LTL

Syntaxe: $P \in AP$

$\phi,\psi ::= P \mid \neg\phi \mid \phi\vee\psi \mid \phi\wedge\psi \mid \mathbf{X}\,\phi \mid \psi\mathbf{U}\phi \mid \mathbf{X}^{-1}\,\phi \mid \psi\mathbf{S}\phi$

U = until **S** = since

$\rho,i \models \mathbf{X}\,\phi$ ssi $\rho,i+1 \models \phi$

$\rho,i \models \psi\mathbf{U}\phi$ ssi ($\exists\, j \geq i.$ ($\rho,j \models \phi$ et $\forall\, i\leq k<j$ on a $\rho,k \models \psi$)

$\rho,i \models \mathbf{X}^{-1}\,\phi$ ssi ($i>0$ et $\rho,i-1 \models \phi$)

$\rho,i \models \psi\mathbf{S}\phi$ ssi ($\exists\, j \leq i.$ ($\rho,j \models \phi$ et $\forall\, j<k\leq i$ on a $\rho,k \models \psi$)

Exemples de formules

$\mathbf{F}\,\phi$ = ?

$\mathbf{F}\,\phi = \top\,\mathbf{U}\,\phi$

def:

$\rho,i \models \mathbf{F}\,\phi$ ssi ($\exists\, j \geq i.$ $\rho,j \models \phi$)

$\rho,i \models \psi\mathbf{U}\phi$ ssi ($\exists\, j \geq i.$ ($\rho,j \models \phi$ et $\forall\, i\leq k<j$ on a $\rho,k \models \psi$)

$\rho,i \models \top\mathbf{U}\phi$ ssi ($\exists\, j \geq i.$ ($\rho,j \models \phi$ et $\forall\, i\leq k<j$ on a $\rho,k \models \top$)

$\iff$

($\exists\, j \geq i.$ $\rho,j \models \phi$)

$\iff$

$\rho,i \models \mathbf{F}\phi$

Exemples de formules

$\mathbf{F}^{-1}\,\phi = \top\,\mathbf{S}\,\phi$

def:

$\rho,i \models \mathbf{F}^{-1}\,\phi$ ssi ($\exists\, j \leq i.$ $\rho,j \models \phi$)

$\rho,i \models \psi\mathbf{S}\phi$ ssi ($\exists\, j \leq i.$ ($\rho,j \models \phi$ et $\forall\, j<k\leq j$ on a $\rho,k \models \psi$)

$\rho,i \models \top\mathbf{S}\phi$ ssi ($\exists\, j \leq i.$ ($\rho,j \models \phi$ et $\forall\, j<k\leq i$ on a $\rho,k \models \top$)

$\iff$

($\exists\, j \leq i.$ $\rho,j \models \phi$)

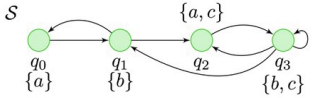
$\iff$

$\rho,i \models \mathbf{F}^{-1}\,\phi$

Exercice 2 :
On considère le STE $\mathcal{S}$ de la figure ci-contre (q_0 est l'état initial). Pour chacune des formules suivantes, dire si la formule est vraie pour $\mathcal{S}$ (*ie* pour toutes ses exécutions). Justifier les réponses.

$\mathbf{X}a$ $\mathbf{X}b$ $\mathbf{F}a$ $\mathbf{G}a$ $\mathbf{GF}a$ $\mathbf{GF}c$ $\mathbf{G}(a \Rightarrow \mathbf{X}b)$
$\mathbf{FG}c \Rightarrow \mathbf{G}(a \Rightarrow \mathbf{X}b)$ $\mathbf{FG}c \Rightarrow \mathbf{G}(b \Rightarrow \mathbf{X}a)$ $\mathbf{GF}b$

Evaluer les formules



Xa ✗ $q_0 \xrightarrow{b} q_1 \ldots$

Xb ✓ q_0 est tjs suivi par q_1 (b)

Fa ✓ q_0 vérifie a !

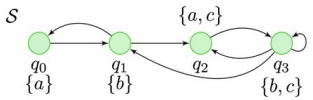
Ga ✗ $q_1 \not\models a$

GFa ✗ on peut boucler en q_3 ...

GFc ✗ on peut boucler en $q_0 q_1$.

Exercice 2 : *Evaluer les formules*

On considère le STE $\mathcal{S}$ de la figure ci-contre (q_0 est l'état initial). Pour chacune des formules suivantes, dire si la formule est vraie pour $\mathcal{S}$ (*ie* pour toutes ses exécutions). Justifier les réponses.



$\mathbf{X}a \quad \mathbf{X}b \quad \mathbf{F}a \quad \mathbf{G}a \quad \mathbf{GF}a \quad \mathbf{GF}c \quad \mathbf{G}(a \Rightarrow \mathbf{X}b)$
$\mathbf{FG}c \Rightarrow \mathbf{G}(a \Rightarrow \mathbf{X}b) \quad \mathbf{FG}c \Rightarrow \mathbf{G}(b \Rightarrow \mathbf{X}a) \quad \mathbf{GF}b$

G (a ⇒ X b) ✓ tout état ⊨ a est suivi par un état ⊨ b
$q_0 \to q_1$ $q_2 \to q_3$

FG c ⇒ G(a ⇒ X b) ✓ car G(a ⇒ Xb) est vrai !

FG c ⇒ G(b ⇒ X a) ✗ car boucle q_3^ω

GF b ✓ car tout circuit contient b
→ passe par q_1 et/ou q_3.

$S \models \phi \Rightarrow \psi$?

Attention…

$S \models (\phi \Rightarrow \psi)$

si et seulement si

toute exécution de S qui vérifie ϕ, vérifie aussi ψ.

Pourquoi utiliser la logique temporelle ?

- une bonne expressivité
- → on peut exprimer beaucoup de choses
- une sémantique naturelle,
- → facilement et succinctement
- de bonnes propriétés de décision
- → des algorithmes et des outils
- beaucoup d'extensions
- → pour les systèmes probabilistes, temps-réel, les jeux, les données,…

Problèmes de vérification

Model-checking:
input: un modèle (STE) **S** et une formule φ
output: oui ssi $\mathbf{S} \models \varphi$.

Satisfaisabilité:
input: une formule φ
output: oui ssi il existe un modèle **S** t.q. $\mathbf{S} \models \varphi$.
(+ **S** si il existe !)

Synthèse de contrôleur:
input: un modèle partiel **S** une formule φ
output: un « controleur » $\boldsymbol{C}$ t.q. $\mathbf{S} \times \boldsymbol{C} \models \varphi$.

Installer PRISM

http://www.prismmodelchecker.org

Spécification d'un ascenseur

Up and Down The Temporal Way

H. BARRINGER*
Department of Computer Science, University of Manchester, Oxford Road, Manchester M13 9PL

A formal specification of a multiple-lift system is constructed. The example illustrates and justifies one of many possible system specification styles based on temporal techniques.

Received September 1985

1. INTRODUCTION

Over the last decade there has been widespread research directed at obtaining techniques for the analysis, specification and development of concurrent systems. Several of these lines of research have led to the belief that temporal logic is a useful tool for reasoning about such systems.[1–4] The use of temporal logic enables, in particular, analysis of both safety and liveness properties in a single uniform logical framework (see Ref. 5 for extensive examples). More recently, techniques have been developed for achieving compositional temporal proof systems.[6,7] Compositionality is an essential requirement for the hierarchical development of implementations from formal specifications. Without compositionality, the check on consistency of a development step would be delayed until all interactions between the developed components are known, essentially, at the implementation level; clearly, it could be rather costly if such a consistency check then showed that the system did not achieve the overall specification. In general, compositionality can be achieved by realising that a specification of any component must include assumptions about the behaviour of the environment in which the component will reside. In the temporal framework, this requires that one can distinguish actions made by a component from those made by its environment; in Refs 6 and 7 the coarse technique of labelling actions is used for just that purpose. Although it is sometimes possible

5 presents the multiple-lift system. Final comments are made in Section 6. For convenience, an appendix contains the semantics of the logic used in this article.

2. INFORMAL REQUIREMENTS

The following is a description of the lift-control system problem as set by Neil Davis.

A Lift-Control System

An *n*-lift system is to be installed in a building with *m* floors. The lifts and the control mechanism are supplied by a manufacturer. The internal mechanisms of these are assumed (given) in this problem.

Design the logic to move lifts between floors in the building according to the following rules.

(1) Each lift has a set of buttons, one button for each floor. These illuminate when pressed and cause the lift to visit the corresponding floor. The illumination is cancelled when the corresponding floor is visited (i.e. stopped at) by the lift.

(2) Each floor has two buttons (except ground and top), one to request an up-lift and one to request a down-lift. These buttons illuminate when pressed. The buttons are cancelled when a lift visits the floor and is either travelling in the desired direction, or visiting the floor with no requests outstanding.

In the latter case, if both floor-request buttons are illuminated, only one should be cancelled. The algorithm used to decide which to service should minimise the waiting time for both requests.