

INPT

DEA Programmation et Systèmes

année 2003/2004

Institut de Recherche en Informatique de Toulouse

# Logique déontique pour la disponibilité

Julien Brunel

directeurs de recherche : Jean-Paul Bodeveix et Mamoun Filali  
projet DISPO, ACI Sécurité Informatique

Résumé : En informatique, le concept de disponibilité met en jeu des notions qui permettent de définir les droits d'accès aux ressources de chaque utilisateur, ainsi que la manière dont le système doit allouer dans le temps les ressources, en fonction des droits d'accès de chacun. Lors d'une analyse de disponibilité (vérification de la satisfaction de propriétés de disponibilité, conformité d'un système vis à vis d'une politique) il faut pouvoir exprimer le fait qu'une formule soit toujours vraie, ou seulement pendant une certaine durée, ou encore qu'elle soit vérifiée à partir d'une certaine date. La logique déontique et la logique temporelle apparaissent naturellement comme des outils adéquats. Notre travail consiste à étudier les différentes logiques faisant intervenir temps et obligation, à trouver un formalisme adapté au concept de disponibilité, et vérifier la disponibilité d'un système.

# Table des matières

<b>1</b>	<b>Concept de disponibilité</b>	<b>4</b>
1.1	Politique et propriété de disponibilité . . . . .	4
1.2	Exemple Simple . . . . .	4
1.3	Problématique . . . . .	5
1.4	Notions à exprimer . . . . .	5
<b>2</b>	<b>Logiques déontiques</b>	<b>7</b>
2.1	Le système “standard” de logique déontique SDL . . . . .	7
2.1.1	Le langage . . . . .	7
2.1.2	Les axiomes . . . . .	7
2.1.3	La sémantique . . . . .	7
2.1.4	Quelques théorèmes . . . . .	8
2.2	Logique déontique ramenée à la logique modale . . . . .	9
2.2.1	Le langage $PDeL^{AM}$ . . . . .	9
2.2.2	Les axiomes de $PDeL^{AM}$ . . . . .	10
2.2.3	La sémantique de $PDeL^{AM}$ . . . . .	10
2.2.4	Quelques théorèmes . . . . .	11
2.3	Aqvist . . . . .	11
2.3.1	Le langage . . . . .	11
2.3.2	Les axiomes . . . . .	12
2.3.3	La sémantique . . . . .	12
<b>3</b>	<b>Logiques avec notion de temps</b>	<b>14</b>
3.1	Logiques temporelles . . . . .	14
3.1.1	Logique temporelle linéaire . . . . .	14
3.1.2	Logiques temporelles arborescentes . . . . .	16
3.2	Logiques temporisées . . . . .	17
3.2.1	Temps discret . . . . .	17
3.2.2	Temps continu . . . . .	18
<b>4</b>	<b>Combinaison déontique/temps</b>	<b>22</b>
4.1	RS5-DS5 . . . . .	22
4.1.1	La syntaxe . . . . .	22
4.1.2	La sémantique . . . . .	22
4.1.3	Sémantique dans $CTL$ . . . . .	23
4.2	Le système $DTL$ de Van Eck . . . . .	25
4.3	Échéances . . . . .	26
4.3.1	Échéance avec la logique d’action . . . . .	26
4.3.2	Utilisation de $CTL$ . . . . .	28

<b>5</b>	<b>Propositions</b>	<b>30</b>
5.1	Relation entre les différents formalismes présentés . . . . .	30
5.2	Modélisation avec une proposition particulière (marqueur) . . . . .	31
5.3	Modélisation avec une relation déontique . . . . .	31
5.4	Analogie entre la relation déontique et une relation de raffinement . . . . .	33
	5.4.1 Dans <i>SDL</i> . . . . .	33
	5.4.2 Dans un modèle de logique temporelle . . . . .	34
5.5	Quelques propriétés des opérateurs d'obligation avec échéance explicite . . . . .	37
5.6	Expression de politiques et propriétés de disponibilité . . . . .	38
5.7	Vérification de la disponibilité . . . . .	39
	5.7.1 Conformité d'un système vis à vis d'une politique . . . . .	39
	5.7.2 Cohérence interne d'une politique . . . . .	40
	5.7.3 Propriété de disponibilité . . . . .	41

# Introduction

La disponibilité est un concept très utile en sécurité informatique. Il permet en effet de gérer des contraintes de temps sur l'accès aux ressources. Saurel et Cupens en propose une modélisation en logique des prédicats dans [22]. L'expressivité de la logique déontique et de la logique temporelle semble adaptée à la disponibilité. Nous étudierons donc comment adapter les formalismes de logique temporelle et de logique déontique pour la disponibilité. Dans le premier chapitre nous parlerons plus en détail de la disponibilité.

**Logique déontique** La logique déontique permet d'exprimer la notion d'obligation (la permission et l'interdiction en dérivent). Elle est l'objet du deuxième chapitre. Le premier à formaliser ceci dans un système logique est Von Wright en 1951 dans [24]. De nombreux autres systèmes verront ensuite le jour, essayant de pallier les lacunes des précédents systèmes, et notamment leurs paradoxes ([25, 16, 12, 5, 19]) ou de donner plus d'expressivité ([7, 6, 18, 11, 10, 9]). Certains de ces derniers mêlent le temps à la notion d'obligation, mais nous verrons qu'ils ne permettent pas d'exprimer les notions essentielles, et notamment l'obligation avec échéance explicite.

**Logiques temporelle et temporisée** Les logiques temporelles ( $CTL, LTL$ ) [8] sont très utilisées pour la spécification des systèmes réactifs en général. Elles utilisent des modèles simples et disposent d'algorithmes de vérification de modèle efficaces. On peut exprimer des propriétés assez complexes, comme "si une alarme se déclenche, une réaction aura lieu" ( $AG(alar\grave{m}e \Rightarrow AF\ reaction)$  en  $CTL$ ). Une propriété classique de disponibilité rentre dans ce cadre : "Si un processus accède à une ressource, alors il la libérera nécessairement" ( $AG(accede \Rightarrow AF\ libere)$ ). On remarquera que les versions arborescentes, comme  $CTL$  permettent d'exprimer la notion de possibilité, exprimée dans la sémantique par "il existe un chemin". Ceci est impossible dans les versions linéaires ( $LTL$ ). Par contre,  $CTL$  perd la notion d'équité : il s'agit, pour un système de transitions donné, d'exprimer "si une transition est infiniment souvent tirable, alors elle est infiniment souvent tirée" (soit, en  $LTL$ ,  $\Box\Diamond enable(\tau) \Rightarrow \Box\Diamond\tau$ ).

Il existe des extensions de certaines logiques temporelles qui permettent d'exprimer le temps de manière explicite. On parle alors de logique temporisée. Il est ainsi possible d'exprimer que "Si un processus accède à une ressource, alors il la libérera avant 5s". Nous travaillerons sur des extensions de  $CTL$  (avec temps discret, comme  $RTCTL$ , ou continu, comme  $TCTL$ ). La propriété précédente s'écrit dans ces logiques :  $AG(accede \Rightarrow AF_{\leq 5s} libere)$ .

Notre travail consiste à construire un formalisme, basé sur la logique déontique et la logique temporelle, dans lequel on peut exprimer les notions intervenant dans la disponibilité, puis à étudier la vérification de la disponibilité d'un système.

# Chapitre 1

## Concept de disponibilité

Nous présentons dans ce chapitre les bases informelles de la disponibilité, permettant de comprendre le type de notion sur lesquelles nous cherchons à travailler. nous nous baserons essentiellement sur les définitions présentées par Saurel et Cuppens dans [22].

### 1.1 Politique et propriété de disponibilité

La disponibilité des systèmes constitue un des concepts clé de la sécurité informatique. Il s'agit de la manière dont le système va gérer les demandes de réalisation de tâches compte tenu de contraintes de temps et de performance.

Une étude a été faite dans [22]. Elle définit la disponibilité, pose la problématique, et propose un formalisme, dans la logique des prédicats, pour traiter ces questions. A la fin de l'étude, une maquette, codée en Prolog, donne des exemples d'analyses de disponibilité pour des spécifications de systèmes, de politiques, et de propriétés. Nous nous basons sur cette étude pour connaître les notions que nous essaierons d'exprimer dans un formalisme plus adapté.

On voit apparaître deux concepts essentiels :

- une politique de disponibilité constitue la spécification des droits, obligations, ou interdictions des différents processus utilisateurs, en terme de ressources pour accomplir des tâches.
- une propriété de disponibilité est une propriété quelconque concernant la disponibilité, que l'on exprime par exemple parce que l'on veut que le système étudié la satisfasse.

### 1.2 Exemple Simple

Voici un exemple très simple qui illustre ces deux notions. On dispose de deux processus  $p_1$  et  $p_2$ , d'une ressource  $r$ .

Un exemple de politique de disponibilité serait :  $p_1$  a le droit d'utiliser la ressource pendant deux minutes d'affilée. Il doit pouvoir accéder à la ressource dix minutes après l'avoir demandée, au plus tard.

$p_2$  a le droit d'utiliser la ressource pendant une minute, et doit pouvoir y accéder au plus tard cinq minutes après l'avoir demandée.

Un exemple de propriété de disponibilité serait : toute demande d'accès à la ressource, de la part d'un processus autorisé, est satisfaite avant dix minutes.

Dans cet exemple, un système qui reste conforme à la politique de disponibilité définie ci-dessus vérifie la propriété.

**Remarque 1** *En fait, dans le modèle de [22], les processus utilisateurs demandent à effectuer des tâches. Celles-ci ont une certaine durée, et nécessitent certaines ressources, mais ceci est transparent du point de vue de l'utilisateur. C'est le système (l'allocateur de ressources) qui décidera qu'un processus pourra accéder à une ressource et ainsi commencer la tâche qu'il avait demandée.*

## 1.3 Problématique

Plus généralement, plusieurs questions peuvent se poser :

- conformité d'un système vis à vis d'une politique de disponibilité :  
Le système (allocateur de ressource dans notre cas) viole-t-il les obligations imposées par la politique ?
- cohérence d'une politique de disponibilité :  
Existe-t-il des contradictions dans les règles qui définissent la politique de disponibilité ?
- politique et propriété de disponibilité :  
Une politique de disponibilité donnée permet-elle de satisfaire une propriété de disponibilité souhaitée ?

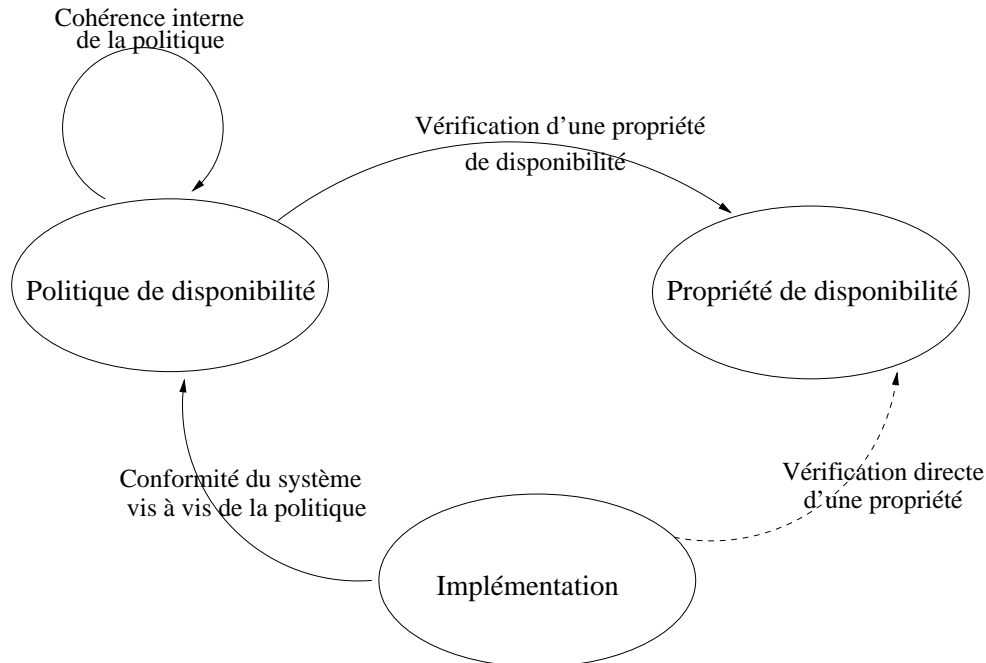


FIG. 1.1 – Problématique

Notre travail se situe dans ce contexte, mais concerne en grande partie le formalisme à utiliser pour réaliser ce genre d'études.

## 1.4 Notions à exprimer

Les notions qui apparaissent dans [22] sont modélisées par des prédicats parfois assez complexes, admettant plusieurs arguments. Pour exprimer une durée, une permission, il faut introduire un nouveau prédicat. Si l'on dispose des notions temporelles et déontiques au sein du formalisme utilisé, les prédicats utilisés dans [22] deviennent fortement redondants.

Dans ce modèle, les processus demandent à réaliser des tâches (prédicat *demande\_realisation*), ayant chacune une certaine durée, et nécessitant certaines ressources. L'allocateur de ressources est chargé de fournir la ressource nécessaire à un processus pour qu'il puisse réaliser la tâche voulue.

Voici les prédicats nécessaires à la définition d'une politique de disponibilité :

- *droit\_utilisation*( $p, r, d$ ) : le processus  $p$  a le droit d'utiliser la ressource  $r$  pendant  $d$  unités de temps d'affilée, au maximum.
- *droit\_disposer*( $p, r, d$ ) : le processus  $p$  doit pouvoir accéder à la ressource  $d$  unités de temps après l'avoir demandée, au plus tard.
- *droit\_realisation*( $s, t, d$ ) : le processus  $p$  a le droit de réaliser la tâche  $t$  dans un délai maximum de  $d$  unités de temps après en avoir fait la demande.

- *obligation\_dure\_max*( $p, t, d$ ) : lorsque  $p$  réalise  $t$ , cette réalisation doit durer moins de  $d$  unités de temps.

D'un point de vue logique, on a besoin de la notion de temps, et de l'obligation (la permission et l'interdiction en dérivent). Plus précisément, il faut exprimer l'obligation avec échéance, c'est à dire l'obligation de faire quelque chose avant une certaine date.

On pourrait alors diminuer sensiblement le nombre de prédicats, puisqu'il ne serait plus nécessaire d'ajouter un nouveau prédicat pour chaque notion.

Par exemple, *droit\_utiliser* peut être vu comme une permission d'utiliser, d'une part, et l'obligation de libérer la ressource au bout d'un certain délai (ou échéance) d'autre part.

On pourrait alors exprimer l'essentiel avec les prédicats de base *utilise*( $p, r$ ) ("le processus  $p$  utilise la ressource  $r$ "), *realise*( $p, t$ ) ("le processus  $p$  réalise la tâche  $t$ "), et *demande\_realisation*( $p, t$ ) ("le processus  $p$  demande à réaliser la tâche  $t$ ").

# Chapitre 2

## Logiques déontiques

Il ne s'agit pas ici d'un compte rendu exhaustif de l'état de l'art. Nous montrons les diverses représentations de l'obligation, à travers quelques uns des principaux systèmes logiques. Nous ciblerons plus tard de plus en plus les formalismes par rapport aux attentes dues à la disponibilité. Ainsi, après la présentation des trois principales idées pour représenter l'obligation dans ce chapitre, nous nous intéressons à l'ajout du temps, puis à l'expression de l'obligation avec échéance (cf chapitres 3 , 4).

### 2.1 Le système “standard” de logique déontique SDL

Un des premiers systèmes logiques à tenter de capturer la notion d'obligation dans un formalisme est le système de Von Wright paru en 1951 [24]. Sa version plus récente avec modèle de Kripke est appelé *SDL* (*Standard Deontic Logic*). On ne dispose que d'une modalité, l'obligation. (L'interdiction et la permission ne sont que des raccourcis basés sur l'obligation.)

#### 2.1.1 Le langage

On dispose d'un ensemble de propositions atomiques *Prop*, des opérateurs usuels de la logique propositionnelle, ainsi que de l'opérateur déontique *O*.

#### 2.1.2 Les axiomes

- Tous les axiomes de la logique propositionnelle
- $O(A \Rightarrow B) \Rightarrow (O(A) \Rightarrow O(B))$  (axiome K)
- $O(A) \Rightarrow P(A)$  (axiome D)
- $\frac{A, A \Rightarrow B}{B}$  (Modus Ponens)
- $\frac{A}{O(A)}$  (O-nécessitation)

Les opérateurs de permission (*P*) et d'interdiction (*I*) sont définis à partir de l'obligation (ces définitions sont valables pour la majorité des systèmes de logique déontique) :

$$I(A) \stackrel{def}{=} O(\neg A)$$

$$P(A) \stackrel{def}{=} \neg I(A)$$

#### 2.1.3 La sémantique

La sémantique de *SDL* est basée sur un ensemble de mondes (ou états) possibles *S*, une fonction  $\Pi$  attribuant à chaque état les propositions qu'il satisfait, et une relation *R* qui relie chaque monde avec un ensemble de mondes (ses alternatives “parfaites”). Les propositions qui sont obligatoires dans un monde sont vérifiées dans toutes ses alternatives “parfaites”. Un modèle est donc un triplet  $\mathcal{M} = (S, \Pi, R)$ .

#### Définition 1 (Sémantique de *SDL*)

Étant donné un tel modèle  $\mathcal{M}$  et un état  $s \in S$ , la sémantique des opérateurs de *SDL* est donnée par les règles suivantes :

- $\mathcal{M} \models_s p$  ssi  $p \in \Pi(s)$
- $\mathcal{M} \models_s \neg A$  ssi  $\mathcal{M} \not\models_s A$
- $\mathcal{M} \models_s A \wedge B$  ssi  $\mathcal{M} \models_s A$  et  $\mathcal{M} \models_s B$
- $\mathcal{M} \models_s O(A)$  ssi  $\forall t \in S (sRt \Rightarrow \mathcal{M} \models_t A)$

**Définition 2 (Satisfaction d'une formule)**

Un modèle  $\mathcal{M}$  satisfait une formule si tout état de  $\mathcal{M}$  la satisfait. Plus formellement :

$$\mathcal{M} \models A \text{ ssi } \forall s \in S \mathcal{M} \models_s A$$

Une formule est dite valide si elle est satisfaite par tous les modèles.

**Propriété 1** Si la relation  $R$  est totale, c'est à dire si

$$\forall w \exists w' \text{ tel que } wRw'$$

alors l'ensemble des règles d'inférence de *SDL* (muni de la sémantique et des axiomes vus plus haut) est complet et sain.

**Remarque 2** Pour qu'un modèle satisfasse  $p$ , il faut, par définition, que tous les états satisfassent  $p$ . En particulier, tous les "bons" états satisfont alors  $p$ . Le modèle vérifie donc aussi  $O(p)$ .

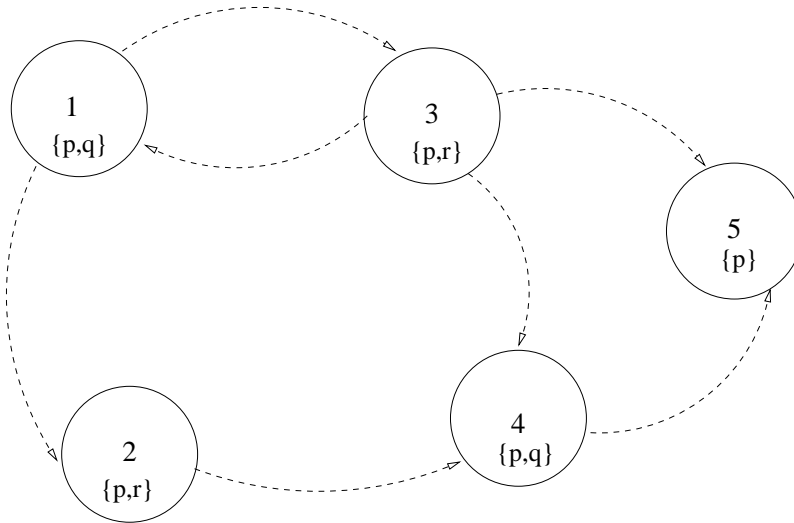


FIG. 2.1 – Modèle de *SDL*

La figure 2.1 représente un exemple simple de modèle de *SDL*. On note sur chaque état l'ensemble des propositions atomiques qu'il satisfait. L'ensemble des "bons" états associés à un état est repéré grâce aux flèches en pointillés. Dans cet exemple, l'état 1 se voit ainsi associés les états 2 et 3 comme "bonnes" alternatives. L'état 1 satisfait donc  $p$ ,  $q$ , et aussi  $O(p)$  et  $O(r)$ . Le modèle satisfait  $p$  - puisque tous les états le satisfont - et donc aussi  $O(p)$ .

**2.1.4 Quelques théorèmes**

Voici plusieurs théorèmes dérivables dans *SDL* (voir [21] par exemple pour un traitement détaillé) :

- $O(A \wedge B) \Leftrightarrow O(A) \wedge O(B)$
- $P(A \vee B) \Leftrightarrow P(A) \vee P(B)$
- $O(A) \vee O(B) \Rightarrow O(A \vee B)$

- $\neg(A \wedge O(\neg A))$
- $O(A) \Rightarrow O(A \vee B)$  (Paradoxe de Ross)
- $P(A \wedge B) \Rightarrow P(A) \wedge P(B)$
- $F(A) \Rightarrow F(A \wedge B)$  (Paradoxe du pénitent)
- $\neg A \Rightarrow (A \Rightarrow O(B))$  (Paradoxe de l'obligation conditionnelle)
- $\frac{A \Rightarrow B}{O(A) \rightarrow O(B)}$  (Paradoxe du bon samaritain)

On voit que plusieurs de ces théorèmes sont contre-intuitifs. (Déterminer ceux qui correspondent à l'intuition n'est pas toujours chose aisée étant donné l'ambiguïté de la notion déontique dans le langage naturel.) Cela vient essentiellement du lien important entre l'obligation et les faits, induit par le système d'axiomes - et les règles de la sémantique. En effet, la règle de  $O$ -nécessitation impose que si  $p$  est un théorème, alors  $O(p)$  aussi. Autrement dit, tout ce qui est à l'obligation d'être. Ceci, qui est un choix arbitraire sur la notion d'obligation, interdit la présence de formules valides qui ne seraient pas obligatoires.

L'axiome  $D$  implique quant à lui l'absence d'interdictions contradictoires : on ne peut obliger  $p$  et  $\neg p$  à la fois. Combiné à la règle de  $O$ -nécessitation, il interdit d'avoir une formule  $A$  comme théorème, en même temps que  $O(\neg A)$ . Au niveau sémantique, aucun modèle ne peut satisfaire  $A$  et  $O(\neg A)$  à la fois, mais un état le peut. Pour raisonner sur la violation d'une obligation, il faut donc se situer au niveau des états d'un modèle et pas sur le modèle entier.

On pourrait imaginer plus d'indépendance pour l'obligation. On suivrait l'idée qu'on se place implicitement à un niveau différent lorsqu'on parle d'obligation qu'une formule soit établie, et lorsqu'on parle de la formule elle-même (sans opérateur déontique). En effet, l'obligation d'avoir certaines formules devrait alors être fixée indépendamment des formules elles-mêmes. Mais une indépendance totale reviendrait à raisonner dans deux modèles différents, pour les obligations d'une part, et les faits d'autre part. Pour éviter les paradoxes précédents, tout en gardant un même système pour les obligations et les faits, il faudrait donc changer radicalement les axiomes, et la sémantique...

Nous allons parcourir les propositions qui nous paraissent les plus intéressantes pour représenter la notion déontique, et remarquerons (cf 5.1) qu'aucun changement radical n'a été proposé par rapport à *SDL*.

## 2.2 Logique déontique ramenée à la logique modale

De nombreux systèmes ont ensuite introduit une proposition atomique particulière ( $V$ ) qui est vraie lorsqu'une obligation est violée. Il est possible de définir  $O$  à partir de cette proposition, dans un système connu, comme par exemple la logique modale avec les opérateurs aléthiques  $\square$  et  $\diamond$ .

### 2.2.1 Le langage $PDeL^{AM}$

Dans [9], Altan, Meyer, et Wieringa proposent un formalisme qui intègre les notions d'obligation de faire, et d'obligation d'être. Il est basé d'une part sur la réduction, proposée par Anderson dans [4], de l'obligation d'être à la logique modale aléthique, et d'autre part, sur l'utilisation de la logique dynamique pour exprimer l'obligation de faire, proposée par Meyer dans [19].

L'expression de l'obligation passe ici par l'introduction d'une proposition atomique particulière, notée  $V$ , indiquant qu'une obligation est violée. La définition suivante exprime alors l'obligation d'être à l'aide de l'opérateur modal classique  $\square$  :

$$O(A) \stackrel{def}{=} \square(\neg A \Rightarrow V)$$

Meyer exprime quant à lui l'obligation de faire dans une logique d'action de la manière suivante :

$$\hat{O}(\alpha) \stackrel{def}{=} [\bar{\alpha}]V$$

On dispose d'un ensemble d'actions élémentaires dans le langage, ainsi que d'opérations de composition sur les actions ( $+$ ,  $\&$ ,  $\bar{\cdot}$ ). Le concept de négation d'une action, noté  $\bar{\alpha}$ , est traité en détail dans [20]. Intuitivement, cela correspond à toutes les actions au sein desquelles  $\alpha$  n'intervient pas. L'opérateur  $[\cdot]$  indique que la formule qui le suit est nécessairement satisfaite après l'exécution de l'action située entre les  $[\cdot]$ . Par exemple,  $[\alpha]A$  exprime que  $A$  est forcément satisfaite après l'exécution de  $\alpha$ .

Le langage de  $PDeL^{AM}$  comprend donc un ensemble de propositions atomiques  $Prop$ , les opérateurs usuels de la logique propositionnelles, un ensemble d'actions de bases  $A$  et l'opérateur usuel  $[.]$  sur les actions, et enfin les opérateurs déontiques d'état et d'action cités précédemment.

## 2.2.2 Les axiomes de $PDeL^{AM}$

- Les axiomes de la logique propositionnelle
- $\Box(A \Rightarrow B) \Rightarrow (\Box A \Rightarrow \Box B)$  (K)
- $\Box A \Rightarrow A$  (T)
- $\neg \Box A \Rightarrow \Box \neg \Box A$  (5)
- $\Diamond \neg V$  (D)
- $[\alpha](A \Rightarrow B) \Rightarrow ([\alpha]A \Rightarrow [\alpha]B)$  (AK)
- $OA \Leftrightarrow \Box(\neg A \Rightarrow V)$  (O)
- $\hat{O}\alpha \Leftrightarrow [\bar{\alpha}]V$  (\hat{O})
- $\Box A \Rightarrow [\alpha]A$  (\Box[\alpha])
- $\neg[\alpha]\perp \wedge [\alpha]\Box A \Rightarrow \Box A$  ([\alpha]\Box)

$\Diamond A$  est utilisé comme raccourci pour  $\neg \Box \neg A$ .

On définit de plus les opérateurs  $F, \hat{F}$  (interdiction d'être et de faire), et  $P, \hat{P}$  (permission d'être et de faire) de la manière suivante :

$$\begin{aligned} F(A) &\stackrel{def}{=} O(\neg A) \\ \hat{F}(\alpha) &\stackrel{def}{=} \hat{O}(\bar{\alpha}) \\ P(A) &\stackrel{def}{=} \neg F(A) \\ \hat{P}(\alpha) &\stackrel{def}{=} \neg \hat{F}(\alpha) \end{aligned}$$

On dispose des trois règles d'inférence suivantes :

$$\begin{aligned} &\frac{A, A \Rightarrow B}{B} \text{ (Modus Ponens)} \\ &\frac{A}{\Box A} \text{ (\Box-Nécessitation)} \\ &\frac{A}{[\alpha]A} \end{aligned}$$

## 2.2.3 La sémantique de $PDeL^{AM}$

Un modèle de  $PDeL^{AM}$  est donné par  $\mathcal{M} = (A, W, \Pi, \|\cdot\|, \models, opt)$ , où :

- $A$  est un ensemble d'actions de base, tel que si  $\alpha \in A$ , alors  $\bar{\alpha} \in A$
- $W$  est l'ensemble des mondes possibles
- $\Pi$  est la fonction qui indique, pour chaque monde, les propositions atomiques qui sont vérifiées
- $\|\cdot\| : A \times W \rightarrow \mathcal{P}(W)$  est une fonction qui associe à une action  $\alpha$  et un monde  $w$ , l'ensemble des mondes auxquels l'exécution de  $\alpha$  peut mener à partir de  $w$
- $\models$  est la relation usuelle de satisfaction d'une formule par un monde
- $opt$  représente l'ensemble des "bons" mondes de  $W$ , c'est à dire ceux qui ne vérifient pas  $V$ .

### Définition 3 (Sémantique de $PDeL^{AM}$ )

Étant donné un monde  $w$  d'un modèle  $\mathcal{M}$ , les règles sont les suivantes :

- $\mathcal{M} \models_w p$  (pour  $p \in prop$ ) ssi  $p \in \Pi(w)$
- $\mathcal{M} \models_w \neg A$  ssi  $\mathcal{M} \not\models_w A$
- $\mathcal{M} \models_w A \wedge B$  ssi  $\mathcal{M} \models_w A$  et  $\mathcal{M} \models_w B$
- $\mathcal{M} \models_w [\alpha]A$  ssi  $\forall w' (w' \in \|\alpha\|(w) \Rightarrow \mathcal{M} \models_{w'} A)$
- $\mathcal{M} \models_w \langle \alpha \rangle A$  ssi  $\exists w' (w' \in \|\alpha\|(w) \ \& \ \mathcal{M} \models_{w'} A)$

- $\mathcal{M} \models_w V$  ssi  $w \notin opt$
- $\mathcal{M} \models_w \Box A$  ssi  $\forall w' (w' \in W \Rightarrow \mathcal{M} \models_{w'} A)$

#### Définition 4 (Satisfaction d'une formule)

Un modèle satisfait une formule si tous ses mondes la satisfont. Une formule est valide si elle est satisfaite par tout modèle.

**Propriété 2** L'ensemble de règles de déduction énoncé ci-dessus est sain et complet pour  $PDeL$  muni des axiomes et de la sémantique définis ci-dessus.

### 2.2.4 Quelques théorèmes

Voici des théorèmes de  $PDeL^{AM}$ . On donne la version état et la version action lorsque c'est possible.

	état	action
1		$\vdash \hat{O}(\alpha_1; \alpha_2) \Leftrightarrow \hat{O}(\alpha_1) \wedge [\alpha_1]\hat{O}(\alpha_2)$
2	$\vdash O(A) \vee O(B) \Rightarrow O(A \vee B)$	$\vdash \hat{O}(\alpha_1) \vee \hat{O}(\alpha_2) \Rightarrow \hat{O}(\alpha_1 + \alpha_2)$
3	$\vdash O(A) \wedge O(B) \Leftrightarrow O(A \wedge B)$	$\vdash \hat{O}(\alpha_1) \wedge \hat{O}(\alpha_2) \Leftrightarrow \hat{O}(\alpha_1 \& \alpha_2)$
4	$A \Rightarrow B \vdash O(A) \Rightarrow O(B)$	
5	$\vdash \Box A \Rightarrow O(A)$	
6	$\vdash O(A) \Rightarrow \Box O(A)$	$\not\vdash \hat{O}(\alpha) \Rightarrow \Box \hat{O}(A)$
7	$\vdash O(A) \Rightarrow P(A)$	$\not\vdash \hat{O}(\alpha) \Rightarrow \hat{P}(\alpha)$
8	$\vdash \neg(O(A) \wedge O(\neg A))$	$\not\vdash \neg(\hat{O}(\alpha) \wedge \hat{O}(\bar{\alpha}))$

Ces théorèmes sont intéressants. En effet, on remarque que certains des théorèmes de  $SDL$  qui pouvaient poser problème, comme les 7 et 8, qui expriment tous deux qu'on ne peut pas avoir d'obligations contradictoires, ne sont plus vérifiés dans leur version action. Cela permet donc d'exprimer des notions qui ne peuvent pas l'être dans la plus part des systèmes.

On note aussi que la définition de  $O$  en fonction de  $\Box$  entraîne une grande dépendance, et notamment que  $\Box A \Rightarrow O(A)$ . Ceci est toutefois cohérent avec la règle de nécessité de  $SDL$ ,  $\frac{A}{O(A)}$  qui exprime "on ne fait que ce qu'on est obligé de faire", ou "n'existe que ce qui doit exister".

Le théorème 6, par contre, indique qu'une obligation doit forcément rester vraie. Mais encore une fois, la version action n'est pas un théorème.

La discordance entre les deux versions de chaque théorème peut s'interpréter comme une opportunité d'exprimer des notions qui n'étaient pas accessibles avec la seule version état, ou comme un manque de cohérence entre les deux visions.

## 2.3 Aqvist

### 2.3.1 Le langage

Le système de Aqvist,  $G$ , est basé sur la logique déontique dyadique (voir [25]), qui introduit l'opérateur d'obligation avec deux arguments,  $A$  et  $B$ , noté  $O(A/B)$ . Cette notation signifie que  $A$  a l'obligation d'être satisfaite si  $B$  est vérifiée.

Il enrichit le langage du système de Hansson  $DSDL3$  [16] et fournit une sémantique plus forte. Dans le système  $G$ , on dispose de :

- un ensemble dénombrable de propositions atomiques  $Prop$ , et les constantes  $\top$  et  $\perp$
- les opérateurs booléens classiques ( $\neg, \wedge, \vee, \dots$ )
- les opérateurs aléthiques  $\Box$  et  $\Diamond$
- les opérateurs déontiques  $O$  et  $P$

L'ensemble des  $G$ -formules est défini récursivement, de façon usuelle. On n'a aucune restriction sur la composition des opérateurs aléthiques et déontiques.

### 2.3.2 Les axiomes

Le système d'axiomes de  $G$  comprend les axiomes de la logique propositionnelle, ceux du système modal  $S5$ , huit axiomes reliant les opérateurs déontiques et aléthiques :

$$S5 \left\{ \begin{array}{l} \Box(A \Rightarrow B) \Rightarrow (\Box A \Rightarrow \Box B) \\ \Box A \Rightarrow \Box \Box A \\ \Diamond A \Rightarrow \Box \Diamond A \end{array} \right.$$

- $P(A/B) \Rightarrow \neg O(\neg A/C)$  (DfP)
- $O(A/A)$  (H)
- $O(A \Rightarrow B/C) \Rightarrow (O(A/C) \Rightarrow O(B/C))$  (K)
- $\Diamond B \Rightarrow (O(A/B) \Rightarrow P(A/B))$  (D\*)
- $P(A/C) \Rightarrow (O(B/A \wedge C) \Leftrightarrow O(A \Rightarrow B/C))$  (S)
- $\Box A \Rightarrow O(A/C)$  (N)
- $O(A/C) \Rightarrow \Box O(A/C)$  (A)
- $\Box(B \Leftrightarrow C) \Rightarrow (O(A/B) \Leftrightarrow O(A/C))$  (AEC)

et deux règles de déduction :

$$\frac{A \Rightarrow B, A}{B} (\text{Modus Ponens})$$

$$\frac{A}{\Box A} (\text{Nécessitation})$$

L'axiome (DfP) relie la permission à l'obligation de manière classique (on peut donc voir  $P$  comme un raccourci syntaxique). (K) est l'axiome modal classique (présent dans  $SDL$ ) adapté à l'opérateur déontique dyadique, et (D\*) est la version restreinte de l'axiome modal (D). (S) a été introduit par Spohn en 1975. (N) est une version axiomatique de la règle de  $O$ -nécessitation. (A) correspond à un des théorèmes étudiés de  $PDeL^{AM}$  (le 6). Et (AEC) est appelé axiome d'extension pour les circonstances.

### 2.3.3 La sémantique

Un  $G$ -modèle est un triplet  $M = (W, V, \succcurlyeq)$  où :

- $W$  est l'ensemble (non vide) des mondes possibles
- $V : W \rightarrow 2^W$  est une fonction qui affecte à chaque monde  $w \in W$  l'ensemble des atomes de  $Prop$  satisfaits par  $w$ .
- $\succcurlyeq : W \times W \rightarrow W$  est une relation binaire transitive sur les mondes possibles, qui se lit "est meilleur que". Elle satisfait la propriété suivante, qui exprime que tout sous-ensemble non vide de  $W$  possède un élément maximal.

$$\forall X \subseteq W \quad X \neq \emptyset \Rightarrow \{w \in X / \forall v \in X \quad w \succcurlyeq v\} \neq \emptyset$$

**Remarque 3** La propriété satisfaite par  $\succcurlyeq$  est nécessaire à la définition de  $O(A/B)$ .

#### Définition 5 (Sémantique de $G$ )

Les règles pour qu'une  $G$ -formule soit satisfaite dans un monde  $w$  d'un modèle  $\mathcal{M}$  sont les suivantes :

$$\mathcal{M} \models_w p \quad \text{ssi} \quad p \in \Pi(w)$$

$$\mathcal{M} \models_w \top$$

$$\mathcal{M} \not\models_w \perp$$

$$\mathcal{M} \models_w \Box A \quad \text{ssi} \quad \forall v \in W \quad \mathcal{M} \models_v A$$

$$\mathcal{M} \models_w \Diamond A \quad \text{ssi} \quad \exists v \in W \quad \mathcal{M} \models_v A$$

$$\mathcal{M} \models_w O(A/B) \quad \text{ssi} \quad \forall v \in W \quad (\mathcal{M} \models_v B \ \& \ \forall u \in W (\mathcal{M} \models_u B \Rightarrow v \succcurlyeq u) \Rightarrow \mathcal{M} \models_v A)$$

$$\mathcal{M} \models_w P(A/B) \quad \text{ssi} \quad \exists v \in W \quad (\mathcal{M} \models_v B \ \& \ \forall u \in W (\mathcal{M} \models_u B \Rightarrow v \succcurlyeq u) \ \& \ \mathcal{M} \models_w A)$$

Intuitivement, un monde  $w$  satisfait  $O(A/B)$  si tous les “bons” mondes satisfaisant  $B$  - c’est à dire les mondes maximaux au sens de  $\geq$  qui satisfont  $B$  - satisfont aussi  $A$ .

Donc, si  $O(A/B)$  est vrai dans un monde, il est forcément vrai dans tous les mondes.

La transitivité de  $\geq$  permet d’établir des propriétés intuitives pour les conditions des obligations. Par exemple, si  $\mathcal{M} \models_w O(A/B) \wedge O(A/\neg B)$  alors  $\mathcal{M} \models_w O(A/\top)$  (c’est à dire  $O(A)$ ).

**Définition 6 (Satisfaction d’une formule)**

*Une  $G$ -formule est satisfaite par un modèle  $\mathcal{M}$  ssi elle est satisfaite par tous les mondes de  $W$ .*

**Propriété 3** *L’ensemble de règles de déduction énoncé ci-dessus est sain et complet pour  $G$  muni des axiomes et de la sémantique définis ci-dessus.*

Ce système est proche de la vision modale classique. La relation  $\geq$  qui est le cœur de la représentation déontique, se rapproche plus d’une vision figée, où un monde est “bon” ou “mauvais”, indépendamment du monde duquel on se situe, que d’une vision où une relation entre les états donne un sens à l’obligation.

L’aspect dyadique est par contre une vision intéressante si les obligations et permissions sur lesquelles on veut travailler sont essentiellement conditionnelles.

# Chapitre 3

## Logiques avec notion de temps

La logique temporelle est essentielle pour la spécification de systèmes réactifs. Elle étend la logique des prédicats classique avec des opérateurs exprimant l'invariance, ou l'apparition prochaine d'évènements dans le temps. La logique temporisée étend encore la logique temporelle en faisant intervenir le temps de manière explicite. Nous introduirons le langage et la sémantique de la logique temporelle dans un premier temps, puis de la logique temporisée.

On dispose de tous les opérateurs de la logique propositionnelle, et d'un ensemble de propositions atomiques *Prop*.

### 3.1 Logiques temporelles

Il existe principalement deux familles de logiques temporelles : la logique temporelle linéaire, et la logique temporelle arborescente.

#### 3.1.1 Logique temporelle linéaire

Voici la liste des principaux opérateurs utilisés pour la logique temporelle linéaire.

Opérateurs du futur		Opérateurs du passé	
$\Box p$	$p$ sera toujours	$\Box p$	$p$ a toujours été
$\Diamond p$	$p$ sera un jour	$\Diamond p$	$p$ a été
$p \mathcal{U} q$	$p$ jusqu'à $q$	$p \mathcal{S} q$	$p$ depuis $q$
$p \mathcal{W} q$	$p$ jusqu'à $q$ , ou toujours $p$	$p \mathcal{B} q$	$p$ depuis $q$ , ou $p$ a toujours été
$\circ p$	$p$ sera dans l'état suivant	$\ominus$	$p$ était dans l'état précédent

Plusieurs théories logiques (système d'axiomes + sémantique) existent. La plus connue est *LTL* (*Linear Temporal Logic*). Elle ne permet d'exprimer que le futur. Son système d'axiomes est le suivant :

- Les axiomes de la logique propositionnelle
- $\Box(A \Rightarrow B) \Rightarrow (\Box A \Rightarrow \Box B)$
- $\circ \neg A \Leftrightarrow \neg \circ A$
- $\circ(A \Rightarrow B) \Rightarrow (\circ A \Rightarrow \circ B)$
- $\Box A \Rightarrow (A \wedge \circ A \wedge \circ \Box A)$
- $\Box(A \Rightarrow \circ A) \Rightarrow (A \Rightarrow \Box A)$
- $\Box A \Rightarrow A \mathcal{U} B$
- $A \mathcal{U} B \Leftrightarrow B \vee (A \wedge \circ(A \mathcal{U} B))$

Le système est muni des trois règles d'inférence suivantes :

$$\frac{A, A \Rightarrow B}{B} \text{ (Modus Ponens)}$$

$$\frac{A}{\circ A} \text{ } \circ\text{-nécessitation}$$

$$\frac{A}{\Box A} \text{ } \Box\text{-n\u00e9cessitation}$$

**D\u00e9finition 7 (LTL-mod\u00e8le)**

Un mod\u00e8le (pour LTL ou les autres logiques temporelles lin\u00e9aires) est un couple  $(\sigma, \Pi)$ , o\u00f9  $\sigma$  est une s\u00e9quence infinie d'\u00e9tats  $\sigma = \sigma_0\sigma_1\dots$ , et  $\Pi$  est une fonction associe \u00e0 chaque \u00e9tat les propositions atomiques qu'il satisfait.

**D\u00e9finition 8 (S\u00e9mantique de LTL)**

\u00c9tant donn\u00e9 un mod\u00e8le  $\sigma$  et un \u00e9tat  $s_i$  de  $\sigma$ , voici les r\u00e8gles de s\u00e9mantique pour les op\u00e9rateurs du futurs :

- $\sigma \models_i p \text{ ssi } p \in \Pi(s_i)$  (pour  $p \in Prop$ )
- $\sigma \models_i \neg A \text{ ssi } \sigma \not\models_i A$
- $\sigma \models_i A \vee B \text{ ssi } \sigma \models_i A \text{ ou } \sigma \models_i B$
- $\sigma \models_i \Box A \text{ ssi } \forall k \geq i \sigma \models_k A$
- $\sigma \models_i \Diamond A \text{ ssi } \exists k \geq i \sigma \models_k A$
- $\sigma \models_i \circ A \text{ ssi } \sigma \models_{i+1} A$
- $\sigma \models_i A \mathcal{U} B \text{ ssi } \exists k \geq i \text{ tel que } \sigma \models_k B \text{ et } \forall j \ i \leq j < k \Rightarrow \sigma \models_j A$
- $\sigma \models_i A \mathcal{W} B \text{ ssi } \sigma \models_i A \mathcal{U} B \text{ ou } \sigma \models_i \Box A$

**D\u00e9finition 9 (Satisfaction d'une formule)**

Une formule  $A$  est satisfaite par le mod\u00e8le  $\mathcal{M} = (\sigma, \Pi)$  si et seulement si elle est satisfaite par le premier \u00e9tat de  $\sigma$  :

$$\sigma \models A \text{ ssi } \sigma \models_0 A$$

Une formule est valide si elle est satisfaite par tout mod\u00e8le.

**Propri\u00e9t\u00e9 4** L'ensemble des r\u00e8gles d'inf\u00e9rence \u00e9nonc\u00e9 ci-dessus est sain et complet vis \u00e0 vis de LTL muni du syst\u00e8me d'axiomes et de la s\u00e9mantique d\u00e9finis ci-dessus.

La figure 3.1 repr\u00e9sente une s\u00e9quence infinie d'\u00e9tats, qui satisfait  $\Box p$ ; 3.2 montre un exemple de s\u00e9quence satisfaisant  $\Diamond p$ .

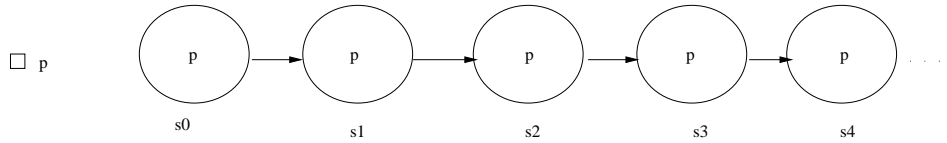


FIG. 3.1 – Mod\u00e8le satisfaisant  $\Box p$

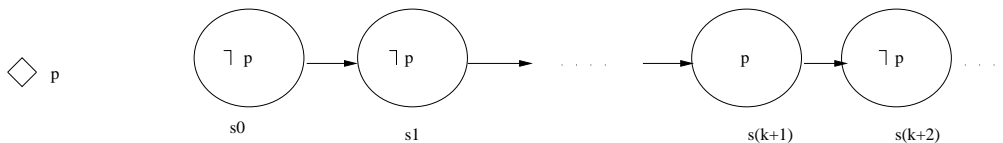


FIG. 3.2 – Mod\u00e8le satisfaisant  $\Diamond p$

### 3.1.2 Logiques temporelles arborescentes

La logique temporelle linéaire ne permet pas d'exprimer la notion de possibilité. En effet, on peut énoncer le fait qu'un évènement arrive forcément dans le futur, ou au contraire jamais, mais on ne sait pas dire qu'un évènement peut arriver. Il faut pour cela une alternative au déroulement linéaire du temps de *LTL*. On parle alors de logique arborescente. La plus connue est *CTL* (*Computation Tree Logic*).

#### Définition 10 (Syntaxe de CTL)

On rappelle qu'une formule  $\phi$  bien formée de CTL est définie ainsi :

$$\phi, \psi, \dots \stackrel{def}{=} p \mid \neg\phi \mid \phi \wedge \psi \mid EX \phi \mid AX \phi \mid E(\phi \mathcal{U} \psi) \mid A(\phi \mathcal{U} \psi)$$

où  $\phi$  et  $\psi$  sont des formules bien formées quelconques, et  $p$  un élément de l'ensemble *Prop* des propositions atomiques.

On dispose des abréviations suivantes :

- $EF\phi \stackrel{def}{=} E(\top \mathcal{U} \phi)$  il existe un instant du futur auquel  $\phi$  est vrai
- $AG\phi \stackrel{def}{=} \neg EF(\neg\phi)$   $\phi$  sera toujours vrai
- $AF\phi \stackrel{def}{=} A(\top \mathcal{U} \phi)$  dans tous les futurs il existe un instant auquel  $\phi$  est vrai
- $EG\phi \stackrel{def}{=} \neg AF(\neg\phi)$  il existe un futur dans lequel  $\phi$  est vrai à tout instant

**Définition 11 (CTL-modèle)** Un modèle pour CTL est un quadruplet  $\mathcal{M} = (S, R, \Pi, I)$ , où  $S$  est un ensemble d'états,  $R \subseteq S \times S$  une relation d'accessibilité sur les états, et  $\Pi \in S \rightarrow \mathcal{P}(\text{Prop})$  la fonction associant à chaque état l'ensemble des propositions atomiques satisfaites par cet état, et  $I \subseteq S$  l'ensemble des états initiaux.

Un chemin  $\sigma$  dans  $\mathcal{M}$  est une séquence  $\sigma = s_0s_1\dots$  telle que

- tout  $s_i$  soit un état de  $S$
- $\forall i \ s_iRs_{i+1}$
- si  $\sigma$  est fini, avec  $s_n$  son dernier état, il n'existe pas d'état  $s_{n+1}$  tel que  $s_nRs_{n+1}$ .

On appelle  $\sigma_i$  l'état  $s_i$  du chemin  $\sigma$ .

#### Définition 12 (Sémantique de CTL)

La validité d'une formule  $\phi$  de CTL dans l'état  $s$  du modèle  $\mathcal{M}$  est défini ainsi (on ne traite ici que les opérateurs temporels) :

- $\mathcal{M} \models_s EX \phi$  ssi  $\exists s' \in S$  tel que  $\mathcal{M} \models_{s'} \phi \wedge sRs'$
- $\mathcal{M} \models_s AX \phi$  ssi  $\forall s' \in S \ sRs' \Rightarrow \mathcal{M} \models_{s'} \phi$
- $\mathcal{M} \models_s E(\phi \mathcal{U} \psi)$  ssi  $\exists \sigma$  chemin dans  $\mathcal{M}$  avec  $\sigma_0 = s$ , et  $\exists n$  tel que :
  - $\mathcal{M} \models_{\sigma_n} \psi$
  - $\forall i \ 0 \leq i < n \Rightarrow \mathcal{M} \models_{\sigma_i} \phi$
- $\mathcal{M} \models_s A(\phi \mathcal{U} \psi)$  ssi  $\forall \sigma$  chemin dans  $\mathcal{M}$  tel que  $\sigma_0 = s$ ,  $\exists n$  tel que :
  - $\mathcal{M} \models_{\sigma_n} \psi$
  - $\forall i \ 0 \leq i < n \Rightarrow \mathcal{M} \models_{\sigma_i} \phi$

#### Définition 13 (satisfaction d'une formule)

Une formule  $A$  est satisfaite par un modèle  $\mathcal{M}$  ( $\mathcal{M} \models A$ ) si et seulement si elle est satisfaite par un état initial de  $\mathcal{M}$ .

Elle est valide si et seulement si tout modèle la satisfait.

**Remarque 4** On a gagné l'expressivité de la possibilité, par rapport à *LTL*, mais on ne peut plus exprimer des propriétés d'équité (cf introduction).

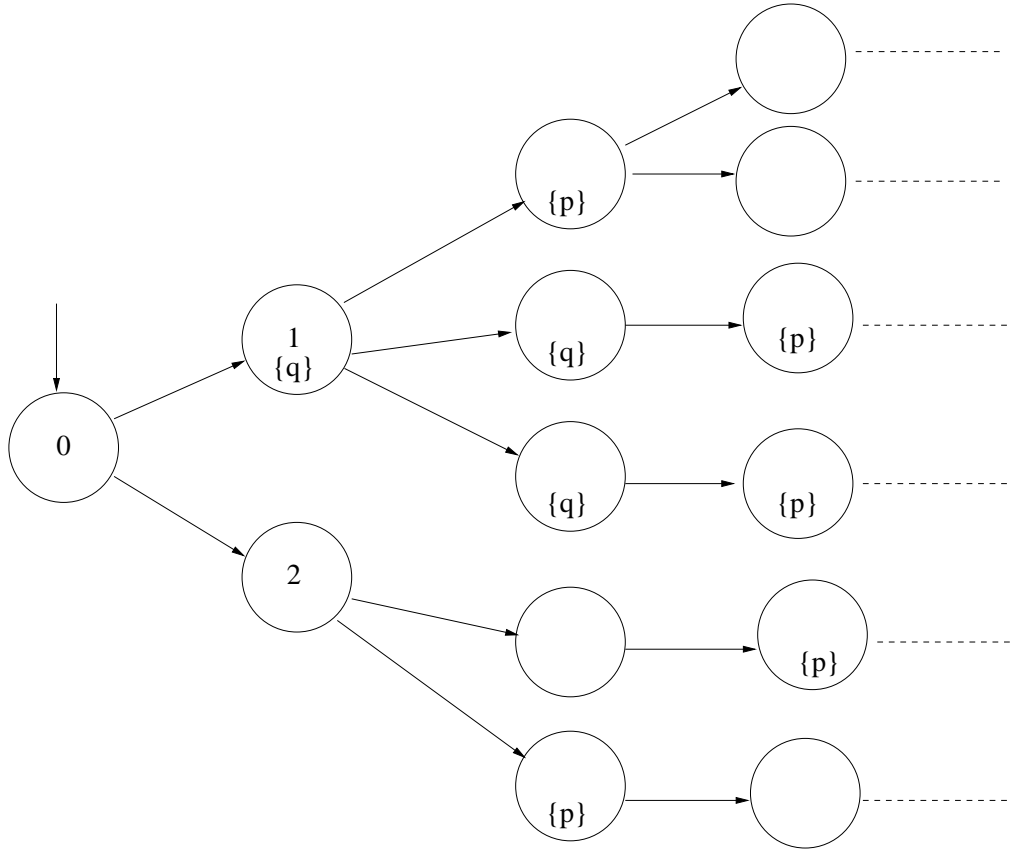


FIG. 3.3 – Exemple de modèle *CTL*

La figure 3.3 représente un exemple de modèle *CTL*, comportant un seul état initial (l'état 0). Celui-ci - donc le modèle - satisfait  $AFp$  puisque tous les chemins partant de 0 passent par un état satisfaisant  $p$ . Voici quelques propriétés satisfaites par les premiers états :

- $\mathcal{M} \models_1 A(q \mathcal{U} p)$
- $\mathcal{M} \models_1 EX q \wedge EX p$
- $\mathcal{M} \models_2 AF p$
- $\mathcal{M} \models_2 EX p$
- $\mathcal{M} \models AFp \wedge EF q$

## 3.2 Logiques temporisées

Pour exprimer des propriétés où le temps intervient de manière explicite (qu'on appelle propriétés temps-réel ou temporisées), comme "la formule  $A$  sera vraie dans 4 unités de temps", ou "pendant 5 unités de temps", on utilise des logiques dites temporisées. Les plus connues sont basées sur *CTL*. On distingue les versions dans lequel le temps est discret (*RTCTL*) et celles où le temps est continu (*TCTL*).

### 3.2.1 Temps discret

Nous prendrons dans cette partie l'exemple de *RTCTL* (*Real-Time Computation Tree Logic*) [14]. Cette logique s'interprète sur la même structure de Kripke que *CTL*, beaucoup plus simple que les structures basés sur les automates temporisés, sur lesquelles s'interprètent les formules de *TCTL* (cf 3.2.2).

### Définition 14 (Le Langage)

La syntaxe de *RTCTL* est donnée récursivement de la manière suivante :

$$\phi \stackrel{def}{=} p \mid \neg\phi \mid \phi \wedge \phi \mid E(\phi \mathcal{U}_{\leq k} \phi) \mid A(\phi \mathcal{U}_{\leq k} \phi)$$

où  $p$  est un élément de l'ensemble *Prop* des propositions atomiques,  $k \in \mathbb{N} \cup \{+\infty\}$ .

On dispose des abréviations suivantes :

$$\begin{aligned} EF_{\leq k} \phi &\stackrel{def}{=} E(\top \mathcal{U}_{\leq k} \phi) & AG_{\leq k} \phi &\stackrel{def}{=} \neg EF_{\leq k}(\neg\phi) \\ AF_{\leq k} \phi &\stackrel{def}{=} A(\top \mathcal{U}_{\leq k} \phi) & EG_{\leq k} \phi &\stackrel{def}{=} \neg AF_{\leq k}(\neg\phi) \end{aligned}$$

On étend en fait l'expressivité du langage de manière artificielle. En effet, une formule de *RTCTL* peut s'exprimer en *CTL*, mais la hauteur temporelle de la formule augmente de manière exponentielle (cf [23]).

Les opérateurs de *CTL*  $E(\phi \mathcal{U} \psi)$  et  $A(\phi \mathcal{U} \psi)$  se retrouvent respectivement avec  $E(\phi \mathcal{U}_{\leq \infty} \psi)$  et  $A(\phi \mathcal{U}_{\leq \infty} \psi)$ .

### Définition 15 (La sémantique)

Un modèle  $\mathcal{M}$  de *RTCTL* est un modèle de *CTL* (cf 11). Les règles de sémantique sont définies récursivement de la manière suivante :

- $\mathcal{M} \models_s E(\phi \mathcal{U}_{\leq k} \psi)$  ssi  $\exists \sigma$  tel que  $\sigma_0 = s$ , et  $\exists i \leq k$  tel que
  - $\mathcal{M} \models_{\sigma_i} \psi$
  - $\forall j \ 0 \leq j \leq i \Rightarrow \mathcal{M} \models_{\sigma_j} \phi$
- $\mathcal{M} \models_s A(\phi \mathcal{U}_{\leq k} \psi)$  ssi  $\forall \sigma$  tel que  $\sigma_0 = s$ ,  $\exists i \leq k$  tel que
  - $\mathcal{M} \models_{\sigma_i} \psi$
  - $\forall j \ 0 \leq j \leq i \Rightarrow \mathcal{M} \models_{\sigma_j} \phi$

## 3.2.2 Temps continu

Nous traiterons ici de *TCTL* (*Timed Computation Tree Logic*) [2, 1] une version temporisée de *CTL* avec temps continu, dont les formules s'interprètent sur des automates temporisés. Nous utiliserons la définition formelle de [17].

### *TCTL*

*TCTL* est une généralisation de *CTL*. On dispose d'une représentation explicite du temps. L'opérateur d'initialisation  $z$ . introduit une horloge auxiliaire  $z \in \mathcal{H}$  qui mesure le temps écoulé depuis l'état présent. Ainsi,  $z. A(p \mathcal{U} (q \wedge z < 2))$  exprime que  $p$  est vrai jusqu'à ce que  $q$  le soit et que  $q$  deviendra vrai avant 2 unités de temps. Plus formellement, voici la définition de *TCTL*.

### Définition 16

On dispose d'un ensemble *Prop* de propositions atomiques, d'un ensemble  $\mathcal{H}$  de variables appelées horloges, prenant leurs valeurs dans  $\mathbb{R}^+$ . La syntaxe de *TCTL* est donnée récursivement de la manière suivante :

$$\phi \stackrel{def}{=} p \mid x < c \mid x - y < c \mid z. \phi \mid \neg\phi \mid \phi \wedge \phi \mid E(\phi \mathcal{U} \phi) \mid A(\phi \mathcal{U} \phi)$$

où  $p \in \text{Prop}$ ,  $x, y, z \in \mathcal{H}$ ,  $c \in \mathbb{Z}$ , et  $< \in \{ <, \leq \}$ .

On trouve souvent des opérateurs temporels bornés à la place de l'opérateur d'initialisation. Ces opérateurs sont en fait des raccourcis utilisant les opérateurs que nous avons définis plus haut.

$$E(\phi \mathcal{U}_{*k} \psi) \stackrel{def}{=} z. E(\phi \mathcal{U} (\psi \wedge z * k))$$

$$A(\phi \mathcal{U}_{*k} \psi) \stackrel{def}{=} z. A(\phi \mathcal{U} (\psi \wedge z * k))$$

où  $* \in \{ \leq, <, \geq, >, = \}$

Par exemple,  $A(\phi U_{\leq 4} \psi)$  signifie que  $\phi$  est vrai jusqu'à ce que  $\psi$  le soit, et  $\psi$  est vrai au bout de moins de 4 unités de temps.

On trouve dans la littérature certaines définition de *TCTL* à partir de ces opérateurs bornés à la place de l'opérateur d'initialisation de l'horloge.

**Remarque 5** *La définition de TCTL avec les opérateurs temporels bornés est moins expressive que celle que nous utilisons.*

Différents raccourcis sont couramment employés.

$$EF_{**k}\phi \stackrel{def}{=} E(\top U_{**k} \phi) \quad AG_{**k}\phi \stackrel{def}{=} \neg EF_{**k}(\neg\phi)$$

$$AF_{**k}\phi \stackrel{def}{=} A(\top U_{**k} \phi) \quad EG_{**k}\phi \stackrel{def}{=} \neg AF_{**k}(\neg\phi)$$

$AG_{\leq 3}p$  exprime ainsi que  $p$  reste vrai jusqu'à 3 unités de temps.  $EF_{> 2}p$  indique qu'il existe une exécution dans laquelle  $p$  deviendra vrai dans plus de 2 unités de temps.

### Automates temporisés

Nous ne donnerons ici qu'une notion intuitive des automates temporisés. Pour une définition formelle, on se référera à [3, 26].

La figure 3.4 représente un exemple d'automate temporisé. Il modélise un système qui surveille que

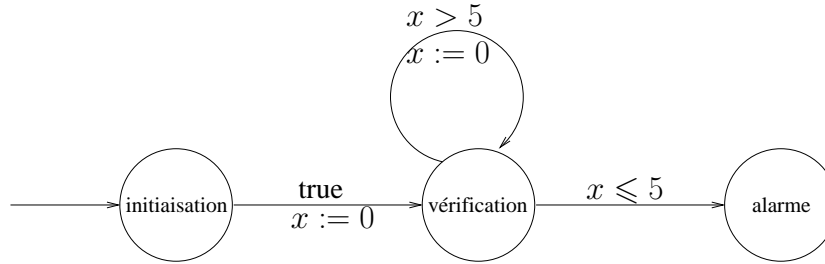


FIG. 3.4 – Exemple d'automate temporisé

l'intervalle entre deux évènements soit supérieur à cinq unités de temps, et déclenche une alarme le cas échéant. Ici, les transitions sont franchies lorsque l'évènement à surveiller arrive.

Chacune comporte une garde (éventuellement *true*) et un ensemble d'horloges qui sont réinitialisées (ici la seule horloge est  $x$ ). Les horloges augmentent toutes uniformément, de manière synchrone, et indiquent le temps écoulé depuis leur dernière initialisation. Elles valent toutes 0 dans l'état initial.

Les sommets de l'automate peuvent eux aussi comporter des conditions que le système respecte forcément (appelées invariants d'état). La présence de ces conditions entraîne la possibilité de rester bloqué dans un état. Dans l'exemple de la figure 3.5, il est possible de voir une exécution dans laquelle on reste entre l'état 0 et l'état 1 pendant 3 unités de temps, et de ne plus pouvoir franchir la transition menant à l'état 2 à cause de la garde  $x \leq 2$ . Une telle exécution ne peut plus progresser, après 3 unités de temps, à cause des contraintes sur les sommets 1 et 2.

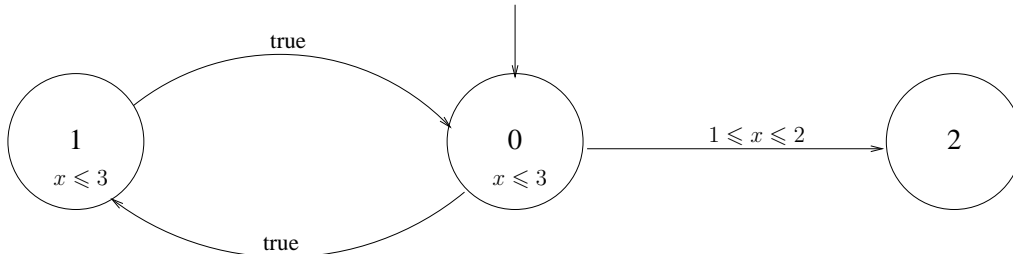


FIG. 3.5 – Exemple d'automate mal temporisé

Lors de l'existence d'exécutions ne pouvant se poursuivre dans le temps (on parle de *deadlock*), on dit que l'automate est mal temporisé.

Pour corriger le problème de l'automate de la figure 3.5, on peut renforcer les conditions sur les sommets comme indiqué sur la figure 3.6. Sans donner la définition formelle de la satisfaction d'une

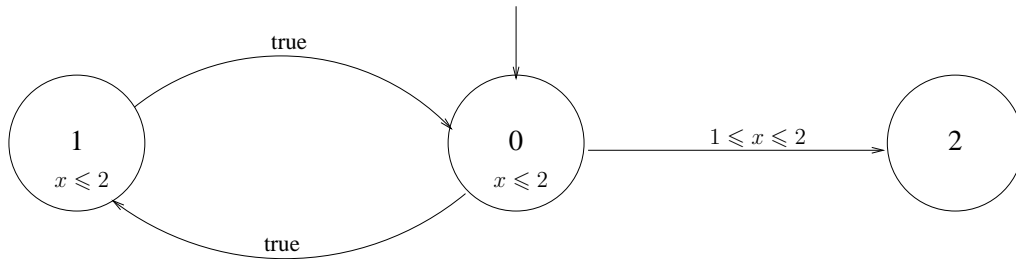


FIG. 3.6 – Automate bien temporisé

formule *TCTL* pour le modèle découlant d'un automate temporisé, nous allons fournir quelques exemples pour illustrer cette notion.

Étant donné un ensemble de propositions atomiques *Prop*, on dispose d'une fonction donnant pour chaque sommet l'ensemble des propositions atomiques qu'il satisfait. Ainsi l'automate de la figure 3.7 satisfait  $AG AF_{\leq 3}p$ . C'est à dire "À tout instant, *p* sera vrai dans moins de trois unités de temps". En effet, on reste dans l'état 1, qui ne satisfait pas *p*, au plus trois unités de temps.

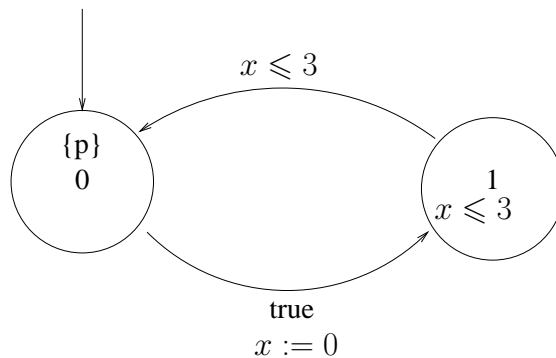


FIG. 3.7 – Automate satisfaisant  $AG AF_{\leq 3}p$

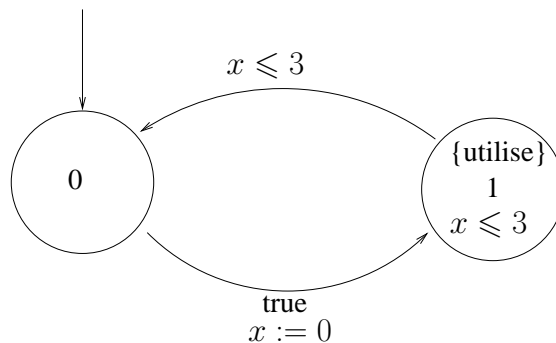


FIG. 3.8 – Processus utilisant une ressource

On peut se servir de cet automate pour modéliser le bon comportement d'un processus vis à vis d'une politique qui l'oblige à libérer la ressource au bout de trois unités de temps. Plus formellement, la

politique impose  $AG (utilise \Rightarrow AF_{\leq 3} \neg utilise)$ . L'automate de la figure 3.8 satisfait bien cette contrainte. Il représente les exécutions d'un processus pouvant utiliser une ressource (état 1) ou pas (état 0). Toute exécution commence dans l'état 0 (état initial), et peut à un moment quelconque utiliser une ressource (état 1). Elle doit alors repasser dans l'état 0 avant trois unités de temps.

**Remarque 6** *On s'aperçoit alors que cette formule est équivalente à celle satisfaite par l'automate 3.7. Il suffit donc de vérifier  $AG AF_{\leq 3} \neg utilise$  pour assurer que le comportement du processus respecte une telle politique.*

# Chapitre 4

## Combinaison déontique/temps

Dans ce chapitre nous présenterons les principaux formalismes existant qui font intervenir à la fois la notion de temps et celle d'obligation. Tout d'abord, le système  $RS5 - DS5$  qui dispose d'un temps continu, et d'une relation déontique fixant les "bons" mondes associés à un état parmi les futurs possibles de cet état. Ensuite, le système  $DTL$ , qui représente le temps de manière discrète mais possède une relation déontique plus fine - et donc plus complexe - que les précédents formalismes. Enfin nous nous intéresserons à deux propositions de logiques déontiques adaptées à l'expression des échéances.

### 4.1 RS5-DS5

#### 4.1.1 La syntaxe

Cette logique, présentée par Bailhache dans [6], puis dans [7] mélange des opérateurs temporels ( $\square$  et  $\diamond$ ), un opérateur déontique ( $O$ ), et un opérateur temporisé ( $R_t$ ), où  $R_t(A)$  indique que la formule  $A$  est satisfaite à l'instant  $t$  ( $t \in \mathbb{R}$ ).

On peut donc ici combiner tous les opérateurs de nature différente, dans n'importe quel ordre, et un nombre quelconque de fois. La présence de l'opérateur temporisé  $R_t$  fournit une expressivité intéressante. En contrepartie, on ne dispose pas de procédure de décision, la décidabilité n'étant pas établie.

Un autre inconvénient pour notre application vient du fait qu'on ne dispose pas de temps relatif :  $R_t A$  signifie que  $A$  est satisfaite à l'instant absolu  $t$ . Par exemple, on ne peut pas exprimer "si  $A$  est vrai,  $B$  est vrai 2 heures après".

On n'a pas non plus à disposition l'obligation avec échéance : " $A$  doit être vrai avant 2 heures".

#### 4.1.2 La sémantique

##### Définition 17 ( $RS5 - D5$ Modèle)

Un modèle  $\mathcal{M}$  de  $RS5 - D5$  est un triplet  $(W, R, S)$  où

- $W$  est l'ensemble des mondes ; chaque monde est un ensemble d'états  $w = \{w_t/t \in \mathbb{R}\}$  où  $t$  désigne un instant.
- $R, S \subseteq W \times \mathbb{R} \times W$  sont deux relations liant un instant (réel), et deux chemins, permettent respectivement de donner un sens aux opérateurs temporels et déontiques.
  - $wR_t w'$  indique que  $w$  et  $w'$  sont deux chemins ayant le même passé, avant  $t$ .
  - $wS_t w'$  indique que  $w$  et  $w'$  sont deux chemins ayant le même passé avant  $t$ , et que  $w'$  est "bon" pour les instants  $t' > t$ .

C'est cette notion de "bon" monde qui va caractériser la notion déontique, et différencier les opérateurs  $\square$  et  $O$ .

En effet,  $\square A$ , vu de l'état  $w_t$ , signifie que dans tous les états futurs (les  $(w_{t'})$  avec  $t' > t$  et  $w'$  connecté à  $w$  à l'instant  $t$ )  $A$  est vrai.

$O(A)$  signifient que  $A$  est vrai dans tous les "bons" états du futur.

**Définition 18 (Sémantique de  $RS5 - D5$ )**

$$\begin{aligned} & \models_{w,t} R_t A \quad ssi \quad \models_{w,t'} A \\ & \models_{w,t} \Box A \quad ssi \quad \forall w' \in W \ (wR_t w' \Rightarrow \models_{w',t} A) \\ & \models_{w,t} OA \quad ssi \quad \forall w' \in W \ (wS_t w' \Rightarrow \models_{w',t} A) \end{aligned}$$

Les relations  $R$  et  $S$  possèdent différentes propriétés. À tout instant  $t$ ,  $R_t$  est une relation d'équivalence (réflexive, transitive, et symétrique). Elle a de plus la propriété suivante qui traduit la notion intuitive de ramification :

$$\forall t, t', w, w' \ (t' < t \ \& \ wR_t w') \Rightarrow wR_{t'} w'.$$

$S$  possède trois propriétés :

- $\forall t, w \ \exists w' \ wS_t w'$  (relation totale)
- $\forall t, w, w' \ wS_t w' \Rightarrow w'S_t w'$  (post-réflexivité)
- $\forall t, t', w, w', w'' \ (wS_{t'} w' \ \& \ w'S_t w'' \ \& \ t' < t) \Rightarrow w'S_{t'} w''$  (post-ramification)

La première propriété exprime, que dans tout monde, et à tout instant, il existe une bifurcation “bonne”.

Trois propriétés additionnelles lient  $R$  et  $S$  :

- $\forall t, w, w' \ wS_t w' \Rightarrow wR_t w'$  ( $RS$ -implication)
- $\forall t, w, w', w'' \ (wR_t w' \ \& \ w'S_t w'') \Rightarrow wS_t w''$  ( $RS$ -transitivité)
- $\forall t, t', w, w', w'' \ (t' < t \ \& \ wS_{t'} w' \ \& \ w'R_t w'') \Rightarrow w'S_t w''$  ( $RS$ -post-implication)

( $RS$ -post-implication) impose que lorsqu'un chemin devient “bon” à partir d'un instant  $t'$ , il le reste pour tous les instants  $t > t'$ .

La figure 4.1 illustre un exemple de modèle de  $RS5 - DS5$ . On remarque que lorsqu'un chemin devient bon (représenté en gras sur la figure) il le reste dans toutes les bifurcations futures. De plus, à chaque bifurcation, au moins une branche est bonne.

On ne peut pas représenter toutes les bifurcations sur un intervalle, puisqu'il en existe une infinité à cause de la densité du temps.

**Remarque 7**  $RS5 - DS5$  ne possède pas de système d'axiomes. Les vérifications ne peuvent donc se faire que dans la sémantique.

### 4.1.3 Sémantique dans $CTL$

La structure arborescente de la sémantique pose naturellement la question de l'expression d'une partie du langage dans  $CTL$ . En considérant un temps discret, on peut exprimer la sémantique de l'opérateur  $O$  en  $CTL$ . Pour garder un équivalent discret de  $R_t$ , on doit disposer d'une proposition particulière marquant le temps à travers les états. On ne s'intéressera ici qu'à l'opérateur déontique.

Un monde de  $RS5 - DS5$  correspond alors à une suite d'états (ou chemin) de  $CTL$ . Un état de  $CTL$  est associé à un instant d'un monde de  $RS5 - DS5$ .

Il faut donc transcrire dans  $CTL$  :

- la relation  $R$  et ses propriétés
- la relation  $S$  et ses propriétés
- la règle de satisfaction de  $O$
- (éventuellement) la règle de satisfaction de  $R_t$

$R$  est directement traduit par la nature arborescente de  $CTL$ . En effet, le futur d'un état (dans  $CTL$ ) sélectionne les mondes (du point de vue de  $RS5 - DS5$ ) qui ont le même passé que le monde courant (avant l' instant courant).

La notion de “bon” monde exprimée au travers de la relation  $S$  peut se traduire dans  $CTL$  à l'aide d'un “marqueur” qui sera satisfait par tous les “bons” états. Dans  $RS5 - DS5$ , un monde est “bon” à partir d'un certain instant. On représente cela par une suite d'états marqués, à partir de l'instant où le monde devient “bon”.

Les propriétés de la relation  $S$  peuvent se traduire facilement. En fait, la plupart des propriétés traduisent l'aspect arborescent du modèle de  $RS5 - DS5$ , qui est automatiquement traduit de part la

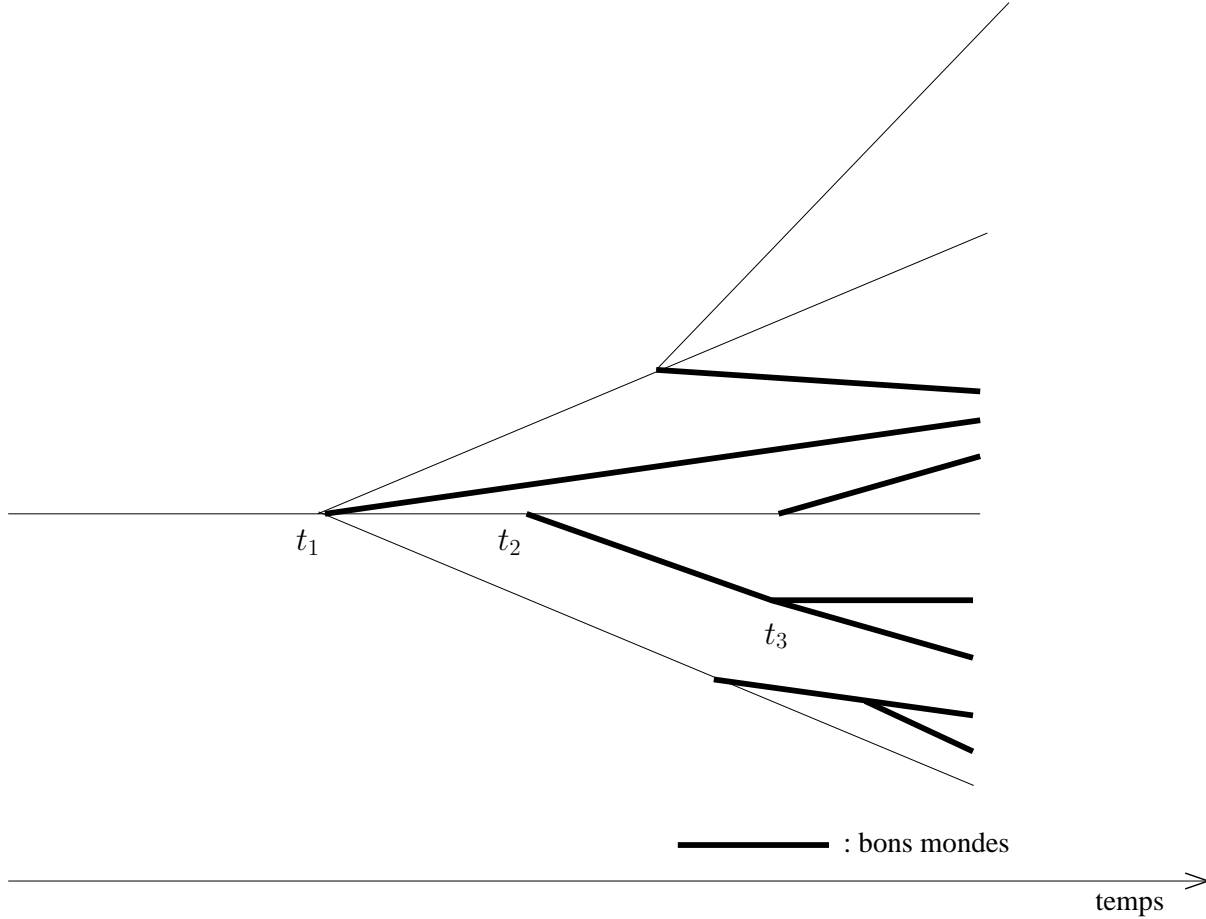


FIG. 4.1 – Modèle de  $RS5 - DS5$

structure d'un modèle  $CTL$ . Par exemple, la propriété de *post-ramification*, assez complexe à exprimer, signifie : si un monde  $w''$  est bon à partir d'un instant  $t$ , et qu'un monde  $w'$ , ayant le même passé que  $w''$  avant  $t$ , est bon à partir d'un instant  $t'$  antérieur à  $t$ , alors  $w''$  est bon dès l'instant  $t'$ . En  $CTL$ , les états correspondant à  $w''$  après  $t$  seraient marqués ; par ailleurs, les états correspondant à  $w'$  avant  $t'$  seraient marqués. Et la conclusion de l'implication est évidemment satisfaite dans le cas où  $w''$  et  $w'$  constituent le même chemin jusqu'à  $t$ .

Il reste à traduire deux propriétés :

- (*RS-post-implication*) (un chemin "bon" reste "bon")
- la relation  $S_t$  est totale (de tout monde, et à tout instant, il existe une bifurcation "bonne").

Appelons  $\delta$  la proposition particulière servant de marqueur. Les deux propriétés se traduisent de la manière suivante :

- $AG(\delta \Rightarrow AX\delta)$  *RS-post-implication*
- $AG(EF\delta)$  *S\_t totale*

**Définition 19** On arrive à la définition suivante de l'obligation, d'après la règle de sémantique donnée dans  $RS5 - DS5$

$$O(A) \stackrel{def}{=} AG(\delta \Rightarrow A)$$

Cette définition correspond à l'obligation dans  $RS5 - DS5$ , étant donné un modèle. Pour établir la validité de  $O(A)$ , il faudrait vérifier  $AG(\delta \Rightarrow A)$  pour tout  $\delta$  vérifiant les deux propriétés précédentes.

On verra plus loin (cf 5.1) que cette définition correspond à la vision classique de l'obligation exprimée en logique modale.

## 4.2 Le système *DTL* de Van Eck

Ce système est un des premiers qui combinent des opérateurs déontiques et temporels. Il apparaît en 1981 dans [12] et [13]. On ne traitera ici que de son fragment le plus intéressant, étudié par Hansen dans [15]. Cette logique ressemble (notamment les modèles) au formalisme de Aqvist,  $G$  (cf partie 2.3), mais son alphabet est plus riche que celui de  $G$ . On dispose en effet d'un ensemble dénombrable de constantes de temps  $TIC = \{t_1, t_2, \dots\}$  ainsi que de deux opérateurs  $<$  et  $=$  sur ces constantes. L'ensemble des *DTL*-formules  $\Sigma_{DTL}$  est défini de manière récursive :

- $\top \in \Sigma_{DTL}$  et  $\perp \in \Sigma_{DTL}$
- si  $p \in Prop$  et  $t \in TIC$  alors  $p(t) \in \Sigma_{DTL}$
- si  $A, B \in \Sigma_{DTL}$ , et  $t, t' \in TIC$ , alors  $\neg A, \Box_t A, O_t(A/B), P_t(A/B), A \& B, A \rightarrow B, A \leftrightarrow B, t < t', t = t' \in \Sigma_{DTL}$

L'instant auquel on se situe pour évaluer la formule se situe dans la syntaxe, et non dans la sémantique. Ainsi, pour évaluer l'atome  $p$  dans le monde  $w$  et à l'instant  $t$ , on évalue en fait  $p(t)$  dans le monde  $w$ . Cela revient en fait au même puisque la fonction  $V_2$  fournit pour chaque monde, et pour chaque instant, l'ensemble des propositions atomiques satisfaites.

**Remarque 8** *Il n'existe pas pour l'instant de système d'axiomes pour *DTL*. Van Eck a par contre défini une sémantique.*

### Définition 20 (*DTL*-modèle)

Un *DTL*-modèle  $\mathcal{M}$  est un sextuplet  $(Z, \ll, W, V_1, V_2, Q)$  où :

- $Z = \{z_1, z_2, \dots\}$  est un ensemble de points de temps
- $\ll$  est une relation d'ordre stricte totale sur  $Z$
- $W$  est l'ensemble des mondes possibles
- $V_1 : TIC \rightarrow Z$  est une fonction affectant un point du temps à toute constante de *TIC*
- $V_2 : W \times Z \rightarrow \mathcal{P}(Prop)$  est une fonction assignant à chaque monde de  $W$ , étant donné un instant de  $Z$ , l'ensemble des propositions atomiques satisfaites.
- $Q : W \times Z \times \mathcal{P}(W) \rightarrow \mathcal{P}(W)$  est une fonction qui choisit, étant donné un certain monde  $w$  et un instant  $z$ , les mondes qui sont "bons" (parmi un ensemble de mondes), c'est à dire ceux qui serviront à connaître les formules qui "devront" être satisfaites du point de vue de  $w$  à l'instant  $t$ .

La fonction  $Q$  doit vérifier plusieurs propriétés qui garantissent qu'elle correspond à l'intuition dont on vient de parler. Pour exprimer ces propriétés de manière plus légère, nous allons définir  $R$  telle que  $wR_z v$  soit vrai si les mondes  $w$  et  $v$  ont le même passé avant  $z$ . C'est à dire

$$wR_z v \stackrel{def}{=} \forall p \in Prop \forall z' \in Z (z' \ll z \Rightarrow V_2(w, z', p) = V_2(v, z', p))$$

Les propriétés que  $Q$  doit vérifier sont les suivantes :

- (i)  $Q(w, z, X) \subseteq X$  :  $Q$  ne choisit que parmi le sous-ensemble  $X$  ;
- (ii)  $Q(w, z, X) = Q(w, z, X \cap \{u \in W / wR_z u\})$  :  $Q$  ne peut choisir que parmi les mondes accessibles depuis  $w$  à l'instant  $z$  ;
- (iii)  $Q(w, z, X) = \emptyset \Rightarrow X \cap \{u \in W / wR_z u\} = \emptyset$  :  $Q$  peut toujours proposer des "bons" mondes parmi ceux qui sont accessibles ;
- (iv)  $z' \ll z \Rightarrow (wR_z v \Rightarrow Q(w, z', X) = Q(v, z', X))$  : deux chemins ayant le même passé avant  $z$  se voient proposer, à n'importe quel moment de leur passé commun, les mêmes "bons" chemins.

### Définition 21 (Sémantique de *DTL*)

Les conditions pour qu'une *DTL*-formule soit vraie dans un monde  $w$  d'un modèle  $\mathcal{M}$  sont définies récursivement de la manière suivante :

- $\mathcal{M} \models_w \top$
- $\mathcal{M} \not\models_w \perp$
- $\mathcal{M} \models_w p(t)$  ssi  $p \in V_2(w, V_1(t))$
- $\mathcal{M} \models_w \Box_t A$  ssi  $\forall v \in W (wR_{V_1(t)} v \Rightarrow \mathcal{M} \models_v A)$

- $\mathcal{M} \models_w \diamond_t A \quad ssi \quad \exists v \in W (wR_{V_1(t)}v \Rightarrow \mathcal{M} \models_v A)$
- $\mathcal{M} \models_w O_t(A/B) \quad ssi \quad \forall v \in Q(w, V_1(t), \| B \|) \quad \mathcal{M} \models_v A$
- $\mathcal{M} \models_w P_t(A/B) \quad ssi \quad \exists v \in Q(w, V_1(t), \| B \|) \quad \mathcal{M} \models_v A$
- $\mathcal{M} \models_w t' < t \quad ssi \quad V_1(t') \ll V_1(t)$
- $\mathcal{M} \models_w t' = t \quad ssi \quad V_1(t') = V_1(t)$

où  $\| A \|$  désigne  $\{w / \mathcal{M} \models_w A\}$

### Définition 22 (Satisfaction d'une formule)

$\mathcal{M} \models A \quad ssi \quad \forall w \in W \quad \mathcal{M} \models_w A$

Une DTL-formule est valide si tout modèle la satisfait.

La représentation de l'obligation dans la sémantique est intéressante. C'est la fonction de choix  $Q$  qui détermine les "bons" états. Ici, ce concept n'est pas absolu. Comme pour *SDL*, un "bon" état vu de l'état courant peut ne pas être "bon" vu d'un autre. En plus, ici, la fonction  $Q$  prend en argument un ensemble d'états candidats, et doit déterminer lesquels sont "bons" parmi ceux-ci. La définition de l'obligation est donc plus fine que dans les autres formalismes, mais plus complexe, ce qui empêche de disposer d'un système d'axiomes.

## 4.3 Échéances

Nous avons besoin d'exprimer l'obligation avec échéance (cf 1). Or, jusqu'ici, les logiques étudiées ne le permettent pas. Nous allons donc nous intéresser à deux formalismes qui ont fait cet effort. L'un est basé sur la logique d'action, l'autre est inclus dans *CTL*.

### 4.3.1 Échéance avec la logique d'action

Dignum et Kuiper ont combiné dans [10] une logique déontique dynamique (basée sur les actions) introduite par Meyer dans [19], avec la logique temporelle. Ils l'ont nommée *logique déontique dynamique temporelle temporisée linéaire (DDLTLB)*.

Cette logique combine la logique propositionnelle, la logique d'actions, la logique temporelle, et la logique déontique.

L'ensemble des actions *Act* est défini récursivement de la manière suivante :

$$\alpha ::= \underline{a} | \mathbf{any} | \mathbf{fail} | \alpha_1 + \alpha_2 | \alpha_1 \& \alpha_2 | \bar{\alpha}$$

$\underline{a}$  désigne une action atomique, **any** et **fail** sont des action spéciales représentant respectivement "n'importe quelle action" et "échec".  $\&$  dénote l'exécution parallèle, et  $+$  le choix entre deux actions.

On dispose d'un ensemble de propositions atomiques *Prop*. L'ensemble des formules  $\Sigma_{DDLTLB}$  de *DDLTLB* (que l'on notera  $\Sigma$  pour alléger l'écriture) est défini ainsi :

*Logique propositionnelle*

- $\top, \perp \in \Sigma$
- si  $A \in Prop$  alors  $A \in \Sigma$
- si  $A, B \in \Sigma$  alors  $A \wedge B, \neg A \in \Sigma$

*Logique d'actions*

- si  $\alpha, \alpha' \in Act$  alors  $[\alpha]A, PREFER(\alpha, \alpha'), DO(\alpha), DONE(\alpha) \in \Sigma$

*Logique temporelle*

- si  $A \in \Sigma$  alors  $\circ A, \ominus A \in \Sigma$
- si  $A, B \in \Sigma$  alors  $A \cup B, A \cup B \in \Sigma$

*Logique déontique*

- si  $A \in \Sigma$  alors  $O(A) \in \Sigma$

On définit ensuite les opérateurs temporels classiques suivants :

- $\diamond A \stackrel{def}{=} \top \mathcal{U} A$
- $\square A \stackrel{def}{=} \neg \diamond \neg A$

et les opérateurs déontiques :

- $P(A) \stackrel{def}{=} \neg O(\neg A)$
- $F(A) \stackrel{def}{=} O(\neg A)$

### Définition 23 (DDLTLB-modèle)

Un modèle  $\mathcal{M}$  pour DDLTLB est défini de la manière suivante :  $\mathcal{M} = (\mathcal{A}, \Sigma, \Pi, \leq, R, R_o)$  où

- $\mathcal{A}$  est un ensemble fini d'événements
- $\Sigma$  est un ensemble d'états
- $\Pi : \Sigma \rightarrow Prop$  est une fonction attribuant à chaque état l'ensemble des propositions atomiques vérifiées par cet état.
- $\leq$  est une relation "est meilleur que" permettant de comparer les états
- $R : \mathcal{A} \times \Sigma \times \Sigma \rightarrow \{0, 1\}$  est une relation indiquant comment les événements transforment les états :  $R_t(\sigma, \sigma')$  indique que l'événement  $t$  transforme l'état  $\sigma$  en  $\sigma'$ .
- $R_o : (\Sigma \times Path(\mathcal{M})) \times (\Sigma \times Path(\mathcal{M})) \rightarrow \{0, 1\}$  relie un couple (état/chemin)  $(\sigma, \rho)$  aux couples  $(\sigma', \rho')$  qui constituent son idéal déontique : les obligations de  $\sigma$  sont satisfaites dans tous les  $\sigma'$ .

On note  $\|\alpha\|$  l'ensemble des événements de  $\mathcal{A}$  qu'on associe, dans la sémantique, à l'action  $\alpha$ .

Les formules s'interprètent ici sur des chemins. Nous avons besoin de quelques définitions pour donner les règles de la sémantique.

Soit  $\rho = (\dots, \sigma_i, t_i, \sigma_{i+1}, t_{i+1}, \dots)$  un chemin d'états/transitions à travers le modèle  $\mathcal{M}$ . L'ensemble de tous les chemins  $\rho$  possibles à travers  $\mathcal{M}$  est appelé  $Path(\mathcal{M})$ . Le  $i$ ème état de  $\rho$  est  $\rho(i)$ .  $\rho^-(\sigma_i)$  désigne le préfixe de  $\rho$  qui finit par  $\sigma_i$ , et  $\rho^+(\sigma_i)$  le suffixe de  $\rho$  qui commence par  $\sigma_i$ .

### Définition 24

$Path(\mathcal{M}, \sigma) = \{\rho / \rho \in Path(\mathcal{M}) \text{ et } \exists i \sigma = \rho(i)\}$

$Path^+(\mathcal{M}, \sigma) = \{\rho / \rho \text{ commence par } \sigma\}$

$Path^-(\mathcal{M}, \sigma) = \{\rho / \rho \text{ se termine par } \sigma\}$

On définit de plus la concaténation sur des chemins ( $\circ$ ).

### Définition 25 (Sémantique de DDLTLB)

Les règles pour qu'une formule soit satisfaite par l'état  $\sigma$  du chemin  $\rho$ , dans le modèle  $\mathcal{M}$ , sont les suivantes :

- $\mathcal{M} \models_{\rho, \sigma} p$  ssi  $p \in \Pi(\sigma)$
- $\mathcal{M} \models_{\rho, \sigma} A \wedge B$  ssi  $\mathcal{M} \models_{\rho, \sigma} A$  et  $\mathcal{M} \models_{\rho, \sigma} B$
- $\mathcal{M} \models_{\rho, \sigma} \neg A$  ssi  $\mathcal{M} \not\models A$
- $\mathcal{M} \models_{\rho, \sigma} [\alpha]A$  ssi  $\forall \sigma' \in \Sigma \quad \forall \rho' \in (\rho^-(\sigma) \circ (\sigma, t, \sigma') \circ Path^+(\mathcal{M}, \sigma')) \quad [R_t(\sigma, \sigma') \Rightarrow \mathcal{M} \models_{\rho', \sigma'} A]$
- $\mathcal{M} \models_{\rho, \sigma} O(A)$  ssi  $\forall \sigma' \in \Sigma \quad \forall \rho' \in Path(\mathcal{M}, \sigma) \quad (R_o((\sigma, \rho), (\sigma', \rho'))) \Rightarrow \mathcal{M} \models_{\rho', \sigma'} A$
- $\mathcal{M} \models_{\rho, \sigma} PREFER(\alpha_1, \alpha_2)$  ssi  $\forall t \in \|\alpha_2\| \quad (R_t(\sigma, \sigma_2) \Rightarrow (\exists t' \in \|\alpha_1\| (R_{t'}(\sigma, \sigma_1) \wedge (\sigma_1 \leq \sigma_2))))$
- $\mathcal{M} \models_{\rho, \sigma_i} \circ A$  ssi  $\mathcal{M} \models_{\rho, \sigma_{i+1}} A$
- $\mathcal{M} \models_{\rho, \sigma_i} \ominus A$  ssi  $\mathcal{M} \models_{\rho, \sigma_{i-1}} A$
- $\mathcal{M} \models_{\rho, \sigma_0} A \cup B$  ssi  $\exists k (\mathcal{M} \models_{\rho, \sigma_k} B \wedge \forall j \ 0 < j \leq k \Rightarrow \mathcal{M} \models_{\rho, \sigma_j} A)$
- $\mathcal{M} \models_{\rho, \sigma_0} A \mathcal{S} B$  ssi  $\exists k (\mathcal{M} \models_{\rho, \sigma_k} B \wedge \forall j \ k < j \leq 0 \Rightarrow \mathcal{M} \models_{\rho, \sigma_j} A)$
- $\mathcal{M} \models_{\rho, \sigma_0} DO(\alpha)$  ssi  $\rho = (\dots, \sigma_0, t_0, \sigma_1, \dots)$  et  $t_0 \in \|\alpha\|$

- $\mathcal{M} \models_{\rho, \sigma_0} DONE(\alpha)$  ssi  $\rho = (\dots, \sigma_{-1}, t_{-1}, \sigma_0, \dots)$  et  $t_{-1} \in \|\alpha\|$

**Définition 26 (Satisfaction d'une formule)**

On notera  $\mathcal{M} \models_{\sigma} A$  ssi  $\forall \rho \in Path(\mathcal{M}, \sigma)$   $\mathcal{M} \models_{\rho, \sigma} A$ . De même,  $\mathcal{M} \models A$  signifie  $\forall \sigma \in \Sigma$   $\mathcal{M} \models_{\sigma} A$ .

Enfin, on dit qu'une formule est valide ( $\models A$ ) ssi tout modèle  $\mathcal{M}$  la satisfait.

Nous disposons au sein de cette logique de l'obligation qu'une certaine formule soit établie, appelée obligation d'être. Grâce aux opérateurs *DONE* et *DO*, on peut exprimer l'obligation d'exécuter une certaine action (appelée obligation de faire), et plus précisément, l'obligation de l'exécuter avant une échéance, que l'on modélise alors par une formule. On note  $O(\alpha < A)$  l'obligation d'effectuer l'action  $\alpha$  avant que la formule  $A$  ne devienne vraie.

**Définition 27**

La définition proposée dans [10] est la suivante :

$$O(\alpha < A) \stackrel{def}{=} O(\neg A \mathcal{U} DO(\alpha))$$

Elle exprime l'obligation que  $A$  soit fausse jusqu'à ce que l'action  $\alpha$  soit exécutée.

Il peut être intéressant d'exprimer le fait qu'une action doit être exécutée avant une échéance, mais aussi après un certain instant (c'est à dire après qu'une certaine formule soit vérifiée, faute de représentation du temps explicite).

**Définition 28**

On a alors

$$O(A < \alpha < B) \stackrel{def}{=} (A \Rightarrow O(\neg B \mathcal{U} DO(\alpha))) \mathcal{U} (\ominus A)$$

Le " $\mathcal{U} \ominus A$ " est là pour que l'échéance soit active seulement la première fois que  $A$  est établie.

Pour aller plus loin, on voudrait pouvoir exprimer le temps de manière explicite. Pour cela, Dignum et Kuiper introduisent une variable spéciale, *time*, telle que toute valeur qu'on lui affecte vérifie l'axiome suivant :

$$\models time = k \Rightarrow \circ(time = k + 1)$$

On suppose alors que la formule  $time = k_0$  ne peut être satisfaite - pour chaque chemin - que dans un état. La définition devient alors, grâce à cette propriété, plus simple :

$$O(t_1 < \alpha < t_2) \stackrel{def}{=} \square(time = t_1 \Rightarrow O(time < t_2 \mathcal{U} DO(\alpha)))$$

Pour des échéance relatives, la définition devient

$$O(now + t_1 < \alpha < now + t_2) \stackrel{def}{=} time = k \Rightarrow \square(time = k + t_1 \rightarrow O(time < k + t_1 + t_2 \mathcal{U} DO(\alpha)))$$

Tout ce travail est fait avec une représentation discrète du temps. Dignum et Kuiper ont repris le même travail et l'ont adapté au temps continu dans [11]. On peut regretter de ne pas disposer de système d'axiomes et de manipuler une sémantique très complexe. L'utilisation d'un temps explicite mérite de plus d'être approfondie.

### 4.3.2 Utilisation de CTL

Broesen, Dignum, et Meyer, ont mis en évidence dans [18] la possibilité d'exprimer les échéances avec la logique temporelle arborescente *CTL* (cf partie 3.1.2). La notion d'obligation sera introduite par une proposition particulière représentant la violation, appelée *Viol*. Plus généralement, on se donne un ensemble  $A$  d'agents qui peuvent chacun avoir des obligations. On affecte alors à chaque agent  $a \in A$  une proposition *Viol*( $a$ ).

**Définition 29 (Obligation à partir de la violation)**

L'obligation pour l'agent  $a$  de respecter une échéance est définie au sein de  $CTL$  à l'aide de la proposition  $Viol(a)$  (on la note  $O_a^V$  car elle est définie à partir de la notion de violation).

$$\begin{aligned} \mathcal{M} \models_s O_a^V(\rho \leq \delta) \quad \text{ssi} \quad & \forall \sigma \text{ tel que } \sigma_0 = s, \quad \forall j \\ & \text{si } \mathcal{M} \models_{\sigma_j} \delta \text{ et } \forall 0 \leq i \leq j \quad \mathcal{M} \models_{\sigma_i} \neg \rho \\ & \text{alors } \mathcal{M} \models_{\sigma_j} Viol(a) \end{aligned}$$

Ceci correspond à la formule :

$$O_a^V(\rho \leq \delta) \stackrel{def}{=} \neg E(\neg \rho \mathcal{U} (\delta \wedge \neg Viol(a)))$$

Cette définition fait apparaître la propriété contre-intuitive suivante :

$$\models \rho \Rightarrow O_a(\rho \leq \delta)$$

C'est à dire que si une formule est vraie dans l'état présent, on a l'obligation de la satisfaire avant n'importe quel échéance. Cela correspond à la vision de l'obligation de  $SDL$  : "on ne fait que ce qu'on est obligé de faire".

Cette propriété est en fait un cas particulier de la propriété suivante :

$$\models \neg E(\neg \rho \mathcal{U} \delta) \Rightarrow O_a(\rho \leq \delta)$$

Pour pallier ce problème, les auteurs ont introduit une deuxième variable particulière,  $Idl$ , représentant le succès d'une obligation.

**Définition 30 (Obligation à partir du succès et de la violation)**

En combinant  $Idl$  et  $Viol$ , ils sont parvenus à la définition suivante :

$$O_a(\rho \leq \delta) \stackrel{def}{=} \neg E(\neg(\rho \wedge Idl(a)) \mathcal{U} (\delta \wedge \neg Viol(a)))$$

Cette définition respecte bon nombre de propriétés intuitives. Elle bénéficie en outre du fait que  $CTL$  est un formalisme connu, très utilisé, pour lequel il existe des algorithmes de vérification de modèle efficaces. Mais le temps n'apparaît pas de manière explicite. Il ne s'agit donc pas du même type d'échéance que celui que l'on désirait. On ne peut pas exprimer "Le processus  $p$  a l'obligation de libérer la ressource avant 3 ms".

# Chapitre 5

## Propositions

Dans ce chapitre, nous ferons tout d'abord le lien entre les différentes manières, vues dans les précédents chapitres, de représenter l'obligation. Deux modélisations ressortent (avec une proposition “marqueur”, et avec une relation déontique). Pour chacune nous proposerons un modèle adapté à une logique à la fois temporelle et déontique, dans lequel nous exprimerons l'obligation avec élégance.

### 5.1 Relation entre les différents formalismes présentés

On distingue deux approches générales pour représenter la notion d'obligation :

- par une relation déontique indiquant les “bons” mondes (une propriété vraie dans tous ces mondes est considérée comme obligatoire)
- par une variable particulière marquant la violation d'une obligation

La première vision se situe directement au niveau sémantique, la seconde au niveau syntaxique. Mais la traduction dans la sémantique, qui correspond plus à l'intuition, est directe, puisque dans la définition d'un modèle, on donne les états qui correspondent à un échec (ceux qui satisfont la variable “spéciale”).

Il s'agit en fait dans les deux cas du même principe : marquer certains états qui joueront un rôle particulier dans l'évaluation d'une obligation.

Nous avons déjà vu dans la partie 4.1.3 que la sémantique du système  $RS5 - D5$ , basée une notion de “bons” mondes, peut s'exprimer, dans  $CTL$ , en perdant toutefois la densité du temps. On marque alors les “bons” mondes à l'aide d'une proposition particulière au lieu d'utiliser une relation sur les mondes.

Regardons l'interprétation de  $O(A)$  dans les deux cas, que nous exprimerons dans  $CTL$  par souci de comparaison.

- Dans le premier,  $O(A)$  est vrai si  $A$  est vrai dans tous les “bons” mondes, c'est à dire en  $CTL$  :

$$O(A) \stackrel{def}{=} AG (\delta \Rightarrow A)$$

où  $\delta$  marque un “bon” monde.

- Dans le deuxième,  $O(A)$  est vrai si l'absence de  $A$  entraîne la violation, représentée par la proposition  $V$ . Ceci se note en  $CTL$  :

$$O(A) \stackrel{def}{=} AG (\neg A \Rightarrow V)$$

La définition est donc équivalente dans les deux cas, si l'on considère que  $\delta = \neg V$ , c'est à dire que les “bons” mondes sont ceux au sein desquels il n'y a pas de violation.

Il faut tout de même remarquer que la relation déontique de  $SDL$  fournit des “bons” mondes différents pour chacun des états. Le “marquage” par une proposition particulière devient alors impossible. Toutefois, si l'on définit la relation déontique  $R$  d'un modèle  $\mathcal{M}$  de  $SDL$  par

$$\forall w, w' \ wRw' \text{ ssi } \mathcal{M} \not\models_w V \tag{5.1}$$

alors l'obligation de  $SDL$  devient équivalente à  $AG(\neg A \Rightarrow V)$ . En effet,  $\mathcal{M} \models_w O(A)$

$$\Leftrightarrow \forall w' \ wRw' \Rightarrow \mathcal{M} \models_{w'} A$$

$$\begin{aligned} \Leftrightarrow \forall w' \mathcal{M} \not\models_{w'} V &\Rightarrow \mathcal{M} \models_{w'} A && \text{(d'après 5.1)} \\ \Leftrightarrow \forall w' \mathcal{M} \models_{w'} \neg V &\Rightarrow A \end{aligned}$$

La dernière ligne revient à  $\mathcal{M} \models_{w'} AG(\neg A \Rightarrow V)$  en faisant l'analogie entre un modèle de *SDL* et un modèle de *CTL*.

## 5.2 Modélisation avec une proposition particulière (marqueur)

Dans les travaux -peu nombreux- déjà réalisés sur l'obligation mêlée au temps, les deux notions sont toujours extrêmement imbriquées. En fait, les “bons” mondes qui vont caractériser la notion déontique sont choisis parmi les futurs possibles du monde courant [6, 18]. On voit bien qu'il s'agit là d'un choix philosophique à faire sur ce que l'on entend par obligation.

Nous allons dans ce paragraphe chercher à exprimer une obligation avec échéance explicite (“on a l'obligation de libérer la ressource avant  $2ms$ ”) en restant dans cet esprit. C'est à dire que les “bons” mondes qui serviront à définir l'obligation appartiendront au futur du monde courant. On se place à nouveau dans un cadre temporisé avec temps discret.

Notre langage sera celui de *RTCTL* (cf 3.2.1), auquel on ajoute une proposition particulière  $\delta$  servant à repérer les “bons” mondes (on aurait pu choisir une proposition représentant la violation d'une obligation).

On ajoute les règles habituelles portant sur les bons mondes :

- Il existe toujours un “bon” futur (AG EF  $\delta$ )
- Lorsqu'un monde est “bon” il le reste (AG ( $\delta \Rightarrow AG \delta$ ))

Pour exprimer l'obligation de faire  $p$  avant  $k$  unités de temps au sein de ce formalisme, on veut traduire la notion intuitive suivante :

*Dans tous les “bons” futurs (antérieurs à  $k$  unité de temps)  $p$  sera vrai dans moins de  $k$  unités de temps*

Autrement dit, pour tout chemin partant de l'état courant qui devient “bon”,  $p$  doit être vrai au moins une fois, entre le moment où il devient “bon”, et l'échéance fixée depuis l'état courant. Cela revient à dire, dans la sémantique que nous avons choisie :

$$\begin{aligned} \mathcal{M} \models_s O_{\leq k}(A) \quad \text{ssi} \quad \forall \sigma \text{ tel que } \sigma_0 = s, \forall i \leq k \\ \text{si } \mathcal{M} \models_{\sigma_i} \delta \quad \text{et} \quad \forall i' 0 \leq i' < i \Rightarrow \mathcal{M} \not\models_{\sigma_{i'}} \delta \\ \text{alors } \exists j \text{ tel que } i \leq j \leq k \text{ et } \mathcal{M} \models_{\sigma_j} A \end{aligned}$$

La figure 5.1 illustre cette notion. L'état initial vérifie  $O_{\leq 3}(p)$ . En effet, pour tout chemin,  $p$  devient vrai entre le premier “ bon” état rencontré et l'échéance, du moins s'il existe un “bon” état avant trois unités de temps (c'est à dire trois transitions).

Cela peut se traduire par une formule de *RTCTL* que l'on définit récursivement de la manière suivante :

**Définition 31 (Obligation avec échéance explicite)**

$$\begin{cases} O_{\leq k}(A) \stackrel{def}{=} (\delta \Rightarrow AF_{\leq k} A) \quad \wedge \quad (\neg \delta \Rightarrow AX O_{\leq k-1}(A)) \\ O_{\leq 0} A \stackrel{def}{=} \delta \Rightarrow A \end{cases}$$

Dans la partie 5.5, une étude des propriétés que l'opérateur d'obligation avec échéance devrait satisfaire, d'après l'intuition, sera faite en détail.

## 5.3 Modélisation avec une relation déontique

Nous allons utiliser une logique temporisée avec temps discret comme dans 3.2.1. Nous ajoutons au langage *RTCTL* un opérateur déontique  $O$ , et à la sémantique une relation déontique  $R_o$  qui associe chaque état à ses “idéaux”.

**Définition 32 (La syntaxe)**

*On dispose d'un ensemble de propositions atomiques Prop, de tous les opérateurs de RTCTL, et de l'opérateur déontique O.*

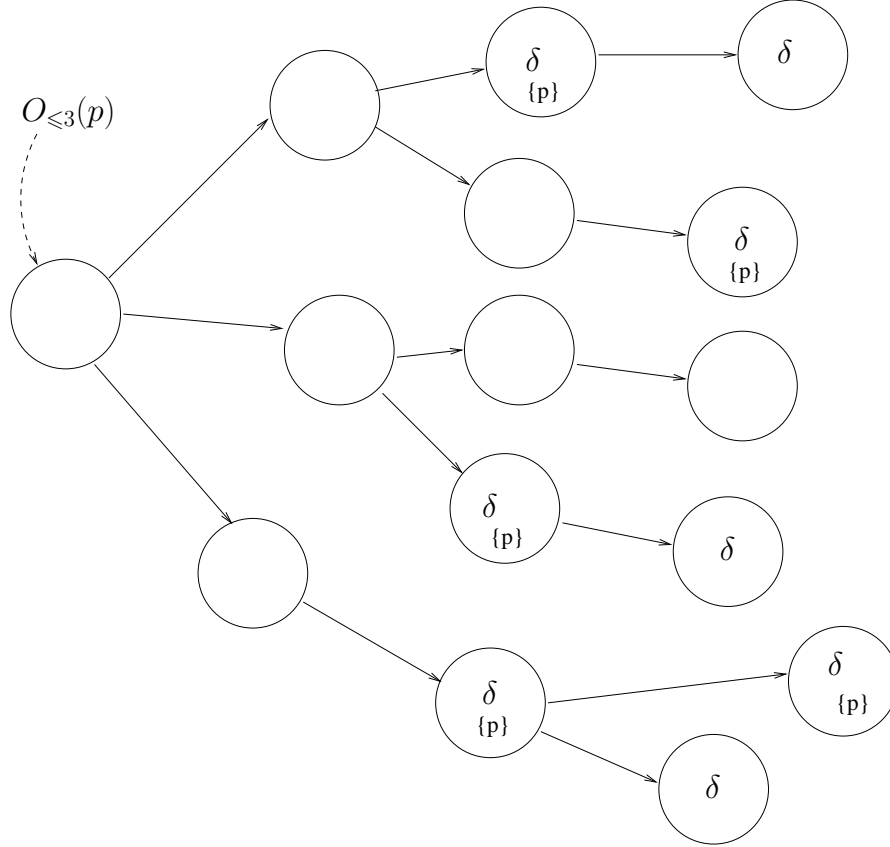


FIG. 5.1 – Obligation avec échéance explicite

**Définition 33 (Modèle)** Un modèle dans notre nouveau formalisme (*RTCTL* + relation déontique) est un quintuplet  $\mathcal{M} = (W, R, \Pi, I, \tau)$  où :

- $W$  est l'ensemble des états, ou mondes
- $R$  est une relation sur  $W \times W$  associant chaque monde à un ensemble de “bons” mondes
- $\Pi$  est la fonction associant à chaque état les propositions atomiques qu'il satisfait
- $\tau$  est une relation sur  $W \times W$  qui marque les transitions temporelles entre les états
- $I \subseteq W$  est l'ensemble des états initiaux

La sémantique des opérateurs est celle de *RTCTL* pour les opérateurs temporels, et celle de *SDL* pour les opérateurs déontiques.

On peut donc exprimer le fait que  $p$  soit vrai dans moins de  $k$  unités de temps (comme dans *RTCTL*) :

$$AF_{\leq k} p$$

L'obligation avec échéance se déduit donc directement de cette formule. En effet, on veut exprimer le fait qu'on a l'obligation que  $p$  soit vrai avant  $k$  unités de temps, c'est à dire  $O(AF_{\leq k}(p))$ .

**Définition 34 (Obligation avec échéance explicite)**

On définit l'opérateur d'obligation avec échéance explicite de la manière suivante :

$$O_{\leq k}(A) \stackrel{def}{=} O(AF_{\leq k} A)$$

Les propriétés de l'opérateur déontique - notamment son comportement vis à vis de formules temporelles - dépend des contraintes que nous fixons sur la relation déontique  $R$ . L'étude de ces propriétés est essentielle puisque la conformité avec l'intuition n'est à ce stade absolument pas garantie. Elle sera faite en détail dans les parties suivantes (5.4,5.5).

## 5.4 Analogie entre la relation déontique et une relation de raffinement

Nous nous intéressons dans cette partie à la relation déontique  $R$ , qui constitue le cœur de la représentation de l'obligation dans la sémantique des formalismes vus jusqu'ici. Nous avons fait le rapprochement entre cette relation et une relation de raffinement pour des systèmes de transition.

Dans un premier temps, nous nous intéresserons juste à un modèle de  $SDL$ , sans transition temporelle, puis nous étendrons l'étude au formalisme "temporel + déontique", vu précédemment (cf 5.3).

### 5.4.1 Dans $SDL$

Au niveau sémantique, l'obligation est représentée dans  $SDL$  à travers des mondes dits "bons". À chaque monde "quelconque" on associe des alternatives idéales. Une relation est mise en évidence entre mondes quelconques et bons mondes.

Celle-ci met donc en évidence la distinction entre deux niveaux :

- les mondes "quelconques" :  
On vérifie directement sur ces mondes n'importe quelle propriété.
- les mondes "idéaux" ou "bons"  
On vérifie que ces mondes satisfont une propriété uniquement lorsqu'il faut déterminer si le monde quelconque associé satisfait l'obligation de cette propriété.

La vérification qu'une obligation n'est pas violée, c'est à dire la vérification d'une propriété du type " $O(A) \Rightarrow A$ " revient à vérifier certaines propriétés sur la relation  $R$  entre les mondes "bons" et "quelconques".

Plus précisément, si un modèle  $\mathcal{M}$  de  $SDL$  satisfait  $O(A) \Rightarrow A$  pour toute formule  $A$ , alors la relation  $R$  "conserve les propriétés" de  $SDL$  entre deux ensembles d'états.

#### Définition 35 (Satisfaction pour les modèles abstraits et concrets)

Soit un modèle  $\mathcal{M} = (W, \Pi, R)$  de  $SDL$ .

- Soit  $\mathcal{M}_a$  l'ensemble des "bons" mondes, ou l'ensemble des mondes abstraits.

$$\mathcal{M}_a \stackrel{def}{=} \{w' \in W \mid \exists w \in W \ wRw'\}$$

- Soit  $\mathcal{M}_c$  l'ensemble des mondes auxquels sont associés des "bons" mondes, appelés ici mondes concrets.

$$\mathcal{M}_c \stackrel{def}{=} \{w \in W \mid \exists w' \in W \ wRw'\}$$

On définit une nouvelle relation de satisfaction sur  $\mathcal{M}_a$  et  $\mathcal{M}_c$ , notée  $\models$  pour alléger la notation de la manière suivante :

$$\mathcal{M}_* \models A \quad \text{ssi} \quad \forall w \in \mathcal{M}_* \ \mathcal{M} \stackrel{w}{\models} A$$

où  $*$  =  $a$  ou  $c$ .

**Remarque 9** Si  $R$  est totale, ce qui est le cas dans  $SDL$ , alors  $W_c = W$ , c'est à dire que les mondes concrets correspondent à l'ensemble des mondes.

On a alors la propriété suivante :

#### Propriété 5 (Conservation des propriétés de $SDL$ )

$R$  "conserve" les propriétés  $SDL$  au sein d'un modèle  $\mathcal{M}$  si  $O(A) \Rightarrow A$  est satisfaite par  $\mathcal{M}$ , c'est à dire :

si

$$\mathcal{M} \models O(A) \Rightarrow A \tag{5.2}$$

alors

$$\mathcal{M}_a \models A \Rightarrow \mathcal{M}_c \models A \tag{5.3}$$

En effet,  $\mathcal{M} \models_w O(A) \Rightarrow A$  revient à :

$$(\forall w' wRw' \Rightarrow \mathcal{M} \models_{w'} A) \Rightarrow \mathcal{M} \models_w A$$

Donc, vérifier que  $\mathcal{M} \models O(A) \Rightarrow A$  revient à :

$$\forall w (\forall w' wRw' \Rightarrow \mathcal{M} \models_{w'} A) \Rightarrow \mathcal{M} \models_w A \quad (5.4)$$

Or (5.3) signifie

$$\forall w' (\exists w wRw' \Rightarrow \mathcal{M} \models_{w'} A) \quad (5.5)$$

$\Rightarrow$

$$(\forall w \mathcal{M} \models_w A) \quad (5.6)$$

c'est à dire "si tous les bons mondes satisfont  $A$ , alors tous les mondes satisfont  $A$ ".

Supposons (5.4).

Supposons de plus que "tous les bons mondes satisfont  $A$ " (5.5).

Soit  $w \in W$ .

Alors, d'après (5.5)  $\forall w' wRw' \Rightarrow \mathcal{M} \models_{w'} A$ .

Donc, d'après (5.4),  $\mathcal{M} \models_w A$ .

On a donc montré que si (5.4) alors ((5.5)  $\Rightarrow$  (5.6)).

C'est à dire si (5.2) alors (5.3).

On a donc montré que si l'obligation d'une propriété n'est pas violée, alors il y a conservation de cette propriété entre modèle abstrait et concret.

Donc, si toutes les obligations sont respectées, alors toutes les propriétés sont conservées entre modèle abstrait et modèle concret.

## 5.4.2 Dans un modèle de logique temporelle

Considérons maintenant une relation supplémentaire (transitions temporelles) dans notre modèle, pour représenter une évolution discrète du temps, et un ensemble d'états appelés états initiaux. Nous avons alors affaire à un modèle du formalisme de la partie 5.3 (version "temporelle + déontique").

On peut considérer le modèle - sans la relation  $R$  - comme un système de transitions, et se poser la question des propriétés qu'il faudrait imposer pour que  $R$  soit une relation de raffinement sur les deux systèmes de transition suivants :

- celui constitué des états abstraits (les "bons" mondes)
- celui constitué des états concrets (les mondes quelconques)

Dans le second système, toutes les transitions temporelles sont considérées ; dans le premier, seules celles dont les deux extrémités sont des "bons" mondes.

Rappelons tout d'abord ce qu'est une relation de raffinement sur deux systèmes de transitions.

### Définition 36 (Relation de raffinement et abstraction)

Soient  $S_c = (\Sigma_c, I_c, \tau_c)$  et  $S_a = (\Sigma_a, I_a, \tau_a)$  deux systèmes de transitions, et  $R$  une relation sur  $\Sigma_c \times \Sigma_a$ , dite de raffinement.  $S_c$  raffine  $S_a$  noté  $S_a \sqsubseteq_R S_c$  (ou  $S_a$  est une abstraction de  $S_c$ ) si

- $\forall \sigma_c \in I_c \exists \sigma_a \in I_a \sigma_c R \sigma_a$
- $\forall \sigma_c, \sigma'_c \in \Sigma_c \forall \sigma_a \in \Sigma_a \sigma_c R \sigma_a \wedge \tau_c(\sigma_c, \sigma'_c) \Rightarrow \exists \sigma'_a \in \Sigma_a \sigma'_c R \sigma'_a \wedge \tau_a(\sigma_a, \sigma'_a)$

$$\begin{array}{ccc} \sigma_c & \xrightarrow{\tau_c} & \sigma'_c \\ \uparrow R & & \uparrow R \\ \sigma_a & \xrightarrow{\tau_a} & \sigma'_a \end{array}$$

Définissons maintenant plus précisément les systèmes de transition abstrait et concret que nous allons utiliser.

**Définition 37** Soit  $\mathcal{M}$  un modèle précédemment défini. Nous appelons ici  $S_a = (W_a, I_a, \tau_a)$  le système de transitions abstrait, et  $S_c = (W_c, I_c, \tau_c)$  le système de transitions concret où :

- $W_a$  est l'ensemble des "bons" mondes (voir ci-dessus)
- $W_c = W$  est l'ensemble des mondes
- $I_c = I$
- $I_a = \{w' \in W_c / \exists w \in I \ wRw'\}$
- $\tau_a = \{(\sigma, \sigma') \in \tau / \sigma \in W_a \wedge \sigma' \in W_a\}$
- $\tau_c = \tau$

Le choix de  $I_a$  est fait de manière à ce que la première des deux propriétés d'une relation de raffinement soit toujours satisfaite pour  $R$ . (L'existence d'états dans  $I_c$  est garantie si  $R$  est totale.)

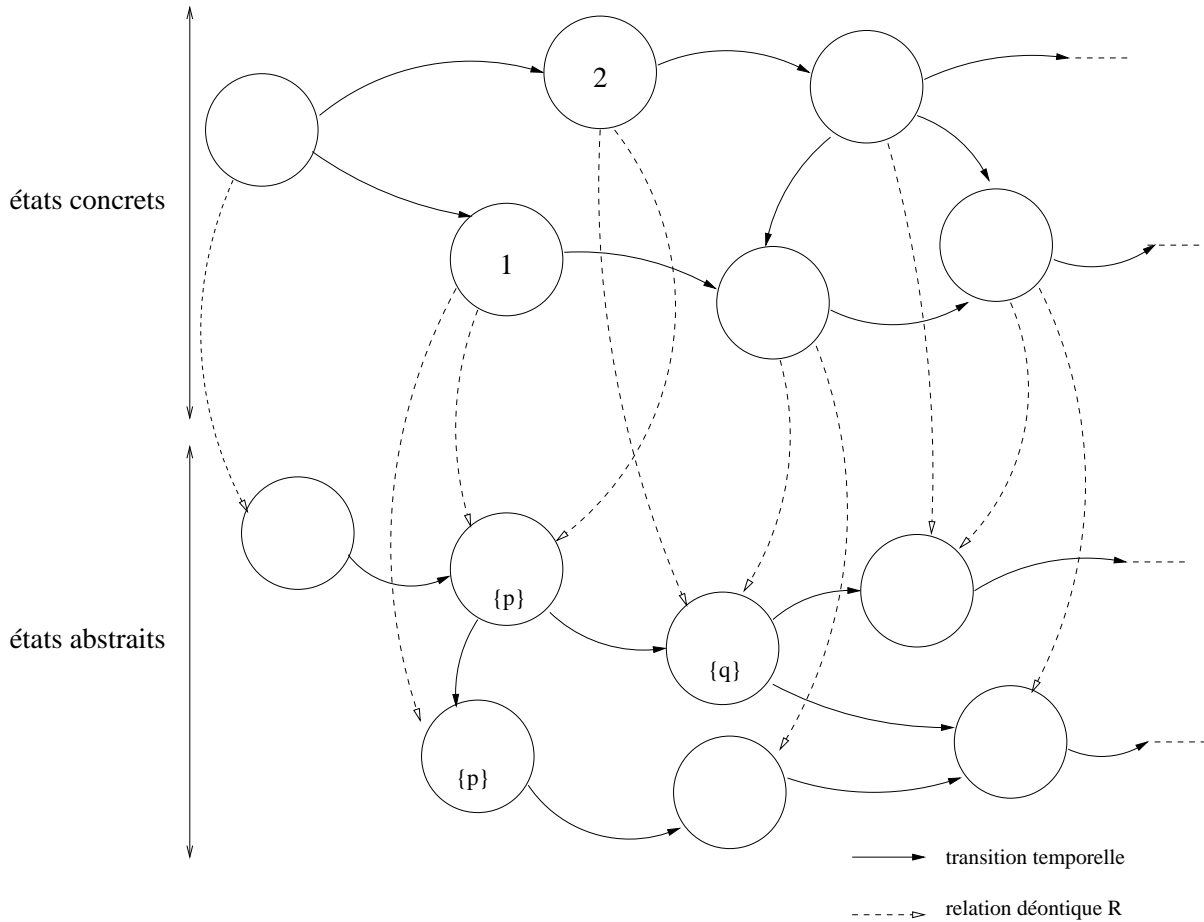


FIG. 5.2 – Exemple de modèle déontique/temporel

La figure 5.2 représente un exemple de modèle sur lequel sont présentes la relation déontique  $R$ , et les transitions temporelles. Par souci de clarté, on ne représente pas les transitions temporelles entre les états concrets et les états abstraits, ni les idéaux déontiques associés à des états déjà "bons".

On remarque sur cet exemple que l'état 1 satisfait  $O(p)$  puisque tous les "bons" états qui lui sont associés satisfont  $p$ . De même, l'état 2 satisfait  $P(p)$  et  $P(q)$ , puisqu'un de ses "bons" états satisfait  $p$ , et un autre  $q$ .

Supposons que  $S_a \sqsubseteq_R S_c$ . On a alors

$$\forall \sigma_c, \sigma'_c \in W_c \quad \forall \sigma_a \in W_a \quad \sigma_c R \sigma_a \wedge \tau_c(\sigma_c, \sigma'_c) \Rightarrow \exists \sigma'_a \in W_a \quad \sigma'_c R \sigma'_a \wedge \tau_a(\sigma_a, \sigma'_a) \quad (5.7)$$

Intuitivement, étant donné deux états concrets  $\sigma_c$  et  $\sigma'_c$  ( $\sigma'_c$  étant le successeur de  $\sigma_c$ ) et un état abstrait  $\sigma_a$  en relation déontique avec  $\sigma_c$ , on impose l'existence d'un successeur  $\sigma'_a$  de  $\sigma_a$ , en relation déontique avec  $\sigma'_c$ .

Cette propriété établit un lien entre la relation déontique  $R$  et la relation de transition temporelle  $\tau$ . On obtient par exemple les propriétés suivantes, qui relient la permission et les opérateurs temporels.

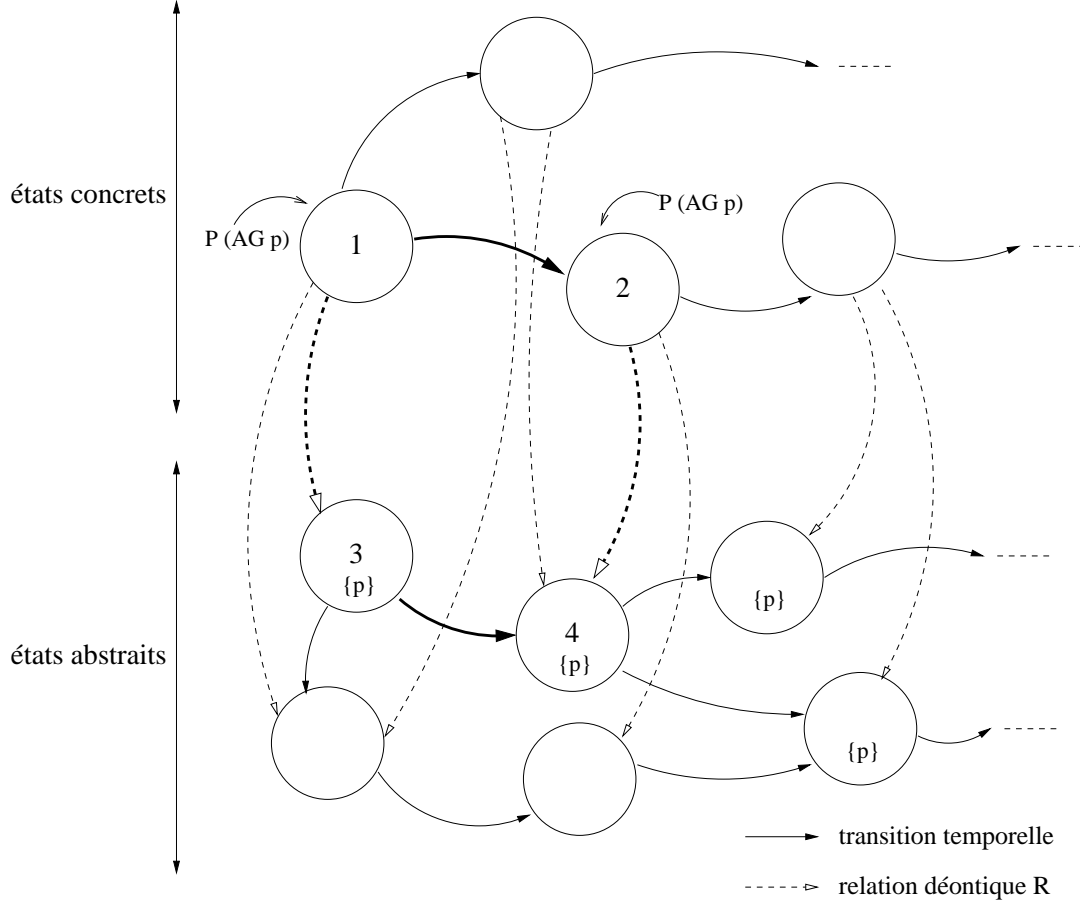


FIG. 5.3 – Raffinement et lien déontique/temporel

- si  $\mathcal{M} \models_w P(AG A)$  alors  $\mathcal{M} \models_w AX P(AG A)$  et donc  $\mathcal{M} \models_w AG P(AG A)$
- si  $\mathcal{M} \models_w P(AF A)$  alors  $\mathcal{M} \models_w P(A) \vee EX P(AF A)$

La première propriété est illustrée avec la figure 5.3. L'état concret 1 satisfait  $P(AG p)$  puisque un de ses "bons" états, l'état 3, satisfait  $AG p$ . L'état 2 étant le successeur de 1, la propriété (5.7) impose qu'il existe un état 4, successeur de 3, et en relation déontique avec 2. Or, si 3 satisfait  $AG p$ , alors 4 aussi, en tant que successeur. Donc 2 admet un "bon" état qui satisfait  $AG p$ .

Ce petit exemple illustre pourquoi  $P(AG p) \Rightarrow AX P(AG p)$ .

Si nous voulons une propriété analogue concernant l'obligation plutôt que la permission, il faut imposer une propriété très proche à la relation déontique  $R$ . Étant donné deux états concrets  $\sigma_c$  et  $\sigma'_c$  ( $\sigma'_c$  étant le successeur de  $\sigma_c$ ) et un état abstrait  $\sigma'_a$  en relation déontique avec  $\sigma'_c$ , on impose l'existence d'un prédécesseur  $\sigma_a$  de  $\sigma'_a$ , en relation déontique avec  $\sigma_c$ . C'est à dire

$$\forall \sigma_c, \sigma'_c \in W_c, \forall \sigma'_a \in W_a (\tau_c(\sigma_c, \sigma'_c) \wedge \sigma'_c R \sigma'_a) \Rightarrow \exists \sigma_a \in W_a \tau_a(\sigma_a, \sigma'_a) \wedge \sigma_c R \sigma_a \quad (5.8)$$

Nous avons alors, avec la propriété 5.8

- si  $\mathcal{M} \models_w O(AG A)$  alors  $\mathcal{M} \models_w AX O(AG A)$  et donc  $\mathcal{M} \models_w AG O(AG A)$
- $\mathcal{M} \models (O_{\leq k}(A) \wedge I(A)) \Rightarrow AX O_{\leq k-1}(A)$

La première propriété indique que si on est obligé à un moment donné d'avoir toujours  $A$ , alors l'obligation persiste dans le futur.

La seconde exprime que si aujourd'hui je suis obligé d'avoir  $A$  avant trois jours, alors demain je serai obligé d'avoir  $A$  avant deux jours, à condition d'avoir l'interdiction de  $A$  aujourd'hui. En effet, rappelons que si l'état courant satisfait  $O_{\leq 3 \text{ jours}} A$  alors tous ses idéaux satisfont  $AF_{\leq 3 \text{ jours}} A$ . Or, si un de ceux-ci satisfait  $A$  à l'instant présent, alors il n'est pas garanti qu'il satisfasse  $A$  plus tard, et il se peut donc que l'obligation ne tienne pas dans le futur.

## 5.5 Quelques propriétés des opérateurs d'obligation avec échéance explicite

Nous pouvons vérifier si un certain nombre de propriétés attendues d'après l'interprétation intuitive de l'obligation avec échéance sont vérifiées par nos deux formalismes : celui avec "marqueur" et celui avec une relation "déontique".

Une des premières concerne la cohérence de la notion temporelle d'échéance : *Si on a l'obligation de partir avant 10 heures, a-t-on aussi l'obligation de partir avant 11 heures ?*

$$\models O_{\leq k_1}(A) \Rightarrow O_{\leq k_2}(A) \quad (\text{avec } k_1 \leq k_2)$$

Dans les deux cas la réponse est oui.

Posons-nous maintenant la question du maintien de l'obligation jusqu' à l'échéance. *Si j'ai aujourd'hui l'obligation de laver mon linge avant trois jours, serai-je toujours obligé de laver mon linge demain ?* Dans le formalisme avec "marqueur", la réponse est non. C'est à dire

$$\not\models O_{\leq k}(A) \Rightarrow AX O_{\leq k-1}(A)$$

En fait, on a (si la propriété "un bon monde restera toujours bon" est vraie)

$$\models (O_{\leq k}(A) \wedge \neg A) \Rightarrow AX O_{\leq k-1}(A)$$

L'obligation peut ne plus tenir demain si j'ai déjà accompli mon devoir. Dans l'exemple, si je lave mon linge aujourd'hui, l'obligation ne reste pas pour les jours à venir.

Par contre, dans la version avec relation déontique, si aucun lien n'existe entre les deux relations, les obligations dans l'état présent et dans les états futurs sont décorélées. Cependant, nous avons vu dans la partie 5.4 que l'équation 5.8 donne lieu à des propriétés intéressantes.

Une autre propriété concerne la conjonctivité de  $O_{\leq d}$ . En effet, pour l'obligation classique, elle est vérifiée par tous les systèmes. Or, ici dans les deux systèmes proposés :

$$\not\models O_{\leq k}(A \wedge B) \Leftrightarrow O_{\leq k}(A) \wedge O_{\leq k}(B)$$

Plus précisément,

$$\models O_{\leq k}(A \wedge B) \Rightarrow O_{\leq k}(A) \wedge O_{\leq k}(B)$$

mais

$$\not\models O_{\leq k}(A) \wedge O_{\leq k}(B) \Rightarrow O_{\leq k}(A \wedge B)$$

Mais ceci est en fait tout à fait conforme à l'intuition. En effet,  $O_{\leq k}(A \wedge B)$  impose que  $A$  et  $B$  soient vrais en même temps, à un certain instant du futur, avant  $k$  unités de temps. C'est une propriété plus forte que d'imposer  $O_{\leq k}(A)$  d'une part, et  $O_{\leq k}(B)$  d'autre part.

Qu'en est-il de la violation d'une obligation ?

$$O_{\leq k}(A) \wedge AG_{\leq k} \neg A$$

Dans un modèle avec “marqueur”, les “bon” mondes font partie du futur de l’état courant. Donc si  $O_{\leq k}(A)$  est satisfait dans un état, alors  $A$  est satisfait au moins dans un futur de cet état (avant  $k$  unités de temps), à condition qu’il existe un “bon” état avant  $k$  unités de temps. On a donc  $\models EF_{\leq k}A$  (s’il existe un “bon” état futur avant  $k$  unités de temps), mais pas  $AF_{\leq k}A$ . Donc, dans cette version,

$$O_{\leq k}(A) \wedge AG_{\leq k}\neg A \text{ est satisfiable et non valide}$$

et

$$\models EF_{\leq k}\delta \Rightarrow (O_{\leq k}(A) \Rightarrow EF_{\leq k}A)$$

C’est à dire “Si on a l’obligation que  $A$  soit vrai avant  $k$  unités de temps, alors  $A$  peut arriver avant  $k$  unités de temps, à condition qu’il existe des bons états dans un futur antérieur à  $k$  unités de temps.”

Dans un modèle avec relation déontique, le fait qu’un état vérifie une obligation avec échéance n’a pas de conséquence sur les formules sans obligation qu’il satisfait. Un état peut donc satisfaire à la fois  $O_{\leq k}(A)$  et  $AG_{\leq k}\neg A$ .

## 5.6 Expression de politiques et propriétés de disponibilité

On va supposer dans cette partie que l’on dispose de l’opérateur d’obligation avec échéance explicite ( $O_{\leq k}(A)$ ). Le but est d’exprimer une politique ainsi que des propriétés de disponibilité.

Nous disposons de  $n$  processus,  $m$  tâches, et  $N$  ressources. Nous nous référons toujours à la définition d’une politique de disponibilité de [22]. Cela passe par les quatre prédicats  $droit\_utiliser(s, r, d)$ ,  $droit\_disposer(s, r, d)$ ,  $droit\_realisation(p, t, d)$ , et  $obligation\_duree\_max(p, t, d)$  (cf partie 1). Chacun de ces prédicats a une double signification : une obligation et une permission. La figure 5.4 rappelle

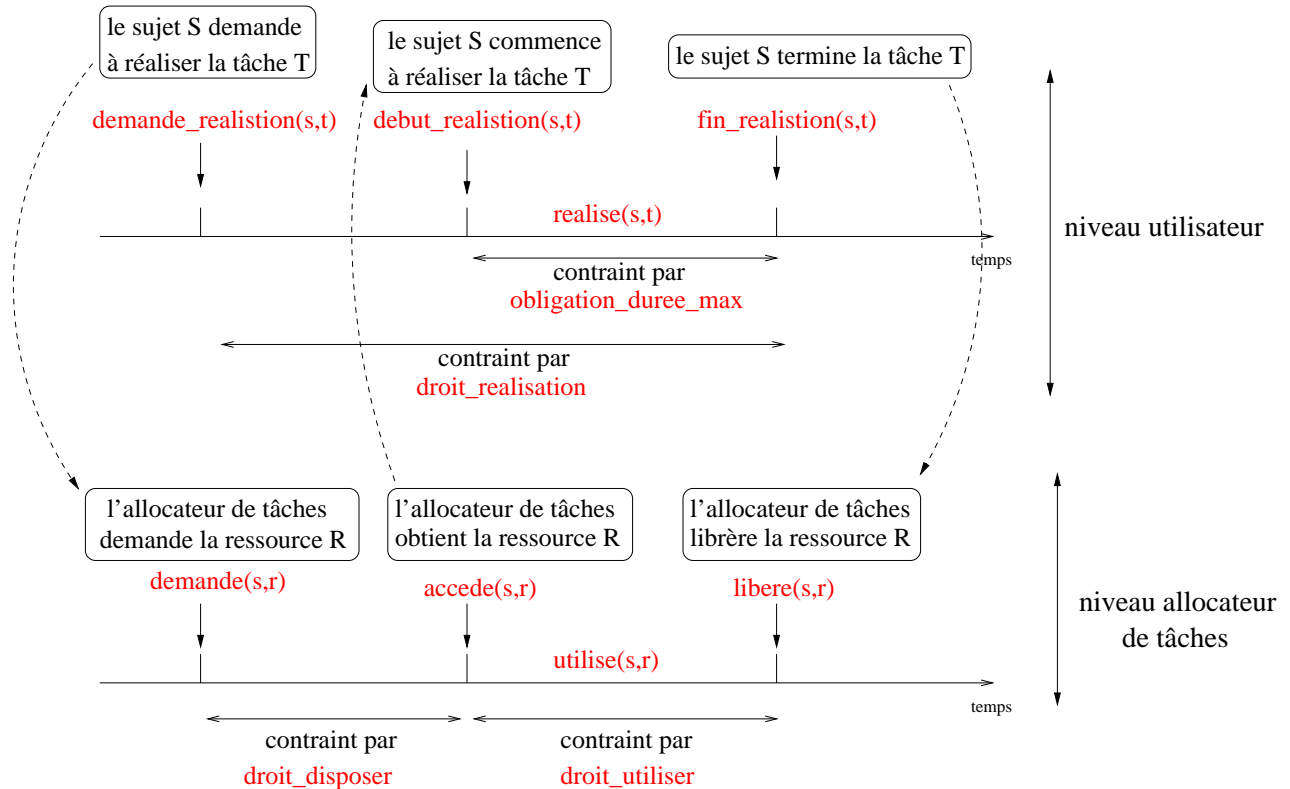


FIG. 5.4 – Contraintes de réalisation de tâches et d’accès aux ressources

le rôle des différents prédicats utilisés dans [22] pour décrire le comportement du système, et spécifier la politique de disponibilité.

Voici la traduction des obligations induites par chaque prédicat, dans notre formalisme.

$$\begin{aligned}
\text{droit\_utiliser}(s, r, d) &\equiv AG(\text{accede}(s, r) \Rightarrow O_{\leq d}(\neg \text{utilise}(s, r))) \\
\text{droit\_disposer}(s, r, d) &\equiv AG(\text{demande}(s, r, t) \Rightarrow O_{\leq d} \text{utilise}(s, r)) \\
\text{droit\_realisation}(s, t, d) &\equiv AG(\text{demande\_realisation}(s, t) \Rightarrow O_{\leq d} \text{fin\_realisation}(s, t)) \\
\text{obligation\_duree\_max}(p, t, d) &\equiv AG(\text{debut\_realisation}(s, t) \Rightarrow O_{\leq d} \text{fin\_realisation}(s, t))
\end{aligned}$$

La définition des permissions est plus complexe. En effet, des questions peuvent se poser sur la durée pendant laquelle une permission se maintient. Si le prédicat  $\text{droit\_utiliser}(s, r, d)$ , par exemple, est vrai à l'instant  $T$ , alors le processus  $s$  a la permission d'utiliser la ressource  $r$ , au même instant  $T$ . Mais qu'en est-il des instants suivants? Si on ne répond pas à cette question, il faut définir les permissions dans le temps, indépendamment des faits et des obligations. On s'expose alors à des incohérences dont il faudra vérifier l'absence ultérieurement.

Pour donner la permission à un processus d'utiliser une ressource (ce qui est fait dans [22] à travers  $\text{droit\_utiliser}$  et  $\text{droit\_disposer}$ ) on voudrait donner la permission à un processus jusqu'à ce qu'il libère la ressource, ou que le délai  $d$  fixé par  $\text{droit\_utiliser}$  soit dépassé. On aurait donc des formules de la forme

$$z.A(P \text{ utilise}(s, r) \mathcal{U} (\text{libere}(s, r) \vee z > d))$$

avec la syntaxe temporelle de  $TCTL$ .

Pour donner la permission à un processus de réaliser une tâche, (ce qui est fait dans [22] à travers  $\text{droit\_realisation}$  et  $\text{obligation\_duree\_max}$ ), il faut de la même manière définir temporellement les permissions ( $P \text{ realise}(s, t)$ ).

**Remarque 10** *Les prédicats  $\text{fin\_realisation}$ ,  $\text{debut\_realisation}$ ,  $\text{accede}$ , et  $\text{libere}$ , peuvent être évités si l'on dispose des opérateurs temporels  $\circ$  ("dans l'état suivant") et/ou  $\ominus$  ("dans l'état précédent"). En effet  $\text{fin\_realisation}$ , par exemple, peut être vu comme  $\ominus \text{realise} \wedge \neg \text{realise}$ , ou comme  $\text{realise} \wedge \neg \circ \text{realise}$ .*

*On peut donc décrire le comportement du système et une politique de disponibilité à l'aide des prédicats de base suivants :  $\text{demande}(s, t)$ ,  $\text{utilise}(s, r)$ ,  $\text{demande\_realisation}$ ,  $\text{realise}(s, t)$ .*

## 5.7 Vérification de la disponibilité

La problématique, vu dans la partie 1, entraîne plusieurs types de vérifications :

- Conformité d'un système vis à vis d'une politique de disponibilité
- Cohérence interne de la politique de disponibilité
- Satisfaction d'une propriété de disponibilité par un système conforme à une politique donnée

### 5.7.1 Conformité d'un système vis à vis d'une politique

Avec la spécification d'une politique vue plus haut, la vérification de la conformité (ou non-violation) d'un système vis à vis d'une politique devient claire. Il s'agit de vérifier les obligations qui sont explicitées dans la définition de la politique, et que les faits sont permis par la politique..

Cela simplifie grandement la solution présentée dans [22]. Pour vérifier les obligations, une première méthode consiste à transformer les différentes formules constituant la politique, en supprimant les modalités déontiques. Par exemple, si la politique comporte

$$AG(\text{accede}(s_1, r) \Rightarrow O_{\leq d}(\neg \text{utilise}(s_1, r)))$$

il faudra vérifier

$$AG(\text{accede}(s_1, r) \Rightarrow AF_{\leq d}(\neg \text{utilise}(s_1, r)))$$

**Remarque 11** *En fait, on a vu dans 3.2.2 que vérifier  $AG AF_{\leq d} \neg \text{utilise}(s_1, r)$  suffit.*

De même, pour vérifier que l'obligation induite par le prédicat  $\text{droit\_disposer}$  est satisfaite, il faut vérifier la formule  $AG(\text{demande}(s_1, r) \Rightarrow O_{\leq d} \text{accede}(s_1, r))$ .

En fait, si  $Pol$  représente la conjonction des formules présentes pour définir une politique de disponibilité, et  $Sys$  une modélisation décrivant un algorithme d'allocation de ressources, alors vérifier que

$Sys$  est conforme à  $Pol$  revient à vérifier que les obligations présentes dans  $Pol$  ne sont pas violées par  $Sys$  d'une part, et que les faits présents dans  $Sys$  sont autorisés dans  $Pol$  d'autre part. C'est à dire que pour tout prédicat quelconque  $p$ , si  $O(p)$  est présent dans  $Pol$ , alors  $p$  doit être présent dans  $Sys$ , et, de même, si  $p$  apparaît dans  $Sys$ , il faut vérifier que  $P(p)$  apparaît dans  $Pol$ .

L'allocation de ressources doit être représenté formellement pour pouvoir vérifier des formules. Il peut être représenté par un système de transitions ou une formule de logique temporelle. Dans les deux cas, on le nommera  $Sys$ . On distingue deux méthodes :

- Suppression des opérateurs déontiques avant la vérification

On transforme alors les formules de la politique contenant des obligations par des formules temporelles, appelées contraintes de sécurité. Ainsi  $AG(accede(s_1, r) \Rightarrow AF_{\leq d}(\neg utilise(s_1, r)))$  et  $AG(demande(s_1, r) \Rightarrow AF_{\leq d} accede(s_1, r))$  sont des contraintes de sécurité, correspondant aux prédicats *droit\_utiliser* et *droit\_disposer*. Il faut alors vérifier

$$Sys \models Cont$$

où  $Cont$  désigne la conjonction des contraintes de sécurités, c'est à dire les formules de la politique correspondant aux obligations, au sein desquelles on remplace les opérateurs déontiques avec échéance par des opérateurs temporels.

**Remarque 12** *Seulement les obligations présentes dans la politique sont alors vérifiées. On ne peut pas vérifier que tous les prédicats de base qui sont vrais dans la modélisation du système, sont aussi permis dans la politique.*

- Conservation des obligations et permissions :

On peut alors utiliser des formules comportant des modalités déontiques en utilisant les formalismes proposés plus haut. Les obligations présentes dans une politique de disponibilité sont des obligations avec échéance. Vérifier que  $O_{\leq d}(p)$  n'est pas violé revient naturellement à vérifier que  $AF_{\leq d}(p)$  se déduit de  $Sys$ . Il s'agit donc de vérifier toutes les formules de la forme

$$((Sys \wedge Pol) \Rightarrow O_{\leq d}(p)) \Rightarrow (Sys \Rightarrow AF_{\leq d} p)$$

et

$$(Sys \Rightarrow p) \Rightarrow ((Sys \wedge Pol) \Rightarrow P(p))$$

pour  $p$  prédicat de base.

Une autre approche amène à vérifier que des formules du type  $O(p) \Rightarrow p$ ,  $I(p) \Rightarrow \neg p$ , et  $p \Rightarrow P(p)$  se déduisent de la conjonction de  $Sys$  et  $Pol$ . On obtient alors des formules du type

$$(Sys \wedge Pol) \Rightarrow (O_{\leq d} p \Rightarrow AF_{\leq d} p)$$

$$(Sys \wedge Pol) \Rightarrow (p \Rightarrow P(p))$$

**Remarque 13** *Les deux formules concernant la permission (première et deuxième approche) sont équivalentes.*

Si on dispose d'une description du système et de la politique sous forme de modèle, les deux formules à vérifier deviennent :

$$(Sys \wedge Pol) \models (O_{\leq d} p \Rightarrow AF_{\leq d} p)$$

$$(Sys \wedge Pol) \models (p \Rightarrow P(p))$$

## 5.7.2 Cohérence interne d'une politique

Dans [22], sept axiomes sont fixés, permettant de vérifier partiellement la cohérence d'une politique. De part la traduction dans notre formalisme, certains deviennent implicites. Il reste :

- Si un processus a la permission de réaliser une tâche  $t$ , alors il doit avoir la permission d'utiliser les ressources nécessaires à la réalisation de  $t$ .

- Si un processus a l'obligation de réaliser une tâche  $t$  au bout d'une durée de  $d$  après l'avoir demandée, alors  $t$  doit avoir une durée inférieure à  $d$ .
- Si un processus  $p$  a la permission de réaliser une tâche, et que celle-ci se décompose en sous-tâches, alors  $p$  doit avoir la permission de réaliser les sous-tâches.

Il se traduisent tous directement dans notre formalisme à partir des prédicats *demande\_realisation*, *realise*, *utilise*, et *sous\_tache* qui indique la décomposition d'une tâche en sous-tâches.

D'autre part, vérifier la satisfiabilité de la conjonction des formules définissant la politique permet d'assurer la cohérence entre toutes les obligations et les permissions, puisque les deuxièmes sont définies à partir des premières.

### 5.7.3 Propriété de disponibilité

Pour vérifier une propriété de disponibilité isolée  $p$ , dans le cas où le système est conforme à une politique, il faut démontrer que les contraintes de sécurité (cf 5.7.1) impliquent  $p$ . Si ce n'est pas le cas, on ne peut pas conclure, et il faut essayer de démontrer directement que le système *Sys* satisfait la propriété  $p$ .

# Conclusion et perspectives

Nous avons donc, après avoir présenté l'état de l'art en logique déontique, proposé des modèles et des sémantiques permettant d'exprimer à la fois l'obligation, les principales notions temporelles ou temporisées, et surtout l'obligation avec échéance. Deux types de propositions ont été mises en évidence :

- une modélisation comportant une relation déontique indépendante de la relation temporelle
- une modélisation ne comportant qu'une relation temporelle et une proposition "marqueur" pour modéliser les "bons" mondes

Celles-ci permettent d'exprimer les notions nécessaires pour définir un politique de disponibilité, au sens de Saurel et Cuppens dans [22]. Nous avons ensuite vu quels types de formules vérifier, dans ces formalismes, pour assurer la disponibilité d'un système.

L'intérêt de l'utilisation de logiques déontique et temporelle paraît clair pour spécifier une politique ou une propriété de disponibilité. Se doter d'une sémantique aussi proche que possible de l'intuition que l'on se fait des opérateurs déontiques et temporels est alors indispensable. En revanche, l'aspect vérification n'est pas satisfaisant. Elle nécessite en effet

- soit la transformation des formules déontiques en formules temporelles : on perd la vérification des permissions, et l'automatisation n'est pas garantie
- soit la construction d'un modèle dans lequel on représenterait à la fois la politique et le système, qui n'est pas encore au point.

## Perspectives

Nous allons mettre au point un formalisme analogue à celui développé dans 5.3, mais basé sur *TCTL*, permettant de mêler un temps continu à la notion d'obligation.

Une étude de cas sera faite pour permettre de comparer concrètement les méthodes basées sur la logique des prédicats, développées dans [22], à des méthodes basées sur des formalismes déontiques et temporels. L'objectif est de trouver des moyens concrets de valider la disponibilité. Il s'agit de trouver des procédures de décision pour un formalisme combinant les notions temporelle et déontique, ou d'automatiser la transformation en logique temporelle, des formules déontiques qui définissent une politique.

Toutes les obligations, tous les faits qui décrivent le système ont ici un caractère universel. Ils ne permettent pas de distinguer la partie utilisateur de la partie système. Il serait intéressant de développer la compositionnalité qui est sous-jacente à notre problème. Des logiques déontiques utilisant des opérateurs indexés par des agents (cf 4.3.2) existent. Ainsi  $O_a(p)$  signifie que l'agent  $a$  doit faire  $p$ . Il serait intéressant d'utiliser cet aspect pour la disponibilité.

Enfin, les logiques d'action (cf 2.2,4.3.1) permettent d'exprimer des obligations sur des événements ponctuels, ce qui correspond notamment aux prédicats *accède* ou *libère* que nous utilisons pour décrire le système, et paraît donc intéressant pour notre application.

# Bibliographie

- [1] R. Alur. *Techniques for automatic verification of real-time systems*. PhD thesis, Stanford University, 1991.
- [2] R. Alur, C. Courcoubetis, and D. Dill. Model-checking for real-time systems. In *Proc. 5th LICS*, pages 414–425. IEEE Computer Society Press, 1990.
- [3] R. Alur and D. Dill. The theory of timed automata. *Theoretical Computer Science*, 1994.
- [4] Alan Ross Anderson and Omar K. Moore. The formal analysis of normative concepts. *The American Sociological Review*, pages 9–17, 1957.
- [5] L. Aqvist. Some results on dyadic deontic logic and the logic of preference. *Synthese*, 66 :95–110, 1986.
- [6] P. Bailhache. The deontic branching time : two related conceptions. *Logique et Analyse*, 141-142 :159–175, 1993.
- [7] P. Bailhache. Canonical models for temporal deontic logic. *Logique et Analyse*, pages 3–21, 1995.
- [8] B. Bérard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, and Ph. Schnoebelen. *Systems and Software Verification*. Springer, 1999.
- [9] P. d’Altan, J. Ch Meyer, and M. Wieringa. An integrated framework for ought–to–be and ought–to–do constraints, 1998. <http://citeseer.ist.psu.edu/daltan98integrated.html>.
- [10] F. Dignum and R. Kuiper. Combining dynamic deontic logic and temporal logic for the specification of deadlines. In *30th Hawaii International Conference on System Sciences (HICSS) Volume 5 : Advanced Technology Track*, 1997.
- [11] F. Dignum and R. Kuiper. Obligations and dense time for specifying deadlines. In *Thirty-First Annual Hawaii International Conference on System Sciences (HICSS)-Volume 5*, 1998. "<http://csdl.computer.org/comp/proceedings/hicss/1998/8245/00/82450186abs.htm>".
- [12] J. A. Van Eck. A system of temporally relative modal and deontic predicate logic and its philosophical applications. Dissertation, Department of Philosophy, University Groningen, 1981.
- [13] J. A. Van Eck. A system of temporally relative modal and deontic predicate logic and its philosophical applications. *Logique et Analyse*, 25 :249–290, 339–381, 1982.
- [14] E. A. Emerson, A. K. Mok, A. P. Sistla, and J. Srinivasan. Quantitative temporal reasoning (extended abstract). In E. M. Clarke and R. P. Kurshan, editors, *Computer-Aided Verification : Proc. of the 2nd International Conference CAV’90*, pages 136–145. Springer, Berlin, Heidelberg, 1991.
- [15] Jorg Hansen. On relation between aqvist’s deontic system g and van eck’s deontic temporal logic. workshop DEON’98, 1998. "<http://www.hh.shuttle.de/win/Joerg.Hansen/Deontic.html>".
- [16] B. Hansson. An analysis of some deontic logics. *R. Hilpinen (ed.) : Deontic Logic : Introductory and Systematic Readings. Dordrecht, Holland : D. Reidel Publishing Company*, pages 121–147, 1971.
- [17] T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model-checking for real-time systems. In *Proc. 7th LICS*, pages 394–406. Computer Society Press, 1992.
- [18] J. Broersen, F. Dignum, V. Dignum, and J-J Ch Meyer. Designing a deontic logic of deadlines. In *7th International Workshop on Deontic Logic in Computer Science (DEON’04)*, Madeira, Portugal, 26-28 May 2004.
- [19] J-J Ch Meyer. A different approach to deontic logic : Deontic logic viewed as a variant of dynamic logic. *Notre Dame Journal of Formal Logic*, 1988.

- [20] J-J Ch Meyer. *Semantics and Contextual Expression*, pages 117–145. FORIS publications, Dordrecht/Riverton, 1989.
- [21] John-Jules Ch. Meyer, Roel Wieringa, and Frank Dignum. The role of deontic logic in the specification of information systems. In *Logics for Databases and Information Systems*, pages 71–115, 1998.
- [22] Claire Saurel and Frédéric Cuppens. Modélisation du concept de disponibilité. Technical report, ONERA Département Traitement de l' Information et Modélisation, 1999.
- [23] M. Turiani. Logique temporele temporisée pour la vérification des programmes :expressivité et complexité. Master's thesis, ENS Cachan, Laboratoire Spécification et Vérification, 1999.
- [24] G. H. Von Wright. Deontic logic. *Mind*, 1951.
- [25] G. H. Von Wright. A note on deontic logic and derived obligation. *Mind*, 1956.
- [26] S. Yovine. *Méthodes et Outils pour la Vérification Symbolique de Systèmes Temporisés*. PhD thesis, Institut National Polytechnique de Grenoble, 1993.