a future formula of L_1 we mean a formula $\varphi(x)$ in which all quantifiers are restricted to $[x,\infty)$, i.e. are of the form $(\forall y \geqslant x)$ or $(\exists y \geqslant x)$. Such formulas refer only to the present and future of the moment x.

Theorem 2.2. For every future formula $\varphi(x)$ of L_1 we can find an $L(U)$-formula θ such that $\varphi \equiv \theta$ in every discrete complete model at every point.

This is closely related to the main result of [Kl], which asserts that the language $L(S,U)$ with Since and Until is expressively complete, with respect to the full language L_1, over complete models. Our approach to Theorem 2.2 can easily be adapted to give a simpler proof of Kamp's theorem, which in fact follows from Theorem 2.2 and its dual for the case of discrete complete ordering. Conversely, Theorem 2.2 follows from Kamp's theorem together with the following:

Theorem 2.3. Every $L(S,U)$ formula is equivalent, over discrete complete models, to a Boolean combination of $L(S)$- and $L(U)$ formulas.

In a sense 2.2 is equivalent to the conjunction of 2.3 and Kamp's result. In [G2], Theorem 2.3 is proved by a direct syntactical argument and shown to imply Kamp's theorem (and hence Theorem 2.2).

The proof of 2.2 presented here follows [SS].

Proof of 2.2 (outline). First we need a normal form for L_1-formulas with any number of free variables, in order to restrict later inductions to formulas $\varphi(x,y)$ which refer only to what happens between x and y (allowing $y=\infty$ or $x=-\infty$, this includes also formulas about the future of x or the past of y). For every variable z let $At(z)$ be the set of all formulas of the form $\pm q_i(z)$ (i.e. $q_i(z)$ or $\sim q_i(z)$), $i \in \omega$. If x and y are distinct variables define the set $Bet_n(x,y)$ of formulas by induction on n as follows:

$Bet_0(x,y) = \emptyset$; $Bet_{n+1}(x,y) = \{\pm\exists z(x<z<y \wedge \varphi_1 \wedge \ldots \wedge \varphi_m) \mid m \in \omega$ and for each i, $\varphi_i \in Bet_n(x,z) \cup At(z) \cup Bet_n(z,y)\}$.

Let $Bet(x,y) = \bigcup_{n\in\omega} Bet_n(x,y)$;

$Aft(x) = Bet(x,\infty)$; $Bef(x) = Bet(-\infty,x)$.

Call a formula $\psi(x_1,\ldots,x_k)$ special on $x_1,\ldots,x_k$ when it has the form $\pi \wedge \psi_1 \wedge \ldots \wedge \psi_m$, where π determines the relative order of $x_1,\ldots,x_k$ (say $x_1<\ldots<x_k$), $m \in \omega$ and each ψ_i is in the set $Bef(x_1) \cup At(x_1) \cup Bet(x_1,x_2) \cup \ldots \cup At(x_k) \cup Aft(x_k)$.

Lemma 1. Each L_1-formula $\varphi(x_1,\ldots,x_k)$ is equivalent to a finite disjunction of special formulas on $x_1,\ldots,x_k$.

This is proved by induction on φ and is essentially [Kl, Ch. II, Theorem 1]. The next task is to analyze the expressive power of a formula in $Bet(x,y)$, where x or y may also be $-\infty$ or $+\infty$. It is at this point that our proof begins to diverge from Kamp's. We define the notion of a Decomposition Formula. This is a formula $\delta(x,y)$ of the form:

$$x<y \wedge \exists z_0,\ldots,z_n[x=z_0<z_1<\ldots<z_n=y] \wedge \bigwedge_{i=1}^{n-1} \theta_i(z_i) \wedge \bigwedge_{i=1}^{n} \forall u(z_{i-1}<u<z_i \supset \varphi_i(u)]$$

where each θ_i, φ_i is a translation into L_1 of an $L(U)$ formula. If $y = \infty$ we allow additional conjuncts of the form

$$\forall u(\exists v>u)\psi_j(v)$$

where each ψ_j is an L_1-translation too.

Main Lemma. Every formula in Bet(x,y) is equivalent to a disjunction of decomposition formulas.

Applying this lemma to a Bet(x,∞) formula yields theorem 2.1 since a dcomposition of (x,∞) is easily translatable into an L(U) formula about x.

The Lemma itself is proved by induction on n for formulas in $\mathrm{Bet}_n(x,y)$. The difficult part involves negation and is overcome by:

Prop. 2.4. A negation of a decomposition formula is equivalent to a finite disjunction of decomposition formulas.

In order to prove this proposition we introduce the concept of event expressions. These are terms of an auxiliary language which describe first or last occurrences of events given by a temporal formula. A typical event expression may be:

"The last q occurrence before the first p occurrence in the interval (x,y)".

A main lemma is the proof states that for each decomposition formula we can find a finite set of event expressions, such that their relative order determines the truth value of the decomposition formula. Conversely, every statement of the relative order of some event expressions is equivalent to a disjunction of decomposition formulas. Since a finite set of event-expressions has a finite number of possible order arrangements, and any interval in a given model realizes exactly one of these arrangements, the negation of some of them is equivalent to the disjunction of the rest. This leads to proposition 2.4.

This proof can be modified to apply to an arbitrary complete linear order (not isomorphic to natural numbers). We can then obtain an alternate proof of Kamp's result about expressive completeness of U and S. Moreover, we can add to U, S two more connectives U' and S' handling gaps and obtain the following theorem:

Theorem 2.5 Every formula $\varphi(x) \in L$, can be translated into a formula θ of L(S,U,S',U') such that $\varphi \equiv \theta$ in every linearly ordered model at any point.

3. DX and DUX are Deductively Complete

In this section we present two axiomatic systems, DX for L(F,G,X) and DUX for L(F,G,X,U) and prove their deductive completeness, i.e., every temporal formula of the language which is valid in all ω-models is provable in the appropriate system. Validity in this section should be taken as validity in all ω models.

The axiomatic system can be taken as follows:

Axioms: Take G,X,U as primitive and define $Fw = \sim G(\sim w)$.

A1. $\vdash G(p\supset q)\supset(Gp\supset Gq)$

A2. $\vdash X(\sim p) \equiv \sim Xp$

A3. $\vdash X(p\supset q)\supset(Xp\supset Xq)$

A4. $\vdash Gp\supset Xp\wedge XGp$

A5. $\vdash G(p\supset Xp)\supset(Xp\supset Gp)$

U1. $\vdash pUq\supset Fq$

U2. $\vdash pUq \equiv Xq\vee(Xp\wedge X(pUq))$

Rules of Inference.

R1. If w is a tautology then $\vdash w$.

R2. If $\vdash w_1\supset w_2$ and $\vdash w_1$ then $\vdash w_2$ (Modus Ponens)

R3. If $\vdash w$ then $\vdash GW$ (Generalization)

The system consisting of A1-A5, R1-R3 is the system DX (I), while the full system (adding U1, U2) is the system DUX(II). The semantical interpretation is the one given in the introduction.

Theorem 3.1 (Soundness)

(1) If $\vdash_I w$ then w is valid.

(2) If $\vdash_{II} w$ then w is valid

(For w in the two appropriate languages: L(F,G,X) and L(F,G,X,U)).

Theorem 3.2 (Completeness)

(1) If $w \in L(F,G,X)$ is valid then $\vdash_I w$

(2) If $w \in L(F,G,X,U)$ is valid then $\vdash_{II} w$.

Proof:

Let w_0 be a $\vdash$ consistent wff. We will construct a finitely representable ω-model satisfying w_0.

The proof proceeds through several

major steps: We construct first a model S for w_0 whose elements, Δ are each a maximal consistent theory (a maximal set of formulas which are consistent relative to I or II). S is not necessarily an ω model, but it satisfies w_0.

Next we take a finite set of formulas Γ, which contains w_0 and all of its subformulas and is closed under some appropriate requirements (e.g. if $Fw\in\Gamma$, then also $Xw\in\Gamma$, $XFw\in\Gamma$).

We construct a finite model $\bar{S}$ obtained by identifying states (points) of S which agree over Γ. Up to this point the process is very similar to the proof of completeness for Propositional Dynamic Logic ([G1], see also [FL]).

The problem with $\bar{S}$ is that a state may have more then one successor. We thread through $\bar{S}$ an infinite path which has the property that it traverses infinitely often every node of one of the strongly connected components of $\bar{S}$. In other words, if a node $n\in\bar{S}$ appears infinitely often in the path then so does each of its $\bar{S}$ successors. This ultimately periodic path serves as our final model.

A more detailed description follows:

Let S be the set of all maximal consistent theories of the system. A standard argument shows that for some Δ_0 in S, $w_0\in\Delta_0$. Define the following relations $<$ and $+$ on S.

$$\Delta^+ = \{w \mid Xw\in\Delta\}$$

$$\Delta<\Theta \text{ iff for all } w,\ Gw\in\Delta\rightarrow w\in\Theta.$$

It can be shown, using the axioms and theorems that:

$$\Delta^+\in S \text{ for every } \Delta\in S \text{ and}$$

$$\forall x,y,z\in S\ (x<y\wedge x<z\wedge y\neq z\rightarrow y<z\vee z<y)$$

If $Fw\in\Delta$ then for some Θ, $\Delta<\Theta$ and $w\in\Theta$. Thus $<$, restricted to the set of all Δ, $\Delta\geq\Delta_0$, is a linear order consistent with F.

The next step is to extend $\{w_0\}$ to a set Γ of formulas having the following closure properties:

(a) $Fw\in\Gamma\rightarrow Xw,\ XFw\in\Gamma$,
$Xw\in\Gamma\rightarrow X{\sim}w\in\Gamma$
$w_1Uw_2\in\Gamma\rightarrow Fw_2, Xw_1, Xw_2, X(w_1Uw_2)\in\Gamma$

(b) Γ is closed under subformulas and boolean operations.

Moreover, Γ can be chosen so as to contain only finitely many distinct formulas, up to logical equivalence.

Let $\bar{S}=\{\Delta\cap\Gamma \mid \Delta\in S\}$. Clearly $\bar{S}$ is a finite set.

Definition. Define $\rho_+,\rho_<$ on $\bar{S}$ by:

$t\rho_+s$ iff for some $\Delta\in S$, $t=\Delta\cap\Gamma$ and $s=\Delta^+\cap\Gamma$.

$t\rho_<s$ iff for some $\Delta,\Theta\in S$, $\Delta<\Theta$ and $t=\Delta\cap\Gamma$, $s=\Theta\cap\Gamma$.

Lemma $\rho_<$ is the transitive closure of ρ_+.

The proof uses the induction axiom scheme A5.

We are ready now to define the model $(\omega,<,\{d_i\})$. Let $s_0=\Delta_0\cap\Gamma$, which is an element of $\bar{S}$ containing w_0. Let $\{S_n\}$ be a sequence of states from $\bar{S}$ such that for any $t\in\bar{S}$, if $t=s_i$ for infinitely many i then every t' such that $t\rho_+t'$ is also equal to infinitely many s_i. We use $\{s_n\}$ to form our final model. The truth assignment for the atoms in this model is given by $Q_i=\{n \mid q_i\in S_n\}$.

Lemma For any $w\in\Gamma$, $\|w\|_n=$ true iff $w\in S_n$. The proof proceeds by induction on w. For each connective we use the axioms involving that connective.

We thus get that $\|w_0\|_0=$ true.

We have shown that for any consistent w_0 there exist $\{Q_i\}$ such that $\|w_0\|_0=$ true in the model $(\omega,<,\{Q_i\})$.

This proves the completeness theorem.

Corollary Both DX and DUX are decidable.

Since the final model constructed was finitely representable we can deduce from the proof a bound, depending on w_0, for the size of its finite representation. Hence if a formula is satisfiable it is satisfiable by a model whose complexity does not exceed this bound and a decision procedure exists.

Comment.

In the earlier work ([P1],[P2], [MY]) the temporal operators were defined in a somewhat different way than in this work. The definitions here conform more closely to the conventions used by temporal logicians. Denoting the temporal operators under the interpretation given in [P1] etc. by F^0, G^0, U^0 respectively they can be related to the operators used here as follows:

$$F^0 w \equiv w \vee Fw$$

$$G^0 w \equiv w \wedge Gw$$

$$w_1 U^0 w_2 \equiv Gw_1 \vee w_2 \vee (w_1 \wedge (w_1 U w_2))$$

Note that the difference is whether we include the present as a part of the future or exclude it. Also, U has an existential character while U^0 has a universal character. In discussing program properties the F^0, G^0, U^0 operators seem more natural, as the attentive reader of section 1. might have realized.

The axiom systems DX and DUX described above can easily be transformed to complete axiomatix systems D^0X and D^0UX for $L(G^0,X)$ and $L(G^0,X,U^0)$ respectively.

A^01. $\vdash G^0(p \supset q) \supset (G^0 p \supset G^0 q)$

A^02. $\vdash X(\sim p) \equiv \sim Xp$

A^03. $\vdash X(p \supset q) \supset (Xp \supset Xq)$

A^04. $\vdash G^0 p \supset p \wedge XG^0 p$

A^05. $\vdash G^0(p \supset Xp) \supset (p \supset G^0 p)$

U^01. $\vdash G^0 p \supset pU^0 q$

U^02. $\vdash pU^0 q \equiv q \vee (p \wedge X(pU^0 q)$

The inference rules are the same, using G^0 in the generalization. A^01-A^05, R1-R^03 constitute D^0X, while the full system is D^0UX.

A system equivalent to D^0UX appears in [MY].

References

[FL] - Fischer, M.J. and R.L. Ladner: "Propositional Modal Logic of Programs", Proc. of the 9th ACM Symp. on the Theory of Computing, Boulder, Col. May (1977).

[G1] - Gabbay, D.: "Axiomatization of Logic of Programs", Manuscript, Nov. (1977).

[G2] - Gabbay, D.: "The Separation Property of tense Logics", Manuscript, Sept. (1979).

[GR] - Gries, D.: "A Proof of Correctness of Reim's Semaphore Implementation of the with-when statement", Technical Report TR 77-314, Cornell University, Ithaca, N.Y.

[K1] - Kamp, H.W.: "Tense Logic and the Theory of Linear Order", Ph.D. Thesis University of California, Los Angeles 1968.

[K2] - Kamp H.W.: Completeness Results in Temporal Logic", Preprint (1978).

[KR] - Krablin, L.: "A Temporal Analysis of Fairness," M.Sc. Thesis, University of Pennsylvania (1979)

[L1] - Lamport, L: "Proving the Correctness of Multiprocess Programs", IEEE Trans. Software Engineering, SE-3, 7 (March 1977) pp. 125-132.

[L2] - Lamport, L.: "Proving the Correctness of Multiprocess Programs", ACM TOPLAS, Vol. 1, No. 1 (July (1979) pp. 84-97.

[MP] - Manna, Z., Pnueli, A.: "The Modal Logic of Programs", Int. Conference in Automata, Languages and Programming, Graz (July 1979).

[MY] - Myers, T.J. and A. Pnueli: "The Temporal Situation Until Now", Technical Note, University of Pennsylvania (1979).

[P1] - Pnueli, A.: "The Temporal Logic of Programs", 19th Annual Symp. on Foundations of Computer Science, Providence, R.I.

[P2] - Pnueli, A.: "The Temporal Semantics of Concurrent Programs", Int. Symp. Semantics of Concurrent Computation, Evian, July 1979.

[SS] - Shelah, S., J. Stavi: "Expressive Completeness for Temporal Logic", Forthcoming.