Conception et validation d'un protocole avec le modèle AltaRica

Alain Griffault

LaBRI (UMR CNRS 5800), Université Bordeaux I
351, cours de la libération
33405 Talence Cedex
Alain.Griffault@labri.fr

Résumé. Le langage AltaRica est issu d'une collaboration industrielle avec le LaBRI. Un des objectifs principaux était de pouvoir décrire, dans un même formalisme, les aspects dysfonctionnels et les aspects comportementaux d'un système complexe. Divers compilateurs vers des outils classiques permettant ensuite d'analyser soit qualitativement, soit quantitativement le système.

AltaRica permet la description hiérarchique de systèmes constitués de composants qui peuvent interagir : soit par la synchronisation des actions des composants pour les aspects événementiels, soit par la coordination des flux des composants pour les aspects flots de données.

Le papier présente l'utilisation couplée d'AltaRica et du vérificateur de modèle Mec dans la cadre d'un contrat industriel. L'étude a permis de concevoir et de valider un protocole dans le domaine de la téléphonie dont l'objectif est de réaliser un système de commande pour désactiver et réactiver des signaux émis par un téléphone portable.

1 Introduction

Le projet AltaRica est né en 1996 de la volonté d'industriels et de chercheurs d'établir divers ponts entre : la sûreté de fonctionnement et les méthodes formelles, les analyses quantitatives des dysfonctionnements et les analyses qualitatives des comportements fonctionnels, les différents outils et méthodes d'aide à la modélisation, afin de fournir aux ingénieurs concepteurs de systèmes complexes où critiques un atelier outillé.

Les partenaires : les sociétés IXI et ELF Aquitaine Production d'une part ; les laboratoires LaBRI et LADS d'autre part ont, lors de la première phase, défini et validé par l'expérimentation sur des cas tests le langage AltaRica .

La deuxième phase a permis de fixer la sémantique du langage [AGPR00], de développer des prototypes d'outils nécessaires à la simulation d'un modèle AltaRica , de développer des prototypes de compilateurs vers les outils Aralia et Mec et d'écrire un premier manuel méthodologique.

Depuis, le projet AltaRica a donné naissance à plusieurs projets.

- Dassault Aviation a développé l'atelier Cécilia OCAS dédié métier dans lequel certains concepts du langage non pertinents pour eux, et algorithmiquement coûteux sont interdits, mais dans lequel quelques *patterns* parmi les plus utilisés sont devenus des *macros*.

– Le LaBRI, avec d'autres laboratoires de recherches, continue d'étudier et d'étendre le modèle. (`http://altarica.labri.fr/`)

Parallèlement, plusieurs expérimentations d'utilisation ont étés menées par les différents partenaires du projet [PR99,CS02].

La société WideWave S.A. est une jeune entreprise bordelaise. Un de ses projets est la réalisation du brevet suivant déposé par un de ses créateurs : *Un système de commande à distance pour désactivation et réactivation des signaux émis par un téléphone portable et procédé de mis en œuvre dudit système.* Avant de réaliser un prototype de son système, la société a souhaité le valider, d'où cette étude sur la conception et la validation d'un protocole lié à la téléphonie.

La première partie de l'article décrit le système étudié et les objectifs de l'étude. La seconde partie décrit les outils utilisés : le langage AltaRica , le vérificateur de modèle Mec et la méthodologie utilisée. L'étude réalisée est détaillée dans la troisième, et une conclusion fait le bilan de cette expérience.

2 Les objectifs de l'étude

L'étude porte sur un système de commande à distance pour désactiver et activer des signaux émis par un téléphone portable. Plus précisément, nous devons analyser le protocole proposé par le client, et éventuellement lui en proposer.

De façon très générale, l'environnement (un complexe hospitalier, hôtelier, cinématographique ...) peut être assimilé à un ensemble de salles reliées par des portes, certaines donnant accès à l'extérieur. A chaque salle, est associé un niveau de contrôle, correspondant à des droits différents pour le propriétaire d'un portable s'y promenant. A chaque porte est associé un équipement dont le but est de changer l'état interne du portable.

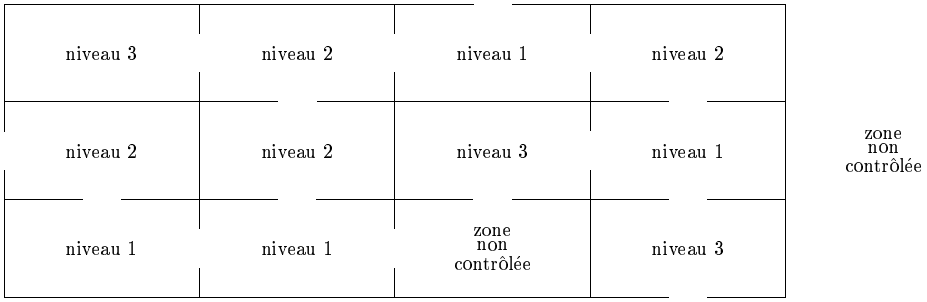

Fig. 1 – L'environnement

Plusieurs types de résultats sont envisageables :

Sur les protocoles, une étude qualitative des comportements attendus, et ce en absence et en présence de défaillances de l'environnement.

Sur les portes, une étude qualitative des besoins en types d'émetteurs.

Sur les labyrinthes formés par les salles, des règles de construction.

En fait, l'objectif peut être vu comme la synthèse d'un contrôleur pour le portable, et ce pour un environnement quelconque du type de la figure 1.

3 Les outils utilisés

MEC [ABC94] est un vérificateur de modèle développé au LaBRI depuis 1986. Un système y est décrit soit directement par un système de transition étiqueté, soit calculé comme un produit de synchronisation de systèmes de transition, chacun représentant un composant du système. Les propriétés à vérifier sont décrites dans une logique avec point fixe sans alternance.

MEC a été utilisé avec succès dans de nombreux projets industriels, mais toujours par des universitaires comme experts. Dans ce projet, nous cherchons également à montrer que la vérification formelle reste un des outils les plus efficaces pour concevoir des systèmes complexes, c'est pourquoi nous avons utilisé pour cette étude :

- le langage de description ALTARICA ,
- le simulateur ALTARICA développé par Gérald Point [POI00], pour la mise au point des composants de base, et la compréhension de certains scénarios,
- le compilateur `alta2mec` qui permet d'obtenir la sémantique d'un composant ALTARICA au format MEC .
- le vérificateur de modèle MEC pour réfuter ou comparer les propositions.

Nous présenterons le langage ALTARICA au travers de l'étude réalisée, le concept de diffusion ne sera donc pas abordé dans cet article. De même la présentation de MEC sera restreinte au strict nécessaire pour l'étude.

4 La solution proposée par la société

La proposition consiste à équiper chaque porte de deux émetteurs de signaux (A et B) capturés et interprétés par le téléphone portable (cf figure 2).

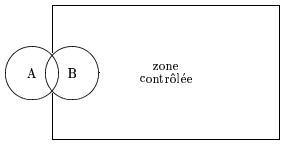

Fig. 2 – La configuration SallePorteAB

La combinaison des deux signaux, en délimitant cinq zones, doit permettre de définir une zone sous contrôle et de distinguer entre le demi-tour et le franchissement réel d'une porte. Dans la figure 3, d'une part les étiquettes `+X` (resp. `-X`) indiquent la réception (resp. la perte) du signal `X`; d'autre part les états indiquent la zone `[0-4]` et si le portable est sous contrôle (`[F|T]`).

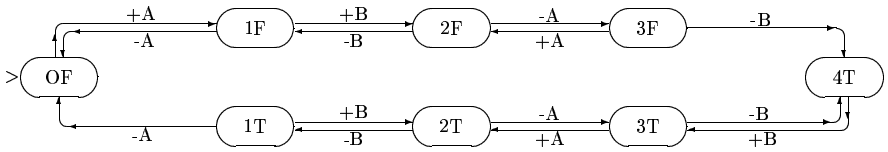

Fig. 3 – L'idée du protocole de contrôle (zone, contrôle)

5 L'étude de la solution proposée

Dans cette partie, nous allons décrire le langage AltaRica au travers de la description des divers composants du système. Pour chacun, nous donnons : l'idée par des commentaires et/ou un schéma ; le code AltaRica ; et la sémantique, non pas sous la forme textuelle obtenue par le compilateur `alta2mec`, mais par un système de transitions dessiné lorsque sa complexité le permet.

AltaRica est un langage qui permet de décrire un automate à contrainte. Une configuration est une valuation des variables, celles-ci sont de deux natures : les flux qui font d'AltaRica un langage flots de données, et les états qui changent en fonction d'événements comme dans un automate classique. Une partie `assert` permet de contraindre les flux en fonction des états. Plus précisément, à partir d'une configuration, lorsqu'un événement se produit, les variables d'états sont d'abord affectées à leurs nouvelles valeurs, puis les variables de flux sont calculées afin que les contraintes définies par les assertions soient respectées.

5.1 Les émetteurs

Un émetteur sûr (figure 4)

```
/* Cet émetteur "sécurisé" n'est jamais en panne,
 * il émet un signal "vrai" en continu.
*/
node EmetteurSur
  flow   signal : bool;
  assert signal = true;
edon
```

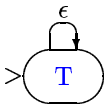

Fig. 4 – Le composant EmetteurSur (signal)

Un émetteur pouvant être en panne (figure 5)

```
/* Cet émetteur émet un signal "vrai" en continu ssi
 *  - il n'est pas en panne.
*/
node Emetteur
  flow   signal : bool;
  state  ok : bool;
  event  panne;
  trans  ok |- panne -> ok := false;
  assert signal = ok;
  extern initial_state = ok = true;
edon
```

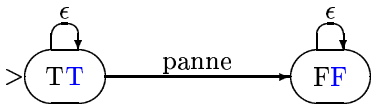

Fig. 5 – Le composant Emetteur (ok, signal)

5.2 Les portes

AltaRica permet une description hiérarchique d'un système en déclarant des instances d'autres composants à l'intérieur d'un composant. Le composant père peut être vu comme un contrôleur de ses fils : d'une part il peut synchroniser leurs événements, d'autre part il peut contraindre leurs flux.

Une porte équipée d'émetteurs sûrs (figure 6)

```
/* Cette porte est équipée d'émetteurs "sûrs".
 * Elle émet donc un signal A "vrai" en continu ssi
 *  - "quelqu'un" est dans la zone couverte par l'émetteur A
 * un signal B "vrai" en continu ssi
 *  - "quelqu'un" est dans la zone couverte par l'émetteur B
 * Hypothèse :
 * - elle n'est correcte que pour un seul "passant".
*/
node PorteSur
  sub    A, B : EmetteurSur;
  flow   signalA, signalB : bool;
  state  inA, inB : bool;
  event  entreeA, entreeB, sortieA, sortieB;
  trans
```

```
    true |- entreeA -> inA := true;
    true |- sortieA -> inA := false;
    true |- entreeB -> inB := true;
    true |- sortieB -> inB := false;
  assert
    signalA = (inA & A.signal);
    signalB = (inB & B.signal);
  extern initial_state = inA = false, inB = false;
edon
```

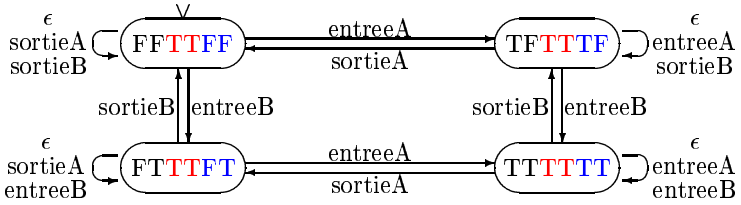

Fig. 6 – PorteSur (inA, inB, A.signal, B.signal, signalA, signaB)

Une porte équipée d'émetteurs non sûrs Par rapport à la porte précédente, il faut ajouter les synchronisations des défaillances.

```
/* Cette porte émet
 * un signal A "vrai" en continu ssi
 *  - l'émetteur A n'est pas en panne
 *  - "quelqu'un" est dans la zone couverte par l'émetteur A
 * un signal B "vrai" en continu ssi
 *  - l'émetteur B n'est pas en panne
 *  - "quelqu'un" est dans la zone couverte par l'émetteur B
 * Hypothèses :
 * - les pannes des deux émetteurs peuvent être simultanées.
 * - elle n'est correcte que pour un seul "passant".
*/
node Porte
  sub    A, B : Emetteur;
  flow   signalA, signalB : bool;
  state  inA, inB : bool;
  event  panne, entreeA, entreeB, sortieA, sortieB;
  trans
    true |- panne ->;
    true |- entreeA -> inA := true;
    true |- sortieA -> inA := false;
```

```
    true |- entreeB -> inB := true;
    true |- sortieB -> inB := false;
  sync
    <panne, A.panne>;
    <panne, B.panne>;
    <panne, A.panne, B.panne>;
  assert
    signalA = (inA & A.signal);
    signalB = (inB & B.signal);
  extern initial_state = inA = false, inB = false;
edon
```

Le système de transitions obtenu contient 16 états et 100 transitions.

5.3 Le portable

La première version est celle de la figure 7. Elle s'obtient facilement en décrivant les transitions du graphe une à une.

```
/* Le portable mémorise une situation qui code
 * son appartenance aux différentes zones A et B.
 * Hypothèses :
 * - le portable reçoit tous les signaux émis,
 * - il détecte ainsi la présence et l'absence des signaux.
*/
node Portable
  flow   signalA, signalB : bool;
  event  reaction;
  state  situation : [0,4];
         controle : bool;
  trans
    // protocole d'entrée
    situation = 0 & signalA  |- reaction -> situation := 1;
    situation = 1 & signalB  |- reaction -> situation := 2;
    situation = 2 & ~signalA |- reaction -> situation := 3;
    situation = 3 & ~signalB |- reaction -> situation := 4,
                                            controle := true;
    // protocole de sortie
    situation = 4 & signalB  |- reaction -> situation := 3;
    situation = 3 & signalA  |- reaction -> situation := 2;
    situation = 2 & ~signalB |- reaction -> situation := 1;
    situation = 1 & ~signalA |- reaction -> situation := 0,
                                            controle := false;
  extern initial_state = situation = 0, controle = false;
edon
```

Le système obtenu qui comprend 32 états et 240 transitions, est fidèle à la description de la figure 3.

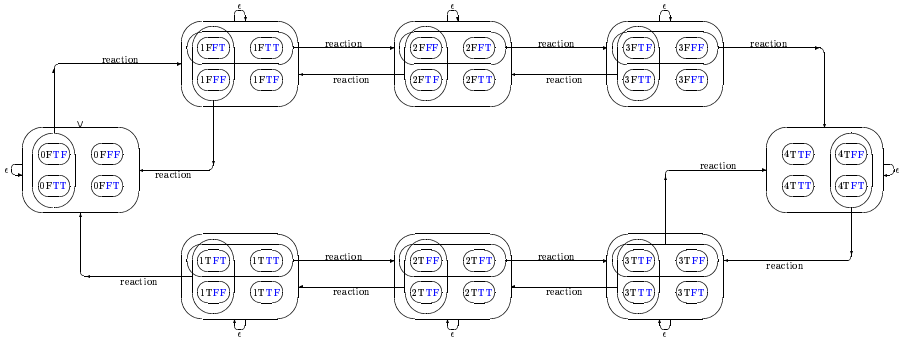

Fig. 7 – Le composant Portable (situation, contrôle, signalA, signalB)

5.4 Le premier modèle étudié (cf figure 2)

Nous faisons l'hypothèse (réaliste) que les réactions du portable aux changements de zones sont très rapides, notamment suffisamment rapides pour qu'elles soient effectives avant un nouveau changement de zone. Le concept AltaRica de priorité entre les événements (cf le symbole < dans la déclaration des événements) permet de modéliser aisément cette hypothèse.

```
/* Exemple étudié :
 * - un portable, - une salle, - une porte
*/
node SallePorteAB
  sub    tel : Portable;
         P : Porte;
  state  zone : [0,4];
  event  reaction > {entreeA, entreeB, sortieA, sortieB, panne};
  trans
    true |- reaction, panne ->;
    // protocole d'entrée
    zone = 0 |- entreeA -> zone := 1;
    zone = 1 |- entreeB -> zone := 2;
    zone = 2 |- sortieA -> zone := 3;
    zone = 3 |- sortieB -> zone := 4;
    // protocole de sortie
    zone = 4 |- entreeB -> zone := 3;
    zone = 3 |- entreeA -> zone := 2;
    zone = 2 |- sortieB -> zone := 1;
    zone = 1 |- sortieA -> zone := 0;
  sync
    <reaction, tel.reaction>;
    <panne, P.panne>;
    <entreeA, P.entreeA>;
```

```
    <entreeB, P.entreeB>;
    <sortieA, P.sortieA>;
    <sortieB, P.sortieB>;
  assert
    tel.signalA = P.signalA;
    tel.signalB = P.signalB;
  extern initial_state = zone = 0;
edon
```

6 La spécification du système

Dans la démarche *vérification de modèles* :

- le système réel est généralement modélisé par un système de transitions qui est une abstraction possible de l'ensemble des comportements du système,
- les propriétés attendues du système sont appelées spécifications et sont généralement décrites par des formules logiques,
- un *vérificateur de modèles* calcule pour chaque propriété de la spécifications les états et/ou les transitions qui la satisfont. Un contre exemple (une exécution) est exhibé lorsqu'une propriété n'est pas satisfaite.

Pour notre étude, nous distinguons les propriétés de validation, les propriétés dysfonctionnelles et les propriétés fonctionnelles du système.

6.1 Les propriétés de validation

Elles servent d'une part à vérifier des propriétés générales ; d'autre part à convaincre le donneur d'ordre que le modèle *ressemble* à son système. Elles ont pour objectif de montrer que le modèle est réaliste, et constituent un préalable obligatoire avant la suite de l'étude. Il n'existe malheureusement pas d'ensemble de propriétés permettant de valider un système. Voici un extrait de celles que nous avons traitées.

```
\* les transitions qui correspondent aux pannes. *\
panne              := !label="panne";
\* les transitions qui correspondent au delai du portable. *\
finDelai           := !label="finDelai";
\* les transitions qui correspondent au reaction du portable. *\
reaction           := !label="reaction";
\* les états stables (n'évoluent pas sans un vrai événement). *\
stable             := * - src(reaction \/ finDelai);
\* les états où le portable devrait être contrôlé. *\
zoneControlee      := stable /\ !state="*zone_4*";
\* les états où le portable ne devrait pas être contrôlé. *\
zoneHorsControle   := stable /\ !state="*zone_0*";
\* les états où le portable est contrôlé. *\
portableControle   := stable /\ !state="*tel_controle_true*";
\* les cfc (composantes fortement connexes) : les circuits. *\
```

```
cfc                := loop(*,*);
\* la cfc qui passe par l'etat initial. *\
cfcInitial         := loop(rsrc(initial),*);
```

6.2 Les propriétés dysfonctionnelles

Un objectif pour cette étude est de savoir si le protocole proposé pour le portable est *résistant* vis à vis des pannes de l'environnement. Ces propriétés doivent donc être satisfaites même si des pannes des composants surviennent.

D-blocage : le système doit être exempt de situation de blocage ;

D-réinitialisable : de toute situation, le portable doit pouvoir revenir naturellement (par une succession d'événements normaux) dans la situation initiale ;

D-contrôle dans les zones sous contrôle, le portable est effectivement contrôlé.

D-hors-contrôle dans les zones hors du contrôle, le portable est dans sa configuration initiale.

```
\* les états sans successeurs : les états de blocage. *\
Dblocage         := * - src(*);
\* les cfc qui interdisent la ré-initialisation du portable. *\
Dreinitialisable := cfc - cfcPortableInitial;
\* les états controlés alors que le portable est non contrôlé. *\
Dcontrole        := zoneControlee - portableControle;
\* les états non controlés avec le portable non ré-initialiser. *\
DhorsControle    := zoneHorsControle - portableInitial;
```

6.3 Les propriétés fonctionnelles

Le système doit satisfaire ces propriétés en l'absence de panne. Le second objectif de cette étude est de vérifier si le protocole proposé pour le portable est correct vis à vis de celles-ci.

Les propriétés à vérifier sont donc identiques aux précédentes, hormis pour la seconde car la ré-initialisation ne concerne alors pas seulement le portable, mais tous le système.

F-blocage : le système doit être exempt de situation de blocage ;

F-réinitialisable : de toute situation, le système doit pouvoir revenir naturellement (par une succession d'événements normaux) dans la situation initiale ;

F-contrôle : dans les zones *déclarées* contrôlées du système, le portable est effectivement contrôlé ;

F-hors-contrôle dans les zones *déclarées* non contrôlées, le portable est dans sa configuration initiale.

Avec MEC , il suffit de calculer le sous-système correspondant aux comportements sans panne, puis d'effectuer des calculs similaires aux précédents.

```
\* les états accessibles sans panne. *\
EFonct          := reach(initial,*-panne);
\* les transitions possibles sans panne. *\
TFonct          := (rsrc(EFonct)\/rtgt(EFonct))-panne;
\* les états sans successeurs du sous automate fonctionnel. *\
Fblocage        := EFonct - src(TFonct);
...
```

7 Les modèles étudiés

L'outil MEC a fourni les résultats du tableau 1. Le système est donc fonctionnellement correct, mais **ne résiste pas aux pannes.**

Tableau 1 – Les propriétés du modèle SallePorteAB

Propriété	#		Un contre-exemple
États	86		
Transitions	250		
D-blocage	0		
D-réinitialisable	77	Bug	entreeA, reaction, P.A.panne
D-contrôle	3	Bug	P.A.panne, entreeA, entreeB, sortieA, sortieB
D-hors-contrôle	3	Bug	entreeA, reaction, entreeB,reaction, P.A.panne, reaction, sortieB, reaction, sortieA
États Fonctionnels	22		
Transitions Fonctionnelles	50		
F-blocage	0		
F-réinitialisable	0		
F-contrôle	0		
F-hors-contrôle	0		

Afin de mieux répondre au client, nous avons prolongé l'étude par plusieurs variantes pour étudier :

- des anomalies comme des émetteurs montés à l'envers (les signaux `A` et `B` inversés) ; ou bien distants créant ainsi une zone supplémentaire,
- la nécessité d'avoir des émetteurs sans pannes dans le système,
- des environnements avec deux ou trois salles et trois ou quatre portes.

Nous avons également proposé et étudié de nouveaux protocoles :

un protocole temporisé pour lequel, afin d'être certain de revenir dans une situation initiale, le passage en mode contrôlé arme une horloge qui lorsqu'elle se déclenchera, remettra le portable dans son état initial. (cf figure 8)

un protocole correcteur des pannes A et B basé sur l'idée que si le site est bien conçu, il n'y a aucune raison de recevoir : ni le signal A dans une

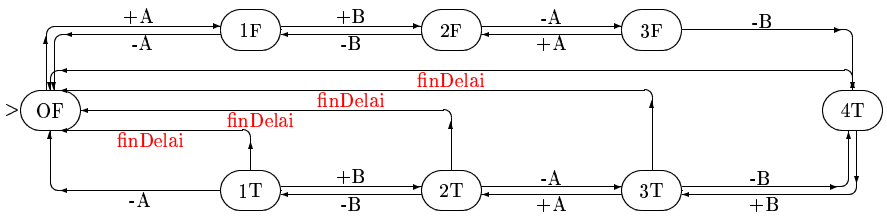

Fig. 8 – L'idée du composant PortableTemporise

zone contrôlée ; ni le signal B dans une zone non contrôlée. La réception du signal A dans l'état 4T (resp. B dans 0F) peut donc être interprété comme le fait que l'émetteur B (resp. A) est en panne. (cf figure 9)

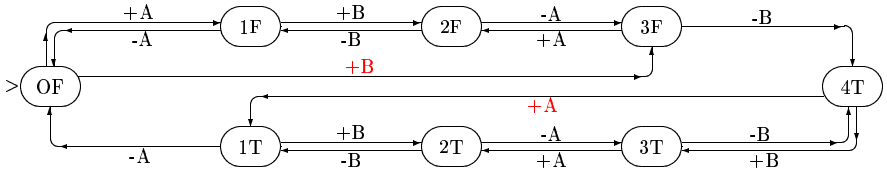

Fig. 9 – L'idée du composant PortablePannesAB

un protocole multi-niveaux qui compte les signaux captés, puis prend le niveau de contrôle correspondant au dernier signal perdu. (cf figure 10)

8 Les résultats et leurs interprétations

8.1 Comparaison des différentes propositions

Nous avons montré que si tous les émetteurs sont sûrs, seul le protocole temporisé peut poser des problèmes. Du fait qu'un émetteur sûr est certainement beaucoup plus onéreux qu'un qui puisse tomber en panne, nous avons montré l'intérêt de disposer d'émetteurs sûrs pour les seules portes donnant sur l'extérieur du complexe contrôlé. Les tableaux 2, 3 et 4 résument les résultats.

La solution temporisée peut provoquer des pertes de contrôle dans des zones contrôlées ; et aucune est satisfaisante en cas de pannes des émetteurs.

La sortie sécurisée nous permet de garantir une propriété cruciale : **le propriétaire du portable n'a pas à subir les conséquences des pannes de l'environnement.** Les deux propriétés non satisfaites nous paraissent moins graves : en cas de pannes de l'environnement, le contrôle n'est pas toujours correctement assuré.

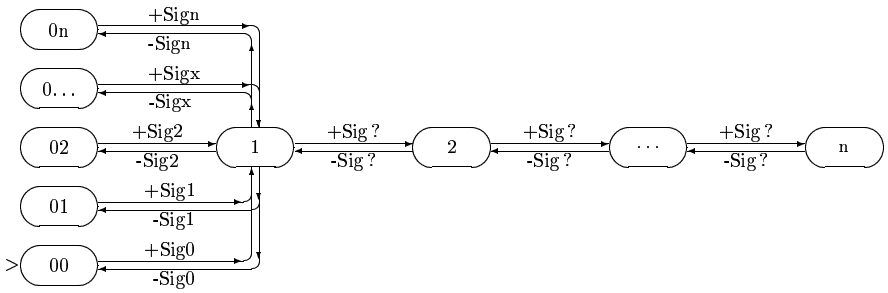

Fig. 10 – L'idée du Portable muti-niveaux (nombre signaux, niveau contrôle)

Tableau 2 – Les propriétés des protocoles sans sortie sécurisée

Modèle	Initial	Temporisé	Pannes A, B	Niveaux
D-blocage				
D-réinitialisable	Bug	Bug	Bug	Bug
D-contrôle	Bug	Bug	Bug	Bug
D-hors-contrôle	Bug		Bug	Bug
F-blocage				
F-réinitialisable				
F-contrôle		Bug		
F-hors-contrôle				

Nous nous sommes limités aux deux anomalies du tableau 4, car elles correspondent à des situations réalistes : une erreur de montage, et un déplacement des émetteurs suite à un choc. **Le fait qu'un éloignement des émetteurs ne perturbe pas le protocole à plusieurs niveaux est un résultat positif.**

8.2 Interprétation des résultats

Le protocole à plusieurs niveaux que nous proposons montre que le projet initial de contrôler un portable est réalisable. Cependant certaines conditions doivent être acceptées :

Le gestionnaire de l'environnement doit accepter qu'en cas de pannes d'un au moins de ses émetteurs, il existe des zones de l'environnement, dans lesquelles le portable n'est pas contrôlé correctement.

Le gestionnaire de l'environnement doit accepter d'installer au moins un passage de sortie sécurisée.

Les fabriquants de portables doivent intégrer le protocole dans leurs appareils, et définir les types de signaux qui peuvent être utilisés.

Tableau 3 – Les propriétés des protocoles avec une sortie sécurisée

Modèle	Initial	Temporisé	Pannes A, B	Niveaux
D-blocage				
D-réinitialisable	Bug	Bug	Bug	Bug
D-contrôle	Bug	Bug	Bug	Bug
D-hors-contrôle				
F-blocage				
F-réinitialisable				
F-contrôle		Bug		
F-hors-contrôle				

Tableau 4 – Les résistances aux anomalies vis à vis de la propriété F-contrôle

Modèle	Initial	Temporisé	Pannes A, B	Niveaux
A et B inversés	Bug	Bug	Bug	Bug
A et B éloignés	Bug	Bug		

8.3 Conseils aux exploitants

Pour un bon fonctionnement, nous invitons l'opérateur téléphonique à :
- choisir le protocole à plusieurs niveaux,
- ne pas mettre de temporisation.

De même, nous conseillons au gestionnaire d'un environnement de :
- mettre, entre deux salles de niveaux différents, un couple d'émetteurs ; l'intersection des zones couvertes étant vide ou non,
- mettre une sortie sécurisée composée d'un seul émetteur de niveau 0,
- ajouter éventuellement des émetteurs redondants à l'intérieur des zones.

9 Conclusion

La démarche a été facile à mettre en œuvre, mais afin de pouvoir exploiter correctement l'outil MEC à partir du source généré par le traducteur `alta2mec`, il est nécessaire de savoir comment ce source a été obtenu. Cette restriction fait que le couple (ALTARICA , MEC) n'est pas encore facilement utilisable par les industriels. Cela justifie le projet actuel du LaBRI de développement d'un nouveau vérificateur de modèles pour le langage ALTARICA .

En suivant nos préconisations, la figure 11 fourni une solution pour notre exemple introductif de la figure 1.

Remarques A posteriori, les résultats obtenus sont sans réelles surprises, et sont mêmes certainement assez pauvres pour un spécialiste de théorie des langages. En réalité, comme de coutume pour les études formelles de systèmes, la principale

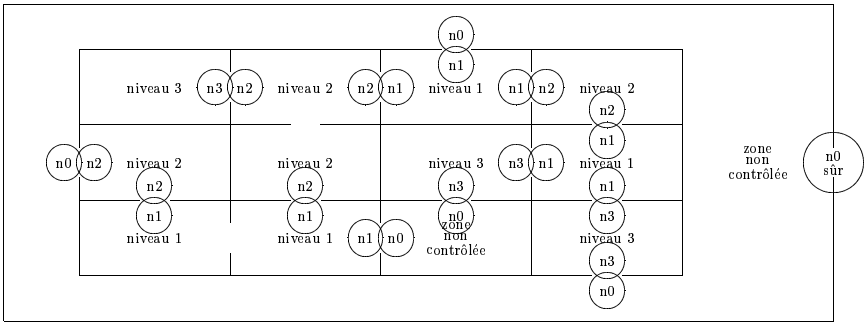

Fig. 11 – Une solution du problème initial

difficulté fut la compréhension du brevet, pour pouvoir en extraire les différentes fonctionnalités du système.

Références

[ABC94] A. ARNOLD, D. BEGAY, and P. CRUBILLE. *Construction and analysis of transition systems with MEC.* World Scientific, 1994.

[AGPR00] A. ARNOLD, A. GRIFFAULT, G. POINT, and A. RAUZY. The altarica formalism for describing concurrent systems. *Fundamenta Informaticae*, 40 :109–124, 2000.

[CS02] C. CASTEL and C. SEGUIN. Modèles formels pour l'évaluation de la sûreté de fonctionnement des architectures logicielles d'avionique modulaire intégrée. In *AFADL*, 2002.

[POI00] G. POINT. *Altarica : Contribution à l'unification des méthodes formelles et de la sûreté de fonctionnement.* PhD thesis, LaBRI, Université Bordeaux I, Janvier 2000.

[PR99] G. POINT and A. RAUZY. Altarica - constraint automata as a description language. *European Journal on Automation*, 1999. Special issue on the *Modelling of Reactive Systems.*