Conflits et contre-mesures dans l'activité de pilotage
Conflicts and countermeasures in the pilot's activity

Fr. Dehais Ch. Lesire C. Tessier L. Chaudron

Onera - DCSD
2 avenue Édouard-Belin
31055 Toulouse cedex 4

{dehais,lesire,tessier,chaudron}@cert.fr

Résumé

Dans le cadre de la sécurité aéronautique, nous proposons une méthodologie de suivi, prédiction et assistance dans l'interaction pilotes/systèmes ; il s'agit d'anticiper la détection des conflits de pilotage afin d'aider à assurer le maintien du niveau de sécurité du vol par la mise en œuvre de contre-mesures adaptées. Notre démarche a reposé sur des expérimentations avec des pilotes en simulateur de vol et sur l'analyse formelle des paramètres de vol. La modélisation des conflits est fondée sur des théories issues à la fois des facteurs humains et de l'intelligence artificielle. Nous présentons le système GHOST, environnement expérimental qui permet de tester différentes contre-mesures aux conflits de pilotage, et notamment au syndrome dit de « persévération ».

Mots-clés

Conflits, suivi de situations, pilotage, expérimentations.

Abstract

In the framework of a study on lessons learned in aeronautics, we propose a methodology to analyze the pilot's activity so as to detect and prevent cognitive conflict in man-artefact interactions. We detail our approach based on an experimental work with pilots in a flight simulator and on the analysis of flight parameters. Conflict modeling is built both on human factors and artificial intelligence approaches. We present GHOST, an experimental environment designed to test on-line countermeasures to help pilots facing cognitive conflicts.

Keywords

Conflicts, situation tracking, aeronautics, experiments.

1 Introduction

La revue d'incidents aéronautiques a montré que l'apparition de conflit dans la gestion du vol est un facteur de dégradation de l'activité et qu'elle est à l'origine de nombreux accidents [1]. Ces conflits peuvent se produire entre l'équipage et le contrôle aérien (Air Philippines, avril 1999), entre l'homme et les systèmes embarqués (collision aérienne entre un Tupolev 154 et un Boeing 757 au-dessus de la Suisse, juillet 2002), dans la représentation de la position de l'aéronef (Korean Air, août 1997), dans la compréhension des consignes (Streamline, mai 2000), ou encore dans l'utilisation des procédures de vol (Crossair, janvier 2000). Pour améliorer la sécurité aérienne, l'idée est de développer un système capable de détecter les conflits suffisamment tôt et d'envoyer des contre-mesures en temps réel pour assister le pilote. Un tel système suppose de disposer d'un moyen de suivi et de prédiction de l'activité du pilote, d'un moyen de détection des conflits et d'un moyen d'élaboration de contre-mesures adaptées à la situation dans laquelle se trouve le pilote (figure 1).

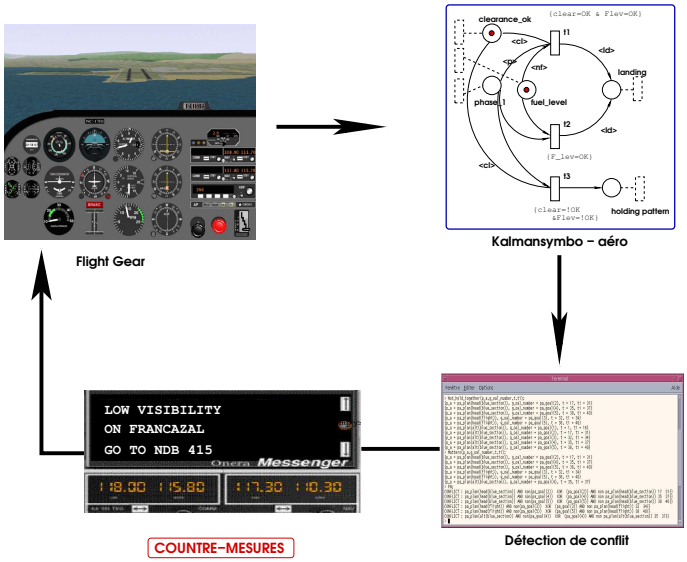

FIG. 1 – *Outil d'assistance au pilotage*

Après avoir défini la notion de conflit, nous décrivons le scénario expérimental de base, puis examinons successivement le suivi de situation et la détection de conflits (paragraphe 4) puis l'envoi de contre-mesures et les résultats expérimentaux (paragraphe 5).

2 Le conflit

Les travaux relatifs au concept de conflit sont essentiellement menés dans le domaine des systèmes multiagents et visent surtout à éviter, résoudre ou à se débarrasser du conflit par négociation [2, 3, 4], argumentation [5], satisfaction de contraintes [6, 7, 8] plutôt qu'à le modéliser et à comprendre son apparition. Par ailleurs, dans cette discipline, les conflits sont assimilés à une incohérence formelle, réduction peu compatible avec le monde réel : dans le domaine du pilotage il est admis que l'apparition d'incohérences (panne, météo, données erronées de l'interface) n'est pas nécessairement conflictuelle, et ne s'oppose pas forcément à la poursuite de la mission. Ces incohérences deviennent conflictuelles seulement si elles sont *cruciales* pour la réalisation d'un but particulier de la mission ou pour la sécurité de l'équipage. Ce point de vue se rapproche de la définition d'Easterbrook [9] mais aussi des travaux en sciences sociales [10, 11] qui voient dans le conflit l'impossibilité pour un agent ou un groupe d'agents d'atteindre un but *qui a une importance*.
Nous avons proposé dans [12, 13], la définition suivante pour les conflits :

Définition 2.1 (Conflit) *un conflit est un ensemble de connaissances qui n'est pas cohérent*[1] *et le fait qu'il ne soit pas cohérent importe. Des connaissances* cruciales *détermineront le fait qu'une incohérence importe.*

3 La mission *Ravitaillement*

La mission qui a servi à mettre en place et à expérimenter les hypothèses pour élaborer l'outil d'assistance à la résolution des conflits est une mission simple de gestion du carburant (figure 2).

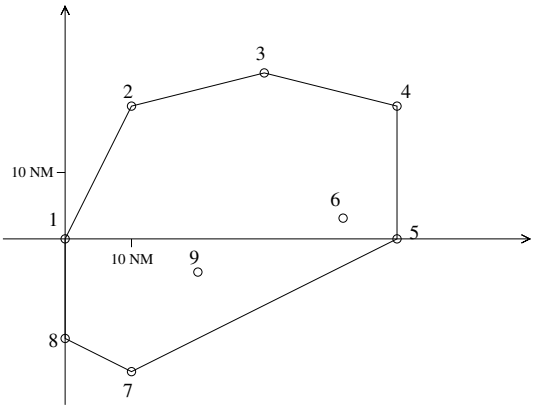

FIG. 2 – *Mission* Ravitaillement *(NM: Mille Nautique)*

Le pilote doit suivre le plan de vol suivant : décoller (la piste se trouve entre 1 et 8 et orientée vers 1), passer, dans l'ordre, par les points tournants 1, 2, 3, 4, 5, 7 et 8, et atterrir en 1. Le pilote a la possibilité de se ravitailler en passant à la verticale des points 6 et 9. Il peut se diriger vers 6 s'il est entre 4 et 5 ou entre 5 et 7. Il peut se diriger vers 9 s'il est entre 5 et 7.

1. La notion de cohérence dépend du formalisme utilisé pour représenter les connaissances.

4 Suivi et prédiction de l'activité du pilote, détection de conflit

Afin de suivre et prédire l'activité de pilotage lors d'une mission, nous adaptons l'estimateur symbolique *Kalmansymbo* [14] au domaine aéronautique - *Kalmansymbo-aéro*. Le suivi de l'activité du pilote est fondé sur le principe du suivi de situation classiquement utilisé en diagnostic, pour la surveillance ou le renseignement [15, 16, 17]. Quel que soit le formalisme employé pour représenter la connaissance, le suivi de situation repose sur une mise en correspondance entre des faits observés (observations) et des situations connues (modèles de référence), et son objet est de réaliser des postdictions, des prédictions ou de diagnostiquer l'état courant du système. La détection de conflit intervient lorsque la mise en correspondance révèle une inadéquation entre ce qui est attendu et ce qui est observé.

4.1 Modèle de référence

Le formalisme choisi est celui des réseaux de Petri à objets, dont la définition est issue de [18]. Une mission, selon sa complexité et le niveau de détail du plan de vol, pourra être représentée par un ou plusieurs réseaux de Petri. Les paramètres de vol (qui seront directement issus du simulateur de vol lors des expérimentations) sont encapsulés dans le *jeton-mission*, unique instance de la classe $Jeton$.
Exemple: la figure 3 montre les paramètres de vol utilisés pour la mission *Ravitaillement*.

Jeton
horloge
latitude
longitude
vitesse
fuel
consommation

FIG. 3 – *La classe Jeton de la mission* Ravitaillement

Modélisation des propriétés attendues. Les éléments de connaissance qui sont associés aux places et transitions du réseau de Petri correspondent chacun à un ensemble de faits ou propriétés qui doivent être vérifiés ensemble à un moment donné de la mission. Il en est de même des observations. Cette question est très générale : qu'il s'agisse de représenter la connaissance de perception ou celle de l'action, ce sont des *conjonctions de propriétés et de contraintes* qu'il faut modéliser. Or, traditionnellement orientée vers les techniques de résolution et de déduction, l'IA s'appuie sur le modèle logique des clauses, qui est une représentation disjonctive de la connaissance. C'est pourquoi un modèle dual de celui des clauses a été défini, prouvé et programmé : le modèle des *cubes* [19].
La structure des *cubes* est bâtie sur un modèle algébrique

spécifique de conjonctions de littéraux du premier ordre (et de contraintes numériques).
Exemple : le cube `c = {waypoint(5,unreached), fuel(OK), alt(3100)}` est constitué de la conjonction logique de trois littéraux dont la sémantique est que les trois propriétés sont simultanément vérifiées.
Formellement (modulo un opérateur de réduction, [20]) on peut munir l'ensemble des cubes, dénoté C, de deux opérateurs sup. $\sqcup_c$ et inf. $\sqcap_c$ (et par conséquent d'une relation d'ordre $\leq_c$ induite qui quotiente l'implication logique) et montrer que $(C, \leq_c, \sqcup_c, \sqcap_c)$ est un treillis non-complet et non-modulaire.
Ces bonnes propriétés permettent d'utiliser les cubes dans une gamme variée d'applications (analyse d'incidents, représentation de performances humaines...)

Places du réseau de Petri. Aux places du réseau de Petri sont associées les *activités attendues* au cours du déroulement de la mission. Une activité est représentée par le prédicat $act(c)$, c étant un cube.
Exemple : en ce qui concerne la mission *Ravitaillement*, ces activités correspondent au ralliement ou au survol d'un point tournant ; `act(waypoint(2,unreached), [t1,t2], fuel(OK), alt(3000))` est l'activité de ralliement du point tournant 2, qui doit se dérouler dans l'intervalle de temps [t1,t2], avec une quantité de carburant suffisante et à une altitude de 3000 ft.
Remarque : par souci de lisibilité des figures et de l'interface graphique de *Kalmansymbo-aéro*, seule apparaît aux places la propriété `waypoint(n,e)`, avec `n` numéro du point tournant et `e` son état (*reached* ou *unreached*) – figure 4.

Transitions du réseau de Petri. Aux transitions du réseau de Petri sont associées les *conditions cruciales* de passage entre activités, exprimées sous la forme d'un cube.
Exemple : `fuel(2,OK)` associé à une transition signifie que le paramètre carburant est crucial pour la réalisation de l'activité en aval de la transition (ralliement du point tournant 2).
Les transitions portent également des actions, représentées par des prédicats, qui agiront sur le *jeton-mission* pour modifier ses attributs.
Exemples : `goToWp(2)` simule le déplacement vers le point tournant 2, `refuel()` simule le ravitaillement.
Graphiquement les transitions sont étiquetées sous la forme *cond / a* où *cond* est le cube des conditions cruciales et *a* une conjonction d'actions – figure 4.

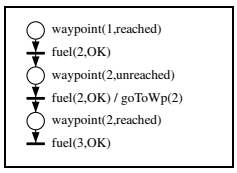

FIG. 4 – *Modèle partiel de la mission* Ravitaillement

Les transitions de changement de but. Certaines transitions sont particulières car elles représentent un changement de but potentiel en cours de mission. Pour les différencier des transitions classiques représentées par un rectangle plein, on représente les transitions de changement de but par un rectangle vide. Des conditions cruciales sont également associées à ces transitions.
La particularité de ces transitions en phase de prédiction des activités sera expliquée au paragraphe 4.4.
Exemple : pour la mission *Ravitaillement*, une telle transition correspond à un déroutement vers un point tournant différent de celui qui est en cours de ralliement ; sur la figure 5, la transition de changement de but est présente pour modéliser l'abandon du point tournant 5 (`waypoint(5, unreached)`) pour se diriger vers le point tournant 6 (`waypoint(6,unreached)`).

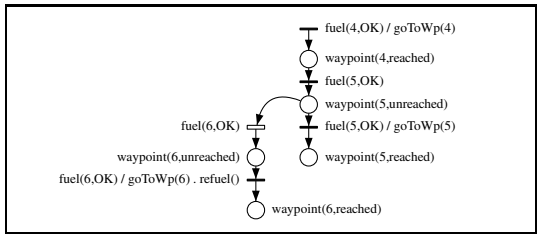

FIG. 5 – *Transition de changement de but*

Les boucles. Lors d'une mission, un pilote peut être amené à accomplir plusieurs fois la même activité ; il est donc indispensable de gérer les ≪ boucles ≫ dans les réseaux de Petri. Cependant ces boucles vont poser des problèmes lors de la prédiction d'activités, les activités des boucles pouvant théoriquement être prédites une infinité de fois. On décide donc provisoirement, dans le cadre de cet article, de limiter le nombre de parcours des boucles à 1. On peut alors ≪ casser ≫ les boucles en dupliquant certaines places, et ainsi rendre ≪ linéaire ≫ le réseau de Petri.
Exemple : la figure 6 est un exemple de sous-réseau comportant une boucle : ralliement du point tournant 5 (`waypoint(5,unreached)`), changement de but pour rallier 6 (`waypoint(6,unreached)`), ravitaillement (`waypoint(6,reached)`), et reprise du ralliement de 5 (`waypoint(5,unreached)`). On remplace ce modèle par celui de la figure 7, dans lequel l'activité `waypoint(5,unreached)` a été dupliquée pour distinguer les phases avant et après ravitaillement.

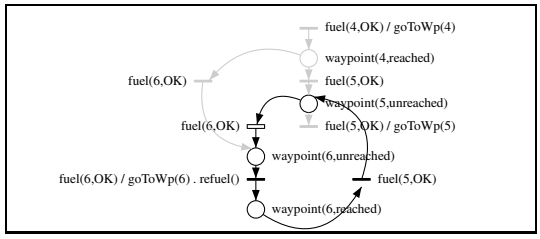

FIG. 6 – *Boucle*

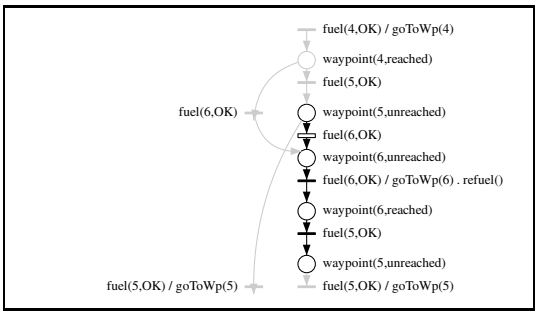

FIG. 7 – *Boucle dépliée*

Observation. L'activité du pilote est observée à travers ses actions sur l'interface du simulateur et donc des paramètres de vol enregistrés. À chaque pas de temps, ces paramètres sont traités et traduits [12] pour construire une *observation*, représentée par le prédicat $obs(C)$, C étant un cube.
En ce qui concerne la mission *Ravitaillement*, une observation est construite à partir des paramètres issus du simulateur de vol toutes les 2 secondes.
Exemple : `obs(waypoint(5, unreached), [45,45], fuel(OK), alt(3100))` signifie que l'activité observée au temps 45 est le ralliement du point tournant 5, que la quantité de carburant est suffisante pour ce ralliement et que l'altitude est 3100 ft.

4.2 Mise en correspondance

La mise en correspondance d'une observation avec une activité attendue est le processus de base de *Kalmansymboaéro* et de la détection de conflits. Cette mise en correspondance se fait en deux étapes : (1) détermination des mises en correspondance possibles à l'aide de propriétés discriminantes ; (2) qualification des mises en correspondance possibles.

Détermination des mises en correspondance possibles. On fait l'hypothèse – raisonnable pour le domaine aéronautique concerné – qu'une ou plusieurs propriétés discriminantes caractérisent une activité donnée.

Définition 4.1 (Propriété discriminante) *une propriété discriminante est la signature d'une activité. Elle permet d'identifier sans ambiguïté sa nature.*

Exemples : la propriété d'enfoncement des amortisseurs de trains caractérise l'activité de toucher des roues.
Dans le cas de la mission *Ravitaillement*, les propriétés `waypoint(n,unreached)` ou `waypoint(n,reached)` d'indication de ralliement ou de survol d'un point tournant sont discriminantes.

Définition 4.2 (Mise en correspondance possible) *Une mise en correspondance est* possible *entre une observation $obs(C)$ et une activité $act(c)$ si et seulement si les propriétés discriminantes de c sont présentes dans C. Dans le cas contraire, la mise en correspondance entre $obs(C)$ et $act(c)$ est dite* impossible.

Exemple : la mise en correspondance entre l'observation `obs(waypoint(5, unreached), [45,45], fuel(OK), alt(3100))` et l'activité attendue `act(waypoint(5,unreached), [50,60], fuel(OK), alt(3000))` est possible ; en revanche la mise en correspondance entre `obs(waypoint(5, reached), [45,45], fuel(OK), alt(3100))` et `act(waypoint(5,unreached), [50,60], fuel(OK), alt(3000))` est impossible.

Marquage du réseau de Petri. Une mise en correspondance possible entre une observation et une activité attendue se traduit par le marquage, par le *jeton-mission*, de la place correspondant à cette activité dans le réseau de Petri. On obtient donc autant de réseaux de Petri marqués que de mises en correspondance possibles. On notera par $\bullet$ le marquage d'une place.

Qualification des mises en correspondance possibles. Dans le cas où la mise en correspondance est possible, l'observation ne va cependant pas, en général, correspondre exactement aux activités attendues en ce qui concerne les propriétés non discriminantes. Cela est dû en particulier à la simplification introduite par la modélisation, et à la manière personnelle d'agir de chaque pilote.
Exemple : en ce qui concerne l'observation `obs(waypoint(5, unreached), [45,45], fuel(OK), alt(3100))` et l'activité attendue `act(waypoint(5,unreached), [50,60], fuel(OK), alt(3000))`, le pilote a parcouru sa route ≪ plus vite ≫ que prévu en volant ≪ plus haut ≫.

Définition 4.3 (Mise en correspondance parfaite ou partielle) *Une mise en correspondance possible entre une observation $obs(C)$ et une activité $act(c)$ est* parfaite *si et seulement si les propriétés non discriminantes de c correspondent exactement aux propriétés non discriminantes de C. Dans le cas contraire, la mise en correspondance entre $obs(C)$ et $act(c)$ est dite* partielle.

Une mise en correspondance peut être partielle pour différentes raisons : des valeurs numériques ou temporelles ne correspondent pas (voir l'exemple précédent), des valeurs symboliques sont contradictoires (par exemple `fuel(OK)` et `fuel(nonOK)`), des propriétés attendues ne sont pas observées ou au contraire des propriétés observées n'étaient pas attendues, etc., avec toutes les combinaisons possibles.

4.3 Détection de conflit

Un algorithme en deux étapes a été développé [12] pour détecter les conflits à partir de la mise en correspondance entre observations et activités attendues. Cet algorithme, dans un premier temps, analyse les mises en correspondance possibles et détecte parmi elles les mises en correspondance partielles (incohérences) ; dans un deuxième temps, il analyse ces incohérences au vu des conditions cruciales de la mission, afin de mettre en évidence des conflits.

Détection des mises en correspondance partielles. Étant donné une observation et l'ensemble des mises en correspondance possibles de cette observation avec les activités attendues, le prédicat `not_hold_together` détecte les mises en correspondance partielles. Dans l'état actuel des travaux, ce prédicat indique tous les cas de différences entre observation et activité, qui seront considérées comme des incohérences, sans aucune tolérance par exemple sur les valeurs numériques.

Exemples : deux observations au temps 45 et au temps 55 vont être mises en correspondance partiellement avec l'activité `waypoint(5,unreached)` :

```
→not_hold_together(obs(waypoint(5, un-
reached), [45,45], fuel(OK), alt(3100)),
act(waypoint(5, unreached), [50,60],
fuel(OK), alt(3000)));
→not_hold_together(obs(waypoint(5, unrea-
ched), [55,55], fuel(nonOK), alt(3000)),
act(waypoint(5, unreached), [50,60],
fuel(OK), alt(3000))).
```

Détection des conflits. Il s'agit maintenant d'identifier, parmi les incohérences détectées, celles qui *importent* pour le déroulement de la mission. Le prédicat `matters` va donc récupérer, pour chaque activité concernée dans une incohérence, les *conditions cruciales* associées à sa ou ses transition(s) aval et indiquer si l'incohérence importe à travers ces conditions cruciales.

Exemple : la condition `fuel(5,OK)` est cruciale pour l'activité `act(waypoint(2,unreached), [50,60], fuel(OK), alt(3000))` (voir figure 5). Le conflit suivant est donc détecté au temps 55 :

```
→matters(obs(waypoint(5, unreached),
[55,55], fuel(nonOK), alt(3000)),
act(waypoint(5,unreached), [50,60],
fuel(OK), alt(3000))),
```

ce qui signifie que le pilote n'a pas suffisamment de carburant pour atteindre le point tournant 5.

Remarque : dans cette phase de vol, l'altitude et le temps ne sont pas cruciaux (et n'apparaissent donc pas sur les transitions), on n'a donc pas de conflit pour l'observation au temps 45.

4.4 Prédiction

La prédiction permet, à partir d'une situation reconnue représentée par un réseau de Petri marqué, de calculer une situation prédite, c'est-à-dire un ensemble d'activités susceptibles d'être observées par la suite. Ce calcul se fait en construisant l'arbre des marquages accessibles du réseau. Nous détaillons ici les particularités de ce calcul liées à la modélisation choisie. On notera par $\times$ une place correspondant à une activité prédite.

Les transitions de changement de but. Dans le cas idéal, un pilote ne change pas de but : il prévoit sa route lors de la préparation de mission. Ainsi, lors de la phase de prédiction, on va suivre ce modèle « idéal » et ne pas prédire le franchissement des transitions de changement de but, sauf dans le cas où la place immédiatement en amont d'une telle transition est *marquée* (avec le jeton •).

Exemple : dans le cas de la mission *Ravitaillement*, la première transition de changement de but rencontrée est celle qui permet au pilote de se dérouter pour aller se ravitailler en 6. Ainsi, lorsque l'activité `waypoint(5,unreached)` est prédite, la transition de changement de but n'est pas franchissable pour la prédiction (figure 8). Elle le devient lorsque cette activité est marquée (figure 9).

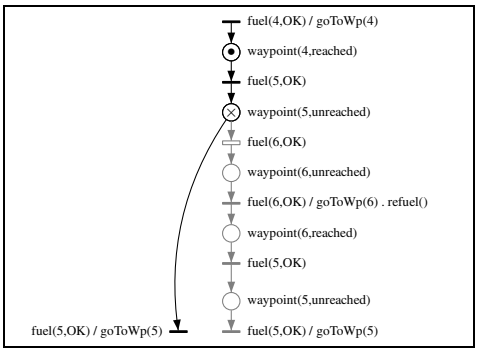

FIG. 8 – *Transition de changement de but non franchissable en prédiction*

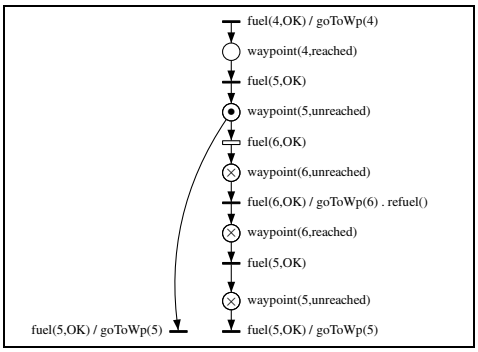

FIG. 9 – *Transition de changement de but franchissable en prédiction*

Propriétés des activités prédites. Lors de la prédiction, le *jeton-mission* franchit virtuellement des transitions, et ses paramètres sont modifiés à chaque franchissement virtuel.

Exemple : lors du franchissement virtuel de la transition qui porte l'action `goToWp(3)`, les paramètres prédits du *jeton-mission*, sont les suivants :

- modification de ses coordonnées

$position := coordonnées(Wp3)$

- modification de son horloge

$horloge := horloge + tempsPourAllerA(Wp3)$

- consommation de carburant

$niveau := niveau - conso * tempsPourAllerA(Wp3)$.

En prédiction, l'horloge indique l'heure à laquelle l'avion est prévu dans une activité. Cette heure est communiquée à la place correspondante. On obtient donc un intervalle construit à partir de l'ensemble des heures où le *jeton-*

mission est prévu dans la place. On nommera ces intervalles les *intervalles de prédiction*.
Exemple : la figure 10 illustre ce principe.

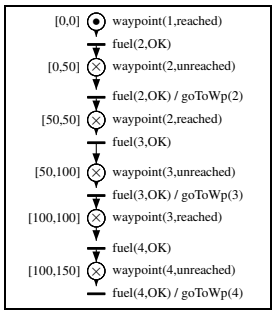

FIG. 10 – *Intervalles de prédiction*

L'état initial correspond à l'activité `waypoint(1,reached)` à $t = 0$. La première transition ne modifie pas la valeur de l'horloge du jeton car elle ne porte pas d'action. L'intervalle de la première place est donc $[0,0]$. En prédiction, le jeton entre dans la deuxième place à $t = 0$, puis franchit la deuxième transition : son horloge est modifiée, $t = 50$, temps nécessaire pour aller de 1 à 2, étant donné la vitesse de l'avion. On considère donc que le jeton quittera cette place à $t = 50$, et on construit l'intervalle $[0,50]$. On construit de la même façon les intervalles suivants.
Lors des phases de prédiction successives, on agrandit ces intervalles en y ajoutant les valeurs de l'horloge. Ainsi, une place possédant l'intervalle $[0,50]$ et à laquelle la prédiction indiquerait un intervalle de $[20,60]$ se verrait attribuer l'intervalle de prédiction $[0,60]$.

L'intervalle de prédiction, ainsi que les autres paramètres prédits, viennent instancier le cube de l'activité prédite concernée.

Horizon temporel de prédiction. La définition d'un *horizon temporel de prédiction* permet de limiter la construction de l'ensemble des activités prédites : on arrête la construction lorsque l'horloge prédite du *jeton-mission* va au-delà de l'horizon.
Exemple : dans le cas de la mission *Ravitaillement*, on fixe la valeur de l'horizon à 3 mn. Cette valeur est pertinente car elle permet de prévoir les activités du pilote sur environ deux sélections de points tournants.
Remarque : cet horizon est une notion variable qui peut être adaptée selon les missions ou phases de mission considérées (horizon court pour des phases très dynamiques, horizon plus long pour des phases comme la croisière).

Prédiction et transitions concurrentes. Les réseaux faisant apparaître des transitions concurrentes, une place peut être prédites plusieurs fois (*via* des chemins différents) et donc posséder plusieurs intervalles de prédiction. On pourrait, comme on le fait lors de phases successives de prédiction, faire l'union de ces intervalles. Cependant, une place souvent prédite risque d'avoir un intervalle de prédiction beaucoup trop grand, ce qui enlève tout l'intérêt de l'intervalle lors de la mise en correspondance. On décide, plutôt que d'agrandir l'intervalle de prédiction, de dupliquer le réseau : lorsqu'on est confronté à des transitions en concurrence, on crée une copie du réseau pour chaque transition franchissable.
Le problème qui subsiste alors est le suivant : lors de la phase de recalage suivante, il est possible que la mise en correspondance se fasse sur une place en amont des transitions en concurrence. On est alors amené à réitérer le processus, créant ainsi encore plus de réseaux (figure 11). On remédie à ce problème en supprimant de chaque réseau les transitions concurrentes, excepté celle qui est franchie. On obtient ainsi plusieurs réseaux sans interférence des intervalles de prédiction.

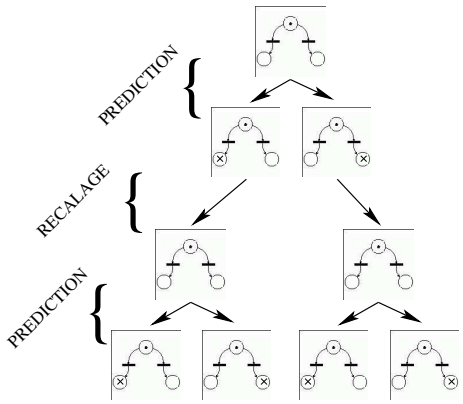

FIG. 11 – *Multiplication des réseaux en présence de transitions en concurrence*

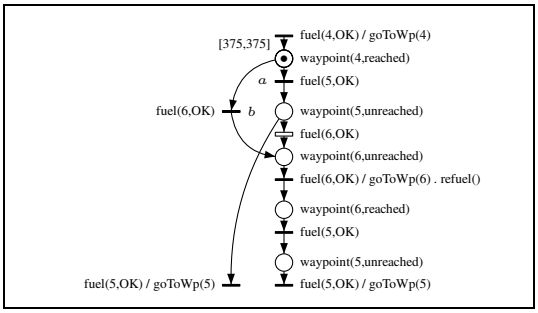

FIG. 12 – *Prédiction multiple : état initial*

Exemple : les figures 12, 13 et 14 illustrent ce principe. À partir de la place marquée, on est confronté à deux transitions en concurrence (notées a et b). On va donc construire deux réseaux correspondant chacun à une de ces deux transitions.
Le réseau de la figure 13 a été privé de la transition b. On effectue donc la prédiction en franchissant la transition a. On procède inversement pour le réseau de la figure 14.

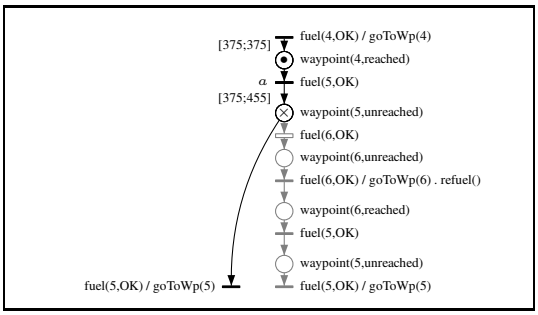

FIG. 13 – *Prédiction multiple : premier chemin*

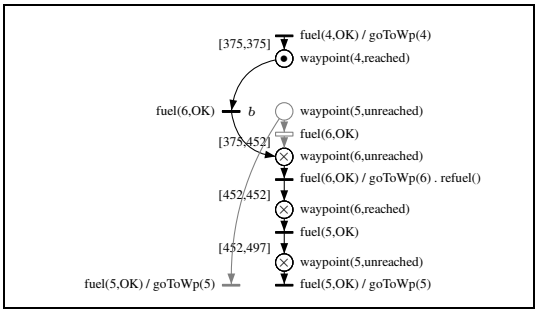

FIG. 14 – *Prédiction multiple : second chemin*

Mise en correspondance et prédiction. La prédiction est effectuée à partir de chaque réseau marqué, donc à partir de chaque mise en correspondance possible (parfaite ou partielle). Ces mises en correspondance étant au préalable passées au crible des propriétés discriminantes, leur nombre reste faible. En ce qui concerne les conflits et les activités prédites calculées à partir de ces conflits, ce sont eux qui seront à l'origine de l'envoi des contre-mesures.

4.5 Recalage

Le recalage est la phase de mise en correspondance d'une observation avec les activités prédites. Il permet le suivi de l'activité, c'est-à-dire de vérifier que les actions perçues du pilote sont en accord - ou non - avec ce qui est prévu. Il permet également de vérifier, en cas de conflit, l'efficacité des contre-mesures envoyées (c'est-à-dire que le pilote ne va pas persévérer dans ce conflit).

Lors de l'arrivée d'une observation, celle-ci est mise en correspondance avec les activités prédites (dans le cas particulier de la première observation, toutes les activités des réseaux de Petri représentant les différents plans de vol que le pilote peut suivre sont *a priori* candidates en tant qu'activités attendues). Les mises en correspondance impossibles sont écartées, tandis que les mises en correspondances possibles (parfaites et partielles) produisent des activités *recalées*. Dans l'état actuel des travaux, on conserve l'ensemble des activités recalées (comme dit plus haut, leur nombre reste faible du fait de l'action des propriétés discriminantes).

Cet ensemble d'activités recalées est ensuite livré à la phase de prédiction en attendant l'arrivée d'une nouvelle observation.

4.6 Logiciel de suivi d'activités

Afin de valider le modèle et les principes définis, nous avons mis au point un logiciel (figure 15) qui met en œuvre les phases successives de prédiction et recalage de *Kalmansymbo-aéro* et permet de visualiser leurs résultats.

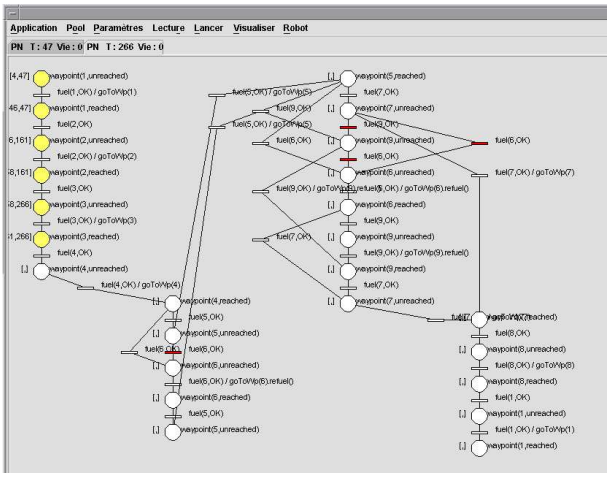

FIG. 15 – *Logiciel de suivi d'activité*

Le suivi et la prédiction d'activité ont été testés avec succès sur les paramètres de vol provenant des expérimentations menées sur la mission *Ravitaillement* (voir paragraphe 5.1).

5 Élaboration des contre-mesures et résultats

5.1 Comportement des pilotes sur la mission *Ravitaillement*

Le scénario de la mission *Ravitaillement* a été proposé à dix pilotes sur simulateur de vol. Durant les expérimentations, le suivi d'activités et la détection de conflit décrits précédemment ont détecté que deux pilotes avaient eu une gestion de vol conflictuelle qui les avait menés au crash. L'analyse de ces deux situations conflictuelles montre que les pilotes ont subi un phénomène psychologique de persévération : ils ont pris à chaque fois une mauvaise décision pour résoudre leur conflit, décision dans laquelle ils ont persisté à mesure que la pression temporelle augmentait et que la situation se dégradait. Le plus surprenant est qu'ils disposaient des informations nécessaires sur leur interface pour sortir de cette situation.

Exemple : un des pilotes se dirigeait vers le point tournant 5 (au lieu d'aller d'abord se ravitailler en 6), et s'est mis à tourner autour de 5 jusqu'à ce qu'il tombe en panne de carburant.

Lors du debriefing, le pilote a expliqué qu'il était persuadé que le point 5 était une zone de ravitaillement (en dépit de la carte dont il disposait) et alors qu'il l'avait atteint, il a

constaté que le ravitaillement ne s'était pas effectué : pensant l'avoir manqué, il a décidé d'y retourner. Il a reproduit cette manœuvre plusieurs fois, devenant de plus en plus stressé jusqu'à ce qu'il tombe en panne.

5.2 Idée des contre-mesures

Ces premières expériences ont permis de montrer que les pilotes pris dans un conflit ont tendance à persévérer et à s'enfermer dans la résolution du problème au détriment de la surveillance des paramètres vitaux. Ce type de comportement est étudié en neuropsychologie [21, 22] et en psychologie [23] sous le nom de *syndrome de persévération*. Ce syndrome est connu pour concentrer toute l'attention du pilote vers un objectif unique : focalisation excessive des mécanismes attentionels sur un cadran ou focalisation du raisonnement sur une tâche unique. Une fois pris dans cette logique, le pilote mobilise toutes ses ressources pour la réalisation de cet objectif, même s'il est dangereux pour la sécurité. Plus grave encore, l'analyse d'événements aériens a montré que les équipages qui persévèrent sont également insensibles aux alarmes classiques visuelles et auditives. Ce comportement est extrêmement dangereux et le BEA (Bureau Enquête Accidents) lui attribue *quarante pour cent des victimes d'accidents dans l'aéronautique civile* (transport et aviation légère).

Sans nécessiter de définition formelle, le concept de *contre-mesure* consiste en la mise en œuvre d'un mécanisme spécifique d'information du pilote selon une modalité adaptée à la situation et ce, dans le but de le prémunir ou l'aider à faire face à la dégradation de son activité (erreur, persévération...) Dans l'application concrète de pilotage présentée ici, le mécanisme "mesure → contre-mesure" va au-delà d'une alarme « classique ». Il s'agit d'un principe qui est spontanément utilisé par les instructeurs : dans telle phase de vol où la situation de pilotage se dégrade, l'élève pilote se trouve par exemple obnubilé par tel cadran et ne parvient plus à surveiller correctement les autres élements du vol. Dès lors, souvent, l'instructeur détourne l'attention du pilote piégé par son conflit en masquant le cadran responsable de la focalisation et l'amène à reporter son attention sur les indicateurs pertinents. On n'est donc plus dans la mécanique habituelle d'ajout systématique d'information (alarmes) à l'opérateur. Les contre-mesures peuvent ainsi se matérialiser par différentes familles de changements de mode d'interaction (retrait partiel d'information, modification d'un contenu, co-occurrence de tels changements...)

Dans nos travaux actuels, l'idée des contre-mesures repose sur le principe d'un *retrait ciblé de l'information* : le cadran sur lequel se focalise le pilote disparaît puis est remplacé momentanément par un message pertinent pour la sécurité à ce moment de la mission (paramètre à surveiller d'urgence ou solution optimale du conflit). Ainsi le pilote reçoit directement dans son champ visuel l'information pertinente.

5.3 GHOST

Afin de tester ces contre-mesures, l'environnement expérimental GHOST a été développé. Il se compose de :

• Flightgear[2], simulateur de vol qui propose de nombreux modèles d'avions et intègre de manière réaliste l'ensemble des aéroports ainsi que les balises de radionavigation (VOR et NDB) ;

• Atlas[3], logiciel de suivi de route ;

• une interface de magicien d'Oz, que nous avons développée, qui permet à un opérateur humain de déclencher à distance des événements au cours des expérimentations (pannes, dégradations météorologiques et envois de contre-mesures) *via* une connection locale TCP/IP. Le magicien d'Oz dispose de plusieurs options pour envoyer des contre-mesures au pilote (figure 16) :

• *remplacer* le cadran sur lequel se focalise le pilote ;

• *faire clignoter* le cadran ;

• *envoyer un message* à la place du cadran qui a disparu ou qui clignote;

• *envoyer* un message dans l'écran central de l'aéronef pour expliquer la résolution du conflit.

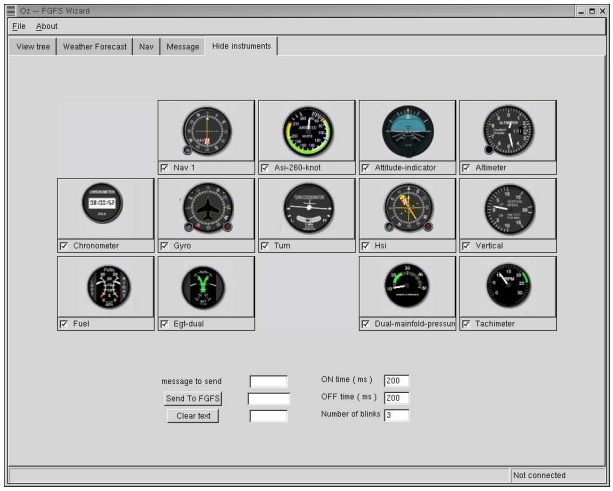

FIG. 16 – *Interface du magicien d'Oz : contre-mesures*

5.4 Expérimentations

Scénarios expérimentaux. Puisqu'un conflit apparaît lorsqu'un but désiré ne peut être atteint, nous avons conçu deux scénarios où un but de la mission ne pouvait être réalisé :

• le scénario 1 consiste en une tâche de navigation depuis l'aéroport de Toulouse-Blagnac jusqu'à l'aéroport de Francazal en passant par trois points tournants (VOR 117.70, NDB 331 et NDB 423). Une zone de brouillard est placée de telle manière que le pilote ne puisse pas voir la piste d'atterrissage sauf au dernier moment, soit trop tard pour

2. http://www.flightgear.org/
3. http://atlas.sf.net

se poser ;
• le scénario 2 consiste en une tâche de navigation depuis l'aéroport de Toulouse-Blagnac jusqu'à l'aéroport de Francazal en passant par trois points tournants (VOR 117.70, NDB 415 et NDB 423). Cette fois-ci une zone de brouillard est située en seuil de piste, de manière que le terrain soit visible de loin, mais à mesure que le pilote s'en approche, la piste disparaît dans le brouillard.
Dans ces scénarios, le conflit du pilote peut se résumer par "*dois-je tenter un atterrissage dangereux à Francazal conformément à mes ordres ou dois-je retourner à Blagnac pour un atterrissage en toute sécurité ?*"

Contre-mesures. Lorsque la visibilité se dégrade (ce qui est le cas dans les deux scénarios), le pilote se focalise sur un cadran nommé H.S.I. (Horizontal Situation Indicator) qui le guide vers sa destination.
Pour sortir le pilote de sa persévération à vouloir se diriger vers l'aéroport, les contre-mesures consistent à :
• enlever le cadran H.S.I. pendant une courte période ;
• afficher deux courts messages l'un après l'autre à la place du H.S.I (*"Atterrissage dangereux"..."Regarder l'écran central"*) ;
• afficher sur l'écran central l'explication du conflit (*"Visibilité dégradée sur Francazal, atterrissage impossible, retourner à la balise NDB 415*).
L'idée développée ici est de choquer les mécanismes attentionnels en faisant disparaître momentanément le H.S.I., d'introduire un conflit cognitif (*"si j'atterris, je peux m'écraser"*) de manière à modifier le raisonnement du pilote et enfin de proposer une solution à partir du cadran central.

5.5 Résultats

21 expérimentations ont été conduites avec GHOST en décembre 2002 avec des pilotes privés et des pilotes militaires. Dans les deux scénarios, les pilotes avaient à faire face à la décision d'abandonner la mission (atterrissage à Francazal) et de retourner à Blagnac. Les scénarios ont été testés avec les conditions expérimentales suivantes : scénarios 1 et 2 sans envoi de contre-mesures et scénarios 1 et 2 avec envoi de contre-mesures.

Sans contre-mesures. 7 pilotes ont testé les scénarios 1 ou 2 sans les contre-mesures (voir tableau suivant). Le terme « Circuits » correspond au nombre de tours de piste autour de Francazal (c'est-à-dire au nombre de fois où le pilote persévère) avant que le pilote ne heurte le sol ou se pose.

Pilote	Scénario	Circuits	Résultats
Pilote1	1	3	crash
Pilote2	1	3	crash
Pilote3	1	1	atterrissage chanceux
Pilote4	1	1	atterrissage chanceux
Pilote5	2	1	crash
Pilote6	2	1	atterrissage chanceux
Pilote7	2	1	crash

Les résultats montrent que sans les contre-mesures, aucun des pilotes n'a pris la décision adéquate (retour à Blagnac), tous ont persévéré. Quatre d'entre eux se sont écrasés, et les trois autres ont réussi un « atterrissage chanceux », c'est-à-dire que pendant qu'ils tournaient autour de Francazal, la piste leur est miraculeusement apparue entre deux bancs de nuages et ils ont réussi à se poser tant bien que mal (souvent en abîmant l'appareil). Lors du debriefing, tous ont admis avoir pris une décision mauvaise et dangereuse.

Avec contre-mesures. 12 pilotes ont testé les scénarios 1 ou 2 avec les contre-mesures (voir tableau suivant). « Circuits » correspond au nombre de circuits réalisés par le pilote autour de Francazal avant l'envoi de contre-mesures par le magicien d'Oz.

Pilote	Scénario	Circuits	Résultats
Pilote8	1	3	crash à Francazal
Pilote9	1	2	retour à Blagnac
Pilote10	1	2	retour à Blagnac
Pilote11	1	2	retour à Blagnac
Pilote12	2	3	retour à Blagnac
Pilote13	2	2	retour à Blagnac
Pilote14	2	2	retour à Blagnac
Pilote15	2	2	retour à Blagnac
Pilote16	1	2	atterrissage à Francazal
Pilote17	1	2	retour à Blagnac
Pilote18	1	2	retour à Blagnac
Pilote19	1	5	crash à Francazal

Les résultats montrent l'efficacité des contre-mesures pour sortir les pilotes de leur persévération puisque 9 sur 12 ont changé leur décision pour atterrir en toute sécurité à Blagnac. Les pilotes, lors du debriefing, ont également confirmé que c'étaient bien les contre-mesures qui les avaient fait changer d'avis. De plus la disparition momentanée du H.S.I. ne leur avait pas causé de stress.
Les résultats du pilote19 suggèrent que plus un pilote persévère depuis longtemps, plus il est difficile de l'en sortir : ce pilote a déclaré qu'il était tellement stressé après ces 5 tours qu'il n'avait même pas noté les contre-mesures.
Les pilotes 8 et 16 ont également persévéré en dépit des contre-mesures : le pilote 16 a déclaré qu'il savait que son comportement était erroné mais qu'il voulait s'amuser avec le simulateur, en revanche nous n'avons pas d'explication pour le comportement du pilote 8.

6 Conclusion et perspectives

Dans cet article nous présentons une approche de *suivi, prédiction et assistance* à l'opérateur et son application directe à l'aéronautique. L'idée forte est de concentrer la représentation de l'activité de l'opérateur (en l'occurrence le pilote) sur les composantes qui sont déterminantes pour des situations critiques pour la sécurité des vols : ici c'est le mécanisme très grave de *persévération* qui est traité.

En ce qui concerne *Kalmansymbo-aéro* et la détection des conflits, les travaux à venir vont consister à affiner les

mécanismes et intégrer de nouveaux éléments :
• prise en compte de tolérances pour la mise en correspondance, en fonction des phases de vol ;
• prise en compte en prédiction de boucles dont le nombre de parcours est limité à n (introduction de jetons compteurs) ;
• introduction du temps dans les conditions cruciales ;
• amélioration du suivi de plusieurs situations possibles en parallèle : introduction d'une relation de préférence sur les mises en correspondances partielles et les prédictions qui en découlent pour un suivi particulier des situations potentiellement les plus pertinentes.
En ce qui concerne GHOST, un enjeu est maintenant de ≪ fermer la boucle ≫ et de remplacer le magicien d'Oz par une élaboration et un envoi automatiques pertinents des contre-mesures en fonction des situations estimées par *Kalmansymbo-aéro* et des conflits détectés. La première étape consiste à construire, à partir des résultats expérimentaux, une ≪ expertise ≫ des contre-mesures.

Il est actuellement prévu de porter le dispositif expérimental GHOST sur simulateur mobile d'A320, de façon à envisager des expériences :
• en situation plus réaliste pour les pilotes ;
• avec des équipages et non plus avec un seul pilote ;
• pour étudier l'interaction entre pilote automatique et équipage.

Remerciements
Les auteurs remercient Olivier Grisel et Fabrice Schwach pour leur travail remarquable dans la mise au point du dispositif expérimental GHOST.

Références

[1] F. Dehais and L. Chaudron. Conflict modelling in combat flight activity. In *COOP'2000*, Sophia Antipolis, May 2000.

[2] K. Sycara. Multiagent compromise via negotiation. In Les Gasser and Michael N. Huhns, editors, *Distributed Artificial Intelligence*, volume 2, pages 119–137. Pitman Publishing, 1989.

[3] M. Adler, B. Alvah, R. Weihmayer, and R. Worrest. Conflict-resolution strategies for nonhierarchical distributed agents. In Les Gasser and Michael N. Huhns, editors, *Distributed Artificial Intelligence*, volume 2, pages 139–161. Pitman Publishing, 1989.

[4] J. S. Rosenschein and G. Zlotkin. *Rules of encounter. Designing convention for automated negotiation among computers*. Artificial Intelligence. MIT Press, 1994.

[5] H. Jung and M. Tambe. Conflicts in agent teams. In C. Tessier, L. Chaudron, and H.-J. Müller, editors, *Conflicting agents - Conflict management in multi-agent systems*, Multiagent systems, artificial societies and simulated organizations 1, pages 153–167. Kluwer Academic Publishers, 2000.

[6] S.E. Conry, K. Kuwabara, V.R. Lesser, and R.A. Meyer. Multistage negotiation for distributed constraint satisfaction. *IEEE Transactions on SMC*, 21(6):1462–1477, 1991.

[7] T. Khedro and M. Genesereth. Modeling multiagent cooperation as distributed constraint satisfaction problem solving. In *11th ECAI*, pages 249–253, 1994.

[8] M. Hannebauer. Their problems are my problems, the transition between internal and external conflicts. In C. Tessier, L. Chaudron, and H.-J. Müller, editors, *Conflicting agents - Conflict management in multi-agent systems*, Multiagent systems, artificial societies and simulated organizations 1, pages 63–109. Kluwer Academic Publishers, 2000.

[9] S. M. Easterbrook. Handling conflict between domain descriptions with computer supported negotiation. *Knowledge Acquisition: An International Journal*, 3(4):255–289, 1991.

[10] K. Lewin, R. Lippitt, and R. K. White. Patterns of aggression behavior in experimentally created "social climates". *Journal of Social Psychology*, 10:271–299, 1939.

[11] C. Castelfranchi. Conflict ontology. In H.-J. Müller and R. Dieng, editors, *Computational conflicts - Conflict modelling for distributed intelligent systems*, pages 21–40. Springer Verlag, 2000.

[12] F. Dehais. Modelling cognitive conflict in pilot's activity. In *STAIRS 2002: Starting Artificial Intelligence Researchers Symposium*, pages 45–54, July 2002.

[13] F. Dehais, C. Tessier, and L. Chaudron. Ghost: experimenting countermeasures for conflicts in the pilot's activity. In *IJCAI03 Conference*, Acapulco, Mexico, August 2003.

[14] C. Tessier. Towards a commonsense estimator for activity tracking. In *AAAI Spring symposium*, Stanford University, Palo Alto CA, USA, March 2003.

[15] M. Ghallab. On chronicles: representation, on-line recognition and learning. In L.C. Aiello, J. Doyle, and S. Shapiro, editors, *Proceedings of the Fifth National Conference on Artificial Intelligence*, San Francisco, California, 1996.

[16] L. Chaudron, C. Cossart, N. Maille, and C. Tessier. A purely symbolic model for dynamic scene interpretation. *International Journal on Artificial Intelligence Tools*, 6(4):635–664, 1997.

[17] N. Rota and M. Thonnat. Activity recognition from video sequences using declarative models. In *ECAI'00*, pages 673–677, Berlin, 2000.

[18] Robert Valette. *Les Réseaux de Petri*. LAAS, 2000. Support de cours ENAC.

[19] L. Chaudron, N. Maille, and M. Boyer. The CUBE lattice model and its applications. *Applied Artificial Intelligence*, 3(17), March 2003.

[20] N. Maille. *Un modèle logico-algébrique pour la fusion symbolique et l'analyse formelle*. PhD thesis, Supaéro, nov. 1999. `ftp://www.cert.fr/fr/dcsd/PSEVPUB/thnm.gz`.

[21] B. Vand Der Kolk. The body keeps the score: memory and the evolving psychobiology of post traumatic stress disorder. *Harvard Review of Psychiatry*, 1:253–265, 1994.

[22] J. Pastor. Cognitive performance modeling - Assessing reasoning with the EARTH methodology. In *COOP'2000*, Sophia Antipolis, May 2000.

[23] J.L. Beauvois and R.V. Joule. A radical point of view on dissonance theory. In Eddie Harmon-Jones and Judson Mills, editors, *Cognitive Dissonance : Progress on a Pivotal Theory in Social Psychology*. 1999.