Une extension temporisée d'AltaRica

Application à la modélisation d'un système embarqué

Claire Pagetti

IRCCyN
1 rue de la Noë
BP 92101
44321 Nantes Cedex 3
France
email : Claire.Pagetti@irccyn.ec-nantes.fr

RÉSUMÉ. Timed AltaRica *est un langage hiérarchique de modélisation de haut niveau de systèmes réactifs temps réel. Ce langage est une extension temporelle du langage AltaRica dont les caractéristiques majeures sont : la hiérarchie, la synchronisation des événements, les priorités temporelles sur les événements, la manipulation de variables partagées ou non et les contraintes sur ces variables. Nous exposons formellement la syntaxe et la sémantique de Timed AltaRica et nous développons certaines fonctionnalités comme l'urgence. La présentation sera illustrée par un exemple d'avionique, lui même analysé à l'aide du prototype* Tarc.

ABSTRACT. *This paper summarises the recent works on the real time extension of the language AltaRica.* Timed AltaRica *is a high level hierarchical real-time modelling language. The fundamental features are: the hierarchical conception, the events synchronisation, the shared or hidden variables handling, the timed priorities among actions and the logical constraints on variables. We present in the following the syntax and the semantics of timed AltaRica while modelling an avionic example in the language. We also put some attention to a particular modelling method: urgency. The example analysis is managed by the prototype* Tarc.

MOTS-CLÉS : *Timed AltaRica, systèmes temporisés, urgence, vérification, temps réel*

KEYWORDS: *Timed AltaRica, timed systems, urgency, verification, real-time*

1. Introduction

Contexte et problématique. Les systèmes industriels actuels amènent à modéliser des structures complexes de part leur taille, leurs interactions avec l'environnement (systèmes réactifs), la répartition spatiale de leurs composantes (systèmes distribués) et les contraintes temporelles fortes qu'ils doivent respecter (systèmes temps réel). La modélisation peut devenir extrêmement ardue et nous proposons comme alternative un langage temps réel de modélisation de haut niveau *Timed AltaRica*. Ce langage assez généraliste a le pouvoir d'expression des automates temporisés et fournit de nombreuses fonctionnalités de spécification. Notre propos est de présenter ce langage et d'en montrer les atouts.

En tant qu'extension du langage de modélisation *AltaRica*, il repose sur le modèle des automates à contraintes ([POI 99]) temporisés. Un système est vu comme un ensemble de variables contraintes par des formules et une relation de transitions. Les avantages du langage AltaRica [POI 99, ARN 00, POI 00] sont multiples puisqu'il permet une conception hiérarchique des systèmes, des synchronisations événementiels, d'octroyer des priorités sur les événements. Son utilisation est de plus en plus répandue pour la modélisation de systèmes, notamment industriels (Dassault Aviation, Ixi, Total Fina Elf, ...). On peut trouver des études applicatives dans [RAU 02] et [SEG 01]. D'un point de vue logiciel[1], l'idée centrale est la possibilité d'extraire d'un programme AltaRica plusieurs formalismes pour des applications diverses et des logiciels différents. Ainsi, on peut aussi bien transformer un programme AltaRica en MEC [ARN 94] et appliquer une logique temporelle que le transformer en réseau de Petri stochastique ou en arbre de défaillance exploitable par Aralia [ARA 96].

Notre contribution. Face à l'importance accrue des systèmes temps réel, nous avons introduit le temps quantitatif dans AltaRica et étendu en conséquence la syntaxe et la sémantique [CAS 02]. Les opérations proposées sont le produit synchronisé, la possibilité de cacher, d'instancier des variables ou des sous composants, la synchronisation des flux, les priorités temporelles [BOR 00a].

Peu de langages temps réel de haut niveau existent actuellement. Leur utilisation devient nécessaire pour éviter des erreurs de modélisation et faciliter la conception. Ils ont été développés à peu près à la même période que Timed AltaRica : CHARON [ALU 00] est un langage modulaire de modélisation de systèmes hybrides, ARGOS [JOU 96] est étendu à une sous classe de systèmes temporisés. Timed AltaRica est un langage formellement défini dont l'utilisation devient efficace une fois sa syntaxe acquise. Ses caractéristiques autorisent une certaine concision : un composant n'est spécifié qu'une fois et peut être instancié à différents endroits, les vecteurs de synchronisations énumèrent les actions *devant absolument* ou *devant si elles le peuvent* s'exécuter, les variables partagées contraignent et réduisent les comportements, les invariants temporels servent notamment pour les délais, enfin les priorités

1. Les outils sont téléchargeables sur le web, URL : http ://altarica.labri.fr/

temporelles spécifient des comportements comme *cette action ne peut avoir lieu que si telle autre n'est pas exécutable dans un certain délai*.

Le langage est implémenté dans un prototype, Tarc [PAG 03], qui transforme un noeud hiérarchique en un automate temporisé. Celui-ci est ensuite traduit en un automate de UPPAAL [PET 00] ou HYTECH [HEN 97] et peut ainsi être analysé grâce à ces outils. La suite sera illustrée par la modélisation d'une commande de gouverne d'un avion, exemple extrait de [BON 01].

Plan

Dans une première partie, nous rappelons des notions générales qui nous seront utiles pour présenter formellement Timed AltaRica. Dans une seconde partie, nous développons la syntaxe et la sémantique du langage. Enfin, dans une troisième partie, nous nous concentrons sur l'étude de la gouverne d'avion : nous proposons une modélisation alternative à celles développées dans [BON 01].

2. Contexte général

Nous rappelons dans cette partie des notions et des définitions utiles à la construction de Timed AltaRica. En premier lieu, puisque Timed AltaRica manipule des ensembles de variables contraintes par des prédicats, nous rappelons quelques notions sur les formules d'un langage du premier ordre et sur les variables de type *horloge*. Ensuite, nous donnons la sémantique du langage donnée sous forme de *système de transitions temporisé interfacé*. Enfin nous présentons une synthèse des caractéristiques d'AltaRica.

2.1. *Préliminaires*

Variables discrètes

On considère un ensemble fini de variables discrètes Z, ainsi $z \in Z$ peut être un entier ou un booléen etc... Les *expressions* $\mathbb{E}(Z)$ sont construites à partir des variables de Z et des fonctions d'arité $n,\ \forall n \in \mathbb{N}$. On définit également les *formules* $\mathbb{F}(Z)$ sur $\mathbb{E}(Z)$: les formules atomiques sont construites à partir des expressions et des prédicats d'arité $n,\ \forall n \in \mathbb{N}$ (*tt* et *ff* sont des prédicats d'arité 0) ; les formules sont construites à partir des formules atomiques, des connecteurs logiques ($\wedge, \Rightarrow \cdots$), des parenthèses et des quantificateurs (existentiel et universel). Ainsi, $\forall z \in Z, z+3$ est une expression et $z \geq 3$ est une formule. Pour $f \in \mathbb{F}(Z)$ on note *free*(f) l'ensemble des variables libres de f. On suppose que chaque variable de Z prend ses valeurs dans un ensemble commun $\mathcal{D}$. Une *valuation* ν est une application $\nu : Z \rightarrow \mathcal{D}$ qui associe à chaque variable une valeur, on note $\mathcal{D}^Z$ l'ensemble des valuations.

Une *affectation* des variables de Z est une application $a : Z \rightarrow \mathbb{E}(Z)$ qui associe à chaque variable une expression. Ainsi, une affectation de $z \in Z$ de la forme $a(z) = y + 2$ correspond à $z := y + 2$.

Horloges

On considère un ensemble fini X de variables à valeurs réelles positives appelées horloges. Une *valuation d'horloge* sur X est une application $v : X \to \mathbb{R}$ qui assigne une valeur réelle positive à chaque horloge de X.

L'ensemble des *contraintes d'horloge* $\mathcal{B}(X)$ sur l'ensemble X est défini inductivement par :

$$g := x \backsim r | \ x - y \backsim r \ | g \wedge g \ | g \vee g$$

avec $x, y \in X, \backsim \in \{<, \leq, >, \geq, =\}$, $r \in \mathbb{Q}$. On considère également $\mathcal{B}_C(X)$ le sous ensemble de $\mathcal{B}(X)$ des contraintes convexes d'horloges.

Interprétation

Pour exprimer les sémantiques des langages, il est nécessaire d'interpréter les structures précédentes. Dans le cas discret, soit $f \in \mathbb{F}(Z)$, l'interprétation de $[\![f]\!]$ est un sous ensemble de $\mathcal{D}^Z$ et plus précisément $[\![f]\!] \subseteq \mathcal{D}^{free(f)}$. On a de plus $[\![tt]\!] = \mathcal{D}^Z$ et $[\![ff]\!] = \emptyset$.

Dans le cas continu, une contrainte d'horloge g est une formule particulière qui s'évalue à vrai ou faux : $[\![g]\!] \subseteq \mathbb{R}^X_{\geq 0}$ et $g(\nu) = tt \iff \nu \in [\![g]\!]$.

2.2. *Systèmes temporisés*

Automates temporisés

Les automates temporisés ont été introduits par Alur et Dill [ALU 90, ALU 94] et sont rapidement devenus un modèle de référence pour les systèmes temporisés. De nombreux articles ont été écrits afin d'en étudier les propriétés et de nombreux logiciels de vérification, comme UPPAAL [PET 00], KRONOS [DAW 95] ou CMC [LAR 98], ont été conçus à partir des résultats obtenus. On considère un domaine de temps dense $\mathbb{T}$, on suppose que $0 \in \mathbb{T}$ et $\mathbb{T} \subseteq \mathbb{R}_{\geq 0}$, par exemple $\mathbb{T}$ peut être $\mathbb{Q}_{\geq 0}$ ou $\mathbb{R}_{\geq 0}$. Un automate temporisé [ALU 94] sur $\mathbb{T}$ est un sextuplet $\mathcal{A} = (Q, E, X, q_0, I, \rightarrow)$ tel que : Q est un ensemble fini de localités, E est un ensemble fini d'actions, X est un ensemble fini d'horloges, $q_o \in Q$ est l'état initial, $I : Q \to \mathbb{B}_C(X)$ est une application qui associe à chaque localité son invariant temporel et $\rightarrow \subseteq Q \times (\mathbb{B}(X) \times E \times 2^X) \times Q$ est la relation de transition.

La sémantique d'un automate temporisé est donnée par le système de transitions $(S, E \cup \mathbb{R}_{\geq 0}, s_0, \rightarrow_S)$ où $S = Q \times \mathbb{R}^X$, $s_0 = (q_0, \nu_0)$ avec $\forall x \in X, \nu_0(x) = 0$ et $\rightarrow_S \subseteq S \times E \cup \mathbb{T} \times S$ avec $((q, \nu), e, (q', \nu')) \in \rightarrow_S \Leftrightarrow (e \in E \wedge \exists (q, g, e, a, q') \in \rightarrow$, *telle que* $g(\nu) = tt, \nu' = a(\nu), \nu' \in [\![I(q')]\!]$) *ou* $(e = \delta \in \mathbb{T} \wedge q' = q \wedge \nu' = \nu + \delta \wedge \forall \delta' \leq \delta, v + \delta' \in [\![I(q)]\!])$

Systèmes de transitions temporisés interfacés

La sémantique d'AltaRica [POI 99] est basée sur la notion d'automates à contraintes : un système est donné par l'ensemble de ses variables et la relation de transitions

gardée qui le définit. Ainsi, le modèle n'est pas vu comme un ensemble d'états de contrôle mais comme un ensemble de valuations de variables satisfaisant certaines formules. Nous en donnons ici une extension temporisée qui nous permettra d'exprimer la sémantique de Timed AltaRica.

Définition 1 *Un* système de transitions temporisé interfacé *[CAS 02] (STTI) de dimension* (n, m) *sur le domaine temporel* $\mathbb{B}$ *est un quintuplet* $\mathcal{A} = \langle E_t, F_t, S_t, \pi, T \rangle$ *avec :*

1) $E_t = E_+ \cup \{\epsilon\} \cup \mathbb{R}$*, où* E_+ *est un ensemble fini d'événements tel que* $\epsilon \notin E_+$ *;*

2) $F_t = F \times \mathbb{R}^m$ *est un ensemble de valeurs de flux, avec* F *ensemble de valeurs discrètes de flux et* $\mathbb{R}^m$ *ensemble de valeurs continues ;* $S_t = S \times \mathbb{R}^n$ *ensemble des valeurs d'états, avec* S *ensemble des valeurs discrètes et* $\mathbb{R}^n$ *ensemble des valeurs continues ;*

3) $\pi : S_t \to 2^{F_t}$ *est une application qui associe à chaque variable d'état* $q \in S_t$ *toutes les valeurs de flux admissibles en* q*. On suppose que* $\forall q \in S_t,\ \pi(q) \neq \emptyset$*.*

4) $T \subseteq S_t \times F_t \times E_t \times S_t$ *est la relation de transition, elle vérifie :*

a) $(q, g, e, q') \in T \Rightarrow g \in \pi(q)$

b) $\forall q \in S_t, \forall g \in \pi(q),\ (q, g, \epsilon, q) \in T$

c) $\forall q \in S_t, \forall g \in \pi(q),\ (q, g, 0, q) \in T$

Une configuration d'un STTI est un couple $(q, g) \in S_t \times F_t$ *avec* $g \in \pi(q)$*.*

Exemple 1 *On suppose que le domaine des valeurs des variables est* $\mathcal{D} = \{0, 1\}$*, alors toutes les variables discrètes auront au plus deux valeurs. On va modéliser un cas très simple où le système est initialement en état de marche. Si un événement* $failure$ *survient, le système est en panne et peut alors être réparé par un événement* $repair$*. On considère le STTI* $\mathcal{A} = \langle E_t, F_t, S_t, \pi, T \rangle$ *avec* $E_+ = \{failure, repair\}$*. Une valeur visible exprime l'état du système :* 1 *s'il marche,* 0 *sinon, donc* $F_t = F = \{f | f \in \{0, 1\}\}$*. Cette valeur ne peut être modifiée par la relation de transitions (cf 4 dans la Définition 1), on introduit alors une valeur d'état qui coïncide avec la valeur de flux, donc* $S_t = S = \{s | s \in \{0, 1\} \wedge \pi(s) = \{f | f = s\}\}$*. Enfin, il y aura deux transitions* $(0, 0, failure, s := 1)$ *et* $(1, 1, repair, s := 0)$*. Dans ce cas fini, on peut représenter* $\mathcal{A}$ *par un automate équivalent, il est donné dans la Figure 1.*

2.3. *Urgence*

Dans les systèmes temporisés, il est possible d'exprimer une notion d'urgence pour une action. Si un événement e est urgent alors toute transition étiquetée par e doit être tirée dès qu'elle est permise. Le temps ne peut alors plus s'écouler dans une configuration permettant d'exécuter une action e. Ce type de priorité a été introduit et

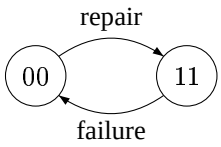

Figure 1. *Représentation sous forme d'automate de $\mathcal{A}$*

étudié pour les systèmes temporisés dans [SIF 96, BOR 98, BOR 00b, BOR 00a] afin de modéliser des réactions critiques.

Définition 2 (Relation d'urgence) *Une relation d'urgence sur un ensemble d'événements E est le sous-ensemble d'événements urgents de $E \setminus \{\epsilon\}$. On note $e > time$ un événement urgent.*

Cette relation est indéterministe en ce sens que dans une configuration où une action urgente est possible, le système doit quitter d'une manière ou d'une autre la configuration, que ce soit en exécutant la transition urgente ou une autre tirable.

On exprime l'effet d'une relation d'urgence sur un STTI :

Définition 3 (Opérateur de restriction d'urgence) *Soient $\mathcal{A} = \langle E_t, F_t, S_t, \pi, T\rangle$ un STTI de dimension (m, n) et $<$ une relation sur E_t. On définit l'opérateur de restriction d'urgence $\restriction$ pour la relation de transition $T \subseteq S_t \times F_t \times E_t \times S_t$ et la relation d'urgence $<$ par : $(q, g, e, q') \in T \restriction < \Longleftrightarrow \big[(q, g, e, q') \in T \wedge \big(e = t \in \mathbb{T}, \forall t' \in \mathbb{T}, t' \leq t, (q, g, t', q') \in T \Rightarrow \forall e' \in E_+, (q', g', e', q'') \in T \Longrightarrow e' \not> time\big)\big]$.*

Cette définition traduit bien l'obligation de quitter une configuration où une action urgente est possible. L'opérateur réduit les transitions temporelles en interdisant celles qui sont inadmissibles. Ainsi, dans l'exemple 1, si on rajoute l'hypothèse $failure > time$, alors l'état de contrôle 00 est toujours traversé et il est impossible d'y rester.

Dans ce papier, nous introduisons uniquement la notion d'urgence, il s'agit d'un cas particulier des priorités temporelles [SIF 96, BOR 98, BOR 00a] ajoutées dans Timed AltaRica (cf [CAS 02]).

2.4. *Un aperçu d'AltaRica*

Le langage AltaRica ([POI 99, ARN 00]) permet de décrire le comportement d'un système réactif de manière modulaire. Ainsi le système global peut être décomposé en plusieurs modules (appelés *nœuds*), eux mêmes éventuellement hiérarchiques. On développe la syntaxe en se référant à la Figure 2. La ligne 1 donne le nom du composant F_R. Un nœud AltaRica est constitué de :

variables : les variables de *flux* ont un rôle de témoin sur l'état partiel de l'environnement, et sont dans ce cas non modifiables par le nœud, ou à l'inverse informent l'extérieur sur l'état interne du nœud. Dans la Figure 2, ligne 2, une seule variable de flux $Panne_R$ est spécifiée. Les variables d'*états* sont modifiables par les transitions du nœud et ne sont pas visibles depuis l'extérieur. Il y a deux variables d'état, ligne 3, $etat$ et x. Nous verrons ultérieurement le type $clock$.

événements : ils étiquettent les transitions, elles-mêmes dépeintes à l'aide de leur garde, leur événement et leur action sur les variables d'états. Les deux événements, $breakdown_R$ et Cmd, sont donnés ligne 4 de l'exemple. AltaRica permet en outre d'ordonner partiellement les événements à l'aide de priorités statiques.

sous composants : ce sont des nœuds AltaRica dont les interactions sont de deux types. D'une part on leur impose un synchronisme sur les événements, spécifié par des vecteurs de synchronisation et d'autre part ils respectent des coordinations de flux explicitées par des contraintes entre variables.

contraintes : il existe une relation de dépendance entre les variables du nœud et les variables visibles des sous composants, exprimée sous forme d'une assertion, qui doit toujours être vérifiée. Elle peut ainsi obliger certaines variables de flux à modifier leur valeur. Fig 2, aux lignes 8 et 9, deux conditions contraignent le composant, la première associe la variable de flux et la variable d'état $etat$ ce qui signifie qu'elles sont toujours égales et que la variable de flux fait partie de la catégorie des flux informant l'extérieur de nœud. La deuxième contrainte correspond à un invariant temporel et sera ultérieurement développée.

L'accent est porté dans ce papier sur la partie applicative puisque un article entier est dévolu à toute l'étude théorique. Par la suite, certaines notions ne seront pas développées mais l'ensemble des définitions, théorèmes et preuves sont accessibles sur les pages d'AltaRica [2]. Nous allons dans la prochaine partie donner l'énoncé de l'exemple traité dans ce papier.

3. Architecture du système de commande de gouverne [BON 01]

Cet exemple est tiré de [BON 01] et représente un fragment du système de pilotage d'un avion. Les processus de navigation, d'un Airbus A340 par exemple, sont implémentés sur plus de cinquante calculateurs répartis et connectés par des bus. On envoie une commande, sous forme d'un événement Cmd, toutes les 20 ms pour asservir une gouverne. Trois fonctions, F_R, F_L et F_B, se relaient pour jouer ce rôle de manière disjointe. Une fonction est dite en mode commande si elle envoie la commande périodique. Initialement, la fonction F_R est en mode commande. Elle peut à tout instant être défaillante, elle est alors remplacée par la première fonction de secours, F_L qui entre en mode commande. Une exception existe, il se peut que F_L soit déjà en panne auquel cas la deuxième fonction de secours F_B intervient. Si F_L remplace F_R et tombe a

2. http ://altarica.labri.fr/

posteriori en panne, F_B passe en mode commande. Les trois fonctions F_R, F_L et F_B sont distribuées sur trois calculateurs positionnés sur l'aéronef et reliés par deux bus. Le rôle des bus est d'envoyer les informations sur l'état des fonctions. Ainsi, le bus C_{LR} transmet l'information de panne de F_R à F_L et le bus C_{LRB} envoie les messages de panne de F_R et F_L à la fonction F_B. Dans [BON 01], le système vérifie certaines hypothèses :

– le délai de communication du bus C_{LR} est dans l'intervalle $[0, 20]$,

– le délai de communication du bus C_{LRB}, est dans l'intervalle $[0, 40]$ (à cause de la passerelle reliant les bus),

– la communication est instantanée entre les calculateurs et la gouverne,

– en cas de défaillance, les calculateurs n'émettent plus de Cmd et ne sont pas réparables,

– il y a au plus deux calculateurs en panne.

4. AltaRica temporisé

Un système est représenté dans le formalisme AltaRica par un ensemble de variables, le temps est naturellement introduit par des variables de type *horloge*. Nous présentons le langage au travers de la modélisation détaillée de la fonction F_R.

4.1. *Syntaxe de Timed AltaRica*

```
1.node FR

2. flow  Panne_R : [0,1]
3. state  etat : [0,1] ; x : clock
4. event  breakdown_R, Cmd
5. trans  x=20 & etat=0 |- Cmd ->
              x:=0;
6.        x<=20 & etat=0 |-
              breakdown_R -> etat:=1
7. extern initial_state =   etat=0, x=0
8. assert etat=Panne_R;
9. tinvariant etat=0 => x <=20

10.edon
```

Figure 2. *Modélisation de la fonction F_R*

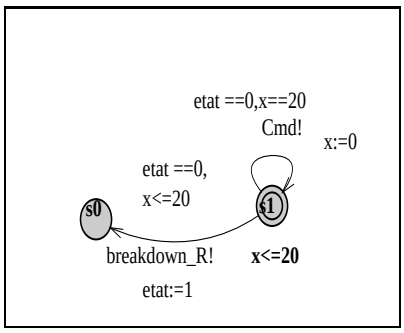

Figure 3. *F_R en UPPAAL*

Nous donnons les versions à la fois Timed AltaRica, Figure 2, et UPPAAL, Figure 3, dans un souci de clarté. L'automate UPPAAL est généré par le prototype Tarc[3]. La fonction F_R, donnée à la Figure 2, génère l'envoi de Cmd toutes les 20 unités de temps. En effet, l'invariant temporel $etat = 0 \implies x \leq 20$ spécifié ligne 9 contraint

3. http ://altarica.labri.fr/CVSweb/cvsweb .cgi/mec/

le système à changer de configuration avant 20 unités de temps (u.t.) et la seule solution consiste à tirer soit la transition $(x = 20 \wedge etat = 0, Cmd, x := 0)$ ligne 5 soit la deuxième transition $(x \leq 20 \wedge etat = 0, breakdown_R, etat := 1)$ ligne 6. La première transition représente l'envoi périodique, de période 20 u.t., de l'événement Cmd, *i.e.* F_R est en mode commande, et la deuxième transition correspond à une panne de la fonction. Dans ce cas, la variable booléenne $etat$ est modifiée, ce qui change également la valeur de la variable de flux $Panne_R$ par respect pour l'assertion $etat = Panne_R$ ligne 8. Ainsi, l'extérieur par le biais de la variable de flux $Panne_R$ connaît l'état de la fonction F_R, à savoir si elle est ou non en dysfonctionnement.

Au travers de l'exemple nous avons esquissé la syntaxe du langage. Nous allons à présent la définir formellement. Nous proposons une extension des composants AltaRica définis dans [POI 00].

Définition 4 (Composant temporisé : syntaxe abstraite [CAS 02]) *Un composant est un sextuplet* $\mathcal{T} = \langle V_S \cup C_S, V_F \cup C_F, E, A, M, < \rangle$ *où :*

1) V_S, V_F *sont des ensembles finis disjoints appelés respectivement ensembles des variables d'*états *et de* flux. C_S, C_F *sont des ensembles finis disjoints appelés respectivement ensembles des variables d'*horloges d'état *et d'*horloges de flux. *On note* $V_T = V_S \cup V_F$ *et* $C_T = C_S \cup C_F$. *On suppose que* $V_T \cap C_T = \emptyset$ *;*

2) $E = E_+ \cup \{\epsilon\}$ *est un ensemble fini d'événements et* $\epsilon \notin E_+$ *est l'action vide ;*

3) $A = A_{V_T} \cap A_{C_T} \in \mathbb{F}$[4] *est l'*assertion *du composant. Elle vérifie free*$(A) \subseteq V_T \cup C_T$ *;* $A_{V_T} \in \mathbb{F}$, *free*$(A_{V_T}) \subseteq V_T$. A_{C_T} *est un ensemble de contraintes de la forme* $P \Longrightarrow I$ *avec* $P \in \mathbb{F}$, *free*$(P) \subseteq V_T$ *et* $I \in \mathbb{F}$, *free*$(I) \subseteq C_T$ *tel que* I *définit une région convexe de* $\mathbb{R}^n$ *si* $|C_S| = n$ *;*

4) $M \subseteq (\mathbb{F} \times \mathcal{B}(C_T)) \times E \times (\mathbb{E}(V_T)^{V_T} \times 2^{C_T})$ *est un ensemble de* macro-transitions $((g, \gamma), e, (a, R))$ *telles que :*

a) (g, γ) *est une* garde *avec* $g \in \mathbb{F}$ *et free*$(g) \subseteq V_C$ *;* $\gamma \in \mathcal{B}(C_T)$ *;*

b) $e \in E$ *est l'événement déclenchant la transition ;*

c) $a : V_S \rightarrow \mathbb{E}(V_C)$[5] *est une affectation pour les variables de* V_S. $R \in 2^{C_T}$ *est l'ensemble des horloges remises à zéro par la transition ;*

5) $<$ *est une relation d'urgence sur les événements.*

Une fois la syntaxe étendue, la sémantique de Timed AltaRica est exprimée sous forme de systèmes de transitions temporisés interfacés.

4.2. *Sémantique d'AltaRica temporisé*

Nous étendons à nouveau les définitions sémantiques d'AltaRica [POI 00]. A partir de la définition de la syntaxe abstraite 4, on voit se dessiner l'intérêt des automates à

4. L'assertion est une formule, cf Partie 2.1
5. Une variable d'état peut prendre une nouvelle valeur donnée par une expression, cf Partie 2.1

contraintes puisqu'un composant est muni de variables et de formules contraignant le système. La sémantique va associer des valeurs aux variables, dans $\mathcal{D}$ pour les discrètes et $\mathbb{R}$ pour les horloges, elle va également transformer l'assertion A en la fonction π des valeurs admissibles. Enfin la relation de transitions est gardée ce qui entraîne que les valeurs des variables permettant une transition sont celles vérifiant la formule $g \wedge \gamma$ et dont l'affectation est admissible.

Définition 5 (Composant temporisé : sémantique [CAS 02]) *On considère $\mathcal{T} = \langle V_S \cup C_S, V_F \cup C_F, E, A, M, < \rangle$ un composant Timed AltaRica avec $|C_S| = n$ et $|C_F| = m$. La sémantique de $\mathcal{T}$ sur le domaine temporel $\mathbb{T}$ est le système de transitions temporisé interfacé $[\![\mathcal{T}]\!] = \langle E_t, F_t, S_t, \pi, T \rangle$ de dimension (n, m) construit de la manière suivante :*

1) $E_t = E \cup \mathbb{T}$

2) $F_t = \mathcal{D}^{V_F} \times \mathbb{R}^m$ *;*

3) $S_t = \{(s, \nu) \in \mathcal{D}^{V_S} \times \mathbb{R}^n | \exists (f, \mu) \in \mathcal{D}^{V_F} \times \mathbb{R}^m, ((s, \nu), (f, \mu)) \in [\![A]\!]^6\}$ *;*

4) $\pi : S_t \to 2^{F_t}$ *tel que* $\pi(q) = \{(f, \mu) | \ (q, (f, \mu)) \in [\![A]\!]\}$ *;*

5) $T \subseteq S_t \times F_t \times E_t \times S_t$ *tel que* $T = [\![M]\!] \restriction <$ *avec :*

a) $[\![M]\!] = \cup_{t \in M} [\![t]\!] \bigcup \cup_{\delta \in \mathbb{T}} [\![\delta]\!]$ *;*

b) $[\![((g, \gamma), e, (a, R))]\!] = \{((s, \nu), (f, \mu), e, (s', \nu')) | ((s, \nu), (f, \mu)) \in [\![A \wedge g \wedge \gamma]\!], s' = a(s, f) \wedge \nu' = R(\nu, \mu)\}$,

c) $\delta \in \mathbb{T}$, $[\![\delta]\!] = \left\{ \left((s, \nu), (f, \mu), \delta, (s', \nu') \right) | ((s, \nu), (f, \mu)) \in [\![A]\!], s' = s \wedge \nu' = \nu + \delta \wedge \forall \delta' \leq \delta, \pi(s, \nu + \delta') \neq \emptyset \right\}$.

Lors d'une transition temporelle, le système ne peut rester dans une configuration que tant que l'assertion est vérifiée. Lors d'une transition discrète, le système doit vérifier l'assertion pour ses valeurs d'arrivée.

Exemple 2 *Reprenons la fonction FR donnée Figure 2 et calculons sa sémantique. Le domaine $\mathcal{D} = \{0, 1\}$ convient et l'ensemble des événements $E_+ = \{breakdown_R, Cmd\}$ reste inchangé. Les valeurs de flux $F_t = \{f | f \in \{0, 1\}\}$ correspondent aux valeurs possibles de $Panne_R$ et les valeurs des variables d'états sont celles respectant l'assertion $S_t = \{(s, t) | s \in \{0, 1\}, t \in \mathbb{R}, (s, x) \in \{0 \times [0, 20], 1 \times \mathbb{R}\}\} = \{0 \times [0, 20], 1 \times \mathbb{R}\}$. La fonction π est donc donnée par l'assertion $\pi(0 \times [0, 20]) = 0$ et $\pi(1 \times \mathbb{R}) = 1$. Enfin les transitions continues sont données par $\forall \delta \in \mathbb{R}$, $(0, t, \delta, 0, t + \delta) \in T \Leftrightarrow t + \delta \leq 20$ et $(1, t, \delta, 1, t + \delta) \in T$; les discrètes sont $(0, t, Cmd, 0, 0) \in T \Leftrightarrow t = 20$ et $(0, t, breakdown_R, 1, t) \in T \Leftrightarrow t <= 20$.*

6. L'interprétation d'une formule est rappelée dans la section 2.1

4.3. *Hiérarchie*

Timed AltaRica est un langage hiérarchique : un nœud de profondeur hiérarchique non nulle (i.e. ayant des sous composants) peut se réécrire en un nœud de profondeur nulle. Ainsi, le produit synchronisé de n nœuds est un nœud.

Correspondance avec les systèmes temporisés existants

On généralise la notion de bisimulation aux STTI (cf [CAS 02]). Un automate temporisé peut être décrit par un composant Timed AltaRica : celui-ci n'aura pas de variable de flux et une variable d'état entière suffit pour énumérer les états de contrôle. Réciproquement, un algorithme de traduction d'un composant Timed AltaRica en un automate temporisé *bisimilaire temporisé interfacé* est développé dans [CAS 02]. L'idée consiste a regroupé les valeurs des variables discrètes vérifiant le même invariant temporel en un état de contrôle puis à énumérer toutes les transitions sans les interpréter entre ces états de contrôle. Dans le cas général, c'est à dire un nœud temporisé, l'algorithme consiste à mettre à plat le nœud sous forme d'un composant temporisé en utilisant la réécriture puis à traduire le résultat sous forme d'un automate temporisé.

5. Etude de cas

Dans [BON 01], l'exemple de la gouverne est modélisé de trois manières différentes : sous forme synchrone, par un problème de satisfaction de contraintes et enfin à l'aide d'automates temporisés. Nous proposons ici une quatrième version du système de gouverne avec Timed AltaRica. La fonction F_R est déjà donnée section 2.

5.1. *Modélisation, une solution avec AltaRica temporisé*

Plutôt que de simplement calquer la représentation du système réalisée sous forme d'automates temporisés, nous utilisons ici les caractéristiques de Timed AltaRica, notamment les coordinations de flux et la notion d'urgence.

Dans la Figure 4, le bus de communication C_{RL} entre la fonction F_R et la première fonction de secours F_R connaît la valeur de la variable de flux $Panne_R$ et agit en urgence dès qu'elle vaut 1. En effet, une priorité est donnée sur l'événement $reception$ qui le rend urgent. Une transition étiquetée par $reception$ prévaut sur tout, même l'écoulement du temps, et doit être exécutée dès que la garde devient vraie. Ainsi, si F_R tombe en panne, instantanément sa variable de flux $Panne_R$ prend la valeur 1 et le nœud C_{LR} doit immédiatement exécuter la transition $t_1 = (Panne_R = 1 \wedge etat = 0, reception, etat := 1, c := 0)$. Ensuite, le nœud respectant son assertion $etat = 1 \wedge panne = 0 \implies c < 20$ a moins de 20 u.t. pour réagir et envoyer l'information de panne, ce qui se traduit ici par : effectuer la transition $t_2 = (c < 20 \wedge etat = 1, sending, panne := 1)$ entraînant instantanément la modification de la valeur de flux $Panne_{RL}$, par respect pour l'assertion.

```
node CLR
  flow Panne_R, Panne_RL : [0,1]
  event sending, reception
  priority reception > time
  state etat, panne : [0,1];c : clock
  trans panne=0 & Panne_R=1 & etat=0
        |- reception -> etat:=1, c:=0;
     c<20 & etat=1 |- sending ->
        panne:=1,etat:=0
  assert Panne_RL=panne
  tinvariant (etat=1) => (c<20)
  extern initial_state = etat=0,
        panne=0, c=0
edon
```

Figure 4. *Modélisation du canal de communication entre F_L et F_R*

```
node FL
  flow Panne_RL, Panne_L : [0,1]
  event breakdown_L, action_L, Cmd
  priority action_L >time
  state etat,panne : [0,1]; y:clock
  trans Panne_RL=1 & panne=0 & etat=0
        |- action_L -> etat:=1, y:=20;
    etat=0 |- breakdown_L -> panne:=1;
    etat=1 |- breakdown_L -> panne:=1,
       etat:=0;
    etat=1 & y=20 |- Cmd ->y:=0
  extern initial_state =
          etat=0, panne=0
  assert Panne_L=panne;
  tinvariant etat=1 ∧ panne=0 => y<= 20
 edon
```

Figure 5. *Modélisation de la fonction F_L*

Dans la Figure 5, on modélise la première fonction de secours qui passe en mode commande si d'une part elle n'est pas déjà en panne et si d'autre part, la fonction F_R est altérée. Sa mise en route est alors urgente et donc une relation d'urgence $action_L > time$ est alors spécifiée.

Les canaux de communication reliant les fonctions F_R et F_B à la dernière fonction de secours F_B sont représentés dans la Figure 6. Ce composant se comporte de manière similaire au canal C_{LR} (Figure 4) : les réceptions sont urgentes et forcent le composant à changer d'état. Ensuite, le flux $Panne_{RLB}$ vaut 2 si les deux fonctions F_R et F_L sont en dysfonctionnement. Le composant doit envoyer les informations avant une durée de 40 u.t. Le système global est décrit dans la Figure 7. Les variables de flux, comme par exemple $Panne_{RLB}$ sont visibles de l'extérieur des sous composants et peuvent à la fois être testées et contraintes. Les valeurs des flux de panne sont transmises par les canaux. Lorsque $Panne_{RLB} = 2$ cela entraîne l'activation de F_B modélisée dans le nœud $main$. Le respect des assertions contraint le système et assure les propriétés requises du système comme nous le verrons dans la prochaine partie.

5.2. *Vérification*

Dans l'article [BON 01], l'objectif était de modéliser des systèmes intégrés asynchrones. Pour des raisons de sécurité évidentes, ces modèles devaient assurer des comportements admissibles comme de la vivacité, par exemple, mais également deux exigences particulières :

P1 : à tout instant, il y a au plus une fonction en mode commande ;

P2 : la gouverne ne doit pas rester non commandée plus de 160 ms.

Nous voulons également garantir ces trois propriétés pour notre modélisation. La méthode consiste ici à traduire chaque composant Timed AltaRica en UPPAAL avec Tarc, puis à utiliser le model checker afin de vérifier les propriétés. Nous traduisons chaque composant en automate temporisé puisque la modélisation est de hiérarchie 2

```
node CLRB
  flow Panne_L , Panne_R : [0,1];
     Panne_RLB : [0,2]
  state etat, panne : [0,2];
     locR, locL : [0,1];
     h1, h2 : clock
  event
       rec_L, rec_R, sending
  priority
       rec_R > rec_L > time
  trans
     Panne_L=1 & etat=0 & locL=0|- rec_L ->
       etat:=1, locL:=1, h1:=0;
     Panne_R=1 & etat=0 & locR=0|- rec_R ->
       etat:=2, locR:=1; h2:=0;
     Panne_L=1 & etat=2 & panne_RLB=0 |-
          rec_L -> etat:=3, locL:=1, h1:=0;
     Panne_R=1 & etat=1 & panne_RLB=0 |-
          rec_R -> etat:=3, locR:=1, h2:=0;
     h1<40 & etat=1 |- sending ->
          panne:=panne+1, etat:=0;
     h2<40 & etat=2 |- sending ->
          panne:=panne+1, etat:=0;
     h1<40 & etat=3 |- sending ->
         panne:=panne+1, etat:=2;
     h2<40 & etat=3 |- sending ->
          panne:=panne+1, etat:=1
  extern initial_state =
          etat=0, panne=0
  assert Panne_RLB=panne;
        etat=1 => h1<=40;
        etat=2 => h2<=40;
        etat=3 => h1<=40 & h2<= 40
 edon
```

Figure 6. *Canaux de communication entre F_R, F_L et F_B*

```
node MAIN
 flow Panne_RLB : [0,2]
 state  etat:[0,2], z:clock
 event  action_B, Cmd
 priority action_B> time
 trans  Panne_RLB=2 & etat=0
  |- action_B -> etat:=1, z:=0;
  z=20 & etat=1 |- Cmd -> z:=0;
 init   etat=0, z=0;
 assert etat=1 => z<=20;
 sub fr : FR, fl : FL,
   clr : CLR, clrb : CLRB
edon
```

Figure 7. *Spécification du système global et de la dernière fonction de secours F_B*

et aucune synchronisation n'est réalisée. Le système global est donc le produit libre des automates -l'automate temporisé associé à $main$ correspond à F_B- contraint par des relations entre variables globales. Cela est équivalent à la mise à plat du système suivi de la traduction. Les automates UPPAAL résultant de la traduction des composants temporisés sont représentés.

La Figure 8 correspond au composant de la Figure 5 d'une part et également à la modélisation de la gouverne. L'introduction de ce nouvel automate temporisé permet d'utiliser correctement UPPAAL en autorisant toutes les transitions, de plus l'action Cmd remet à 0 une horloge locale r. Ainsi, la propriété P2 revient à vérifier que r est toujours inférieur à 160, ce qui s'exprime en TCTL par $A[](Gouverne.r \leq 160)$.Toujours globalement l'horloge de la gouverne est inférieure à 160, ce qui est bien équivalent à la transition Cmd a lieu au plus toutes les 160 u.t. Le canal de communication C_{LRB}, composant 6, reliant F_B aux autres fonctions est donné dans la Figure 9 et les autres automates temporisés du système sont donnés Figure **??**.

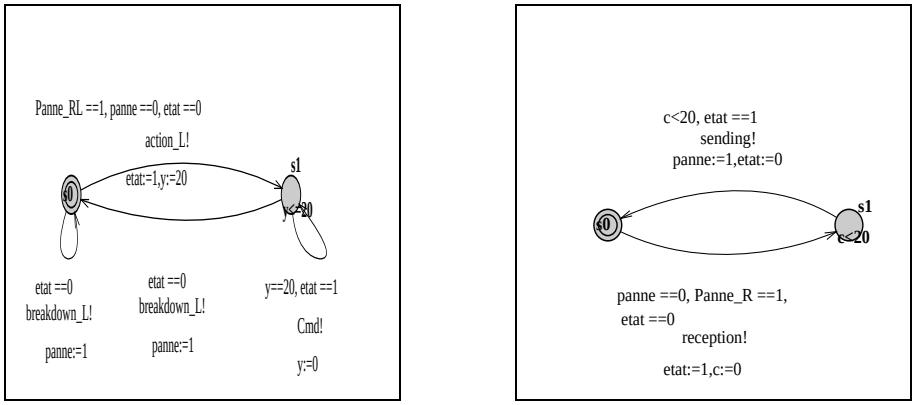

Figure 8. *Les automates UPPAAL de la fonction F_L et de la fonction C_{LR}*

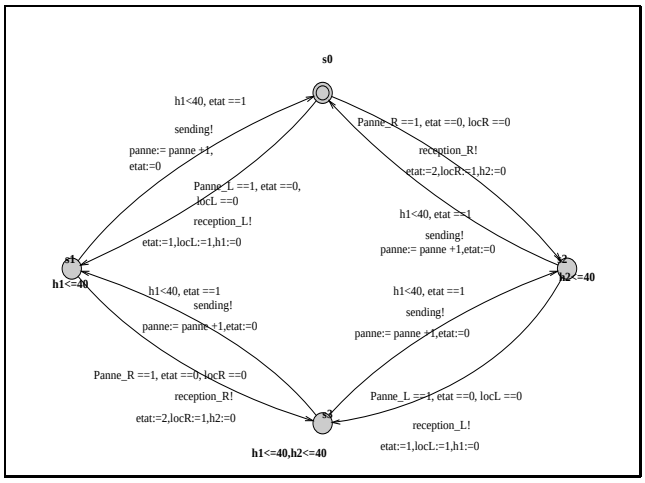

Figure 9. *Automate temporisé UPPAAL du canal de communication C_{LRB} reliant F_B à F_R et F_L*

Dans le système global, la fonction initiale en mode commande est bien F_R et toutes les variables sont initialisées à zéro. Le système est vivant : on a vérifié la propriété $A[](not\ deadlock)$. Enfin, il nous faut exprimer la propriété P1.

$A[](FR.s1\ imply\ (not\ FL.s1\ and\ not\ FB.s1)\ and\ (FL.s1\ imply\ not\ FB.s1))$. Notre modèle vérifie les trois propriétés.

6. Conclusion et perspectives

Timed AltaRica est un langage de haut niveau de modélisation de systèmes temps réel reposant sur des spécifications par contraintes. Ce langage fournit des méthodes efficaces de modélisation comme le produit synchronisé, qui est un élément incontournable, la hiérarchie ou les priorités temporelles. Au cours de ce travail, nous nous sommes concentrés sur la présentation du langage, à la fois syntaxique et sémantique, et son utilisation, à travers l'étude complète d'un exemple d'avionique. Cet exemple a été traité à l'aide du prototype Tarc qui a permis grâce aux liens proposés vers des outils de vérification d'analyser le système.

L'implémentation de la partie théorique est encore en cours puisqu'actuellement seules la translation AltaRica temporisé vers automate temporisé et la mise à plat d'un système sont réalisées. Des passerelles sont proposées vers UPPAAL comme on l'a vu mais aussi vers l'outil HYTECH [HEN 97]. Ainsi une extension du langage vers des systèmes hybrides est envisageable. Un prochaine étape sera d'incorporer les priorités temporelles en utilisant la librairie de polyèdres POLKA [JEA 02] afin de réduire les contraintes. Une fois l'outil achevé, une étude portera sur la sûreté temporisée puisque AltaRica se veut un atelier de sûreté de fonctionnement et de méthodes formelles.

7. Bibliographie

[ALU 90] ALUR R., DILL D., « Automata for modelling real-time systems », *17 International Conference on Automata, Languages, and Programming*, vol. 443, Lecture Notes in Computer Science, SpringerVerlag, 1990, p. 322–335.

[ALU 94] ALUR R., DILL D., « A theory of timed automata », *Theoretical Computer Science B*, vol. 126, 1994, p. 183–235.

[ALU 00] ALUR R., GROSU R., HUR Y., KUMAR V., LEE I., « Modular Specification of Hybrid Systems in CHARON », *HSCC*, 2000, p. 6-19.

[ARA 96] ARALIA G., « Computation of Prime Implicants of a Fault Tree within Aralia », *Reliability Engineering and System Safety*, , 1996, Special issue on selected papers from ESREL'95.

[ARN 94] ARNOLD A., BEGAY D., CRUBILLE P., *Construction and analysis of transition systems with MEC*, 1994.

[ARN 00] ARNOLD A., GRIFFAULT A., POINT G., RAUZY A., « The AltaRica formalism for describing concurrent systems », *Fundamenta Informaticae*, vol. 40, 2000, p. 109–124.

[BON 01] BONIOL F., BEL G., ERMONT J., « Modélisation et vérification de systèmes intégrés asynchrones : étude de cas et approche comparative », *9ème Conférence Internationale sur les Systèmes Temps Réel, RTS'2001*, , 2001.

[BOR 98] BORNOT S., SIFAKIS J., « On the Composition of Hybrid Systems », *HSCC*, 1998, p. 49-63.

[BOR 00a] BORNOT S., GLER G., SIFAKIS J., « On the Construction of Live Timed Systems », *Tools and Algorithms for Construction and Analysis of Systems*, 2000, p. 109-126.

[BOR 00b] BORNOT S., SIFAKIS J., « An Algebraic Framework for Urgency », *Information and Computation*, vol. 163, n° 1, 2000, p. 172-202.

[CAS 02] CASSEZ F., PAGETTI C., ROUX O., « A Timed Extension for AltaRica », rapport n° R 12002-13, 2002, IRCCyN/CNRS, Nantes, URL : `http://altarica.labri.fr/`.

[DAW 95] DAWS C., OLIVERO A., TRIPAKIS S., YOVINE S., « The Tool KRONOS », *Hybrid Systems III : Verification and Control*, vol. 1066, Rutgers University, New Brunswick, NJ, USA, 22–25 October 1995, Springer, p. 208–219.

[HEN 97] HENZINGER T. A., HO P. H., WONG-TOI H., « HYTECH : A model checker for hybrid systems », *Lecture Notes in Computer Science*, vol. 1254, 1997, p. 460– ??

[JEA 02] JEANNET B., « Convex Polyhedra Library », release 1.1.3c édition, mars 2002, Documentation of the "New Polka" library available at `http://www.irisa.fr/prive/Bertrand.Jeannet/newpolka.html`.

[JOU 96] JOURDAN M., MARANINCHI F., « Vérification de systèmes réactifs en Argos temporisé », HERMÈS, Ed., *Actes du Congrés Modélisation des Systèmes Réactifs, MSR'96*, 1996.

[LAR 98] LAROUSSINIE F., LARSEN K., « CMC : A tool for compositional model-checking of real-time systems », *IFIP Joint Int. Conf. Formal Description Techniques & Protocol Specification, Testing, and Verification (FORTE-PSTV'98)*, vol. 1066, Paris, France, November 1998, Kluwer Academic, p. 439–456.

[PAG 03] PAGETTI C., CASSEZ F., ROUX O., « Tarc : A Timed Hierarchical Modelling Tool », *soumis à FACS 03*, 2003.

[PET 00] PETTERSSON P., LARSEN. K. G., « UPPAAL2k », *Bulletin of the European Association for Theoretical Computer Science*, vol. 70, 2000, p. 40–44.

[POI 99] POINT G., RAUZY A., « AltaRica - Langage de modélisation par automates à contraintes », HERMÈS, Ed., *Actes du Congrés Modélisation des Systèmes Réactifs, MSR'99*, 1999.

[POI 00] POINT G., « Altarica : Contribution à l'unification des méthodes formelles et de la sûreté de fonctionnement », PhD thesis, University of Bordeaux I, Janvier 2000.

[RAU 02] RAUZY A., « Mode automata and their Compilation into Fault Trees », *Reliability Engineering and System Safety*, vol. 78, 2002, p. 1–12, URL : `http://iml.univ-mrs.fr/~arauzy/publis/publis.html`.

[SEG 01] SEGUIN C., « Modèles formels pour l'évaluation de la sûreté de fonctionnement des architectures logicielles d'avionique modulaire integrée », *AFADL*, , 2001.

[SIF 96] SIFAKIS J., YOVINE S., « Compositional specification of timed systems », *13th Annual Symp. on Theoretical Aspects of Computer Science, STACS'96*,, vol. 1046 de *lncs*, 1996, p. 347–359, Invited paper.