International Journal of Control
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713393989

Decidability for a temporal logic used in discrete-event system analysis

J. F. Knight [a]; K. M. Passino [b]
[a] Department of Mathematics, University of Notre Dame, IN, U.S.A
[b] Department of Electrical and Computer Engineering, University of Notre Dame, IN, U.S.A

Online Publication Date: 01 December 1990
To cite this Article: Knight, J. F. and Passino, K. M. (1990) 'Decidability for a temporal logic used in discrete-event system analysis', International Journal of Control, 52:6, 1489 - 1506
To link to this article: DOI: 10.1080/00207179008953606
URL: http://dx.doi.org/10.1080/00207179008953606

INT. J. CONTROL, 1990, VOL. 52, NO. 6, 1489–1506

Decidability for a temporal logic used in discrete-event system analysis

J. F. KNIGHT† and K. M. PASSINO‡

The type of plant considered is one that can be modelled by a non-deterministic finite state machine P. The regulator is a deterministic finite state machine R. The closed loop system is formed by connecting P and R in a 'regulator configuration'. Formulae in a propositional temporal language are used to describe the behaviour of the closed-loop system. It is shown that there is a mechanical procedure which, for a given P and R, and a temporal formula Ψ, will determine in a finite number of steps whether or not Ψ must be true. This 'decidability' result could be proven using other known results on temporal logic. The proof given here shows that the behaviour of the closed-loop system may safely be assumed to be ultimately periodic. Formulae of a given complexity, say n, will be true in all possible 'runs' of the system just in case they are true in all ultimately periodic runs, with the period and the onset of periodicity bounded by a certain function of n. A 'synthesis' result follows immediately from the decidability result. The interpretation of time is discussed at some length. The results are illustrated on two discrete-event system examples. This paper is an expanded version of Knight and Passino (1987).

1. Introduction

We imagine a plant P in which the information about current conditions and the mechanisms for control are limited, and there are significant unpredictable, uncontrollable forces at work. Time is discrete. We assume that the plant P acts as a non-deterministic finite state machine. The regulator will be a deterministic finite state machine. For simplicity, we consider only 'full-state feedback' regulator systems here, leaving 'output-feedback' systems for a later paper. Thus, the output of the plant, which is the input to the regulator, is the full plant state; and the output from the regulator, which is the input to the plant, is the full regulator state. Let X denote the set of plant states and Q denote the set of regulator states, both finite. The plant and regulator models have the following form:

(a) $P = (X, Q, \delta, X_0)$, where $\delta : Q \times X \to \mathbb{P}(X) - \{\varnothing\}$ is the plant transition function, and $X_0 \subseteq X$ is the non-empty set of possible initial plant states; and

(b) $R = (Q, X, \xi, q_0)$, where $\xi : X \times Q \to Q$ is the regulator transition function, and $q_0 \in Q$ is the initial regulator state.

The closed-loop system is formed by connecting R and P in the regulator configuration shown in Fig. 1. The regulator system may be thought of as a non-deterministic 'generator', whose output is the output of the plant. There is no regulator system input ('reference input'). As the system runs it generates an infinite string of elements of X. Because of the non-deterministic nature of P there are, in general, many

Received 8 November 1988. Revised 15 May 1989.

† Department of Mathematics, University of Notre Dame, IN 46556, U.S.A.

‡ Department of Electrical and Computer Engineering, University of Notre Dame, IN 46556, U.S.A; currently with the Dapartment of Electrical Engineering, Ohio State University, 2015 Neil Avenue, Columbus, OH 43210, U.S.A.

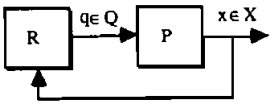

Figure 1. Regulator system.

(up to $2^{\aleph_0}$) different infinite strings of plant states that could be generated by P together with R. We shall refer to these as R-allowable strings. (The reason for attaching R here is that we shall consider different possible regulators R for a fixed plant P.)

As a simple example, consider a liquid-holding surge tank with sensors distinguishing among five liquid levels (empty, low, normal, high, and full) and with a fill valve that can be closed, or open. Unpredictable users operate another valve to take liquid from the tank. In modelling this system we let X be the set of five distinguishable liquid levels and let Q represent the two settings of the fill valve. When it is time to act, the regulator computes the next setting for the fill valve by applying the transition function ξ to the current liquid level and the current setting of the fill valve. The fill valve influences the liquid level but does not determine it completely. The unpredictable users determine which one of the set of possible next water levels will actually be the next plant output.

We use a propositional temporal language to describe the behaviour of the closed-loop system in terms of the plant and regulator states. The symbols of the language will be as follows: propositional variables (these will be the elements of $Q \cup X$), the usual propositional connectives (&, $\neg$, etc.), temporal 'modal' operators $\bigcirc$ (next time), $\Box$ (for all times, now and later), $\Diamond$ (for some time now or later) and parentheses. The language is determined by the pair of sets Q and X.

We assume that a fixed plant P is given. We shall define what it means for a temporal formula φ to be satisfied by a pair (R, α), where R is a possible regulator for P and α is an R-allowable string. We say that R makes the formula φ valid if (R, α) satisfies φ for all R-allowable strings α. Saying that the regulator R makes the specification φ valid is a way of saying that the regulator guarantees that the specification will be met.

Our first main result says that for any R-allowable string α there is an 'ultimately periodic' string β, also R-allowable, such that (R, α) satisfies φ if and only if (R, β) does. Using this, we show that there is a mechanical procedure for deciding in a finite number of steps whether a given regulator R makes a given formula φ valid. From this decidability result we obtain a synthesis result, which says that for a given formula φ we can effectively either find a regulator R making φ valid, or else say for sure that no such R exists. Our decision procedure is not computationally practical, in general. A 'tableau' method would be more efficient (Manna and Wolper 1981). However, most specifications of real interest seem to have a simple form (low 'rank'), and if we carried out a certain part of the decision procedure we could then easily examine a whole family of low-rank specifications.

Temporal logic has been widely utilized in computer science for such purposes as concurrent and sequential program verification, hardware verification and design, and computer communication protocol verification. A good introduction to temporal logic can be found in the work of Manna and Pnueli (1983), and a wide variety of temporal logics and their applications are discussed in Galton (1987) and the references therein. More recently, temporal logic has been utilized in a control theoretic framework (Fusaoka *et al.* 1983, Ostroff and Wonham 1985, Thistle and Wonham 1986, Ostroff 1987, Knight and Passino 1987, Ostroff and Wonham 1987, and Passino and Antsaklis 1988).

The propositional temporal language we use is like that in Manna and Wolper (1981) except that we do not include the 'until' operator. Manna and Wolper obtain decidability and synthesis results using the method of semantic tableaus. Our proof is closer in spirit to the work of Buchi (1962). Thistle and Wonham (1986) use a language with a slightly different syntax. They also allow a more general sort of plant and regulator than we do (in particular, the set of regulator states may be infinite). They do not prove decidability. Instead, they describe a system of rules of proof, and illustrate its use by proving closed-loop specification formulae from axioms for the plant and controller in various specific examples. Our work is also related to recent work by Ostroff (1987), who extended Thistle and Wonham's work and obtained computationally practical decision procedures for a special class of formulae in his language. In our treatment, the plant P and the regulator R are incorporated in a natural way into the semantics of the language. Our decidability proof, based on the fact that an arbitrary string can be replaced by an ultimately periodic one, seems to be new. It is completely elementary and self-contained.

In § 2 we say more about our model and discuss the way that time advances. In § 3 we describe precisely the syntax and semantics of our language and we define a notion of 'rank', which quantifies the complexity of a temporal formula. In § 4 we give the decidability and synthesis results. Section 5 contains examples of regulator system analysis and synthesis, illustrating the various notions and results from the earlier sections.

2. Modelling

Recall that in our model the regulator system is a pair of finite state machines.

Let ω denote the set of natural numbers. A 'run' of the regulator system yields an infinite sequence of pairs $(q_i, x_i)_{i\in\omega}$, where q_i is the regulator state at step i and x_i is the state of the plant at step i. Formally, a run is defined to be a sequence of pairs $((q_i, x_i))_{i\in\omega}$ such that q_0 is the initial state of the regulator, $x_0 \in X_0$ is a possible initial state for the plant, and for each i, $q_{i+1} = \xi(x_i, q_i)$, and $x_{i+1} \in \delta(\xi(x_i, q_i), x_i)$. The plant output sequence derived from the run is $(x_i)_{i\in\omega}$. Note that if we know ξ and q_0, then we can recover the run from the output sequence.

There is a slight asymmetry in the definition of 'run'. If we wished, we could have thought of our plant transition function differently, so that x_{i+1} would be the value of this function at (q_i, x_i) rather than at (q_{i+1}, x_i). In our formulation, if x_i is the input to the regulator, what comes out will be q_{i+1}. If q_{i+1} is the input to the plant, what comes out will be x_{i+1}. We can compute q_{i+1} as soon as we know x_i and q_i, but in general we have to wait to find out what x_{i+1} will be.

In our model of the regulator system, it is clear how time advances. 'Next time' means 'for the next pair of states (q_i, x_i)' (in a run produced by the pair of finite state machines). We shall define satisfaction of temporal formulae in terms of our model. Suppose that we have in mind a real plant and regulator. Our hope is that the temporal formulae say something meaningful about the real system. This means that when we choose a pair of finite state machines to serve as a model we have to think about the way time advances in the real system. We must try to answer the following questions, if we are to arrive at a useful model.

(*a*) When does the regulator act?

(*b*) When do we measure the resulting plant state?

There must be some mechanism in the plant that provides a measurement of the current plant state x_i and triggers the computation of the next regulator state q_{i+1}.

We consider briefly here two versions of the surge tank example, which lead to different models. (We shall discuss these surge tanks more fully in § 5.)

Schedule 1

Suppose that there is a clock in the plant, and each hour, on the hour, there is a fresh reading of the liquid level, after which the regulator adjusts the fill valve to the position it will maintain during the next hour.

Schedule 2

Suppose that regulator activity is tied to changes in the liquid level. Whenever the liquid goes from one level to another a device in the plant alerts the regulator. The regulator then adjusts the fill valve to the position it will maintain until the liquid level changes again.

For the surge tank that operates on Schedule 1, 'next time' means 'next hour'. For the surge tank that operates on Schedule 2, 'next time' means 'next time something happens'. Not surprisingly, the two versions of the surge tank give rise to different models. For Schedule 1, the plant transition function would certainly allow the possibility that if the tank is currently high, and the regulator closes the fill valve, then the tank will next be high again. For the surge tank operating on Schedule 2, the plant transition function would not allow this.

We could have more complicated schedules. There might be one reading and regulator adjustment per minute during certain peak hours and one per hour the rest of the day. Here the model for the plant would require more states. The plant transition might indicate that the plant state '2 a.m. and empty', could be followed, if the fill valve has been opened, by '3 a.m. and full', while '7.02 a.m. and empty', with the fill valve open, could not be followed by '7.03 a.m. and full'. Here 'next time' means 'next minute' for part of the day and 'next hour' the rest of the day. Always 'next time' means 'the next time the regulator has acted and the resulting plant state has been recorded'.

3. Formal language

In this section we describe precisely the syntax and semantics of our propositional temporal language. Actually there are different languages for different plants or, more precisely, for different pairs of sets X and Q. As we said earlier, the symbols are of the following forms. First, there are the propositional variables, which we are taking to be the elements of $Q \cup X$. Next, there are the usual logical connectives $\neg$ (not), $\&$ (and), $\vee$ (or), $\rightarrow$ (implies), and $\leftrightarrow$ (if and only if). Then there are the temporal operators $\bigcirc$, $\Box$ and $\Diamond$. Finally, there are parentheses. Now, we turn to the syntax.

The rules for forming formulae are as follows.

(*a*) A single propositional varibable q for $q \in Q$, or x for $x \in X$ is a formula.

(*b*) If φ is a formula, then so is $\neg\varphi$.

(*c*) If φ and ψ are formulae, then so are $(\varphi\&\psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$ and $(\varphi \leftrightarrow \psi)$.

(*d*) If φ is a formula, then so are $\bigcirc\varphi$, $\Box\varphi$, and $\Diamond\varphi$.

(*e*) Nothing is a formula unless it can be obtained by finitely many applications of (*a*)–(*d*) above.

The temporal formulae can, for instance, be used to quantify various control-theoretic design objectives (see, for instance, Fusaoka *et al.* 1983). Let $X_S \subseteq X$. Let $\varphi_s = (x_1 \vee x_2 \vee \ldots \vee x_n)$ where $x_i \in X_s$. Let $\varphi_0 = (x_{01} \vee x_{02} \vee \ldots \vee x_{0_n})$ where the possible initial plant states are x_{0i}, for $1 \leqslant i \leqslant n$. Let $Q_b \subset Q$ and $\varphi_b = (q_1 \vee q_2 \vee \ldots \vee q_n)$ where $q_i \in Q_b$. The formula $\varphi_0 \rightarrow \Diamond \varphi_s$ can be thought of as characterizing a 'reachability' requirement. The formulae

(i) $\varphi_0 \rightarrow \Diamond \Box \varphi_s$,

(ii) $\varphi_0 \rightarrow \Box \Diamond \varphi_s$, and

(iii) $\Box \varphi_b \rightarrow \Box \varphi_s$

characterize properties that can be thought of as analogous to 'stability' requirements in conventional control theory. The first says that if the plant starts in one of its initial states, then after some point in time the state will always be in X_s. The second says that if the plant starts in one of its initial states it will visit X_s infinitely often. The third states that if the plant inputs always remain in one set Q_b then the plant states will always remain in set X_s.

Recall the example of the surge tank. Let X have elements $x_1, \ldots, x_5$, representing levels of liquid in the tank (from empty to full). Let Q have elements q_0 and q_1, indicating that the fill valve is closed, open, respectively. Here are some sample formulae of the language.

(1) $x_5 \rightarrow \Diamond (x_4 \vee x_3)$

(2) $x_3 \rightarrow \Box \Diamond x_3$

(3) $\Box((x_2 \vee x_4) \rightarrow \bigcirc x_3)$

(4) $(\Box(x_1 \rightarrow \bigcirc q_1) \rightarrow \Box(x_1 \rightarrow \Diamond x_2))$

Formula 1 says that if the tank is initially full, then eventually it will become high or normal. Formula 2 says that if the liquid level is initially normal it will be normal infinitely often (it may in addition be infinitely often in each of the other states). Formula 3 says that any time the level is low or high, it will next be normal. Formula 4 says that if the fill valve opens whenever the tank is empty, then each time the tank becomes empty, it will eventually reach low again.

We next define the rank of a formula φ, denoted by $r(\varphi)$. The definition proceeds by induction on φ.

(*a*) $r(\varphi) = 0$, if φ consists of a single propositional variable,

(*b*) $r(\neg \varphi) = r(\varphi)$,

(*c*) $r((\varphi \& \psi)) = r((\varphi \vee \psi)) = r((\varphi \rightarrow \psi)) = r((\varphi \leftrightarrow \psi)) = \sup \{r(\varphi), r(\psi)\}$,

(*d*) $r(\bigcirc \varphi) = r(\Box \varphi) = r(\Diamond \varphi) = r(\varphi) + 1$

The formulae characterizing reachability and stability and the four sample formulae for the surge tank given above all have a rank less than or equal to two, and, in fact, the specifications that we have thought of in various natural examples all seem to have rank one or two. The temporal operators $\Diamond$ and $\Box$ resemble the quantifiers $\exists$ and $\forall$ from first-order predicate logic in some respects. For predictate logic we can put any formula in 'prenex normal form'. That is, we can find an equivalent formula with all the quantifiers at the front. Then the number of alternations of quantifiers provides a useful measure of complexity. There is nothing like prenex normal form for temporal logic. In fact, what we have expressed by Formula 3 above could not

be expressed by any formula in which all the temporal operators occurred at the front. Our notion of rank measures the complexity of a formula by counting the number of nested temporal operators without bringing them to the front.

Rank does not go up when we form complicated boolean combinations (putting formulae together with $\neg$, &, $\vee$, $\rightarrow$, and $\leftrightarrow$). The formulae of rank 0 are the ones with no occurrences of temporal operators. Let B_0 be the set of all formulae of rank 0. If $n > 0$ and B_n is the set of formulae of the forms $\bigcirc\varphi$, $\Box\varphi$ and $\Diamond\varphi$, where φ has rank $n-1$, then all formulae of rank n are obtained as boolean combinations of formulae from B_n and formulae of lower rank.

Above we described the syntax of the language; we now turn to the semantics. It will be helpful to introduce some notation. For any set Z, let Z^ω denote the set of all infinite strings of elements of Z. (Formally, these are sequences of length ω.) If $z \in Z^\omega$ we shall write z_i for the ith term in the sequence (an element of Z). We let z^i denote the result of dropping the first i terms from z (this is an element of Z^ω). The plant $P = (X, Q, \delta, X_0)$ is fixed. Let R be a regulator, with initial state q_0 and transition function ξ. If $\alpha \in X^\omega$, α is said to be R-allowable if it is the output sequence derived from a run of the regulator system that consists of R and P.

Let α be an R-allowable string, and let $((\sigma_i, \alpha_i))_{i\in\omega}$ be the corresponding run. (As we mentioned earlier, we can recover the run from the output sequence.) Let $q^{R,\alpha,i}$ denote the state σ_i reached by R after i steps in the run from which α is derived. When we define satisfaction we shall consider families of regulators, all having the same transition function as R, but with different initial states. Let $R^{\alpha,i}$ denote the regulator whose transition function is the same as that of R and whose initial state is $q^{R,\alpha,i}$. Note that $R^{\alpha,0}$ is just R.

We fix the plant P. This fixes the sets Q and X. If R is a regulator for P, and α is an R-allowable string, we shall use the notation $(R, \alpha) \models \varphi$ to indicate that the formula φ is satisfied by (R, α). The definition of satisfaction proceeds by induction on φ.

(i) $(R, \alpha) \models p$ if $p \in Q$ and $p = q_0$ or if $p \in X$ and $p = \alpha_0$,

(ii) $(R, \alpha) \models \neg\varphi$ if it is not the case that $(R, \alpha) \models \varphi$,

(iii) $(R, \alpha) \models (\varphi \& \psi)$ if $(R, \alpha) \models \varphi$ and $(R, \alpha) \models \psi$,

(iv) $(R, \alpha) \models (\varphi \vee \psi)$ if $(R, \alpha) \models \varphi$ or $(R, \alpha) \models \psi$,

(v) $(R, \alpha) \models (\varphi \rightarrow \psi)$ if $(R, \alpha) \models \varphi$ implies $(R, \alpha) \models \psi$,

(vi) $(R, \alpha) \models (\varphi \leftrightarrow \psi)$ provided that $(R, \alpha) \models \varphi$ if and only if $(R, \alpha) \models \psi$,

(vii) $(R, \alpha) \models \bigcirc\varphi$ if $(R^{\alpha,1}, \alpha^1) \models \varphi$,

(viii) $(R, \alpha) \models \Box\varphi$ if for all $i \geqslant 0$, $(R^{\alpha,i}, \alpha^i) \models \varphi$,

(ix) $(R, \alpha) \models \Diamond\varphi$ if for some $i \geqslant 0$, $(R^{\alpha,i}, \alpha^i) \models \varphi$.

From the definition of satisfaction it is clear that for any formula φ, $\Diamond\varphi$ is equivalent to $\neg(\Box(\neg\varphi))$, so we could have omitted $\Diamond$ from the language. Similarly, we could have omitted $\vee$, $\rightarrow$, and $\leftrightarrow$.

An equivalence relation on a set Z is a two-place relation $\sim$ (on Z) that is reflexive, symmetric, and transitive. If $z \in Z$, then the *$\sim$-equivalence class* of z is the set $\{y \in Z : y \sim z\}$. We may speak of $\sim$-equivalence classes in general (without naming elements). We define a family of equivalence relations $\sim_{n,R}$ on the set of R-allowable strings, such that $\sim_{n,R}$-equivalence implies satisfaction of the same formulae of rank

at most n. Suppose that α and β are R-allowable. Then

(*a*) $\alpha \sim_{0,R} \beta$ if $\alpha_0 = \beta_0$, and

(*b*) $\alpha \sim_{n+1,R} \beta$ if

(i) $\alpha \sim_{n,R} \beta$,

(ii) $q^{R,\alpha,1} = q^{R,\beta,1}$ and $\alpha^1 \sim_{n,R^{\alpha,1}} \beta^1$, and

(iii) for all i there exists a j (and for all j there exists an i) such that $q^{R,\alpha,i} = q^{R,\beta,j}$ and $\alpha^i \sim_{n,R^{\alpha,i}} \beta^j$.

Note that $\alpha \sim_{0,R} \beta$ if and only if α and β have the same first symbol; $\alpha \sim_{1,R} \beta$ if and only if α and β have the same first symbol and the same second symbol, and the set of pairs representing a current regulator state and plant state is the same when the regulator R is applied to α and to β. It is easy to see that if $m < n$ and $\alpha \sim_{n,R} \beta$, then $\alpha \sim_{m,R} \beta$. In general, each $\sim_{m,R}$-class will contain many $\sim_{n,R}$-classes. For example, consider the strings $\alpha = x_2x_1x_1x_2x_3x_3x_3 \ldots$ and $\beta = x_2x_1x_2x_3x_3x_3 \ldots$. Suppose that these are both R-allowable, and let $\xi(x_i, q_0) = q_0$ for $i = 1, 2, 3$ (where ξ is the transition function for R). Then $\alpha \sim_{0,R} \beta$ and $\alpha \sim_{1,R} \beta$, but not $\alpha \sim_{2,R} \beta$.

4. Decidability and synthesis results

The decidability and synthesis results are given in this section. The proofs are based on three lemmas. The first lemma says that strings that are $\sim_{n,R}$-equivalent will satisfy the same formulae of rank at most n.

Lemma 1

Let φ be a formula such that $r(\varphi) \leqslant n$, and let α and β be R-allowable strings such that $\alpha \sim_{n,R} \beta$. Then $(R, \alpha) \models \varphi$ if and only if $(R, \beta) \models \varphi$.

Proof

We use induction on n. Note that for any regulator R and any R-allowable strings α, β, the set of formulae φ such that $(R, \alpha) \models \varphi$ if and only if $(R, \beta) \models \varphi$ is closed under boolean combinations. Hence, the only rank n formulae we really need to look at are the basic ones; that is, the elements of B_n. Moreover, by symmetry it is enough to show that if $\alpha \sim_{n,R} \beta$ and $(R, \alpha) \models \varphi$, where $\varphi \in B_n$, then $(R, \beta) \models \varphi$.

We start with $n = 0$. Suppose $\alpha \sim_{0,R} \beta$ and $(R, \alpha) \models p$, where $p \in X \cup Q$. If $p \in X$, then by the definition of satisfaction, $p = \alpha_0$. By $\sim_{0,R}$-equivalence, $\alpha_0 = \beta_0$, so $(R, \beta) \models p$. If $p \in Q$, then by the definition of satisfaction, $p = q_0$, and then $(R, \beta) \models p$. Therefore the lemma holds for $n = 0$. Supposing that the lemma holds for n, consider $n + 1$. Let φ have rank n and suppose that $\alpha \sim_{n+1,R} \beta$. If $(R, \alpha) \models \bigcirc\varphi$, then by the definition of satisfaction, $(R^{\alpha,1}, \alpha^1) \models \varphi$. By $\sim_{n+1,R}$-equivalence, $q^{R,\alpha,1} = q^{R,\beta,1}$ and $\alpha^1 \sim_{n,R^{\alpha,1}} \beta^1$. Then by the induction hypothesis, we have $(R^{\beta,1}, \beta^1) \models \varphi$, so $(R, \beta) \models \bigcirc\varphi$. Let $(R, \alpha) \models \Box\varphi$. By the definition of satisfaction, for all $i \geqslant 0$, $(R^{\alpha,i}, \alpha^i) \models \varphi$. By $\sim_{n+1,R}$-equivalence, for each $j \geqslant 0$ there is some $i \geqslant 0$ such that $q^{R,\alpha,i} = q^{R,\beta,j}$ and $\alpha^i \sim_{n,R^{\alpha,i}} \beta^j$. Then by the induction hypothesis, $(R^{\beta,j}, \beta^j) \models \varphi$, so $(R, \beta) \models \Box\varphi$. Finally, let $(R, \alpha) \models \Diamond\varphi$. There is some $i \geqslant 0$ such that $(R^{\alpha,i}, \alpha^i) \models \varphi$. Then for some $j \geqslant 0$, $q^{R,\alpha,i} = q^{R,\beta,j}$ and $\alpha^i \sim_{n,R^{\alpha,i}} \beta^j$, so by the induction hypothesis, $(R^{\beta,j}, \beta^j) \models \varphi$. Therefore, $(R, \beta) \models \Diamond\varphi$. This completes the proof. □

A string $\alpha \in X^\omega$, is said to be *ultimately periodic*, with period n_2 and onset of periodicity n_1, if α has the form $\upsilon\hat{}\tau\hat{}\tau\hat{}\tau\hat{}\ldots$, where υ and τ are finite strings of lengths n_1 and n_2, respectively. We say that α is *determined* by the pair (υ, τ). Then υ gives the transient behaviour and τ gives the behaviour during one period. (It is possible for different pairs (υ, τ) to determine the same α. If we wished, we could avoid this by choosing υ and τ to minimize first n_1 and then n_2.) For ultimately periodic strings it is relatively easy to answer questions about satisfaction.

Lemma 2

We can effectively decide, given a formula φ, a regulator R and a pair of finite strings (υ, τ) determining a string $\alpha \in X^\omega$, whether $(R, \alpha) \models \varphi$.

Proof

First of all, note that we can tell whether the ultimately periodic string α determined by (υ, τ) is R-allowable. Now we proceed by induction on formulae φ, describing a method for deciding whether $(R, \alpha) \models \varphi$ if α is an R-allowable string determined by (υ, τ). We start with φ of the form p for $p \in X \cup Q$. Let α be determined by (υ, τ). If $p \in X$, then $(R, \alpha) \models p$ if and only if $p = \alpha_0$, where α_0 is the first symbol of $\upsilon \wedge \tau$. If $p \in Q$, then $(R, \alpha) \models p$ if and only if $p = q_0$. There is no difficulty in deciding satisfaction for boolean combinations of formulae that we can deal with, so let us turn to the temporal operators.

Suppose that we have a procedure for deciding satisfaction for φ. We must say how to deal with $\bigcirc\varphi$, $\Box\varphi$ and $\Diamond\varphi$; we consider $\bigcirc\varphi$ first. If α is the ultimately periodic string determined by (υ, τ), then α^1 is also ultimately periodic, determined by (υ^1, τ), where υ^1 is the result of dropping the first term of υ if $\upsilon \neq \varepsilon$ and of τ if $\upsilon = \varepsilon$ (ε denotes the empty string). We have $q^{R,\alpha,1} = \xi(\alpha_0, q_0)$. By the definition of satisfaction, $(R, \alpha) \models \bigcirc\varphi$ if and only if $(R^{\alpha,1}, \alpha^1) \models \varphi$, and we know how to decide this. Next, consider $\Box\varphi$. Let $m = n_1 + |Q| \cdot n_2$. By the definition of satisfaction, $(R, \alpha) \models \Box\varphi$ if and only if for all $i \geqslant 0$, $(R^{\alpha,i}, \alpha^i) \models \varphi$. For each $i \geqslant 0$, there is some $j < m$ such that $q^{R,\alpha,i} = q^{R,\alpha,j}$ and $\alpha^i = \alpha^j$, so it is enough to check that $(R^{\alpha,j}, \alpha^j) \models \varphi$ for all $j < m$. In terms of R and (υ, τ), we can figure out what $q^{R,\alpha,j}$ is, and we can also find a variant υ^j of υ such that (υ^j, τ) determines α^j. By our induction hypothesis, we can decide whether φ is satisfied by the pairs $(R^{\alpha,j}, \alpha^j)$ for all $j < m$. Finally, consider $\Diamond\varphi$. Letting m be as above, we have $(R, \alpha) \models \Diamond\varphi$ if and only if $(R^{\alpha,j}, \alpha^j) \models \varphi$ for some $j < m$, and we can check this. This completes the proof of the lemma. □

If R is a regulator and φ is a formula, then R is said to make φ valid if $(R, \alpha) \models \varphi$ for all R-allowable strings α. There may in general be $2^{\aleph_0}$ R-allowable strings. Only countably many are ultimately periodic. The next lemma is our first main result. It implies that to decide whether R makes φ valid, we need not look at all R-allowable α, or even at all ultimately periodic α. It is enough to consider a particular finite collection with bounded period and onset of periodicity.

Lemma 3

For any regulator R, any $n \in \omega$, and any R-allowable string α, there is an ultimately periodic string β such that $\alpha \sim_{n,R} \beta$. Moreover, given n (and $|Q|$ and $|X|$), we can compute bounds on the onset and period.

Proof

We shall define L_n and K_n for $n \in \omega$ such that K_n is an upper bound on the number of $\sim_{n,R}$-classes (for any regulator R), and for any R-allowable string α, there exist υ and τ such that $\upsilon\hat{}\tau$ has length at most L_n and (υ, τ) determines an R-allowable string β such that $\beta \sim_{n,R} \alpha$. For $n = 0$, we can take K_0 to be $|X|$ (since the $\sim_{0,R}$-class of α is determined by the first symbol of α). We must say what L_0 is. For any R-allowable string α, there exist $i < j \leqslant |Q| \cdot |X|$ such that $q^{R,\alpha,i} = q^{R,\alpha,j}$ and $\alpha_i = \alpha_j$. Let $\alpha \restriction k$ denote the finite string consisting of the first k symbols of α. If $\upsilon = \alpha \restriction i$ and $\upsilon\hat{}\tau = \alpha \restriction j$, then (υ, τ) determines an ultimately periodic R-allowable β such that $\beta \sim_{0,R} \alpha$. Therefore, we can take L_0 to be $|Q| \cdot |X|$.

Suppose that for $k \leqslant n$, the lemma holds and we have determined K_k and L_k. If α is R-allowable, we write $C_k^{\alpha,i}$ for the $\sim_{k,R^{\alpha,i}}$-class of α^i. We may refer to the pair $(q^{R,\alpha,i}, C_k^{\alpha,j})$ as the *k-class* of the term α_i. Note that for two R-allowable strings α and β, $\alpha \sim_{n+1,R} \beta$ if and only if the following three conditions hold:

(*a*) $C_n^{\alpha,0} = C_n^{\beta,0}$

(*b*) $C_n^{\alpha,1} = C_n^{\beta,1}$, and

(*c*) $\{(q^{R,\alpha,i}, C_n^{\alpha,i}) : i \in \omega\} = \{(q^{R,\beta,i}, C_n^{\beta,i}) : i \in \omega\}$.

Condition (*a*) says that $\alpha \sim_{n,R} \beta$, and this implies that $q^{R,\alpha,1} = q^{R,\beta,1}$. If $q^{R,\alpha,1} = q^{R,\beta,1}$, then condition (*b*) says that $\alpha^1 \sim_{n,R^{\alpha,1}} \beta^1$. Condition (*c*) says that for each i there is some j (and also for each j there is some i) such that $q^{R,\alpha,i} = q^{R,\beta,j}$ and $\alpha^i \sim_{n,R^{\alpha,1}} \beta^j$. This analysis shows that we can take K_{n+1} to be $K_n^2 \cdot 2^r$, where $r = K_n \cdot |Q| - 1$.

Next, we describe a method for finding an ultimately periodic string β that is $\sim_{n+1,R}$-equivalent to a given string α. Having done this, we shall be able to say what L_{n+1} is. There are at most $|Q| \cdot K_n$ different n-classes. We mark any term α_i such that there is no $j < i$ with α_i and α_j representing the same n-classes. It may be that some n-classes only occur finitely many times, but at least one n-class occurs infinitely often in α. Let α_r be the first term representing an n-class that occurs infinitely often. Of course, α_r is marked. Let α_s be the first term, after all of the marked ones, such that α_s represents the same n-class as α_r.

If we choose υ and τ such that $\upsilon = \alpha \restriction r$, $\upsilon\hat{}\tau = \alpha \restriction s$, and let γ be the ultimately periodic string determined by (υ, τ), we could show (with effort) that $\gamma \sim_{n+1,R} \beta$. However, $\upsilon\hat{}\tau$ might be very long. We now reduce the length. Suppose that we have α_i and α_j representing the same n-class, where $i < j < s$, and for all k such that $i \leqslant k \leqslant j$, α_k is not marked. Then we close up the interval, leaving out of $\upsilon\hat{}\tau$ all terms α_k for $i < k \leqslant j$. When we have closed up as much as possible in this way, $\upsilon\hat{}\tau$ will consist of at most $|Q| \cdot K_n$ marked terms plus at most $|Q| \cdot K_n$ additional symbols after each marked one. We take L_{n+1} to be $|Q| \cdot K_n + (|Q| \cdot K_n)^2$.

Let β be the ultimately periodic string determined by (υ, τ) (after the shortening process). All that remains to be done is to prove that $\alpha \sim_{n+1,R} \beta$. Each term β_m of β corresponds to a particular term α_i from α (one that was included in υ or τ).

Proposition

If β_m corresponds to α_i, then (1) $q^{R,\alpha,i} = q^{R,\beta,m}$, and (2) for each $k \leqslant n$

$$C_k^{\alpha,i} = C_k^{\beta,m}$$

Suppose for the moment that the proposition is true. Then we can easily show that $\beta \sim_{n+1,R} \alpha$. Recall the three conditions above. Condition (*a*) will hold because β_0 corresponds to α_0. If we only marked one term, then α is constant (it consists of infinitely many repetitions of a single symbol) and $\beta = \alpha$. If there is more than one term marked, then we saved α_1 (whether or not we marked it) and β_1 corresponds to α_1; therefore condition (*b*) holds. Condition (*c*) holds because we marked representatives of all n-classes in α. Therefore, if the claim is true, then $\beta \sim_{n+1,R} \alpha$, and to complete the proof of the lemma, all that remains is to prove the proposition.

Proof

Suppose β_m corresponds to α_i and β_{m+1} corresponds to α_j. There are three cases: (i) $j = i + 1$, (ii) $j > i + 1$, and (iii) $j < i$. Case (ii) indicates that we closed up the interval between α_i and α_j. Then α_i and α_{j-1} represent the same $(n-1)$ class. Case (iii) indicates that α_i is the last term of τ and α_j is the first (i.e. $i = s$ and $j = r$). Then α_{i+1} and α_j represent the same n-class.

It is easy to check that if β_m corresponds to α_i, then $q^{R,\alpha,i} = q^{R,\beta,m}$, and the terms β_m and α_i match. Therefore we have part (1) of the proposition in general, and part (2) for $k = 0$. We continue with the proof of part (2) by induction on k. We show that if (2) holds for k, where $k < n$, then it also holds for $k + 1$. Let β_m correspond to α_i. To show that β_m and α_i represent the same $k+1$-class, we verify the three conditions given earlier. By the induction hypothesis, β_m and α_i represent the same k-class. This is condition (*a*).

To verify the condition (*b*), we consider β_{m+1} and show that in the three cases (i), (ii) and (iii) above, β_{m+1} and α_{i+1} represent the same k-class. Suppose β_{m+1} corresponds to α_j. By the Induction Hypothesis, β_{m+1} and α_j represent the same k-class, so we will be done if we can show that α_j and α_{i+1} represent the same k-class; this is trivially true if $j = i + 1$. If $j > i + 1$, then we are in case (ii) described above, and α_{i+1} and α_j represent the same $n-1$-class. Since $k \leqslant n - 1$, it follows that α_{i+1} and α_j represent the same k-class. Finally, if $j < i$, we are in case (iii), and again α_{i+1} and α_j represent the same n-class. We have shown that in all cases β_{m+1} and α_{i+1} represent the same k-class.

To prove the condition (*c*) we must show that the k-classes represented by $\beta_{m'}$ for $m' > m$ are the same as the k-classes represented by $\alpha_{i'}$ for $i' \geqslant i$. First, take $i' \geqslant i$. If τ includes some α_j representing the same k-class as α_i, then we have $\beta_{m'}$ representing this k-class for arbitrarily large m'. Suppose that τ does not include any such α_j. It will follow that the k-class of α_i is represented only finitely many times in α; the reason for this is as follows. If the k-class were represented infinitely often, then some n-class in the k-class would be represented infinitely often. Let $\alpha_{i''}$ be the first representative of this k-class. By our choice of r, we have $r \leqslant i''$, and the k-class is represented in τ, a contradiction. Since the k-class of $\alpha_{i'}$ is represented only finitely many times in α, there is a largest j such that α_j represents the k-class. If $j \geqslant r$, then for each of the infinitely many $r' > r$ such that $\alpha_{r'}$ represents the same n-class as α_r, there would be $j' \geqslant r'$ representing the same $n-1$-class, and, hence, the same k-class, as α_j. This cannot happen, so we have $j < r$.

Since α_j is the last representative of its k-class it is also the last representative of its $(n-1)$-class. Then it is easy to check that α_{j+1} is the first representative of its n-class. This means that α_{j+1} is marked and corresponds to some (first) element of β, which we call $\beta_{m'+1}$. (There cannot be more than one element of β corresponding to

α_{j+1} unless $j+1=r$.) The symbol $\beta_{m'+1}$ has an immediate predecessor, which we call $\beta_{m'}$. We have $i \leqslant i' \leqslant j < j+1 \leqslant r$, and $\beta_{m'}$ corresponds to some $\alpha_{j'}$ where $i \leqslant j' \leqslant j$ (see Fig. 2).

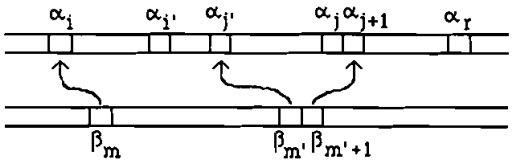

Figure 2.

We may be lucky and have $j'=j$. If $j'<j$, then we close up the interval between $\alpha_{j'}$ and α_{j+1}. Then $\alpha_{j'}$ and α_j represent the same n-class, and, hence, the same k-class.

Now, take $m' \geqslant m$. We must find $i' \geqslant i$ such that $\beta_{m'}$ and $\alpha_{i'}$ represent the same k-class. Let $\beta_{m'}$ correspond to α_j. Then $\beta_{m'}$ and α_j represent the same k-class. If $j \geqslant i$, then we are done. Suppose $j<i$. This means that β_m and $\beta_{m'}$ lie in different copies of τ, and $r \leqslant j < i < s$. Since α_r and α_s represent the same n-class, there must be an $i' \geqslant s$ such that α_j and $\alpha_{i'}$ represent the same k-class (the same $(n-1)$-class, in fact). We have now verified the condition (*c*), completing the proof that if β_m corresponds to α_i, then the $k+1$-classes match. This was all that remained to prove to lemma. □

Theorem 1: Decidability

Given a plant P, a formula φ, and a regulator R for P, we can effectively decide whether an R makes φ valid.

Proof

The decision procedure is as follows. First, compute $r(\varphi)$: say this is n. Next, list the pairs (υ, τ) such that

(*a*) $\upsilon\hat{}\tau$ has length at most L_n (where L_n is as in the proof of Lemma 3), and

(*b*) the ultimately periodic string determined by (υ, τ) is R-allowable.

Then test whether $(R,\alpha) \models \varphi$ for the strings α determined by the pairs on the list, using the procedure from Lemma 2. By Lemmas 1 and 3, R makes φ valid if and only if $(R, \alpha) \models \varphi$ for these strings.

Recall that $P=(X, Q, \delta, X_0)$. Suppose we wish to meet a particular specification φ. Theorem 1 says that for a given regulator $R=(Q, X, \xi, q_0)$ we can determine whether R makes φ valid. If R does not make φ valid, then we may try a different regulator. In fact, since there are only finitely many regulators altogether (differing only in transition function and initial state), we can try them all. By doing so, we will either find one that works or else determine that they all fail. We have established the following.

Theorem 2: Synthesis

Given a formula φ, we can effectivcely either find a regulator R making φ valid, or else say for that that no such R exists.

The theorems above say that we can, in principle, do a really thorough job of regulator system analysis and also mechanically carry out regulator synthesis for the type of plant that we have described.

5. Examples

In this section we describe in detail some examples and determine what the results of the previous section mean for these examples. The first two examples, from chemical process control, are versions of the surge tank described briefly in §§ 1 and 2. In both versions the surge tank, shown in Fig. 3, has sensors distinguishing five liquid levels (empty, low, normal, high, and full). The regulator can only open or close a fill valve. Users operate an empty valve to take liquid from the tank.

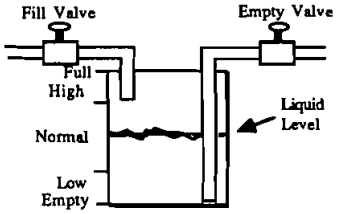

Figure 3. Surge tank.

The first version of the surge tank follows the first schedule described in § 2. Once an hour there is a reading of the liquid level, followed by an adjustment of the fill valve. We assume that in one hour, if the empty valve remains closed and the fill valve is open the liquid level will rise just one level, except that if the tank is full, the level cannot rise. With both fill and empty valves open, the liquid level stays constant, except that if the fill valve is open, so that liquid is coming in, the level will be read as low rather than empty, even when the depth of the liquid is essentially zero.

The set of plant states is $X = \{x_1, x_2, x_3, x_4, x_5\}$, where x_i is the ith liquid level, from lowest to highest. The set of regulator states is $Q = \{q_0, q_1\}$, where q_0 and q_1 stand for 'fill valve closed', and 'fill valve open', respectively. The non-deterministic finite state machine representing the plant is $P = (X, Q, \delta, X_0)$, where the set of initial states is $X_0 = \{x_1, x_3\}$ and the transition function δ may be read from the graph in Fig. 4(*a*).

The graph has nodes representing the plant states. The arrows, labelled by regulator states q, that emanate from a node x lead to the possible next states; i.e. the elements of $\delta(x, q)$. For example, if the plant state is x_3 (normal liquid level) and the

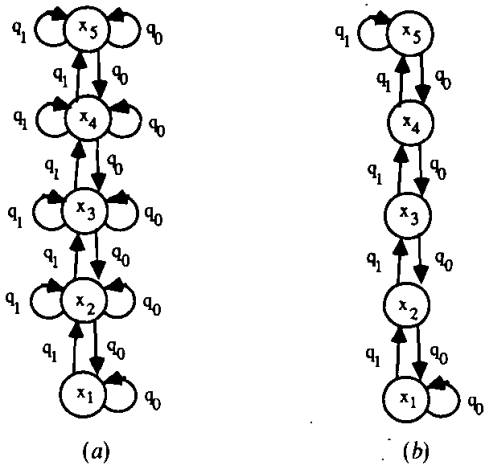

Figure 4. Surge tank state transition graphs.

regulator responds with q_1 (fill valve open), then by the next hour the plant state may still be x_3 (if, for instance, users are drawing liquid from the tank), or the output may be x_4 (high).

One possible regulator is $R = (Q, X, \xi, q_0)$, where the initial state is q_0 (fill valve closed), and the transition function ξ is given in Fig. 5.

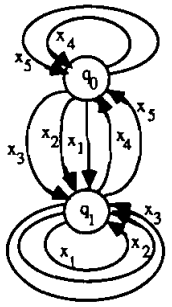

Figure 5. Regulator state transition graph

This graph is interpreted much like the one for the plant transition function. The nodes represent the regulator states. The arrow that comes from node q and is labelled by the plant state x leads to $\xi(x, q)$, the next regulator state. For example, if the regulator state has been q_0 (fill valve closed) and the plant has been in state x_2 (low), then the next regulator state is q_1 (fill valve open).

Let us see what our results mean for rank 1 formulae. By Lemma 3, for any R-allowable string α there is some R-allowable string $\beta \sim_{1,R} \alpha$ such that β is ultimately periodic, and by Lemma 1, for any rank 1 formula φ, $(R, \alpha) \models \varphi$ if and only if $(R, \beta) \models \varphi$. The R-allowable string $\alpha = x_1x_2x_3x_4x_3x_3x_4x_3x_3x_3x_4x_3 \ldots$, with the symbol x_3 repeating, once, twice, three times, etc., is not ultimately periodic. If $\beta = x_1x_2x_3x_4x_3x_4x_3x_4x_3 \ldots$, then β is ultimately periodic—it is determined by the pair (υ, τ), where $\upsilon = x_1x_2$ and $\tau = x_3x_4$. It is easy to see that $\beta \sim_{1,R} \alpha$.

A list of pairs is said to be n-complete (for R) if each R-allowable string is $\sim n,R$-equivalent to the string determined by one of the pairs (υ, τ) on the list. The proof of Lemma 3 tells us that we can obtain an n-complete list by looking at all pairs (υ, τ) such that $\upsilon\hat{}\tau$ has length at most L_n, and taking those which determine R-allowable strings. Since $L_1 = 110$, even the 1-complete list obtained by this method would be quite long, and some of the pairs on the list would be complicated. Here is a simpler list which is 1-complete: (x_1, x_2), (x_1x_2, x_3), (x_1x_2, x_3x_4), $(x_1x_2x_3x_4, x_4x_3)$, $(x_1x_2x_3, x_4)$, (ε, x_3), (x_3, x_3x_4), (x_3x_3, x_4), (x_3x_4, x_3), (x_3, x_4), $(x_3x_4x_3x_3, x_4)$, (ε, x_3x_4). This list has the feature that distinct pairs determine $\sim_{1,R}$-inequivalent strings.

One particular closed-loop specification that we should like to meet is the formula, $\varphi = x_3 \rightarrow \Box(\neg(x_1 \vee x_5))$, which says that if the liquid level is normal initially, then it will never be empty or full. Note that $r(\varphi) = 1$. Therefore, to decide whether the regulator R makes φ valid it is enough to decide whether $(R, \alpha) \models \varphi$ for the strings α determined by the 12 pairs on the 1-complete list above. Suppose (υ, τ) determines the string α. Let n_1 be the length of υ, let n_2 be the length of τ, and let $m = n_1 + 2n_2$. Looking at the form of φ, and using the definition of satisfaction and the proof of Lemma 2, we see that $(R, \alpha) \models \varphi$ if and only if for each $i \geqslant 0$, $(R^{\alpha,i}, \alpha^i) \models \neg(x_1 \vee x_5)$ if and only if for each $i \leqslant m$, $(R^{\alpha,i}, \alpha^i) \models \neg(x_1 \vee x_5)$ for all $i \leqslant m$ if and only if for each $i \leqslant m$,

neither $(R^{\alpha,o}, \alpha^i) \models x_1$ nor $(R^{\alpha,i}, \alpha^i) \models x_5$. For all the pairs (υ, τ) above, $m \leqslant 8$, and there are no occurrences of x_1 or x_5. Hence, R makes φ valid.

Another specification is the formula $\psi = (x_1 \rightarrow \Diamond \Box(x_3 \vee x_4))$, which says that if the liquid level is initially empty, then after some point in time the level will always be normal or high. Note that $r(\psi) = 2$. Certain strings which are $\sim_{1,R}$-equivalent are not $\sim_{2,R}$-equivalent. For example, the string determined by (x_3x_3, x_4) is $\sim_{1,R}$-equivalent, but not $\sim_{2,R}$-equivalent, to the string determined by $(x_3x_3x_3, x_4)$. The strings determined by $(x_3x_3, x_4x_3x_3)$ and $(x_3x_3x_3, x_4x_3x_3)$ are both $\sim_{1,R}$-equivalent to the one determined by (x_3, x_3x_4), but no two of these three are $\sim_{2,R}$-equivalent. We could form a minimal 2-complete list, starting with the members of the 1-complete list above, and adding, for each of these pairs, just enough new pairs to represent all of the $\sim_{2,R}$-classes in the $\sim_{1,R}$-class. In general, using higher-rank formulae, we can describe more subtle features of plant behaviour, and for larger n, we should not be surprised that even the shortest and simplest n-complete lists become unmanageable.

The regulator R makes the formula $\psi = x_1 \rightarrow \Diamond \Box(x_3 \vee x_4)$ valid if and only if $(R, \alpha) \models \psi$ for all α determined by the pairs on a 2-complete list. By the definition of satisfaction, and the proof of Lemma 2, we see that if α is an R-allowable string determined by the pair (υ, τ), then $(R, \alpha) \models \psi$ if and only if either the first symbol of α is not x_1 or else $(R, \alpha) \models \Diamond \Box(x_3 \vee x_4)$, and this holds if and only if either the first symbol of $\upsilon\hat{}\tau$ is not x_1 or else τ has no other symbols than x_3 and x_4. When we come to the pair (x_1, x_2), we find that for the string $\alpha = x_1x_2x_2x_2x_2 \ldots$, determined by this pair (R, α) fails to satisfy ψ. The string α represents a possible output string for the system in which the specification is not met.

We have seen that the regulator R above does not make the formula $\psi = x_1 \rightarrow \Diamond \Box(x_3 \vee x_4)$ valid. If we apply the method of Theorem 2 (testing all possible regulators), we shall either find a regulator that makes ψ valid, or else know for sure that no regulator does this. Even for this relatively simple example that mechanical procedure is long. Allowing ourselves to think about the reason for the failure of the first regulator, we arrive more quickly at the conclusion that no regulator works. Consider the various possibilities for the initial regulator state and for the value of the regulator transition function on (x_1, q_0), (x_1, q_1), (x_2, q_0), and (x_2, q_2). With just this information, we can check that for each regulator R', at least one of the following pairs determines an R'-allowable string: (x_1, x_2), (ε, x_1), (x_1x_1, x_2), (x_1, x_1x_2). If β is an R'-allowable string determined by one of these pairs, then, by the reasoning above, (R', β) fails to satisfy ψ.

As another regulator synthesis example suppose that we wish to satisfy $\varphi' = x_1 \rightarrow \Diamond \Box(x_2 \vee x_3)$, i.e. 'if initially the tank is empty, then eventually there will be a time such that from then on, the liquid level will be low or normal'. This rank 2 formula is invalid for the regulator R above, with a counter-example determined by the pair $(x_1x_2x_3x_4, x_4)$. Theorem 2 says that an exhaustive search will produce a regulator to make φ' valid if one exists. As a synthesis heuristic, we examine some of the (υ, τ) pairs used in the above analysis. For the string $(x_1x_2x_3x_4, x_4)$, the sequence of associated states is $q_0q_1q_1q_1q_0q_0q_0q_0 \ldots$. With this and the plant model P, we see that if we changed ξ so that the regulator responsed to the plant output symbol x_3 with a q_0 rather than a q_1, the specification might be met. Name this new regulator transition function ξ' and the corresponding regulator R'. For R' with initial state q_0 as above, some of the (υ, τ) pairs representing different $\sim_{2,R}$-equivalence classes are: $(x_1, x_2), (x_1, x_2x_3x_3x_2), (x_1, x_2x_2x_3x_2), (x_1x_2x_3, x_3), (x_1x_2x_2, x_3)$. Examining (υ, τ) pairs representing all $\sim_{2,R'}$-equivalence classes would show that the synthesized regulator R' makes the formula φ' valid.

We now consider briefly a second version of the surge tank. The plant states and the regulator states are the same as in the first version, so the possible regulators are the same. However, the schedule is different, and this difference is reflected in the plant transition function. Some device in the plant drives the regulator to respond whenever the liquid level changes from one level to another, and in two additional special circumstances: (*a*) if the level has been empty, and the response was fill valve closed (a situation that would never lead to a change from empty to low), then whenever a frustrated user opens the empty valve and gets nothing, the regulator is notified that the tank is empty; (*b*) if the level has been full, and the response was fill valve open (a situation that would never lead to a change from full to high), then whenever a user closes the empty valve, liquid spills out of the tank, and the regulator is notified that the tank is full. We assume that the empty valve will be opened and closed infinitely often. Again we take the initial plant states to be x_1 and x_3. The plant function is given in Fig. 4(*b*).

Let ψ be $x_1 \rightarrow \Diamond \Box (x_3 \vee x_4)$, as above, and suppose we want to find (if possible) a regulator that will make ψ valid. The proof of Theorem 2 suggests that we make a list of all possible regulators, and test them, one by one, until we find a regulator that works or finish testing them all without finding one that works. As it turns out, the regulator R that was described above makes ψ valid. For the first version of the surge tank, the string determined by the pair (x_1, x_2) served as an example showing that R did not make ψ valid. For the second version of the surge tank, this string is not R-allowable.

The next example is a manufacturing system. In the plant, there is one machine which must process two types of parts. A part of the first type, from producer P1, waits in buffer B1 until it is permitted to enter the processing machine. When the processing is completed, the part goes into the first of two output bins. A part of the second type, from producer P2, waits in buffer B2 until it is permitted to enter the processing machine. When the part comes out of the processing machine, it goes into the second output bin. The regulator must ensure mutual exclusion in the machine, i.e. only one part is processed at a time. The regulator must also ensure that producer P1 gets priority in the use of the machine. The plant state indicates whether there is a part in a given buffer, and whether there is a part of a given type in the machine. The regulator acts to allow parts from the different buffers to enter the machine. The manufacturing system is shown in Fig. 6 below.

We use four-bit binary strings to represent the plant states. The first bit is 1 if there is a P1 part in the machine and 0 otherwise. The second bit indicates the presence or absence of a P2 part in the machine. The third and fourth bits represent

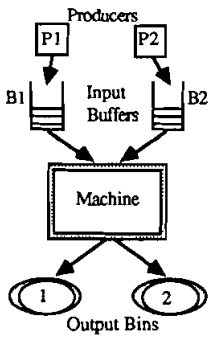

Figure 6. Manufacturing system.

the presence or absence of parts in buffers B1 and B2, respectively. For example, the plant state '1001' indicates that (i) there is a P1 part in the machine, (ii) there is not a P2 part in the machine, (iii) there is not a P1 part in buffer B1, and (iv) there is a P2 part in buffer P2. There are 16 plant states in all. The states of the regulator are q_1 (allowing a part from P1, presently in B1, to enter the machine), q_2 (allowing a part from P2, presently in B2, to enter the machine), and q_3 (not allowing any parts to enter the machine). We have $X = \{0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111\}$ and $Q = \{q_1, q_2, q_3\}$. We suppose that $X_0 = \{0000\}$ and that the initial regulator state is q_3.

We suppose that there is a clock in the plant, and each second, on the second a device provides a measurement of the manufacturing system state and the regulator is triggered to act so as to allow parts to enter the machine. The plant transition function is given by Table 1, and the regulator transition function is given by Table 2.

	Plant input		
Plant state	q_1	q_2	q_3
0000	—	—	0000, 0001, 0010, 0011
0001	—	0100, 0101, 0110, 0111	0001, 0011
0010	1000, 1001, 1010, 1011	—	0010, 0011
0011	1001, 1011	0110, 0111	0011
0100	—	—	0100, 000
0101	—	0100, 0101, 0110, 01111	0001, 0011, 0101, 0111
0110	1000, 1001, 1010, 1011 1100, 1101, 1110, 1111	—	0010, 0011, 0100, 0111
0111	1001, 1011, 1101, 1111	0010, 0011, 0110, 0111	0111, 0011
1000	—	—	0000, 0001, 0010, 0011 1000, 1001, 1010, 1011
1001	—	0100, 0101, 0110, 0111 1100, 1101, 1110, 1111	0001, 0011, 1011, 1001
1010	1000, 1001, 1010, 1011	—	0010, 0011, 1010, 1011
1011	1001, 1011, 1101, 1111	0100, 0101, 0110, 0111 1100, 1101, 1110, 1111	1011, 0011
1100	—	—	*all symbols possible*
1101	—	0100, 0101, 0110, 0111 1100, 1101, 1110, 1111	0001, 0011, 0101, 0111 1001, 1011, 1101, 1111
1110	1000, 1001, 1010, 1011 1100, 1101, 1110, 1111	—	0010, 0011, 0110, 0111 1010, 1011, 1110, 1111
11111	1001, 1011, 1101, 1111	0110, 0111, 1110, 1111	0011, 0111, 1011, 1111

Table 1. Manufacturing system (plant). Possible next plant states are shown.

Regulator state	Regulator input																
	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	
q_1	q_3	q_2	q_1	q_1	q_3	q_3	q_3	q_3	q_3	q_3	q_3	q_3	q_3	q_3	q_3	q_3	q_3
q_2	q_3	q_2	q_1	q_1	q_3	q_3	q_3	q_3	q_3	q_3	q_3	q_3	q_3	q_3	q_3	q_3	q_3
q_3	q_3	q_2	q_1	q_1	q_3	q_3	q_3	q_3	q_3	q_3	q_3	q_3	q_3	q_3	q_3	q_3	q_3

Table 2. Manufacturing system regulator. Next regulator states are shown.

We consider two closed-loop specification formulae. The mutual exclusion specification, which we verify first, is $\varphi = \Box(\neg(1100 \vee 1101 \vee 1110 \vee 1111))$. To avoid confusion, since we now have to deal with output strings whose terms are (finite) strings, we write $x_1, x_2, x_3, \ldots$, instead of $x_1x_2x_3 \ldots$. We also write $(\upsilon; \tau)$ instead of (υ, τ). Some pairs, determining strings in different $\sim_{1,R}$-classes, are: $(\varepsilon; 0000, 0001, 0100)$, $(0000; 0001, 0101)$, $(0000, 0010; 1000)$, and $(0000, 0011; 1011)$. Let α be the string determined by a pair $(\upsilon; \tau)$. Suppose υ has length n_1 and τ has length n_2 and let $m = n_1 + 3n_2$. To show that $(R, \alpha) \models \varphi$ (following Lemma 2, and the definition of satisfaction), it is enough to test that for all $j \leqslant m$ $(R^{\alpha,j}, \alpha^j) \models 1100$ or $(R^{\alpha,j}, \alpha^j) \models 1101$ or $(R^{\alpha,j}, \alpha^j) \models 1100$ or $(R^{\alpha,j}, \alpha^j) \models 1111$. It is clear that for the $(\upsilon; \tau)$ pairs above and all others this is satisfied. Hence the formula is valid.

The priority specification is $\varphi = \Box(0011 \rightarrow \bigcirc(1001 \vee 1011))$. Since this has rank 2, the mechanical procedure would be to examine all pairs on a 2-complete list. Again it is clear just from examining the initial states and transition functions for the plant and regulator that the specification is met. As in the surge tank example, we could take new closed-loop specifications and synthesize regulators to meet them, if possible. For example, we might change the priority from producer P1 to P2. We could also consider a second version of the manufacturing system where the plant states and regulator states are the same as in the first version above but the schedule is different. We could have some device in the plant which drives the regulator to respond whenever the manufacturing system state changes; hence the plant operates in an asynchronous fashion relative to a clock. This will result in a different plant state transition function and interpretation of the priority specification above.

The manufacturing example above is similar to the 'Two Class Parts Processing' example in Thistle and Wonham (1986). There, however, Thistle and Wonham allow an arbitrary finite number of parts of one type to enter the machine. This forces their controller to have an infinite number of states, so that our decision procedure does not apply to their example. There are, however, many practical problems that are finite-state. The examples originally given in Knight and Passino (1987), which are studied above, have also been examined using a branching-time temporal logic framework (Passino and Antsaklis 1988).

Acknowledgments

The authors would like to thank the referee for a number of helpful suggestions and Panos Antsaklis for his help and encouragement. J. F. Knight was partly supported in this work by National Science Foundation grants DMS 85 03353 and DMS 87 01559. K. M. Passino was partly supported by a Jesse H. Jones Foundation research grant, an Arthur J. Schmitt fellowship and the Jet Propulsion Laboratory, under Contract No. 957856.

References

Buchi, J. R., 1962, On a decision method in restricted second order arithmetic. *Proceedings of the 1960 International Congress of Logic, Methodology, and Philosophy of Science*, edited by E. Nagel, P. Suppes, and A. Tarski (Stanford University Press) pp. 1–11.

Fusaoka, A., Seki, H., and Takahashi, K., 1983, A description and reasoning of plant controllers in temporal logic. *Proceedings of the 8th International Conference on Artifical Intelligence*, Vol. 1, pp. 405–408.

Galton, A., (editor), 1987, *Temporal Logics and Their Applications* (New York: Academic Press).

KNIGHT, J. F., and PASSINO, K. M., 1987, Decidability for temporal logic in control theory. *Proceedings of the Twenty-Fifth Allerton Conference on Communication, Control, and Computing*, University of Illinois at Urbana-Champaign, pp. 335–344.

MANNA, Z., and PNUELI, A., 1983, Verification of concurrent programs: a temporal proof system. Report no. STAN-CS-83-967, Department of Computer Science, Stanford University.

MANNA, A., and WOLPER, P., 1981, Synthesis of communicating processes from temporal logic specifications. Report no. STAN-CS-81-872, Department of Computer Science, Stanford University.

OSTROFF, J. S., 1987, Real-time computer control of discrete systems modelled by extended state machines: a temporal logic approach. Ph.D. dissertation, Report No. 8618, Department of Electrical Engineering, University of Toronto, Toronto.

OSTROFF, J. S., and WONHAM, W. M., 1985, A temporal logic approach to real time control, *Proceedings of the 24th Conference on Decision and Control*, Ft. Lauderdale, Florida, pp. 656–657; 1987, State machines, temporal logic and control: a framework for discrete event systems. *Proceedings of the 26th Conference on Decision and Control*, Los Angeles, California, pp. 681–686.

PASSINO, K. M., and ANTSAKLIS, P. J., 1988, Branching time temporal logic for discrete event system analysis. *Proceedings of the Twenty-Sixth Allerton Conference on Communication, Control, and Computing*, University of Illinois at Urbana-Champaign, pp. 1160–1169.

RAMADGE, P. J., and WONHAM, W. M., 1987, Supervisory control of a class of discrete event processes. *SIAM Journal of Control and Optimization*, **25**, 206–230.

THISTLE, J. G., and WONHAM, W. M., 1985, Control problems in a temporal logic framework. *International Journal of Control*, **44**, 943–976.