Model-checking the Preservation of Temporal Properties upon Feature Integration

Dimitar P. Guelev
Section of Logic
Institute of Mathematics and Informatics
Acad. G. Bonchev str.,bl. 8.
1113 Sofia, Bulgaria.
`www.math.bas.bg/~gelevdp/`

Mark Ryan
School of Computer Science,
University of Birmingham,
Birmingham, B15 2TT, UK
`www.cs.bham.ac.uk/~mdr/`

Pierre Yves Schobbens
Institut d'Informatique
Facultés Universitaires de Namur
Belgium
`www.info.fundp.ac.be/~pys/`

February 10, 2005

1 Introduction

The concept of *feature* has emerged as a popular way of structuring user-oriented descriptions of certain kinds of systems. Updating a system by adding new features to it is a technique which enables designs and code to be reused. It started to become popular when telephone companies began to introduce features such as call-forwarding and ring-back-when-free into plain old systems which did not support that functionality. This process of feature addition is well-known to be *non-monotonic*: adding a feature does not necessarily preserve the temporal properties of the system. Usually these features are designed in isolation from one another, and putting several of them together in a phone system may lead to them interfering with each other in undesirable ways. This is known as the 'feature interaction problem', and is currently gaining considerable attention from academic and industrial researchers [CM00, AL03].

Model checking has been used with some success to detect the presence of undesirable feature interactions, or to prove their absence [CM01, PR01, dB99, BZ92]. In a model checking context, feature interaction may be defined as the failure of certain temporal properties of the system incorporating the features. For example, a feature F_2 *breaks a previously introduced feature* F_1 if the system incorporating first F_1 and then F_2 fails to satisfy a temporal property which characterises the correct operation of F_1. This is Type II feature interaction, as defined in [PR01]. Given a system S and a feature F, we write $S + F$ to represent the system S having been updated by the feature F. Type II feature interaction is then written:

$$S + F_1 \models \phi_1 \text{ but } S + F_1 + F_2 \not\models \phi_1$$

where ϕ_1 is a formula which characterises what F_1 was intended to do.

To guarantee that subsequently-introduced features do not break earlier ones, we have to re-check the important properties of earlier features each time we introduce another one. The tool described in [PR01] accepts as input a system S described in the SMV language and a feature F described in the an extension of the SMV language designed to express features. It integrates the feature into the system and outputs SMV code representing $S + F$. SMV can then be used to re-check the important properties of the base system F. Details of this approach and two extended examples (the lift system and the telephone system) can be found in [PR01].

Example 1 *Let POTS stand for the plain old telephone system with no features and consider the following features:*

Call Forward on Busy (CFB): *Whenever the subscriber's line is busy, calls to the subscriber's phone are treated as if they are calls to some other specified phone.*

Terminating Call Screening (TCS): *Calls to the subscriber's phone are rejected if the caller's number appears on a screening list chosen by the subscriber. The caller will hear an announcement to this effect.*

Let φ_{CFB} be the temporal property if the subscriber's phone is busy and another user calls him, the incoming call will terminate at the specified forwarding number.

Then POTS + CFB $\models \varphi_{CFB}$, but POTS + CFB + TCS $\not\models \varphi_{CFB}$. Intuitively, the reason is that in the system POTS + CFB + TCS a user A can set up a call-forward to another subscriber B, who may also set up a call-screen on user C. Now suppose C calls A, who is busy. According to CFB, this call is treated as if it is a call to B. But B screens C, so the call is rejected, violating φ_{CFB}. Details on this example can be found in [PR01].

A difficulty with this approach, however, is that model checking is computationally expensive, and therefore re-checking the same property again and again each time a new feature is introduced is undesirable. It is worthwhile to find methods which avoid these re-checks. Often, the new feature F_2 is intuitively quite orthogonal in function to a previously-introduced feature F_1. Indeed, this should be the case if the features have been well-designed. In this case we expect that the properties established by F_1 will continue to hold after F_2 has been introduced.

In this paper we propose an efficient method to check the preservation of safety properties written in *LTL* upon the addition of features as modelled in terms of finite transition systems. Of course, checking arbitrary *LTL* properties on such systems can be done using the techiques which apply in general. Our aim is to take advantage of the special form of of safety properties and the assumption that the considered basic system has the property in question and develop a more efficient technique. We propose an algorithm which allows to first establish whether the given *LTL* property holds for some given base system. This step of our algorithm produces data which can then be used to establish whether the given property countinues to hold for the combination of the base system with a concrete feature from this class without fully re-examining the transition relation of the base system. We show that a similar method can be applied to certain forms of liveness properties too.

2 Preliminaries

2.1 Descriptions of systems

We assume that observable states of a system S are described as valuations of its set of variables P_S, which we assume to be all boolean for the sake of simplicity. The possible states of S are the valuations of P_S. We denote the set $(P_S \to \{0,1\})$ of these states by W_S. Behaviours of S are infinite sequences

$$b = b_0 b_1 \dots b_n \dots \tag{1}$$

of states $b_i \in W_S$. We define the relation $R_S \subseteq W_S^2$ by putting $R_S(s, s')$ if S can move from s directly to s'. We denote the set of the *initial states* of S by I_S. A sequence of the form (1) is a behaviour of S if and only if $b_0 \in I$ and $R_S(b_i, b_{i+1})$ for all $i < \omega$. To guarantee the infiniteness of behaviours, we require R_S to be *serial*, that is, to satisfy $(\forall s \in W_S)(\exists s' \in W_S) R_S(s, s')$. A system S is described completely by the triple $\langle W_S, I_S, R_S \rangle$. We identify systems with their descriptions of this form.

A state is *accessible*, if it occurs in some behaviour of the respective system. Obviously only accessible states are relevant to the properties of the behaviours of S.

2.2 Linear temporal logic

We assume that the requirements on systems with features are written in propositional Linear Temporal Logic (*LTL*) (cf. e.g. [HR00]) with the past operators included. We need the past operators, in order to use some normal forms for *LTL* requirements which involve them. Past operators are not essential in *LTL* requirements, but avoiding their use can lead to unreasonably long formulations of requirements [LPZ85, LMS02].

Given a vocabulary of propositional variables P, the *LTL* language $\mathbf{L}(P)$ consists of the formulas φ which have the syntax

$$\varphi ::= \top \mid p \mid \varphi \Rightarrow \varphi \mid \bigcirc\varphi \mid \ominus\varphi \mid \varphi \mathsf{U} \varphi \mid \varphi \mathsf{S} \varphi$$

where p stands for a variable P. We consider languages which correspond to systems S and have their respective sets of variables P_S as the vocabulary. The *satisfaction relation* $S, b, n \models \varphi$ is defined between systems S, behaviours b of S of the form (1), *positions* $n < \omega$ in these behaviours and formulas φ from $\mathbf{L}(P_S)$. We omit S and b from $S, b, n \models \varphi$ when they are clear from the context. Given S and b, $\models$ can be defined by the clauses:
$n \not\models \bot$
$n \models p$ iff $b_n(p) = 1$
$n \models \varphi \Rightarrow \psi$ iff either $n \models \psi$ or $n \not\models \psi$
$n \models \bigcirc\varphi$ iff $n + 1 \models \varphi$
$n \models \ominus\varphi$ iff $n \neq 0$ and $n - 1 \models \varphi$
$n \models \varphi \mathsf{U} \psi$ iff there is a $k < \omega$ such that $n + i \models \varphi$ for all $i < k$ and $n + k \models \psi$
$n \models \varphi \mathsf{S} \psi$ iff there is a $k \leq n$ such that $n - i \models \varphi$ for all $i < k$ and $n - k \models \psi$

The symbols $\top$, $\neg$, $\vee$, $\wedge$, $\Rightarrow$ and $\Leftrightarrow$ are used in *LTL* formulas as abbreviations in the usual way. The modalities $\Diamond$, $\Box$, $\diamondminus$ and $\boxminus$ are defined by the clauses:

$$\Diamond\varphi \rightleftharpoons \top \mathsf{U} \varphi, \ \Box\varphi \rightleftharpoons \neg\Diamond\neg\varphi, \ \diamondminus\varphi \rightleftharpoons \top \mathsf{S} \varphi, \ \boxminus\varphi \rightleftharpoons \neg\diamondminus\neg\varphi.$$

We denote the formula $\neg\ominus\top$ by I. I marks the beginning of time:

$$S, b, n \models \mathsf{I} \text{ iff } n = 0.$$

LTL formulas which have no occurrences of $\ominus$, or S, are called *future* formulas. Formulas with no occurrences of $\bigcirc$, or U, are called *past* formulas. Given b and n as above and π a past formula, $b, n \models \pi$ depends only on $b_0, \dots, b_n$. That is why, given a past formula π we put

$$b_0 \dots b_n \models \pi \text{ iff } b_0 \dots b_n \cdot c, n \models \pi$$

for all the (infinite) behaviours of S of the form $b_0 \ldots b_n \cdot c$.

Formulas with no occurrences of temporal operators are called *propositional.* $S, b, n \models \alpha$ depends only on b_n and α, and therefore $S, b, n \models \alpha$ can be abbreviated to $b_n \models \alpha$ for propositional α. Given S and α, we denote the set $\{s \in W_S : s \models \alpha\}$ by $[\![\alpha]\!]_{P_S}$. The subscript $._{P_S}$ indicates that $[\![\alpha]\!]_{P_S}$ depends on the vocabulary of S. We omit it when clear from the context. $[\![\alpha]\!]_{P_S}$ does not depend on other components of S.

S is said to have the *LTL* property φ if $S, b, 0 \models \varphi$ for all behaviours b of S.

2.3 Abbreviations for restrictions of relations and projections of states, etc.

Given a system S, $s \in W_S$ and $P \subseteq P_S$, $s|_P$ stands for the restriction of s to the variables from P. Given a relation $R \subseteq W_S \times W_S$, $R|_U$ and $R|^V$ denote the restrictions $R \cap (U \times W_S)$ and $R \cap (W_S \times V)$ of the binary relation R on W_S to the domain U and the range V, respectively. We denote the complement $W_S \setminus X$ of a subset X of W_S relative to W_S by $\overline{X}$. Similarly, we denote the complement $P_S \setminus P$ of a subset P of P_S relative to P_S by $\overline{P}$.

3 A running example

Here follows an example that we use to illurstrate our methods throughout this paper.

Example 2 *Consider the following simplified version of the n-floor lift system presented in [PR01]. In this version, we suppose there is only two floors. The proposition f inicates whether we are on the upper floor (f) or the lower floor ($\neg f$). We also model the doors of the lift: they can be open (o) or closed ($\neg o$).*

The system $S = \langle W_S, I_S, R_S \rangle$ has four states which we write as pairs representing the values of f and o. In the state 01, f is false and o is true, etc. In the base system, all possible transitions are allowed, as illustrated in Figure 1(a). Thus the system is given by: $W_S = \{00, 01, 10, 11\}$; $I_S = \{00\}$; and $R_S = W_S \times W_S$.

We consider two features of this system:

- *F_1: the lift will not move between floors unless the doors are closed.*
- *F_2: the lift may park on the lower floor. (Real lift systems often have the feature of parking on the ground floor, to improve performance in periods of peak upward traffic.)*

The system $S + F_1$, i.e. the base system with the first feature integrated, is shown in 1(b). We will not treat F_1 in detail. Full details will be given later for F_2.

Figure 1: The lift system. (a) The base system S. (b) S with the first feature F_1 integrated: $S + F_1$.

4 Features

Informally, a feature is an addition to a system of limited calibre meant to improve the functionality of the system. The result of integrating a feature F into a system S, which is an (enhanced) system, is denoted by $S + F$. F can bring in its own variables upon integration into a S. The behaviours of S and $S+F$ can also differ as observed in terms of the variables of S. A system can undergo the successive integration of several features. A feature F which both adds variables and changes behaviour can be seen as a pair of features F_1 and F_2 to be integrated successively, F_1 being just an addition of variables, and F_2 carrying both the description of the behaviour of the new variables and the changes to the behaviour of the base system, but no more new variables. Clearly, properties of $S+F_1+F_2$ written in the vocabulary P_S can only be affected upon adding F_2. In this paper we restrict ourselves to features like F_2, which only change behaviour without contributing variables. If F has this form, then $P_{S+F} = P_S$ and $W_{S+F} = W_S$. We assume $I_{S+F} = I_S$ for the sake of simplicity too. Then the integration of F amounts to replacing R_S by a new transition relation R_{S+F}.

A feature F affects the working of its base system S only at transitions at which it becomes *triggered*. Let the current state of $S+F$ be s and $R_S(s,s')$ for some $s' \in W_{S+F}$. Then, unless F is triggered, $S+F$ can simply make the transition $\langle s, s'\rangle$. F can be triggered by a condition on s, on s', or on both s and s'. In this paper we focus on F which have triggering conditions of the first two kinds and call them *precomposed* and *postcomposed* features, respectively. The triggering condition of such an F is a propositional formula. We denote it by c_F and call it the *guard of* F.

In general it would be too crude to assume that the triggering of a feature F can affect all the variables of $S+F$. That is why we assume that the description of F includes the set of the variables P_F which F can update differently from S when triggered. The effect of a feature F on a pending transition $\langle s_1, s_2\rangle \in R_S$ is as follows:

A *precomposed* F evaluates its guard c_F at state s_1. If $s_1 \models c_F$, then F cancels the transition to s_2 and first takes $S+F$ to some other state s'_1 such that an appropriate relation R_F holds between s_1 and the restriction $s'_1|_{P_F}$ of s'_1 to the variables from P_F which F is allowed to change when triggered. The values of the variables outside P_F remain the same upon the transition from s_1 to s'_1. Then F allows a transition from s'_1 to be made by S. The externally observed transition resulting from this is from s_1 to the state s'_2 to which S takes $S+F$ from s'_1.

A *postcomposed* F evaluates its guard c_F at the destination state s_2 of the pending transition $\langle s_1, s_2\rangle$. If $s_2 \models c_F$, then F prevents the transition to s_2 from being observed. Instead it uses s_2 to choose a state s'_2 such that $R_F(s_2, s'_2|_{P_F})$ and the values of the variables from $\overline{P_F}$ at s'_2 are the same as at S_2. The externally observed transition is from s_1 to s'_2 again.

A feature F can be described as the triple $\langle c_F, P_F, R_F\rangle$, where $R_F \subseteq W_{S+F} \times (P_F \to \{0,1\})$ is the relation describing the F-specific updates of the variables from P_F in transitions which trigger F. It can be assumed that $\mathrm{dom} R_F$ is exactly $[\![c_F]\!]$. Given $\langle c_F, P_F, R_F\rangle$ and S, we can define R_{S+F} by the equalities

$$R_{S+F} = R_S|_{\overline{[\![c_F]\!]}} \cup R'_F \circ R_S \text{ for precomposed } F, \tag{2}$$

and

$$R_{S+F} = R_S|^{\overline{[\![c_F]\!]}} \cup R_S \circ R'_F \text{ for postcomposed } F, \tag{3}$$

where R'_F is defined by the equivalence

$$R'_F(s,s') \leftrightarrow R_F(s, s'|_{P_F}) \wedge s'|_{\overline{P_F}} = s|_{\overline{P_F}}. \tag{4}$$

Note that both the class of precomposed features and that of postcomposed features contain a *neutral* feature, which can be represented using the relation

$$Id_{c_F,P_F}(s,s') \leftrightarrow s \in [\![c_F]\!] \wedge s' = s|_{P_F} \tag{5}$$

as R_F. Note that extending Id_{c_F,P_F} to a relation from W_S to W_S gives the identity relation on W_S.

Example 3 *The feature F_2 adds the capability of the lift to park on the lower floor. It is triggered in the state 10, when the lift is on the upper floor and the doors are closed. It is post-composed with the system: this means that when the system enters the state 10, it may silently move into the parked state 00. Thus, the feature $F_2 = \langle c_{F_2}, P_{F_2}, R_{F_2} \rangle$ is given as follows:*

$$\begin{array}{rcl} c_{F_2} & = & \{10\} \\ P_{F_2} & = & \{f\} \\ R_{F_2} & = & \{(10,10),(10,00)\}. \end{array}$$

The system $S + F_2$ is illustrated in Figure 2.

Figure 2: The lift system, continued. The third bit of the states represents the value of h. (a) The systems S' (solid arrows) and F_2' (dotted arrows). (b) $S' + F_2'$.

4.1 Canonical safety formulas

A set of behaviours B is a safety property iff the possibility to extend every finite prefix $b_0 \dots b_n$ of a behaviour b to an infinite behaviour $b_0 \dots b_n \cdot c$ which is in B implies that b itself is in B. In the rest of the paper we assume that the safety properties in question are written as *canonical safety formulas* which were introduced in [MP89] and have the form

$$\Box\pi \tag{6}$$

where π is a past formula. Every *LTL* formula which expresses a satefy property is equivalent to a canonical safety formula [MP89].

Example 4 *The formula φ_{F_2} representing the desired property introduced by the feature F_2 expresses that the lift cannot have moved during a period in which the doors are open:*

$$\varphi_{F_2} = \bigwedge_{g \in \{f, \neg f\}} \Box(o \wedge g \Rightarrow g\mathsf{S}(\neg o \wedge g)).$$

4.2 Projection in *LTL*

In Section 6 below we argue that it is convenient to make the invisible states which are involved in the working of precomposed and postcomposed features visible, that is, to have these intermediate states occur explicitly in the behaviours of systems with features. The derived operator of *projection* in *LTL* that we introduce below formalises the transformation of properties written with the assumption that the intermediate states are not visible into their equivalent properties of behaviours which include the intermediate states. This operator is analogous to a projection operator introduced to interval temporal logic in [HMM83] where it was denoted by Π. The language FORSPEC [AFF$^+$02] has a similar construct. We present it here in detail for the sake of self-containedness.

Given two *LTL* formulas φ and ψ, we denote the *projection of φ onto ψ* by $\varphi\Pi\psi$. Roughly speaking, a behaviour b satisfies $\varphi\Pi\psi$ if removing from b the states which are at positions i in b such that $b, i \not\models \psi$ produces a behaviour which satisfies φ. This makes sense only if b contains infinitely many positions which satisfy ψ, which is equivalent to $b, 0 \models \Box\Diamond\psi$. Here follows the precise definition:

Definition 1 *Let $\varphi, \psi \in \mathbf{L}(P_S)$ and b be a behaviour of S and $k < \omega$. Let $b, 0 \models \Box\Diamond\psi$. Let the infinite ascending sequence $i_0, i_1, \dots, i_n, \dots$ consist of the natural numbers i such that $b, i \models \psi$. Then*

$$b, k \models \varphi\Pi\psi \textit{ iff } b', k' \models \varphi,$$

where $b' = b_{i_0}, b_{i_1}, \dots, b_{i_n}$ and $k' = \min\{i_n : i_n \geq k, n < \omega\}$.

The operator Π is definable in *LTL*. Indeed, the following equivalences are sufficient to eliminate projection from any *LTL* formula:

$$\begin{array}{lcl}
\bot\Pi\psi & \Leftrightarrow & \bot \\
p\Pi\psi & \Leftrightarrow & (\neg\psi)\mathsf{U}(p \wedge \psi) \wedge \Box\Diamond\psi \\
(\varphi_1 \Rightarrow \varphi_2)\Pi\psi & \Leftrightarrow & (\varphi_1\Pi\psi \Rightarrow \varphi_2\Pi\psi) \wedge \Box\Diamond\psi
\end{array}$$

$$\begin{array}{lcl}
(\bigcirc\varphi)\Pi\psi & \Leftrightarrow & (\neg\psi)\mathsf{U}(\psi \wedge \bigcirc(\varphi\Pi\psi)) \\
(\ominus\varphi)\Pi\psi & \Leftrightarrow & (\neg\psi)\mathsf{S}(\psi \wedge \ominus(\varphi\Pi\psi)) \\
(\varphi_1\mathsf{U}\varphi_2)\Pi\psi & \Leftrightarrow & (\varphi_1\Pi\psi)\mathsf{U}(\varphi_2\Pi\psi) \\
(\varphi_1\mathsf{S}\varphi_2)\Pi\psi & \Leftrightarrow & (\varphi_1\Pi\psi)\mathsf{S}(\varphi_2\Pi\psi)
\end{array}$$

These equivalences suggest an extension of Definition 1 which applies to all ψ and defines $\varphi\Pi\psi$ to be false in case $\Box\Diamond\psi$ is false.

Note that if φ represents a safety property, then so does $\varphi\Pi\psi$, regardless of ψ. This can be easily seen using that φ can be written as a canonical safety formula.

Example 5 *Let us compute the projection of $\bigwedge\limits_{g \in \{f, \neg f\}} \Box(o \wedge g \Rightarrow g\mathsf{S}(\neg o \wedge g))$, which is the requirement φ_{F_2} on F_2, onto some unspecified formula ψ. Since $(\alpha \wedge \beta)\Pi\psi$ is always equivalent to $\alpha\Pi\psi \wedge \beta\Pi\psi$ and $(\Box\alpha)\Pi\psi$ is always equivalent to $\Box\Diamond\psi \wedge \Box(\alpha\Pi\psi)$, $\left(\bigwedge\limits_{g \in \{f, \neg f\}} \Box(o \wedge g \Rightarrow g\mathsf{S}(\neg o \wedge g))\right)\Pi\psi$ is equivalent to*

$$\Box\Diamond\psi \wedge \bigwedge_{g \in \{f, \neg f\}} \Box(((\neg\psi)\mathsf{U}(o \wedge g \wedge \psi)) \Rightarrow (((\neg\psi)\mathsf{U}(g \wedge \psi))\mathsf{S}((\neg\psi)\mathsf{U}(\neg o \wedge g \wedge \psi))). \tag{7}$$

5 Checking safety property satisfaction by base systems

In this section we describe the first part of our model-checking algorithm, which includes checking that the property whose preservation is in question holds for the considered base system.

Let the system $S = \langle W_S, I_S, R_S \rangle$ be fixed for the rest of the section and P_S stand for its vocabulary. Consider a safety property in $\mathbf{L}(P_S)$ written as the canonical safety formula $\Box\pi$. Obviously S satisfies $\Box\pi$ if and only if every finite path in it satisfies π. Note that this condition cannot be straightforwardly simplified to a condition on the individual states of S, because a state can be reachable by many paths, each satisfying different past formulas. However some simplification is still possible due to the following observation:

> Let Φ be the set of the subformulas of π which have either S or $\ominus$ as their main connective, possibly including π itself. Then the relation $b_0 \dots b_{n-1}b_n \models \varphi$ for $\varphi \in \Phi$ depends only on b_{n-1}, b_n and the set of the formulas $\psi \in \Phi$ such that $b_0 \dots b_{n-1} \models \psi$.

Given a subset Ξ of Φ and a pair of states $s, s' \in W_S$, in the sequel we use $\Phi(s, s', \Xi)$ to denote the set of the formulas from Φ which would be satisfied by any behaviour $b_0 \dots b_n s' s$ such that $\{\varphi \in \Phi : b_0 \dots b_n s' \models \varphi\} = \Xi$.

Consider a mapping $l_\Phi : W_S \to 2^{2^\Phi}$. Let $\Xi \in l_\Phi(s)$ if and only if there is a finite behaviour $b_0 \dots b_n$ of S such that $s = b_n$ and $\Xi = \{\varphi \in \Phi : b_0 \dots b_n \models \varphi\}$. Obviously l_Φ can be obtained as the least fixed point of the system of equations:

$$l_\Phi(s) = \{\{\varphi \in \Phi : \vdash_{LTL} \mathsf{I} \Rightarrow \varphi\}\} \cup \{\Phi(s, s', \Xi) : s' \in R_S^{-1}(s), \Xi \in l_\Phi(s')\} \quad \text{for } s \in I_S;$$

$$l_\Phi(s) = \{\Phi(s, s', \Xi) : s' \in R_S^{-1}(s), \Xi \in l_\Phi(s')\} \text{ for } s \in W_S \setminus I_S. \tag{8}$$

Note that $l_\Phi(s) \neq \emptyset$ iff s is a reachable state. Using l_Φ, we can formulate the following obvious criterion for the satisfaction of $\Box\pi$ by S:

Example 6 $\Phi_{F_2} = \{f\mathsf{S}(\neg o \wedge f)\}$. *To compute $l_{\Phi_{F_2}}(s)$, we examine paths $b_0 \dots b_n$ such that $b_n = s$. For each such path, we check whether it satisfies $f\mathsf{S}(\neg o \wedge f)$. If it does, we add $\{f\mathsf{S}(\neg o \wedge f)\}$ to $l_{\Phi_{F_2}}(s)$, otherwise we add $\emptyset$ to $l_{\Phi_{F_2}}(s)$.*

Proposition 1 $S = \langle W, I, R\rangle$ *satisfies $\Box\pi$ if for all $s \in W_S$ either $l_\Phi(s) = \emptyset$ or π is a propositional consequemce of each $\Xi \in l_\Phi(s)$.*

Since π is a boolean combination of formulas from Φ, whether it follows from some $\Xi \subset \Phi$ in propositional logic can be decided immediately.

Our algorithm for checking the preservation of safety properties is based on the way feature-contributed transitions affect the mapping l_Φ defined above. A feature preserves the safety property $\Box\pi$ only if it does not contribute transitions which violate the criterion from Proposition 1. In the rest of the paper we work out the technical details to develop this idea.

6 Separating system- and feature-contributed transitions

The definition (2) of R_{S+F} for precomposed F shows that the states s of S can be partitioned into three subsets with respect to the possible outgoing transitions of $S + F$:

$s \not\models c_F$;
$s \models c_F$ and s triggers F;
$s \models c_F$, but s does not trigger F, because F made the transition *to s*.

In general, states from the second and the third kinds cannot be told apart out of the context of particular behaviours. States from the third set do not occur in observable behaviours, according to our definition of the working of precomposed features. However, (2) suggests that being aware of these states can simplify the separation between the contributions of F and S to the behaviour of $S + F$. We transform the descriptions of S and F so that these states become observable. This facilitates the considered separation at the cost of one additional variable, which we call h (for *h*idden). The components of the transformed descriptions S' and F' of S and F, respectively, are defined as follows:

$P_{S'} = P_S \cup \{h\}$ and $P_{F'} = P_F \cup \{h\}$;
$I_{S'} = \{s \in W_{S'} : s|_{P_S} \in I_s, s \not\models h\}$;
$c_{F'} \rightleftharpoons c_F \wedge \neg h$;
$R_{S'}(s, s') \leftrightarrow R_S(s|_{P_S}, s'|_{P_S}) \wedge (s' \notin [\![h]\!])$;
$R_{F'}(s, s') \leftrightarrow R_F(s|_{P_S}, s'|_{P_F}) \wedge (s \notin [\![h]\!]) \wedge (s' \in [\![h]\!])$.

In words, $R_{S'}$ takes $S' + F'$ from any state to a visible state, F becomes triggered only at visible states and $R_{F'}$ takes $S' + F'$ to hidden states. In all other aspects $R_{S'}$ and $R_{F'}$ are like R_S and R_F, respectively. Obviously a sequence of states $s_0 s_1 \dots s_n \dots$ is a behaviour of $S + F$ iff a behaviour of $S' + F'$ can be obtained from it by appropriately inserting states which satisfy h and setting the value of h at the original states to 0. $S + F$ satisfies an *LTL* property φ iff $S' + F'$ satisfies $\varphi \Pi \neg h$.

Symmetrically, S' and F' can be defined for postcomposed F as follows:

$$\begin{aligned}
&I_{S'} = \{s \in W_{S'} : s|_{P_S} \in I_s, s \models h\};\\
&c_{F'} \rightleftharpoons c_F \wedge h;\\
&R_{S'}(s, s') \leftrightarrow R_S(s|_{P_S}, s'|_{P_S}) \wedge (s' \in [\![h]\!]);\\
&R_{F'}(s, s') \leftrightarrow R_F(s|_{P_S}, s'|_{P_F}) \wedge (s \in [\![h]\!]) \wedge (s' \not\in [\![h]\!]).
\end{aligned}$$

$P_{S'}$ and $P_{F'}$ are as for precomposed F. $S + F$ satisfies an *LTL* property φ iff $S' + F'$ satisfies $\varphi \Pi \neg(h \wedge c_F)$ for postcomposed F.

Moving to S' and F' and the assumption of the visibility of all states leads to the simple form

$$R_{S'+F'} = R_{S'}|_{\overline{[\![c_{F'}]\!]}} \cup R'_{F'} \tag{9}$$

of both (2) and (3), where $R'_{F'}$ is as in (4).

7 Checking preservation of safety properties upon the integration of a feature

In this section we use Proposition 1 to derive criteria for the preservation of given safety property upon the addition of a feature of the form described above. Like in the previous sections, let $S = \langle W_S, I_S, R_S \rangle$ be a fixed system with vocabulary P_S and $F = \langle c_F, P_F, R_F \rangle$ be a fixed feature to be integrated into S. Let $\Box\pi \in \mathbf{L}(P_S)$ be a canonical safety formula for the property in question. Consider the system S' obtained from S by introducing the variable h and defining $I_{S'}$ and $R_{S'}$ as in Section 6. Our basic idea is to use the labelling l_Φ where Φ consists of the subformulas of π which have a temporal operator as their main connective, as defined in Section 4.1. Since S' satisfies the considered safety property $\Box\pi$, the labelling for S' alone should satisfy the conditions of Proposition 1. To check whether $S' + F'$ satisfies $\Box\pi$, one can to use the corresponding labelling for $S' + F'$. However, this would amount to constructing $S' + F'$ and doing the model-checking from scratch. A sufficient condition for F' not to break $\Box\pi$ can be established more easily as follows:

1. Assume that no S'-contributed transitions get cancelled upon the addition of F' and start from a precalculated l_Φ for S'.

2. Add the transitions contributed by F', that is form $R_{S'} \cup R'_{F'}$ instead of the exact transition relation $R_{S'}|_{\overline{[\![c_{F'}]\!]}} \cup R'_{F'}$ of $S' + F'$ given in (9).

3. Check whether the added transitions cause the labelling to be changed by applying the equations (8) with the now extended sets of predecessor states occurring on the right of $=$ in them.

Example 7 *The system with transitions $R_{S'} \cup R'_{F'_2}$ may be visualised by considering both solid and dotted transitions in Figure 2(a). All paths of this system satisfy the projection of φ_{F_2} onto $f \wedge \neg o \wedge h$, that is, (7) with ψ being $f \wedge \neg o \wedge h$, which happens to be logically equivalent to $\Box\Diamond(f \wedge \neg o \wedge h)$.*

If the labelling is not changed, then it can be concluded that the addition of the considered feature preserves the property in question immediately.

Depending on the desired precision, step 3 can be carried out either only on the states which are reachable by a single feature contributed transition, or can be iterated with revising the labelling of each state which is the destination of a transition starting from a state whose labelling has been changed at the previous step, thus obtaining a labelling for an over-approximation of $S' + F'$ with transition relation $R_{S'} \cup R'_{F'}$. Because this is an over-approximation, it may be the case that $S' + F'$ satisfies $\Box\pi$ but this fact cannot be proved by our method. This will arise, for example, if the system choses a path which is a mixture of a path of the original system and a path of the featured system, by executing part but not all of the feature. This can happen if the feature shares states with the system other than the feature's triggering or final state. In practice, such cases are likely to be rare.

The conclusion that if no labels become changed, then the feature preserves the property, follows from the assumption that the basic system satisfies the property, which means that the initial labelling satisfies the conditions of Proposition 1. In case some labels get changed, one needs to iterate step 3, in order to reach a conclusion. If the obtained labelling turns out to satisfy Proposition 1, then again it can be concluded that $S' + F'$ satisfies (the suitably projected counterpart of) $\Box\pi$. However, even if a labelling which does not satisfy Proposition 1 is obtained, it cannot be implied that F breaks $\Box\pi$. The reason for this is that when calculating the extension of the labelling it is not taken in account whether all the considered states are still reachable. States can be rendered unreachable upon adding F', because F' cancels some of the transitions of the base system S'.

As far as properties which can be written without the use of h are concerned, S' is equivalent to S. However, unlike S, S' has plenty of states which can only be reached by adding a feature which, e.g. in the case of precomposed features, adds transitions to states which satisfy h. Since unreachable states are always labelled by $\emptyset$, this may cause an avalanche of otherwise benign changes to the labelling function for S' and thus make it impossible to obtain the new labelling within a reasonable number of steps. That is why, instead of starting from a labelling for S' it is more efficient to start from a labelling for $S' + F'_0$, where $F_0 = \langle c_F, P_F, Id_{c_F,P_F} \rangle$ is the neutral feature with Id_{c_F,P_F} defined as in (5). Adding F_0 would cause all the states which differ from the visible states by just being invisible, that is, by satisfying h, to be labelled in a way that is similar to that for their corresponding visible states, which gives a better initial approximation for the target labelling.

8 Checking the preservation of some kinds of liveness properties

Despite that the above technique cannot be immediately applied to liveness properties, appropriate labelling can help to check the preservation of such properties in certain special cases. In this section we describe a variant of the labelling which works for properties of the form

$$\Box\Diamond\pi \tag{10}$$

where π is propositional or, more generally, a past formula.

Consider a base system $S = \langle W_S, R_S, I_S \rangle$ like before and assume that π is propositional for the sake of simplicity. Let $l_\pi : W_S \to \mathbf{N} \cup \{\infty\}$ be defined by the clauses:

1. If every infinite sequence of transitions starting from s visits a state which satisfies π, then $l_\pi(s)$ is the length of the longest finite such sequence which does not visit a state which satisfies π.

2. If there is an infinite sequence of transitions starting from s and going through states none of which satisfies π, then $l_\pi(s) = \infty$.

Since every sequence of transitions of S with length greater than $|W|$ contains a loop, $\operatorname{ran} l_\pi \subseteq \{0, \dots, |W| - 1\} \cup \{\infty\}$.

The labelling l_π can be easily calculated using the following rule:

$$\text{If } s \models \pi, \text{ then } l_\pi(s) = 0, \text{ otherwise } l_\pi(s) = 1 + \max\{l_\pi(s') : s' \in R_S(s)\} \tag{11}$$

Obviously a system S satisfies (10) iff $l_\pi(s) < \infty$ for all $s \in W_S$.

Now assume that a feature F is given and that $S' = \langle W_{S'}, R_{S'}, I_{S'} \rangle$ has been obtained from S as described in Section 6. The property of S' which corresponds to (10) would be $\Box\Diamond\pi'$, where, depending on whether the features in question are precomposed or postcomposed, π' is either $\pi \wedge \neg h$ or $\pi \wedge (\neg h \wedge c_F)$, c_F being the triggering condition of the considered feature. Let $F' = \langle c_{F'}, P_{F'}, R_{F'} \rangle$ be obtained from F as in Section 6. We have the following sufficient condition the preservation of (10):

Proposition 2 *Let $l_{\pi'}(s) < \infty$ for all $s \in W_{S'}$. Then, if $R_{F'}(s, s'|_{P_{F'}})$ and $s|_{\overline{P_{F'}}} = s'|_{\overline{P_{F'}}}$ imply $l_{\pi'}(s') + 1 \leq l_{\pi'}(s)$ for every $s \in W_{S'}$, $S' + F'$ satisfies $\Box\Diamond\pi'$, and, consequently, $S + F$ satisfies (10).*

The more general case of π being a past formula can be reduced to the propositional case, because past formulas can be modelled by using additional propositional variables and making the transitions update them appropriately. Let us describe this in detail below for the sake of self-containedness.

Let Φ be the set of all the subformulas of π with a temporal operator as the main connective, as in Section 5 and p_φ be a fresh propositional variable for each $\varphi \in \Phi$. Let S_Φ be a system with $P_\Phi = P \cup \{p_\varphi : \varphi \in \Phi\}$ as its vocabulary. Let

$$I_{S_\Phi} = \{s \in W_{S_\Phi} : s|_P \in I_S, s(p_\varphi) = 1 \text{ iff } \vdash_{LTL} \mathsf{I} \Rightarrow \varphi \text{ for } \varphi \in \Phi\}$$

and

$$R_{S_\Phi}(s, s') \leftrightarrow R_S(s|_P, s'|_P) \wedge s'(p_\varphi) = 1 \text{ iff } \varphi \in \Phi(s|_P, s'|_P, \{\varphi' \in \Phi : s(p_{\varphi'}) = 1\})$$

for all $\varphi \in \Phi$ where $\Phi(.,.,.)$ is as in Section 5. A direct check shows that S and S_Φ have the same behaviours as far as properties definable in $\mathbf{L}(P_S)$ are concerned. Furthermore, a finite behaviour $b_0 \ldots b_n$ of S_Φ satisfies a past formula $\varphi \in \Phi$ iff its last state b_n satisfies p_φ. This means that π has a propositional equivalent in S_Φ which can be built using some of the variables from Φ.

9 Concluding remarks

We presented a method for checking whether a system with a feature continues to satisfy a property which held of the base system. This allows us to verify feature interaction of the four types described in [PR01] more efficiently.

The method is sound, meaning that if it concludes that the property is satisfied in the featured system then it really is satisfied there. However, in general it is not complete; that is, it may not be able to conclude that a property holds of a featured system even if it does hold. In the case of safety properties, we gave an intuitive explanation of why it is likely to be complete in practice, though the method is certainly less often complete in the case of liveness properties.

As future work, we intend to improve the reasoning in the case of liveness properties, and to analyse the examples given in our earlier work using this method.

References

[AFF$^+$02] R. Armoni, L. Fix, A. Flaisher, R. Gerth, B. Ginsburg, T. Kanza, A. Landver, S Mador-Haim, Eli Singerman, A. Tiemeyer, M. Y. Vardi, and Y. Zbar. The forspec temporal logic: A new temporal property-specification language. In *Proceedings of TACAS'02*, volume 2280 of *LNCS*, pages 296–311. Springer, 2002.

[AL03] D. Amyot and L. Logrippo, editors. *Feature Interactions in Telecommunications and Software Systems VII.* IOS Press, 2003.

[BZ92] L.G. Bouma and J. Zuidweg. Formal analysis of feature interactions by model checking. In *Proceedings First International Workshop on Feature Interactions in Telecommunications Systems*, St. Petersburg, FL, U.S.A., December 1992.

[CM00] M. Calder and E. Magill, editors. *Feature Interactions in Telecommunications and Software Systems VI.* IOS Press, 2000.

[CM01] M. Calder and A. Miller. Using spin for feature interaction analysis – a case study. In *Proceedings of The 8th International SPIN Workshop on Model Checking of Software (SPIN'2001)*, volume 2057 of *LNCS*, pages 143–162, Toronto, Canada, May 2001.

[dB99] L. du Bousquet. Feature interaction detection using testing and model-checking experience report. In *Proceedings of the Wold Congress on Formal Methods in the Development of Computing Systems*, volume 1 of *Lecture Notes In Computer Science*, pages 622–641, 1999.

[HMM83] J. Halpern, Z. Manna, and B. Moszkowski. A Hardware Semantics Based on Temporal Intervals. In *Proceedings of ICALP'83*, volume 154 of *LNCS*, pages 278–291. Springer, 1983.

[HR00] M. R. Huth and M. D. Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems.* Cambridge University Press, 2000.

[LMS02] F. Laroussinie, N. Markey, and Ph. Schnoebelen. Temporal logic with forgettable past. In *17th Annual IEEE Symposium on Logic in Computer Science (LICS'02)*, pages 383–392. IEEE Computer Society Press, 2002.

[LPZ85] O. Lichtenstein, A. Pnueli, and L. Zuck. The glory of the past. In *Proceedings of the Confenerence on Logic of Programs*, volume 193 of *LNCS*, pages 196–218. Springer, 1985.

[MP89] Z. Manna and A. Pnueli. The anchored version of the temporal framework. In J.W. De Bakker, W.-P. de Roever, and G. Rozenberg, editors, *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, volume 354 of *LNCS*, pages 201–284. Springer, 1989.

[PR01] M. C. Plath and M. D. Ryan. Feature integration using a feature construct. *Science of Computer Programming*, 2001.