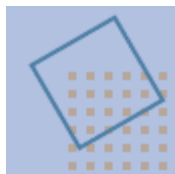


UM2



LIRMM



***Mémoire de stage de Master :***  
**Spécialité Recherche en Informatique**  
**Mention Informatique, Mathématiques, Statistiques**

# **Ordonnancement de tâches phytosanitaires viticoles**

Version révisée  
travaux soutenus le 20 juin 2006

Tu Tuitete

Tuteur de stage : Olivier Naud  
Tuteur LIRMM : Rodolphe Giroudeau

**Département Écotechnologies et Agrosystèmes**

Unité Mixte de Recherche ITAP  
Information et Technologies pour les Agro-procédés

Domaine de Lavalette  
361, Jean François Breton ; B.P. 5095  
34196 MONTPELLIER Cedex 5  
Tél : 04 67 04 63 00 - Fax : 04 67 63 57 95

olivier.naud@montpellier.cemagref.fr  
www.montpellier.cemagref.fr/umritap

30 juin 2006

ACADÉMIE DE MONTPELLIER  
**UNIVERSITÉ MONTPELLIER II**  
— SCIENCES ET TECHNIQUES DU LANGUEDOC —

# **MÉMOIRE DE STAGE DE MASTER**

SPÉCIALITÉ : **Recherche en Informatique**  
*Mention* : **Informatique, Mathématiques, Statistiques**

effectué au **CEMAGREF**

—  
sous la direction de OLIVIER NAUD ET RODOLPHE GIROUDEAU

**Ordonnancement de tâches phytosanitaires vinicoles**

par

**TUITETE Tu**

Soutenu le \*\* 20 Juin 2006 \*\*  
Version corrigée du 30 Juin 2006

**Abstract :** Wine crops need to be protected or cured against various diseases during the whole season. During the spraying operations, chemicals are applied on plants but also reach the ground or are dispersed in the air and so, may pollute the environment. In order to reduce the number of such operations, experts have set up new rules. In these rules, the decision about whether to perform a spraying operation or not is taken at the plot level. In the contrary, most farmers process their whole vineyard on the basis of a single decision. Thus, before promoting these new rules, we need to check out if farms could apply the changes in organization that would be required. This master thesis deals with modelling the problem of scheduling spraying operations, with limited available resources, by the timed automata formalism and model-checking techniques. We show that an expert rule diagram can be transformed in a timed automaton and synchronized with weather forecast events in order to obtain the time intervals of the next operations of the different plots. We used same techniques to model farm constraints and check if schedules can be find out that match the set of temporal constraints, including transport between plots. We solved the appropriate timed logic queries on timed automata with the Kronos model-checking software. Further investigations will include meta-heuristics to reduce the combinatorial problem generated by the high number of plots, and checking various alternatives of expert rules.

**Résumé :** Des traitements chimiques sont nécessaires tout au long de la saison viticole pour protéger la vigne contre diverses maladies. Lors de ces traitements, les produits sont pulvérisés sur les pieds de vigne mais également sur le sol ou dans l'air, et sont donc causes possibles d'une pollution de l'environnement. Afin de réduire le nombre de traitements, des experts ont mis au point de nouvelles règles. Ces règles décident de l'opportunité de traiter une parcelle indépendamment des autres parcelles. Cependant, les viticulteurs traitent le plus souvent leur exploitation entièrement. Donc, avant de promouvoir ces nouvelles règles, il est important de vérifier si les exploitations peuvent supporter les changements dans l'organisation du travail qu'elles susciteraient. Ce mémoire de Master porte sur la modélisation de l'ordonnancement des traitements, avec ressources limitées, par le formalisme des automates temporisés. Nous montrons qu'une règle experte peut être transformée en un automate temporisé et synchronisée avec des prévisions d'événements climatiques pour trouver les intervalles de temps des prochains traitements sur les parcelles. Nous avons utilisé la même technique dans la modélisation des ressources et des contraintes temporelles pour trouver des ordonnancements réalisables prenant en compte les parcours entre parcelles. Pour cela, nous avons appliqué des requêtes de logique temporelle sur les automates à l'aide du logiciel de model-checking Kronos. La recherche sera poursuivie avec deux préoccupations : traiter le problème combinatoire issu de l'analyse simultanée d'un grand nombre de parcelles, par des méta-heuristiques, et vérifier l'applicabilité d'un ensemble d'autres règles expertes.

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>5</b>
1.1	Objectif : réduction des pollutions agricoles . . . . .	5
1.2	Outils de modélisation envisagés . . . . .	6
<b>2</b>	<b>Les automates temporisés</b>	<b>7</b>
<b>3</b>	<b>La logique TCTL</b>	<b>11</b>
3.1	La logique CTL . . . . .	11
3.2	La logique TCTL . . . . .	14
3.3	Le model-checking. . . . .	14
<b>4</b>	<b>Vérifier l'applicabilité de règles de décision à la parcelle</b>	<b>17</b>
4.1	Description détaillée de la problématique . . . . .	17
4.2	Motivations pour le choix des automates temporisés . . . . .	18
4.3	Modélisation . . . . .	19
4.3.1	Identification des processus à considérer . . . . .	19
4.3.2	Méthodologie adoptée et automates . . . . .	21
<b>5</b>	<b>Ordonnancement</b>	<b>26</b>
5.1	Modélisation permettant l'ordonnancement de tâches phytosanitaires . . . . .	26
5.1.1	Motivations pour l'utilisation des automates temporisés . . . . .	26
5.1.2	Hypothèses de départ . . . . .	26
5.1.3	Méthodologie adoptée . . . . .	27
5.2	Première résolution : "lots temporels" . . . . .	28
5.2.1	Automate de tâches . . . . .	28
5.2.2	Les événements de synchronisation pour l'ordonnancement . . . . .	29
5.3	Résolution du problème d'ordonnancement . . . . .	29
5.4	Résolution de problèmes plus importants : "lots géographiques" . . . . .	30
5.5	Résolution avec plusieurs pulvérisateurs . . . . .	30
<b>6</b>	<b>Limites actuelles de la modélisation et perspectives</b>	<b>32</b>
	<b>Bibliographie</b>	<b>35</b>
<b>A</b>	<b>Schéma UML de l'application développée</b>	<b>39</b>
<b>B</b>	<b>Diagramme de règle</b>	<b>40</b>

<b>C</b>	<b>Figures</b>	<b>41</b>
C.1	Automate regle modifiée après réalisation de deux traitements . . . . .	41
C.2	Automate tâches . . . . .	42
<b>D</b>	<b>Résultats Kronos</b>	<b>43</b>
D.1	Résultat d'une requête sur le produit " <i>pluiePrevue</i> × <i>regle</i> " : . . . . .	43
D.2	Résultat d'une requête "on the fly" sur des automates "fenetres temporelles" et "tâches" : . . . . .	44

# Chapitre 1

## Introduction

### 1.1 Objectif : réduction des pollutions agricoles

La réduction des pollutions du sol et de l'eau d'origine agricole s'inscrit dans les perspectives politiques du développement durable. Dans cette optique, des recherches sur les traitements phytosanitaires des vignes ont abouti à un ensemble de règles de décision permettant de réduire considérablement le nombre de traitements chimiques à effectuer par les exploitants [Cle04]. Cependant l'application de ces nouvelles méthodes peut avoir un impact significatif sur l'organisation du travail. Pour que ces règles puissent être adoptées largement par les viticulteurs, il convient donc d'évaluer au préalable leurs conséquences sur les tâches agricoles quotidiennes et sur la mobilisation des ressources disponibles des viticulteurs.

Le travail de ce mémoire s'inscrit dans le projet "*Vins et Environnement*" débuté en octobre 2005 pour une durée de trois ans <sup>1</sup>.

### L'exploitation viticole

Les tâches agricoles d'une exploitation sont soumises à certaines contraintes liées à l'évolution de l'état de la vigne sur une parcelle. Les états de cette parcelle vont eux-même dépendre naturellement du temps qui s'écoule. Elle passera par exemple de l'état "bourgeons" à l'état "en fleur" en une ou plusieurs semaines. Ces considérations sont également valables pour les maladies de la vigne (dynamique de l'épidémie). A partir de ces observations, une exploitation peut-être vue comme un système comprenant divers processus de traitements sur des parcelles de vignes mobilisant diverses ressources humaines ou mécaniques plus ou moins importantes au cours du temps. Cela nous conduit à considérer notre exploitation comme un système régi par des contraintes temporelles visant à agir sur un ensemble de parcelles et à organiser au mieux l'utilisation des ressources dont elle dispose.

---

<sup>1</sup><http://www.inra.fr/Internet/Projets/add-vin/index.htm>

## Evaluer la faisabilité de nouvelles règles pour décider des traitements phytosanitaires

Les nouvelles règles de décision concernant les traitements phytosanitaires sont établies à la parcelle. Autrement dit, une exploitation, vue comme un ensemble de parcelles, pourra traiter ses parcelles indépendamment les unes des autres en fonction de l'évolution respective de leurs vignes. Il nous faut dès lors considérer la parcelle comme unité de base mobilisant les ressources nécessairement finies de l'exploitation.

Pour rendre applicable ces règles au cours d'une saison et afin de garantir certains résultats portant sur la production, le viticulteur devra «surveiller» chaque parcelle au cours de son évolution. L'évolution ou non d'une maladie sur cette parcelle entraînera ou non des traitements à effectuer. De même, des événements pluvieux pourront entraîner des traitements dans les cas où ceux-ci peuvent augmenter la présence d'une maladie. Pour remédier à ce problème, il est préférable de traiter les parcelles concernées avant l'événement pluvieux.

Enfin, l'exploitant et ses ouvriers devront pouvoir effectivement traiter les vignes avec les moyens dont ils disposent. Sachant, que le nombre d'heures de travail dans une journée est limité, cela sera un facteur important dans l'ordonnancement des tâches agricoles. Tout cela nous conduit à considérer finalement plusieurs processus interagissant entre eux afin de garantir la bonne qualité du vignoble tout en appliquant des règles phytosanitaires permettant de limiter la pollution des sols.

### 1.2 Outils de modélisation envisagés

Les automates à états finis sont l'outil de base dans la modélisation des processus à événements discrets. Il existe d'autres formalismes de représentation de ces processus (notamment les réseaux de Petri et les grafsets) mais l'étude de leurs propriétés mathématiques peut se ramener en général à l'étude des propriétés d'automates.

Les automates définissent ce que l'on appelle en informatique théorique des "*langages*". Un automate peut reconnaître ou non un langage par consommation de symboles d'un alphabet, à travers le parcours de ses états et transitions. Cela permet, entre autre, de modéliser le comportement de systèmes ou processus réagissant à des événements internes ou externes. Les événements sont alors les labels/étiquettes des transitions de l'automate.

Les automates temporisés introduit par Alur [AD94] permettent d'intégrer la notion d'écoulement du temps au cours de l'exécution d'un processus. Ainsi, il est possible, par exemple, de borner la durée de réponse d'un processus à un événement auquel il a réagit.

Afin de vérifier certaines propriétés sur ces types d'automates, nous disposons d'outils de model-checking efficaces comme Kronos <sup>2</sup> et Uppaal <sup>3</sup>. Kronos a déjà été utilisé dans d'autres travaux relatifs à l'agriculture ([Hel03],[Lar00]). Pour ces raisons, les automates temporisés constitueront la base de notre modélisation.

Nous reviendrons sur ces choix en section 4.1, après avoir expliqué succinctement les formalismes des automates temporisés et du model-checking de ces automates. Par la suite, en section 5 nous nous concentrerons sur les problèmes d'ordonnancement de tâches phytosanitaires sur un ensemble de parcelles avec ressources limitées.

---

<sup>2</sup><http://www-verimag.imag.fr/TEMPORISE/kronos/>

<sup>3</sup><http://www.uppaal.com/>

# Chapitre 2

## Les automates temporisés

Le formalisme des automates temporisés [AD94] étend celui des automates à états finis en rajoutant des variables réelles positives exprimant des horloges. Les définitions de ce chapitre se trouvent dans la thèse de Yovine [Yov93].

**Définition :** Soit  $X$  un ensemble d'horloges. L'ensemble  $\Phi(X)$  des contraintes temporelles sur  $X$  est défini par la grammaire suivante :

$$\phi := true \mid x \prec c \mid x - y \prec c \mid \neg\phi \mid \phi \wedge \phi$$

avec :

- $x$  et  $y$  des horloges,
- $c \in \mathbb{Z}^+$ ,
- $\prec \in \{=, <, \leq, \geq, >\}$ ,
- $x - y$  : une opération arithmétique sur des horloges.

La disjonction entre deux contraintes temporelles peut être définie par :

$$\phi \vee \psi \equiv \neg(\neg\phi \wedge \neg\psi)$$

**Définition :** Un automate temporisé est représenté par un graphe  $G$  qui est un  $n$ -uplet  $(S, X, E, A, Inv)$  avec :

- $S$  : un ensemble fini de sommets avec  $s_0$  le sommet initial.
- $X$  un ensemble fini d'horloges.
- $E$  un ensemble fini d'étiquettes ou de labels.
- $A$  un sous-ensemble fini du produit cartésien  $S \times E \times \Phi(X) \times 2^X \times S$  définissant les arcs du graphe. Un arc sera donc un  $n$ -uplet  $(s, e, \phi, \delta, s')$  où  $s$  et  $s'$  seront respectivement les sommets source et destination de l'arc,  $\phi \in \Phi(X)$  sera une contrainte temporelle sur un sous-ensemble des horloges de  $X$  appelée **garde** et  $\delta$  un sous-ensemble d'horloges de  $X$  à réinitialiser à 0.
- $Inv(s) : S \rightarrow \Phi(X)$  une application qui associe à chaque sommet de  $S$  une contrainte temporelle appelée **invariant** du sommet.

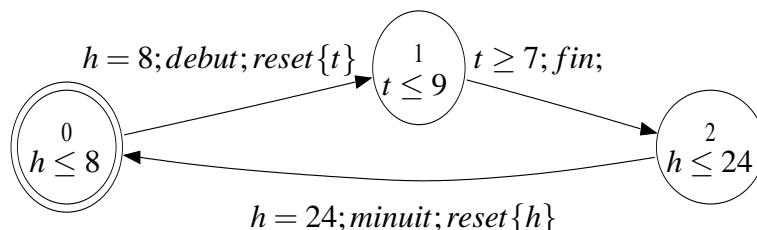
*Remarque :* Les valeurs des horloges croissent uniformément en fonction du temps, c'est à dire que si la valeur d'une horloge augmente de  $\delta t$ , alors toutes les horloges en font de même.

Avec ce formalisme, les contraintes temporelles peuvent être exprimées sur les arcs ainsi que sur les sommets du graphe. Le système pourra franchir un arc seulement si les contraintes temporelles de l'arc exprimées sur les horloges sont satisfaites. De même, le système ne pourra rester dans un état que s'il satisfait l'invariant du sommet. De plus, on impose que le franchissement des arcs se

fasse instantanément, c'est à dire que le temps ne s'écoule que dans les états de l'automate et pas sur les arcs.

**Exemple :**

Prenons l'exemple d'une journée de travail. Supposons qu'une telle journée commence à 8 heures et dure au moins 7 heures de temps mais pas plus de 9 heures. En notant  $h$  et  $t$  les horloges modélisant respectivement l'heure et la durée de travail on obtient ainsi :



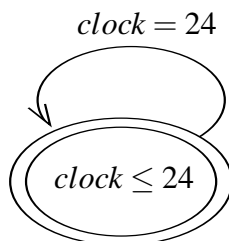
Dans cet exemple on a intentionnellement rajouté un arc permettant une réinitialisation de l'horloge  $h$  représentant l'heure telle que nous la percevons à minuit. Les labels utilisés pour les arcs sont : debut, fin, minuit.

**Définition :** Soient  $G_1 = (S_1, X_1, E_1, A_1, Inv_1)$  et  $G_2 = (S_2, X_2, E_2, A_2, Inv_2)$  deux automates. Le produit  $G = G_1 || G_2$ , en imposant  $X_1 \cap X_2 = \emptyset$ , est l'automate  $(S, X, E, A, Inv)$  avec :

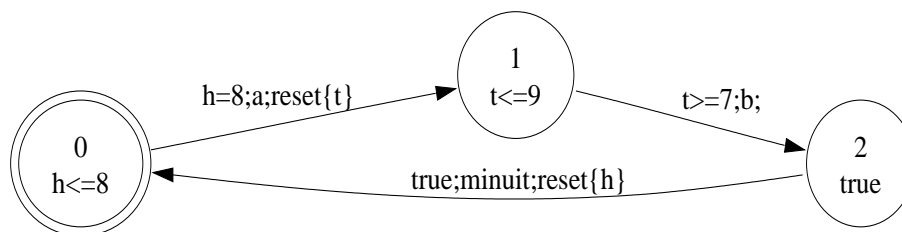
- Les sommets résultent du produit cartésien de  $S_1$  et  $S_2$  i.e  $S = S_1 \times S_2$
- L'ensemble  $X$  des horloges est l'union des ensembles d'horloges  $X_1$  et  $X_2$  i.e  $X = X_1 \cup X_2$ .
- L'ensemble  $E$  des étiquettes est l'union des ensembles d'étiquettes  $E_1$  et  $E_2$  i.e  $E = E_1 \cup E_2$ .
- L'invariant du sommet  $(s_1, s_2) \in S_1 \times S_2$  est constitué de la conjonction des contraintes temporelles  $Inv_1$  et  $Inv_2$  i.e  $Inv(s_1, s_2) = Inv_1(s_1) \wedge Inv_2(s_2)$ .
- Avec  $a_1 = (s_1, e_1, \phi_1, \delta_1, s'_1) \in A_1$  et  $a_2 = (s_2, e_2, \phi_2, \delta_2, s'_2) \in A_2$ , l'ensemble des arcs de  $G$  sont obtenus ainsi :
  - si  $e_1 = e_2$ , alors  $((s_1, s_2), e_1, \phi_1 \wedge \phi_2, \delta_1 \cup \delta_2, (s'_1, s'_2)) \in A$ ,
  - si  $e_1 \neq e_2$ , alors  $((s_1, s_2), e_1, \phi_1 \wedge \phi_2, \delta_1 \cup \delta_2, (s'_1, s_2)) \in A$  ainsi que  $((s_1, s_2), e_2, \phi_1 \wedge \phi_2, \delta_1 \cup \delta_2, (s_1, s'_2)) \in A$ .

**Exemple :**

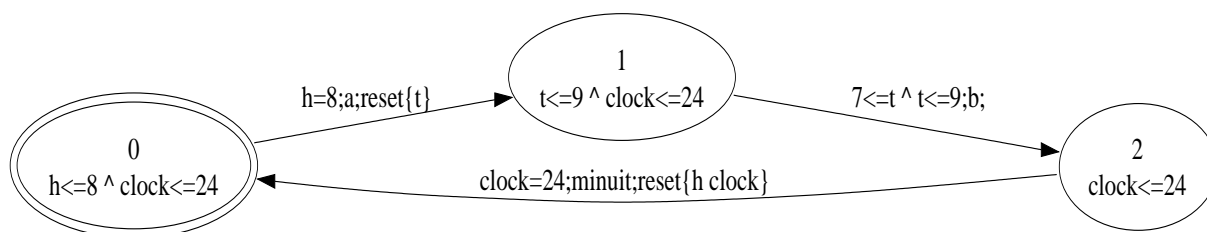
En reprenant notre exemple précédent mais en supposant que la réinitialisation de l'horloge  $h$  se fasse par l'intermédiaire d'un autre automate temporisé composé d'un sommet unique :



Notre automate initial devient :



Et le produit des deux nous donne finalement :



Dans ce nouvel exemple, une action est produite par l'automate "clock" qui est l'action "minuit". Cette action est perçue par le deuxième automate comme un événement d'entrée et lui permet de réinitialiser son horloge  $h$ . Le résultat obtenu est une synchronisation des deux automates à travers l'événement "minuit".

*Remarque :*

1. Le formalisme définit pour le produit de deux automates aurait du nous conduire à mettre l'arc "clock=24;minuit;reset{h clock}" sur les deux sommets 0 et 1 du graphe résultant. Cependant ces arcs n'auraient jamais été franchi puisque sur ces sommets,  $h$  ne peut atteindre la valeur d'horloge 24 sur ces sommets.
2. Ajoutons également cette opération est commutative. Plus de détails concernant le produit d'automates temporisés peuvent être trouvés dans [Tri98].

**Définition :** Une valuation  $v$  est une fonction qui associe à toute horloge de  $X$  une valeur dans  $\mathbb{R}^+$ , autrement dit  $v(X) \in \mathbb{R}^+$ . L'ensemble des valuations est noté  $V$ .

Pour  $t \in \mathbb{R}^+$ ,  $v + t$  est l'interprétation d'horloge  $v'$  définie par  $v'(X) = v(X) + t$ .

Soit  $h \subseteq X$ , on définit enfin  $v[h := 0]$  par :

$$v[h := 0](x) = \begin{cases} 0 & \text{si } x \in h \\ v(x) & \text{sinon} \end{cases}$$

Contrairement à un automate à états finis où un état est déterminé par un sommet de son graphe, l'état d'un automate temporisé sera déterminé par un couple  $q = (s, v)$ ,  $q \in Q$  correspondant à un sommet de  $S \in G$  associé à une valuation  $v$  des horloges de  $X$  sur ce sommet.

**Définition :** Une transition dans un automate temporisé représente le franchissement d'un arc ou la progression du temps dans un même sommet :

- **Transition discrète :** Soit l'arc  $(s, e, \phi, \delta, s')$ . L'état  $q = (s, v)$  a une transition discrète vers  $q' = (s', v')$  si  $v$  satisfait la contrainte  $\phi$  et  $v' = v[\delta := 0]$  permet de prendre en compte la réinitialisation des horloges de  $\delta$ . On note cette transition  $(s, v) \xrightarrow{e} (s', v')$ .

- **Transition temporelle** : Soit  $t \in \mathbb{R}^+$ . L'état  $(s, v)$  a une transition temporelle vers  $(s, v + t)$  notée  $(s, v) \xrightarrow{t} (s, v + t)$  si, pour tout  $t' \leq t$ ,  $v + t'$  satisfait  $Inv(s)$ .

**Définition** : Le triplet  $(Q, q_0, \longrightarrow)$  est le **système de transition** de l'automate temporisé avec  $q_0 \in Q$  un ensemble d'états initiaux. Ce système de transition définit ce que l'on appelle le modèle  $M(G)$  de l'automate temporisé  $G$ .

*Remarque* : La notion de modèle ici est relative à une certaine logique formelle qui sera introduite au 3.

**Définition** : Une séquence est une succession infinie d'états et de transitions  $q_0 \xrightarrow{1} q_1 \xrightarrow{2} q_2 \xrightarrow{i} q_3 \dots$  avec  $\xrightarrow{i} \in \longrightarrow$ .

*Remarque* : L'ensemble de toutes les séquences possibles va représenter finalement le modèle  $M(G)$  de l'automate temporisé de manière extensive. Précisons que le nombre d'états de ces séquences est potentiellement infini puisque que le temps est considéré comme dense c'est à dire qu'entre deux dates  $t_1$  et  $t_2$  il existe toujours une date  $t'$ .

Nous voyons que ce formalisme permet d'une part de modéliser des processus évoluant au cours du temps. De plus plusieurs automates temporisés peuvent se synchroniser par l'intermédiaire d'un mécanisme à événements discrets interprété comme des signaux d'entrée et/ou de sortie. On appelle aussi ces signaux des canaux de communication inter-processus.

Il nous est dès lors possible d'analyser ces processus à travers leurs états respectifs et de les coordonner entre eux en créant des états et/ou des canaux de communication.

Le produit de plusieurs automates donne un système entier comportant plusieurs processus qui peut être vu comme un seul processus lui même. Cependant il serait humainement difficile de lire cet automate système résultant.

# Chapitre 3

## La logique TCTL

Par souci de compréhension nous introduisons d'abord la notion de logique "Computation Tree Logic" (CTL) avant la logique TCTL. Toutes les définitions sont tirées de [Hel03]

**Définition :** Soient  $PA$  un ensemble de propositions logiques atomiques et  $P : S \longrightarrow 2^{PA}$  une application qui à tout sommet  $s \in S$  associe un ensemble de propositions atomiques  $P(s)$  de  $PA$ . L'idée est d'associer aux sommets d'un automate des propriétés que l'on pourra vérifier par la suite tout en intégrant la notion de temps et donc la notion d'état pour un automate temporisé.

### 3.1 La logique CTL

La CTL, introduite au début des années 80 par Clarke et Emerson [CE81], permet de considérer des modèles de systèmes dans lesquels il n'existe non pas une séquence d'exécution mais plusieurs à partir d'un état donné du système, d'où la notion d'arbre. Cette logique s'oppose à la logique "Propositional Linear Temporal Logic" (PLTL) où l'on ne considère à chaque fois qu'une seule séquence d'état du système vérifiant une propriété donnée.

#### Syntaxe CTL

Les formules CTL sont définies par la grammaire suivante :

$$\phi := p \mid \neg\phi \mid \phi \vee \psi \mid \exists \bigcirc \phi \mid \exists [\phi \cup \psi] \mid \forall [\phi \cup \psi]$$

avec  $p \in PA$  une proposition, et les opérateurs suivants :

- $\exists$  : il existe,
- $\forall$  : quel que soit,
- $\bigcirc$  : suivant,
- $\cup$  : jusqu'à.

Les opérateurs usuels booléens de conjonction, d'implication et d'équivalence entre deux formules CTL  $\phi$  et  $\psi$  peuvent être définis comme des macros par :

$$\begin{aligned}\phi \wedge \psi &\equiv \neg(\neg\phi \wedge \neg\psi) \\ \phi \Rightarrow \psi &\equiv \neg\phi \vee \psi \\ \phi \iff \psi &\equiv \phi \Rightarrow \psi \wedge \psi \Rightarrow \phi\end{aligned}$$

## Modèle CTL

**Définition :** Un modèle CTL est ce que l'on appelle une structure de Kripke  $M = (S, P, R)$ <sup>1</sup> où :

- $S$  est un ensemble non vide d'états,  $S_0 \subseteq S$  est l'ensemble des états initiaux,
- $P : S \longrightarrow 2^{PA}$  une application qui à tout sommet  $s \in S$  associe un ensemble de propositions logiques atomiques  $P(s)$  de  $PA$ .
- $R \subseteq S \times S$  est une relation totale sur  $S$ , c'est à dire que pour chaque état  $s \in S$  les états suivants possibles sont donnés par  $R$ .

**Définition :** Une séquence est une succession infinies d'états  $(s_0, s_1, \dots)$  tel que  $\forall i, (s_i, s_{i+1}) \in R$ .

## Sémantique CTL

La sémantique CTL est définie par un relation de satisfaction notée  $\models$  et relative à un modèle  $M$ . On a  $(M, s) \models \phi$  si et seulement si  $\phi$  est satisfaite en  $s$ . On note souvent plus simplement  $s \models \phi$  lorsque  $M$  est implicite. Cette relation est définie par induction de la manière suivante :

- $s \models p$  si et seulement si  $p \in P(s)$  i.e la proposition  $p$  est associée à  $s$ .
- $s \models \neg\phi$  si et seulement si  $\neg(s \models \phi)$  i.e s'il est faux que le sommet  $s$  puisse vérifier la formule  $\phi$ .
- $s \models \phi_1 \vee \phi_2$  si et seulement si  $(s \models \phi_1) \vee (s \models \phi_2)$
- $s \models \exists \bigcirc \phi$  si et seulement si  $(s, t) \in R$  et  $t \models \phi$  i.e si  $\phi$  est vraie pour au moins un successeur de  $s$ .
- $s \models \exists \phi_1 \cup \phi_2$  si et seulement si il existe au moins une séquence  $(s_0, s_1, \dots)$  avec  $s = s_0$  tel que quelque soit  $(i, j)$ ,  $0 \leq i \leq j$ ,  $s_i \models \phi_1$  et  $s_j \models \phi_2$ . Autrement dit, s'il existe au moins une séquence vérifiant  $\phi_1$  en  $s_0$  et  $\phi_2$  plus loin dans la séquence.
- $s \models \forall \phi_1 \cup \phi_2$  si et seulement si pour toutes les séquences  $(s_0, s_1, \dots)$  avec  $s = s_0$  tel que quelque soit  $(i, j)$ ,  $0 \leq i \leq j$ ,  $s_i \models \phi_1$  et  $s_j \models \phi_2$ . Autrement dit, si toutes les séquences vérifient  $\phi_1$  en  $s_0$  et  $\phi_2$  plus loin dans la séquence.

Enfin on note :

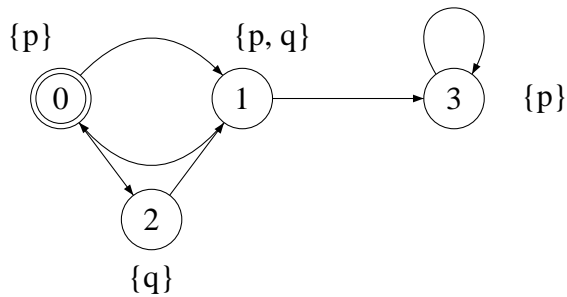
- $\exists \diamond \phi \equiv \exists \text{vrai} \cup \phi$  qui dit qu'il existe une séquence où la formule  $\phi$  est satisfaite au moins une fois.
- $\exists \square \phi \equiv \exists \neg \diamond \neg \phi$  qui dit qu'il existe une séquence où la formule  $\phi$  est toujours vraie (ie  $\phi$  est un invariant).

---

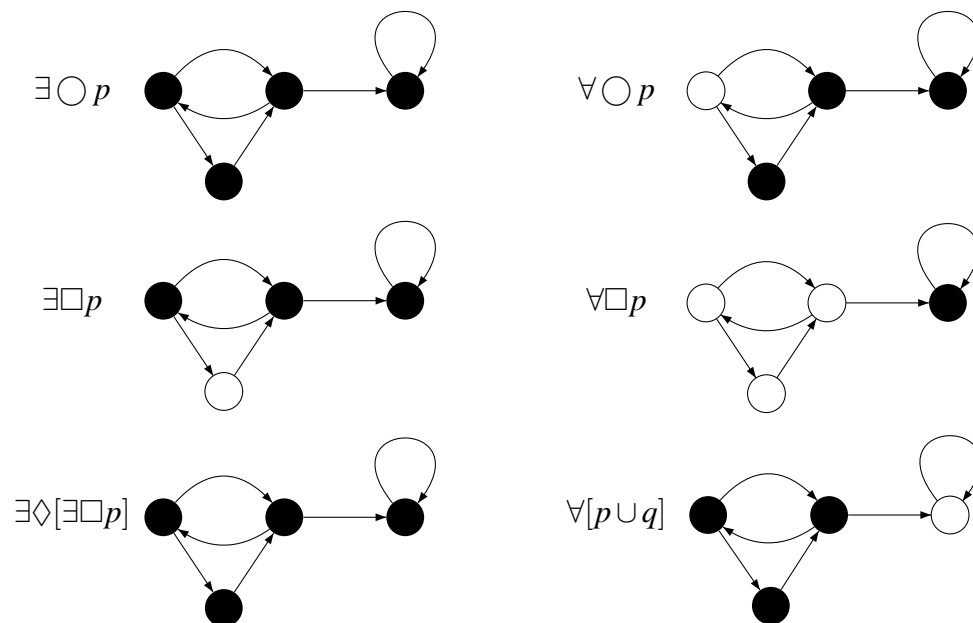
<sup>1</sup>La logique CTL fait partie des logiques modales.

**Exemple :**

L'exemple suivant est tiré de [Kat98]. Soit le modèle  $M$  représenté par la structure de Kripke suivante :



Dans les figures suivante nous avons coloré en noir les sommets vérifiant une formule CTL particulière.



- $\exists \bigcirc p$  est valide sur tous les états puisqu'ils ont tous au moins un successeur vérifiant  $p$ .
- $\forall \bigcirc p$  est valide sur les états 1,2 et 3 puisque leurs successeurs vérifient tous  $p$  alors que l'état 0 possède 1 comme successeur ne vérifiant pas  $p$ .
- $\exists \square p$  n'est pas valide en 2 puisque l'état 2 ne vérifie pas la proposition  $p$  alors que sur les autres états, il existe à chaque fois une séquence sur laquelle la proposition  $p$  est toujours valide.
- $\forall \square p$  n'est valide qu'en 3 puisque tout les séquences commençant dans cet état valident  $p$ . Autrement dit,  $p$  est un invariant à partir de l'état 3.
- $\exists \diamond [\exists \square p]$  est valide dans tous les états puisqu'il existe toujours dans ces états une séquence menant à un état dans lequel il existe une séquence validant toujours  $p$ .
- $\forall [p \vee q]$  n'est pas valide en 3 puisque toutes les séquence issues de l'état 3 ne valident pas  $q$ .

## 3.2 La logique TCTL

La "Timed Computation Tree Logic" étend la CTL en intégrant une notion de temps borné. Ainsi des formules peuvent par exemple être vérifiées dans un état pendant un certain temps  $t$  borné avant qu'elles ne soient invalidées.

Cette logique est très souvent associée aux automates temporisés.

### Syntaxe TCTL

Les formules TCTL sont définies par la grammaire suivante :

$$\phi := p \mid x \prec c \mid x - y \prec c \mid \neg\phi \mid \phi \vee \phi \mid \exists[\phi \cup_{\prec c} \phi] \mid \forall[\phi \cup_{\prec c} \phi]$$

avec  $x$  et  $y$  des horloges,  $c \in \mathbb{Z}^+$  et  $\prec \in \{=, <, \leq, \geq, >\}$ .

### Sémantique TCTL

Pour le modèle  $M(G)$  du système de transition  $(Q, q_0, \longrightarrow)$  de l'automate temporisé  $G$ , on définit la relation de satisfaction d'une formule TCTL en un état  $q = (s, v), q \in Q$  par induction :

- $q \models p$  si et seulement si  $p \in P(s)$  i.e la proposition  $p$  est associée à  $s$ .
- $q \models x \prec c$  si et seulement si  $v(x) \prec c$  i.e la valuation de  $x$  respecte cette contrainte d'horloge.
- $q \models x - y \prec c$  si et seulement si  $v(x) - v(y) \prec c$  i.e la valuation de  $x$  moins celle de  $y$  respecte cette contrainte d'horloge.
- $q \models \neg\phi$  si et seulement si  $\neg(s \models \phi)$  i.e s'il est faux que le sommet  $s$  puisse vérifier la formule  $\phi$ .
- $q \models \phi_1 \vee \phi_2$  si et seulement si  $(s \models \phi_1) \vee (s \models \phi_2)$
- $q \models \exists\phi_1 \cup_{\prec c} \phi_2$  si et seulement si il existe au moins une séquence  $(q_0, q_1, \dots)$  avec  $q = q_0$  tel que quelque soit  $(i, j), 0 \leq i \leq j, q_i \models \phi_1$  et  $q_j \models \phi_2$  avec un temps entre  $q_0$  et  $q_j$  satisfaisant la relation  $\prec c$ . Autrement dit, s'il existe une séquence où  $\phi_1$  est toujours vraie jusqu'à un état, situé dans un temps respectant la contrainte  $\prec c$  où  $\phi_2$  est vraie.
- $q \models \forall\phi_1 \cup_{\prec c} \phi_2$  si et seulement si pour toutes les séquences  $(q_0, q_1, \dots)$  avec  $q = q_0$  tel que quelque soit  $(i, j), 0 \leq i \leq j, q_i \models \phi_1$  et  $q_j \models \phi_2$  avec un temps entre  $q_0$  et  $q_j$  satisfaisant la relation  $\prec c$ . Autrement dit, si pour toutes les séquences  $\phi_1$  est toujours vraie jusqu'à un état, situé dans un temps respectant la contrainte  $\prec c$  où  $\phi_2$  est vraie.

*Remarque :* Ici, l'état  $q$  d'un automate temporisé est un couple  $(s, v)$  alors que dans la logique CTL l'état d'un automate se réduisait à un sommet de son graphe.

## 3.3 Le model-checking.

Le model-checking est une manière générale de valider le modèle d'un système à l'aide d'algorithmes vérifiant certaines propriétés de ce modèle exprimées sous forme logique (comme la logique temporelle). En générale le modèle d'un système est construit en respectant au mieux des spécifications de fonctionnement exprimées dans un cahier des charges. Pour vérifier le modèle au cours de sa construction et après, il est utile de comparer les comportements qu'il a par rapport

aux spécifications du cahier des charges. A l'image de la construction d'un pont, tout est fait, des spécifications à la construction, pour faire en sorte qu'à la fin du chantier, ce dernier soit stable et qu'il puisse soutenir une charge maximale déclarée au préalable.

Les techniques de model-checking sont de plus en plus utilisées en génie logiciel et permettent de prouver ou non la stabilité d'un système sous certaines conditions en explorant de manière exhaustive tous les états possibles du système. Ces techniques permettent par exemple de vérifier des protocoles de communication inter-processus partageant une même ressource afin d'éviter des situations de blocage.

## Classes de propriétés vérifiables sur un modèle

Les définitions suivantes sont tirées de [BBF<sup>+</sup>01] :

- Exactitude : Le système fait au moins ce pourquoi il a été conçu.
- Sûreté : Sous certaines conditions, quelque chose n'arrivera jamais.  
**Exemples :**
  - Absence de blocage.
  - Préservation de l'ordre de message.
  - Accès exclusif à une ressource.
- Vivacité : Sous certaines conditions, quelque chose finira par arriver.  
**Exemples :**
  - Chaque message envoyé sera reçu.
  - Un processus finira par accéder à la ressource.
  - Si je commande un café, je le recevrai.
- Atteignabilité bornée : Sous certaines conditions, quelque chose finira par arriver en un temps borné.  
**Exemples :**
  - Chaque message envoyé sera reçu en moins de 10 secondes.
  - A l'arrivée d'un train à un passage à niveau, les barrières de sécurité se baissent toujours au moins 20 secondes avant.

Comme nous l'énoncions dans l'introduction, une exploitation agricole peut-être vue comme un système réactif. On pourra donc utiliser le model-checking afin de garantir l'existence d'un ordonnancement réalisable des tâches phytosanitaires sur toute la saison viticole en utilisant les règles de décision élaborées par des experts. Et s'il n'existe pas de tel ordonnancement pour une exploitation donnée, on pourra affirmer que les règles ne sont pas applicables dans ce cas de figure.

## Quelques outils de model-checking

HyTech [HHWT97] est dédié à l'analyse des systèmes dynamiques hybrides linéaires<sup>2</sup> et permet donc de représenter le temps.

Uppaal [LPY97] a l'avantage d'avoir une interface graphique et permet également des simulations visuelles des automates temporisés. Il ne permet cependant pas de vérifier l'ensemble des formules de la logique TCTL possible.

Kronos [DOTY95] permet de vérifier l'ensemble des formules TCTL et implémente des algorithmes de recherche en avant et en arrière dans les graphes de simulations construits à partir des automates temporisés. Il implémente également un algorithme dit "à la volée" évitant de construire l'automate résultant du produit d'autres automates sur lequel porte une requête.

Rabbit [BDCA03] est lui adapté aux problèmes de grande taille.

Dans [BS00] une comparaison des outils Hytech, Kronos et Uppaal est faite relativement à un problème de gestion de passage à niveau pour train. Il en ressort que Kronos est le plus efficace en terme de rapidité d'exécution pour requêtes concernant des problèmes de grosse taille en nombre d'états et transitions. Il propose également un algorithme de vérification de formule TCTL dit "à la volée" évitant de construire le produit des automates concernés par la requête. De plus, les formules TCTL sont toutes vérifiables par Kronos et son utilisation se fait par ligne de commande et peut donc facilement être appelé par un script dans un programme. Enfin, Arnaud Hélias [Hel03] connaissant ce logiciel nous a proposé de nous aider concernant son utilisation. Pour toutes ces raisons nous avons choisi Kronos comme model-checker dans notre travail.

## Complexité des vérifications des formules TCTL

En raison de la densité du temps, un automate temporisé possède une infinité d'états. Cependant, Alur [AD94] montre que l'on peut construire un graphe appelé *graphe de région* dans lequel un sommet ne représente pas un état mais un ensemble d'états dans lequel une propriété (i.e. un prédicat) est vrai. Ainsi une relation d'équivalence peut être définie entre des états lorsqu'ils vérifient les mêmes formules TCTL et le graphe obtenu a un nombre fini de sommets.

L'espace mémoire requis pour la construction complète du graphe de région est polynomial en la taille du système à analyser. Mais Henzinger et coauteurs [TXJS92] précisent qu'il est possible de ne construire seulement les états requis en fonction du problème posé.

Dans [Kat98], l'auteur précise que dans le pire des cas, une requête TCTL est décidable en temps exponentiel en fonction du nombre d'horloges présentes dans l'automate.

---

<sup>2</sup>reliant continu (équations différentielles) et discret

## Chapitre 4

# Vérifier l'applicabilité de règles de décision à la parcelle

### 4.1 Description détaillée de la problématique

Nous devons préciser que les règles de décision à la parcelle ne sont pas forcément réalisable pour toutes les exploitations. En effet, l'objet de notre étude est justement de mettre en place un outil permettant de répondre à la question : "Pour une exploitation donnée, correspondant à une certaine configuration, est-il possible d'implanter les règles de décisions concernant des traitements phytosanitaires à la parcelle?". Dans le cas où l'implantation des règles dans une exploitation ne serait pas possible (i.e. s'il n'existe pas d'ordonnancement réalisable des traitements avec les ressources disponibles), l'idéal serait de pouvoir identifier les changements à amener dans cette exploitation pour pouvoir les y intégrer.

Notre travail va consister dans un premier temps à construire un modèle représentant une parcelle de vigne au sein d'une exploitation. Par ailleurs, des règles de décisions concernant les traitements phytosanitaires à apporter à cette parcelle devront être intégrées dans le modèle. Ces règles prennent en compte diverses variables comme le climat, les stades phénologiques des pieds de vigne (stades de développement des plantes) ou encore le taux de maladie (champignons) de la parcelle.

Une règle simple pourrait être par exemple : à l'annonce d'une pluie dans les trois jours et si la parcelle se trouve dans l'état "5 feuilles étalées" alors il faut traiter la parcelle avant les deux jours. Notre parcelle peut donc être considérée comme un système réagissant continuellement au sein de son environnement en fonction d'événements en respectant certains délais de réaction. Ce genre de système est appelé "système réactif" et a été introduit par Pnueli en 1985 [Pnu85].

Comme nous l'avons vu au chapitre 2 en modélisant une parcelle par un automate temporisé, ce dernier peut basculer d'un état à un autre en fonction d'événements. Ces événements pourront être de type temporel tel qu'une date donnée au cours des stades phénologiques de la vigne ou encore un signal de présence de maladie et donc entraînant la nécessité d'un traitement phytosanitaire.

Une exploitation pourra par la suite être modélisée comme un automate résultant du produit des automates modélisant les divers processus internes c'est à dire les parcelles et les ressources matérielles et humaines de l'exploitation.

## 4.2 Motivations pour le choix des automates temporisés

Dans sa thèse, C. Largouët, [Lar00], utilise les automates temporisés pour modéliser des parcelles agricoles pouvant correspondre à différentes cultures. Sa problématique était de raffiner le classement de parcelles agricoles pré-classées par un système de traitements d'images satellites automatique. En effet, après un premier passage par le système de traitements d'images, à une parcelle pouvait correspondre plusieurs classes possibles parmi un ensemble de classes pré-déterminées. Le fait de modéliser les parcelles à l'aide des automates temporisés a permis de réduire encore plus l'ensembles des classes possibles pour une parcelle.

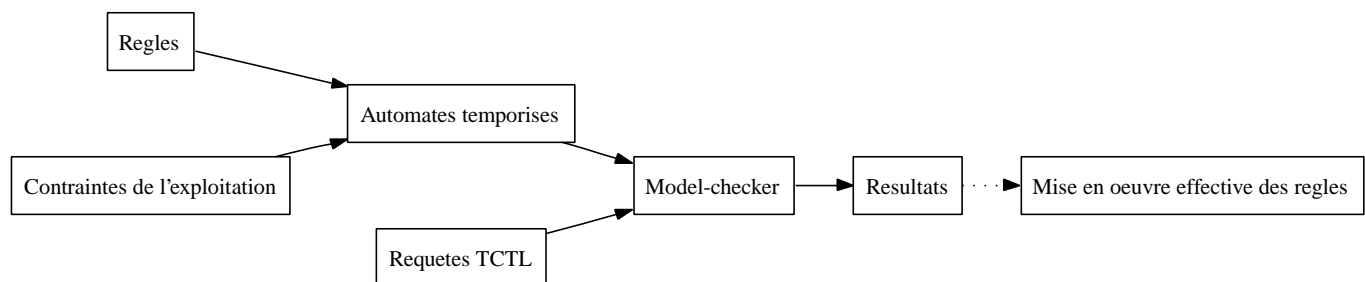
A. Helias propose dans [Hel03] une modélisation générique des contraintes temporelles de transferts entre unités de productions et unités de consommation et l'applique à un système de gestion des effluents d'élevage à la Réunion. Les unités (parcelles agricoles et élevages) sont toutes représentées sous forme d'automates temporisés. Il montre qu'en calculant des dates de disponibilité de la ressource (le lisier ou le fumier d'un élevage), ainsi que des dates de possibilité d'épandage de cette ressource de fertilisation au sein de cultures agricoles variées, on peut simuler les interactions entre plusieurs unités agricoles sur plusieurs années et proposer des solutions au problème de la gestion des effluents d'élevage.

Concernant notre problème, un modèle utilisant les automates temporisés nous permettrait de vérifier l'applicabilité de règles de gestion de parcelles de vignes au sein d'une exploitation, en particulier, les règles de protection phytosanitaire. Nous nous intéressons surtout à l'ordonnancement des tâches liées à ces règles. Actuellement, les traitements viticoles se font souvent de manière systématique, c'est à dire tous les quinze jours bien que certains de ces traitements soient inutiles si la maladie est absente. Dans ces cas là, la pollution par les produits chimiques sur l'environnement pourrait être évitée. De nouvelles règles permettraient de mieux gérer ces situations mais cela a des répercussions sur le travail dans l'exploitation et il faut donc mesurer cet impact. L'idée est de représenter au mieux les différents éléments et contraintes d'une exploitation et alors utiliser un model-checker pour établir les vérifications nécessaires avant mise en oeuvre réelle de règles. La notion de temps étant importante dans l'évolution d'une parcelle de vignes ainsi que dans les règles expertes établies et les ordonnancements de tâches agricoles, il nous a semblé opportun de représenter ces éléments sous forme d'automates temporisés.

Les outils de model-checking tels que Kronos [YO93] ou UPPAAL [LPY97] permettent la vérification de certaines propriétés des automates temporisés et également l'obtention des dates compatibles avec les règles par propagation de contraintes temporelles. Nous avons pris contact avec Arnaud HELIAS au sujet de notre travail et celui-ci nous a apporté son expérience pour la modélisation ainsi que l'utilisation du logiciel Kronos.

Pour ces raisons, nous utilisons les automates temporisés dans notre modélisation.

## Schéma explicatif de la méthode :



## 4.3 Modélisation

Dans l'exposé de notre problématique dans la section 4.1, nous avons introduit la notion de configuration d'une exploitation mais sans pour autant la définir. Dans cette section, nous allons donc clarifier ce point.

Rappelons nous que nous nous intéressons à l'ordonnancement des tâches phytosanitaires engendrées par les règles. Avec un peu de recul, nous pouvons exprimer notre problème comme étant de répondre à la question suivante :

*"Les règles à implanter sont-elles compatibles avec l'organisation du travail dans l'exploitation ?"*

Pour pouvoir y répondre, nous devons d'abord identifier tous les processus permettant la mise en place des règles mais pouvant également perturber l'ordonnancement des tâches agricole dans l'exploitation. Et cela correspondra finalement à notre définition de la "configuration" d'une exploitation. Il nous semble convenable de tester par la suite la mise en oeuvre effective des règles dans une exploitation par simulation en fonction de sa propre configuration et des différents paramètres dont dépendent les règles.

La méthodologie développée peut être appliquée à différentes variantes de règles, au fur et à mesure de leur conception par des pathologistes.

### 4.3.1 Identification des processus à considérer

#### Les parcelles

Un automate parcelle comportera les états phénologiques possibles de la parcelle concernée au cours d'une saison. Les observations concernant les stades phénologiques sont des états dominants sur une parcelle. En effet, il est tout à fait possible que d'un pied de vigne à l'autre, le stade phénologique ne soit pas le même.

Rappelons que les transitions possibles entre deux états seront temporelles puisqu'elles doivent suivre les stades phénologiques. Précisons également que les cépages peuvent varier d'une parcelle à l'autre et que cela induit des stades phénologiques différents d'une parcelle à l'autre. De même, la dispersion géographique des parcelles pourra avoir le même effet.

Les données dont nous disposons ne permettent pas de prédire les stades phénologiques dans tous les cas possibles, c'est à dire obtenir les dates au plus tôt et au plus tard des stades phénologiques pour un cépage et une zone géographique donnée. Cependant, nous disposons d'un historique pour deux parcelles sur deux années qui suffit, au stade actuel, à montrer la faisabilité et la généricité de

notre approche.

## **Ressources humaines et matérielles**

Pour représenter les contraintes sur les ressources matérielles, on peut supposer que si un exploitant possède un tracteur dédié aux traitements phytosanitaires, celui-ci est alors toujours disponible conjointement avec le pulvérisateur au cours de la saison et l'automate correspondant à ce groupe de ressources ne doit comporter que deux états : libre et utilisé pour un traitement sur une parcelle. Pour les ressources humaines, il faudrait prendre en compte les durées de travail journalières, hebdomadaires ainsi que les jours non travaillés tels les dimanches et jours fériés. Cependant, par souci de simplification au stade actuel des travaux, nous allons simplement nous ramener à dix heures pouvant être travaillées dans une journée en considérant que tous les jours (même les dimanches) sont travaillés.

## **La météorologie**

En nous basant sur des données de pluviométrie et de températures moyennes des années 2003 et 2004 du bordelais, nous avons constaté que, d'une année à l'autre, ces paramètres ne sont pas du tout répartis de la même manière au cours des mois. Cela implique que nous ne pouvons pas modéliser la pluie par un automate cyclique simple au cours d'une saison. L'idée serait plutôt de considérer les événements pluvieux comme des événements produits par un "automate chaîne" à partir d'un fichier historique.

De plus il nous faudra distinguer la pluie prévue par la météo de la pluie réelle. Par exemple, une pluie peut être annoncée dans trois jours et effectivement tomber dans deux seulement. Ou encore, une pluie peut ne pas être annoncée du tout et surprendre. Et bien d'autres scénarios sont encore possibles.

Pour nous approcher au mieux de la réalité, nous avons généré un fichier de pluies prévisionnelles basées sur un fichier de pluies réelles. Pour cela, nous avons considéré que les prévisions météorologiques sont exactes dans 50% des cas. La pluie n'est pas prévue du tout ou est prévue un jour avant qu'elle ne tombe effectivement ou est prévue un jour après qu'elle ne soit tombée effectivement dans respectivement 10%,10%,10%,20% des cas.

## **Les pathologies de la vigne**

Contrairement à la pluviométrie au cours d'une année, nous n'avons pas actuellement de données ou de modèles épidémiologiques à l'échelle de la parcelle concernant les maladies de la vigne. Par exemple, à quel période de la saison le Mildiou<sup>1</sup> (un champignon pathologique de la vigne) est plus susceptible d'apparaître ? Sur quelle proportion de l'ensemble des parcelles d'une exploitation apparaît une pathologie donnée ? Nous ne représentons donc pas pour l'instant l'état sanitaire de la parcelle.

## **La rémanence et le lessivage des produits phytosanitaires**

La rémanence d'un produit chimique est sa durée de période active sur une parcelle. Il s'agit donc d'un processus protégeant la vigne à compter de la diffusion du produit et ce, pendant une certaine durée définie par le fabricant de la substance chimique.

---

<sup>1</sup>Nom biologique : *Plasmopara viticola*

Le lessivage est l'annulation de l'effet d'un produit à la surface des feuilles par une pluie dépassant un certain seuil défini par le fabricant du produit. Par ce processus, les feuilles de vignes se retrouvent sans aucune protection chimique contre les pathologies susceptibles de les attaquer. Dans un premier temps, nous ne tiendrons pas compte du lessivage.

## **Les règles**

Une règle à la parcelle est définie à partir d'une combinaison de divers processus afin de générer des traitements de protection vinicole. En effet, une règle s'applique en fonction du stade phénologique d'une parcelle, de la présence ou non d'une pathologie, d'un ou plusieurs processus de rémanence de produits phytosanitaires et des pluies prévues et passées.

On peut l'interpréter comme un processus de "surveillance" d'une parcelle, passant d'un état à l'autre en fonction des événements émis par les processus dont il dépend. Par la suite, il générera des événements de sortie concernant les traitements qui activeront des automates ressources.

Toute la difficulté de la modélisation d'une règle réside dans l'identification des processus impliqués ainsi que l'ordre dans lequel les événements doivent être reçus par l'automate "règle". Par exemple, à la fin de la rémanence d'un produit, s'il y a une annonce de pluie (événement `pluiePrevue`), alors il faut traiter la parcelle avant l'événement pluvieux.

D'autre part, une grande attention doit être portée aux labels pour que le produit des automates donne le résultat escompté. Rappelons également que les labels des transitions sont obligatoires pour respecter leur sémantique.

### **4.3.2 Méthodologie adoptée et automates**

#### **Méthodologie**

En fonction de la question que l'on se pose, plusieurs automates doivent être pris en compte pour la réponse, par produit de ceux-ci. Ce produit d'automates modélisera toutes les contraintes temporelles liées à la question. Nous allons donc dans un premier temps montrer les différentes constructions des automates. Par la suite, une résolution progressive est effectuée : obtention d'un jeu de dates possibles pour les interventions par propagation de contraintes temporelles, choix effectifs des dates et enfin vérification par recherche d'un ordonnancement réalisable au chapitre 5.

#### **Génération des automates**

L'outil de model-checking Kronos ne possédant pas d'interface graphique pour la création des automates, il a été préférable d'en développer un afin de faciliter cette tâche. L'application nous a alors permis d'automatiser la génération de plusieurs automates de même type (plusieurs automates "règle" par exemple) ayant chacun un numéro d'identification nécessaire en cas de produit d'automates.

De plus, la construction d'automates types tels que les pluies réelles ou les pluies prévues ont nécessité l'analyse au préalable de fichiers de données et par la suite la génération des automates correspondant.

Enfin, grâce au logiciel que nous avons développé, il sera possible de générer des automates pour différents autres outils que Kronos, et de rajouter des structures de données spécifiques à notre type d'application (tâches agricoles).

Un schéma UML de l'application est fourni en annexe A.

### Automate règle :

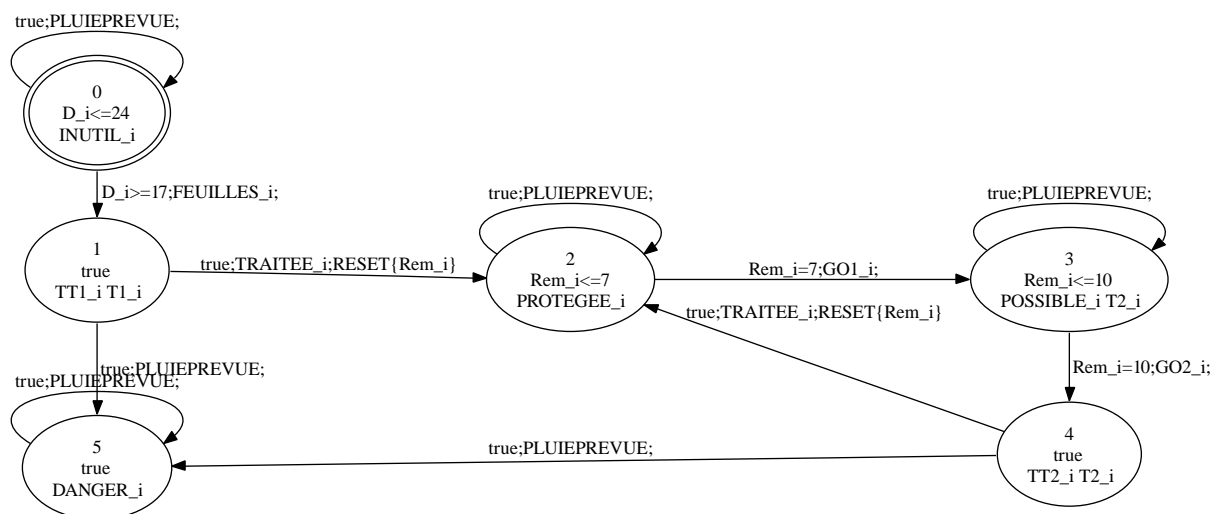
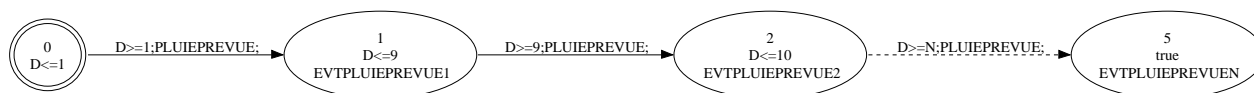


FIG. 4.1 – Règle simple de gestion du Mildiou

Le premier état  $S_0$  correspond à une période de début de saison où un traitement phytosanitaire est inutile du fait que les vignes ne possèdent pas ou peu de feuilles. Le sommet  $S_1$  (vérifiant les prédicats  $TT1$  et  $T1$ ) représente le premier état de traitement possible et il est atteint à partir du moment où un nombre suffisant de feuilles est observé sur la parcelle (événement  $FEUILLES_i$ ) par l'exploitant. Quand une pluie intervient dans cet état, alors on atteint le sommet  $S_5$  de danger (i.e. une évolution de maladie a été entraînée par la pluie). Sinon, c'est que l'on a traité avant la pluie et la parcelle se trouve en l'état  $S_2$  où elle est protégée et le traitement est redevenu inutile. La parcelle reste protégée (prédicat  $PROTEGEE$ ) en  $S_2$  pendant 7 jours au moins. Nous nous donnons la possibilité de traiter une parcelle après ces 7 jours (sommet  $S_3$ ) alors qu'il y a toujours rémanence du produit mais cela n'est pas obligatoire. Ce choix permet de nous donner éventuellement plus de marge dans le temps pour trouver un ordonnancement de tâches de traitement. A la fin de la rémanence (soit 10 jours après le précédent traitement) correspond le sommet  $S_4$ , état dans lequel la parcelle doit obligatoirement être traitée avant la prochaine pluie sous peine de se retourner en danger de propagation de maladie (état  $S_5$ ).

Cette règle est celle que nous avons utilisé dans nos tests. D'autres règles seront testées par la suite(cf 6).

## Automate pluiePrevue :



Cet automate est simplement constitué d'état d'écoulement du temps sans pluie et les transitions nous indiquent des événements de prévisions de pluies considérés comme instantanés. Le fait qu'il puisse pleuvoir une journée entière n'est pas pris en compte ici, cela n'est probablement pas une lacune très gênante dans la région et la période considérée (avril à août dans le Bordelais). Il sera possible de prévoir ce cas de figure par la suite. En tout état de cause, la règle que nous testons dans ce mémoire ne prend pas en compte la durée de pluie prévue, ni les quantités.

Une transition " $D=5 ; \text{pluiePrévue} ;$ " signifie qu'une pluie tombera à la date exacte  $D=5$ . Le dernier sommet correspond à la fin de la saison vinicole où la pluie n'a plus besoin d'être prise en compte et l'invariant du dernier sommet est simplement "*true*".

## Le problème du premier traitement

Comme nous l'avons précisé précédemment, les parcelles n'évoluent pas de la même manière et il est important de situer dans le temps l'état phénologique correspondant aux premières feuilles de chaque parcelle (i.e. état  $S1$  de la règle). En effet, cet état sera observable à une date  $D$  précise du début de la saison et variera en fonction du climat, de la situation géographique des parcelles ou encore des cépages. Cette date  $D$  sera une entrée de notre problème et nous devons le préciser lors des requêtes faites à Kronos. Nous obtenons alors un état de départ pour calculer la première fenêtre temporelle.

### Exemple :

Nous donnons un exemple de requête Kronos avec spécification d'état de départ voulu.

*Kronos - FULLDFS - allpaths - R "TT1" - I "D > 20 and TT1" Res.tg*<sup>2</sup>

Cette requête correspond à la formule TCTL :

$D > 20 \text{ and } TT1 \Rightarrow \exists \diamond (TT1)$

Dans cet exemple, nous spécifions simplement que la date actuelle est le vingtième jour de la saison et que la parcelle  $P$  est vulnérable (i.e. dans un état vérifiant le prédicat  $TT1$ ). La particularité de cet état initial est que l'état de la parcelle vérifie déjà  $TT1$ . Donc à partir de là, Kronos détermine la date échue de validité de  $TT1$  et cela correspond à notre date limite de prochain traitement obligatoire. En fait  $TT1$  est ici juste l'équivalent, pour nous, de vulnérable dans le sens où les feuilles vignes seront atteintes par une les champignons pathologiques en cas de pluie.

Ensuite, il suffit de récupérer et analyser le résultat de Kronos pour le transformer en un système d'inéquations linéaire et enfin exécuter la résolution du programme linéaire avec minimisation de la borne supérieure de l'intervalle cherché. Un exemple de fichier de sortie nommé "dfs.trace" est donné en annexes D.1.

## Obtention des fenêtres temporelles

Il s'agit d'obtenir pour une parcelle  $P_i$  donnée les dates de début et de fin de traitement de celle-ci. Pour cela nous devons poser une requête d'atteignabilité du sommet vérifiant le prédicat  $TTN$  et

<sup>2</sup>-*FULLDFS* entraîne une recherche en profondeur d'abord du graphe et est obligatoire pour obtenir tous les chemins avec *-allpaths*.

une valuation de l'horloge  $Rem_i$  supérieure à 10 (i ; e ; fin de rémanence) au produit de l'automate règle de la parcelle et de celui de prévision des pluies. Plus, concrètement, les lignes de commandes pour Kronos sont les suivantes :

- Pour obtenir le produit des automates dans le fichier " $Res_i.tg$ " :  
 $Kronos - out Res_i.tg pluiePrevue.tg regle_i.tg$
- Pour obtenir la date du prochain traitement de la parcelle  $P_i$  la formule TCTL est :  
 $init \Rightarrow \exists \diamond (TTN \text{ and } Rem_i > 10)$   
 La requête Kronos correspondante est :  
 $Kronos - FULLDFS - allpaths - R "TTN \text{ and } Rem_i > 10" Res_i.tg$

Après l'exécution de la deuxième commande, Kronos nous fournit en sortie un fichier nommé " $dfs.trace$ " contenant le parcours de tous les chemins avec les valuations des horloges correspondantes dans l'automate "res" résultant du produit  $pluiePrevue \times regle_i$ . Ces parcours donnent les diverses transitions prises pour atteindre finalement le ou les sommets vérifiant  $TTN \text{ and } Rem_i > 10$ . Au final, nous sommes juste intéressés par la date échue. Les fichiers générés par Kronos doivent cependant au préalable être analysés et transformés en résultats exploitables pour nous. Pour cela, nous avons adopté la méthode d'Arnaud Hélias consistant en la résolution d'un programme linéaire dont la fonction objectif serait simplement une maximisation de l'horloge  $D$  correspondant à la date calendaire de la saison viticole et incluse directement dans l'automate de pluie prévue. Précisons que cette horloge  $D$  n'est jamais réinitialisée au cours de la saison et qu'elle nous sert donc de référence pour le calcul des fenêtres temporelles.

### Modification obligatoire de l'automate règle

Conceptuellement l'automate règle est celui présenté dans la figure 4.1. Nous pouvons remarquer qu'il existe un cycle entre le sommet  $S5$  et le sommet  $S2$  correspondant à l'arc de traitement de la parcelle. Cependant, nous constatons que lors de l'instanciation de ce traitement à une date  $\delta$  nous devons mettre un invariant de sommet dans le sommet 5 qui est  $D_i \leq \delta$  et également un invariant d'arc sur l'arc "traiteeX" :  $D_i \geq \delta$ . Au final, cela veut dire que l'automate retourne dans l'état  $S2$  à une date précise. Or pour obtenir la date du traitement suivant, il faut réitérer ce processus et donc changer la valeur de  $\delta$  en fonction de la rémanence et d'un futur événement pluvieux. Par conséquent, nous obtenons finalement un automate chaîne tout au long de la saison du fait que cela dépend totalement de la pluie. En effet l'automate règle est bien cyclique mais n'est pas périodique car la pluie ne l'est tout simplement pas. Nous schématisons la règle telle qu'elle est maintenant dans la figure C.1 située en annexes.

## La mémoire des états en fonction du temps et des événements

Par construction itérative d'un automate règle en fonction des traitements effectués sur une parcelle  $P$  correspondante (paragraphe précédent) nous gardons une "mémoire" (i.e. une trace) des états passés de la parcelle. Ceci est important puisque cette mémoire est nécessaire afin de calculer le prochain traitement et connaître l'évolution de  $P$  relativement à une date donnée.

### Algorithme d'obtention des autres fenêtres temporelles

En supposant et que tous les états passés des parcelles (i.e. depuis le jour  $J = 1$  de la saison) ont été mis à jour correctement dans leurs automates règles respectives, nous pouvons alors calculer la date limite  $\delta_2$  du prochain traitement de l'ensemble des parcelles.

Pour ce faire, nous pouvons appliquer l'algorithme suivant :

**for all**  $P_i \in \text{Parcels}$  **do**

*Kronos* – *out res<sub>i</sub>.tg pluiePrevue.tg regle<sub>i</sub>.tg*

*Kronos* – *FULLDFS* – *allpaths* – *R* "TTN and  $Rem_i > 10$ " *res<sub>i</sub>.tg* > *fenetreN<sub>i</sub>*

$\delta_2 \leftarrow \text{Minimize}(\text{fenetreN}_i)$

**end for**

*Minimize(fenetreN<sub>i</sub>)* permet simplement à partir du résultat de *Kronos*, d'obtenir la date limite  $\delta_2$  voulue par résolution de système linéaire tel que présenté pour le problème du premier traitement.

Précisons que les fenêtres temporelles obtenues dans cette section ne permettent pas d'obtenir les dates effectives de traitement des parcelles. En effet, ce n'est qu'après résolution du problème de l'ordonnancement, en tenant compte de ces fenêtres (cf chapitre 5), que ces dates sont obtenues. Nous verrons alors que les résultats de dates obtenus jusqu'à présent nous permettront de spécifier de nouvelles contraintes sous forme d'automates temporisés afin de résoudre l'ordonnancement.

# Chapitre 5

## Ordonnancement

### 5.1 Modélisation permettant l'ordonnancement de tâches phytosanitaires

#### 5.1.1 Motivations pour l'utilisation des automates temporisés

Dans son article "Scheduling with timed automata" [AAM06], Y. Abdeddaïm montre qu'il est possible de résoudre des problèmes d'ordonnancement en modélisant ceux-ci sous forme d'automates temporisés. Plus concrètement, elle prouve qu'en définissant chaque tâche à ordonnancer dans le problème "*job-shop scheduling*" sous la forme d'un automate, on peut trouver une solution exacte à l'aide d'un model-checker.

Ainsi, nous allons définir notre problème d'ordonnancement à partir des contraintes liés à la configuration d'une exploitation et l'application des règles phytosanitaires.

#### 5.1.2 Hypothèses de départ

Les choix faits pour notre modèle sont les suivants :

- Les jours de congés y compris fériés et dimanches sont considérés comme travaillés. Il serait raisonnable de les considérer comme non travaillés en enrichissant le modèle actuel dans une prochaine étude.
- La durée de travail journalière est de 10 heures non-stop.
- Nous ignorons dans notre modèle le parcours entre le lieu de garage (ie l'endroit où il est entreposé lorsqu'il n'est pas utilisé) du tracteur et du pulvérisateur et sa première utilisation. Il en sera de même pour les parcours relatif à un ravitaillement de la citerne contenant un produit à pulvériser.
- Lorsqu'un traitement doit être interrompu du fait du dépassement du nombre d'heures journalières, il est supposé reprendre le lendemain exactement là où il s'est arrêté. C'est à dire que l'on fait comme si la tâche était continue et exécutée sans interruption bien qu'elle soit en réalité préemptive
- L'exploitant ne dispose que d'un seul tracteur de pulvérisation pour le traitement des parcelles.

### 5.1.3 Méthodologie adoptée

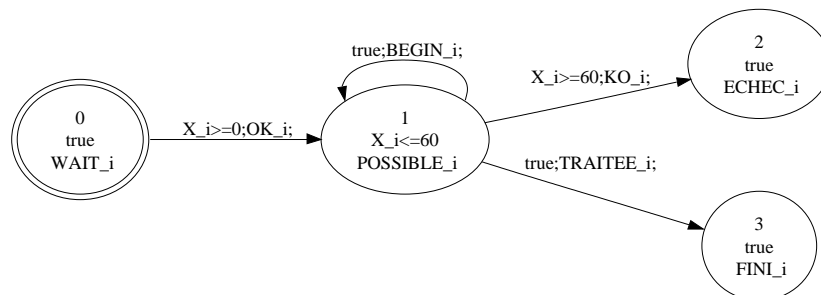
#### Les fenêtres temporelles de pulvérisation

Nous avons montré dans le chapitre précédent comment obtenir à l'aide de l'outil Kronos les fenêtres temporelles de traitements sur les parcelles. Du fait de la modélisation elle-même, ces fenêtres sont constituées de deux dates  $\Delta_1$  et  $\Delta_2$  exprimées en jours. Or un ordonnancement sera calculé en heures de travail passées sur les parcelles puisque le traitement même d'une parcelle est exprimé en heure. Il nous faut par conséquent exprimer nos deux dates  $\Delta_1$  et  $\Delta_2$  en heures. Pour cela nous prenons simplement ces dates et les multiplions par un coefficient 10 correspondant au nombre d'heures effectivement travaillées. Nous obtenons alors une fenêtre temporelle  $[\delta_1, \delta_2]$  relativement aux heures travaillées.

#### Exemple :

Pour  $(\Delta_1, \Delta_2) = (6, 9)$  on obtient  $(\delta_1, \delta_2) = (60, 90)$  qui signifie simplement que le traitement correspondant peut se faire entre la soixantième heure de travail et la quatre-vingt dixième.

#### Automate fenêtre temporelle :



Après obtention d'une date effective de traitement (ie après résolution du problème ordonnancement) nous avons une date de début de traitement que nous convertissons en jours en tronquant tout simplement au jour correspondant. Rappelons que Kronos n'admet que des entiers comme valuations d'horloges. Le résultat est alors reporté dans l'automate règle correspondant à la parcelle.

#### Exemple :

Pour une date de début de tâche  $d = 78$  d'une parcelle  $P$  qui signifie qu'elle débute à la soixante-dix huitième heure de travail,  $\lfloor d \rfloor = 70$  et s'exécute donc le septième jour de la saison pour  $P$ . Et cela est à reporter sur l'automate règle de  $P$ .

#### Les parcours de distances géographiques entre parcelles

Les parcours des distances géographiques entre les parcelles peuvent induire plus ou moins de temps supplémentaire dans un ordonnancement de traitements. En effet, ces temps de parcours ne sont pas négligeables sachant que le temps de traitement d'une parcelle doit être de l'ordre d'une heure et que le trajet entre deux parcelles peut excéder un quart d'heure voire plus. Nous considérerons dorénavant ces parcours comme des tâches de déplacement de durées positives ou nulles. De plus, nous avons encore un problème dû à la granularité des horloges. Nous rappelons que nos fenêtres temporelles sont exprimées en intervalles d'heures sur lesquels des traitements sont possibles alors que les parcours se comptent plutôt minutes. Afin de pallier ce problème nous choisissons une granularité d'une unité d'horloge par tranche de cinq minutes. Maintenant, il nous faut donc exprimer nos durées de parcours et de bornes des fenêtres temporelles en unités de cinq minutes.

## Les surfaces parcellaires

Les parcelles n'ont pas toute la même surface et auront donc une durée de traitement différente d'une parcelle à l'autre. Comme au paragraphe précédent nous exprimerons les temps de traitement des parcelles en unités de cinq minutes. Cela nous donne une granularité que nous estimons suffisante pour modéliser notre problème.

## 5.2 Première résolution : "lots temporels"

Nous pouvons regrouper les parcelles concernées par un traitement sur une période donnée. En effet, si nous décidons d'ordonnancer des traitements sur les 5 jours à venir, il suffit de prendre les parcelles dont les traitements seront à faire sur cette période. Concrètement, les parcelles dont les fenêtres temporelles ont une intersection non vide avec la période  $[o_1, o_2]$  à considérer sont sélectionnées. Cela permet concrètement de se limiter à la recherche d'une solution d'ordonnancement concernant une période spécifique au lieu de s'intéresser directement à un ordonnancement sur la saison entière.

Si l'on veut résoudre un ordonnancement sur une plus grande période et que la combinatoire ne le permet pas, il suffira alors de découper cette période en plusieurs autres de plus petite durée et de résoudre les problèmes séparément avant de "recoller" les ordonnancements obtenus en un seul.

Les grosses difficultés du problème résident dans le fait que les tâches ont des périodes de disponibilité (ie les fenêtres temporelles) et également que les parcours entre parcelles sont obligatoires. En effet, dans un ordonnancement réalisable, l'exécution d'une tâche de traitement est toujours suivie d'une tâche de déplacement à une autre parcelle qui n'est pas forcément de durée nulle (parcelles voisines). En d'autres termes, nous avons des relations de précédence entre les tâches de traitement des parcelles et celles de parcours de distances et cela doit être pris en compte dans notre étude.

### 5.2.1 Automate de tâches

Les parcours sont considérés comme des tâches à effectuer entre deux tâches de traitements sur des parcelles. Il nous faut donc avoir une matrice de distances nous donnant les distances en chaque couple de parcelles. Cette matrice doit évidemment respecter l'inégalité triangulaire.

Nous ajoutons un sommet initial "Start" relié à tous les sommets "traitement" qui modélisera le fait qu'un ordonnancement peut commencer par n'importe quelle parcelle. Ce sommet aura comme invariant "true" puisque la date de début des traitements n'est pas fixée *a priori*, dans cet automate ci. Il sera relié aux sommets de traitements par des arcs sortant.

Un sommet tâche de traitement  $EVTTRAITER_i$  a comme invariant " $T \leq \delta_i$ " pour modéliser le fait que le traitement de la parcelle  $P_i$  dure au plus  $\delta_i$  avec  $T$  étant une horloge.

Un sommet " $idle_i$ " est ajouté en sortie de chaque sommet de tâche de traitement et modélisera le fait qu'il peut s'écouler du temps avant de passer à la prochaine parcelle. Cette considération est évidente lorsque les fenêtres temporelles sont disjointes. L'invariant de ce sommet est donc 'true'. L'arc reliant ce sommet à la tâche de traitement correspondante a comme garde " $T \geq \delta_i$ " pour spécifier la durée minimale de la tâche.

Un sommet "Echec" représentera l'échec dans la résolution du problème d'ordonnancement. Pour chaque sommet  $EVTTRAITER_i$ , on crée un arc sortant le reliant à "Echec". Si au cours de l'exécution d'une tâche de traitement survient la date échue correspondant à sa fenêtre temporelle, on veut pouvoir spécifier que l'ordonnancement de cette tâche a échoué.

Enfin, à chaque entrée dans un sommet tâche, l'horloge  $T$  est remise à zéro.

L'automate de tâches est généré automatiquement et prend en entrée une matrice de distance en durées et les temps de traitements. Il est mis en annexe de ce mémoire à la figure C.2.

## 5.2.2 Les événements de synchronisation pour l'ordonnement

Soit  $P_i$  la parcelle et  $t_i$  la tâche de traitement correspondante.

Les synchronisations se font à l'aide des événements  $BEGIN_i$ ,  $KO_i$  et  $TRAITEE_i$  des automates tâches et fenêtre temporelle de  $P_i$ .  $BEGIN_i$  est émis tant que l'on se situe dans un état où la proposition  $POSSIBLE_i$  de l'automate fenêtre temporelle de  $P_i$  est vérifié. Dès qu'il y a dépassement de la borne supérieure de cette fenêtre, l'événement  $KO_i$  est émis. Sinon la tâche  $t_i$  fait parti d'un ordonnancement réalisable, c'est à dire qu'il existe un ordonnancement dans lequel cette tâche est réalisée. Pour ce dernier cas, un événement  $TRAITEE_i$  sera émis par l'automates de tâches.

## 5.3 Résolution du problème d'ordonnement

Par produit des automates "fenêtres Temporelles" avec l'automate de tâches, nous obtenons tous les états atteignables avec synchronisation des différents événements dans les automates. Pour cinq parcelles, nous obtenons l'automate produit comprend 1787 états et 4210 transitions.

Chaque automate fenêtre temporelle comporte deux sommets vérifiant respectivement les prédicats " $FINI_i$ " et " $ECHEC_i$ " avec  $i$  étant l'indice d'une parcelle. Un ordonnancement réalisable correspond donc à un état où tous les automates fenêtres temporelles passent dans leur état  $FINI_i$  respectifs. Sinon, au moins un d'entre eux vérifiera l'état " $ECHEC_i$ ". Soit "Res" l'automate résultant du produit. Nous pouvons donner une réponse au problème de l'existence ou non d'un ordonnancement par la requête suivante pour trois parcelles :

*Kronos -DFS -R "FINI<sub>1</sub> and FINI<sub>2</sub> and FINI<sub>3</sub>" Res.tg*

Cela correspond simplement à une requête d'atteignabilité des sommets vérifiant " $FINI_1$  and  $FINI_2$  and  $FINI_3$ ". En cas d'échec, cela veut simplement dire qu'un tel état n'est pas atteignable (ie l'ordonnement n'est pas possible). En cas de succès, par analyse du fichier résultats "reach.trace" de Kronos, on retrouve, à l'aide des valuations d'horloges et identification des diverses transitions prises, l'ordonnement correspondant.

Pour obtenir tous les ordonnancement possibles, il faut utiliser l'option "*-FULLDFS - allpaths*") et analyser le fichier résultats "dfs.trace" pour obtenir les différents ordonnancement possibles. Cela donne la possibilité de choisir le parcours de tâches pour l'exploitant en fonction de la durée du travail engendré par ce parcours ou de ses habitudes.

Précisons qu'à partir d'un chemin donné en résultats et afin de reconstituer l'ordonnement correspondant, Kronos fournit les états atteints et les transitions prises dans chaque automate donné en entrée. C'est à dire, que pour connaître l'ordre de parcours de l'automate "tâches" par exemple, il suffit de lire quel état de cet automate est atteint après chaque transition.

Pour obtenir par contre les dates exactes, une analyse plus précise des inéquations fournies par Kronos est obligatoire.

Un exemple de fichier "reach.trace" est mis en annexes.

## 5.4 Résolution de problèmes plus importants : "lots géographiques"

Pour pallier l'explosion combinatoire du nombre d'états et transitions de l'automate permettant la résolution d'un ordonnancement (respectivement 1787 et 4210 dans le cas de 5 parcelles seulement), on propose quelques heuristiques.

**Regrouper en zone :** Pour un nombre de parcelles trop important (générant un grand nombre d'états et transitions), on peut regrouper les parcelles limitrophes ou peu éloignées en "zones géographiques" et par la suite considérer une zone comme une seule et même parcelle et la distance entre deux zones correspondrait à la plus petite distance entre deux parcelles des deux zones considérées. Pour obtenir les zones, des algorithmes d'approximation pour le problème dit "k-clustering" [AP98] peut être utilisé pour déterminer k zones distinctes .

**Simplifier le problème des temps de parcours par un temps borné :** On peut ajouter à la durée de traitement d'une parcelle spécifique une durée correspondant au parcours de la plus grande distance entre cette parcelle et la parcelle la plus éloignée d'elle au sein de l'exploitation. Cela permet de prendre en considération dans l'ordonnancement les durées de parcours entre parcelles et constitue une majoration de la durée réelle de traitement d'une parcelle. Il est clair que si un ordonnancement est réalisable en prenant ces tâches "majorées", il sera également réalisable avec les "vraies" tâches et les réels parcours entre parcelles. Cette méthode permet d'éliminer les sommets "parcours" entre parcelles.

## 5.5 Résolution avec plusieurs pulvérisateurs

L'automate de tâches précédent a été construit en considérant l'existence d'un seul pulvérisateur et sa conception en a été fortement influencée. Nous proposons maintenant une modélisation pour plusieurs pulvérisateurs qui seraient disponibles sur l'exploitation.

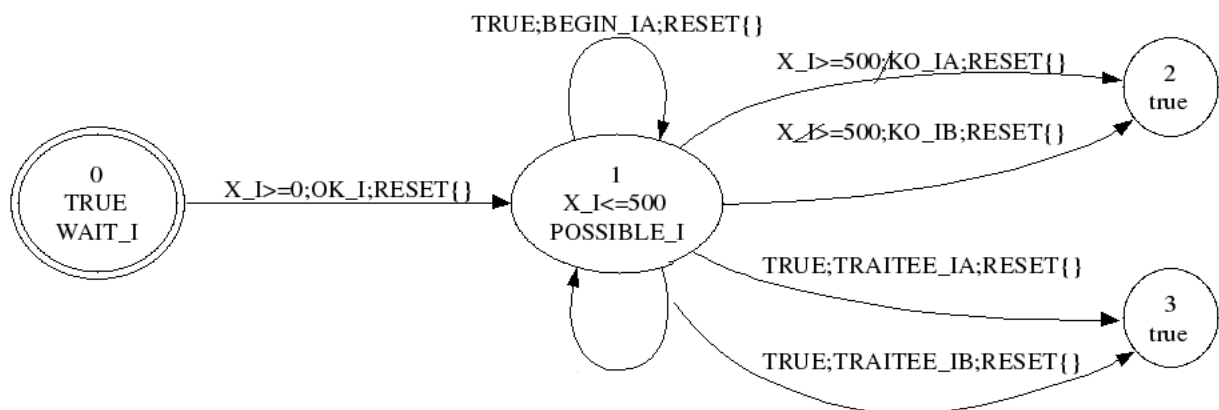
Dans [abdeddaim01], l'auteur préconise une construction en modulaire définie par les tâches préaffectées à des machines numérotées, c'est à dire que l'on connaît à l'avance la machine responsable de l'exécution d'une tâche.

Nous pourrions en faire de même en préaffectant un sous-ensemble de la totalité des parcelles à un pulvérisateur précisément identifié par un numéro et répéter ainsi l'opération pour tous les pulvérisateurs. Ensuite, il suffirait de lancer une résolution d'ordonnancement pour chaque tracteur comme expliqué à la section 5.3.

Nous voulons cependant proposer une autre solution évitant de préaffecter des parcelles aux tracteurs. En effet, en supposant que tous les pulvérisateurs sont complètement équivalents, on voudrait simplement résoudre un ordonnancement avec répartition automatique des parcelles aux tracteurs.

En considérant qu'un tracteur particulier évolue indépendamment d'un autre et qu'il puisse se rendre sur toutes les parcelles de l'exploitation, son automate "tâches" serait le même celui de la sous-section 5.2.1. Il suffit, à partir de là, de différencier les événements de synchronisation des arcs. En effet les événements présentés à la sous-section 5.2.2 seront spécifiques à chaque tracteur. En d'autres termes, les événements  $BEGIN_i$ ,  $KO_i$  et  $TRAITEE_i$  deviennent  $BEGIN_{iA}$ ,  $KO_{iA}$  et  $TRAITEE_{iA}$  pour le tracteur A.

L'automate "fenêtre temporelle" pour deux pulvérisateurs devient alors :



Pour résoudre maintenant un problème d'ordonnancement avec deux pulvérisateurs, il faut deux automates tâches et autant d'automates "fenêtres temporelles" que de parcelles en apportant les modifications précédentes. Nous pouvons donner une réponse au problème de l'existence ou non d'un ordonnancement par une requête "à la volée" pour  $N$  parcelles :

*Kronos -DFS -R "FINI<sub>1</sub> and FINI<sub>2</sub> and ... and FINI<sub>N</sub>" f<sub>1</sub>.tg f<sub>2</sub>.tg ... f<sub>N</sub>.tg tachesA.tg tachesB.tg*

Cependant Kronos fournit en résultat un ordonnancement ne prenant en compte qu'un seul tracteur qui correspond au plus court en nombre de transitions, si le travail est réalisable avec un seul tracteur. Il convient alors d'utiliser la commande *-FULLDFS -allpaths* pour parcourir tous les chemins et déterminer ensuite le plus court en terme de temps si celui-là nous intéresse.

Le gros problème reste la combinatoire : pour 5 parcelles, plus de 200000 chemins sont produits dans le fichier résultat généré par Kronos. Cela est normal puisque l'on peut ramener ce problème au problème "partition" consistant à répartir un ensemble d'objets ayant chacun un poids (ou un temps d'exécution) en deux sous-ensembles de même poids. Et ce problème est connu pour être NP-Complet. On peut alors surmonter l'obstacle de la combinatoire par des heuristiques telles que celles présentées à la sous-section 5.4.

Nous avons montré dans ce chapitre qu'il était possible de résoudre notre problème d'ordonnancement en utilisant encore les automates temporisés et toute leur sémantique. Notre méthode permet une construction modulaire à partir des contraintes liées à notre problème. Cependant, comme dans la majorité des problèmes d'ordonnancement, la combinatoire devient vite un obstacle que nous avons contourné en proposant des heuristiques permettant d'approximer la solution.

## Chapitre 6

# Limites actuelles de la modélisation et perspectives

### Automates règles

Comme nous le disions à la section 4.3.1 l'identification des processus impliqués dans une règle, par lecture d'un diagramme d'une dizaine d'états (cf B.1), créé par des pathologiste n'est pas aisée. Pourtant il s'agit ici d'une règle simple concernant la gestion à la parcelle d'une maladie fongique de la vigne appelé "Mildiou". (*plasmopara viticola*). Dans une autre règle, dont la formalisation mathématique est en cours, et qui vise à gérer simultanément mildiou et oïdium (*erysiphe necator*), il y a déjà plus d'une trentaine d'états dans le diagramme correspondant de type state-charts <sup>1</sup>, qu'il faudra par la suite transformer en règle sous forme d'automate temporisé.

La suite du travail devrait être de comparer, dans différentes configurations d'exploitations viticoles, et avec différents historiques climatiques, l'application de différentes règles et leurs conséquences en termes d'organisation. Un élément d'organisation qui a été peu évoqué dans ce mémoire concerne les observations de maladie, qui sont coûteuses en temps, et donc en ressources humaines. Une des difficultés pour prendre en compte cet aspect observation est d'élaborer des scénarios de développement des maladies couplés avec les réactions aux interventions chimiques.

### Difficultés de la modélisation des automates avec Kronos

1. Pour la modélisation des automates, les invariants de sommets et gardes de transitions ont parfois été orienté par les résultats que l'on voulait obtenir par l'outil Kronos. Par exemple, pour l'automate pluPrevue de la section 4.3.2, les invariants de sommet autres que "true" sont obligatoires afin d'obtenir les dates limites de traitement des parcelles par analyse du fichier résultat de Kronos. En effet, la seule spécification d'une garde de transition de sortie comportant une date de pluie prévue ne suffit pas pour la résolution car pour avoir une trace de valeurs des horloges, l'algorithme implémenté dans Kronos ne se base que sur les valeurs des horloges dans les sommets. Cette apparente lacune dans l'expression des traces s'explique sûrement par le fait qu'une transition est prise de manière instantanée dans la sémantique des automates temporisés et que l'écoulement du temps (à travers les valeurs d'horloges) n'a véritablement de sens que dans les sommets des automates. On notera au passage que les remises à zéro d'horloges ne figurent pas non plus dans les fichiers traces générés par Kronos.

---

<sup>1</sup>Formalisme introduit par Harel en 1987

2. Les différences de granularité entre les horloges des différents automates ont été prises en compte dans notre modèle. Lors de la multiplication par coefficients constant des valeurs numériques des horloges des automates, Kronos a accepté des valeurs au dessus de 16384 (ie  $2^{14}$ ) mais fournit des résultats faux du fait que la représentation des horloges ne devait en fait dépasser 16384. Or cette limitation du logiciel n'était spécifié nulle part.
3. L'explosion combinatoire empêche un traitement du produit de l'ensemble des automates obtenus dans la modélisation. Par exemple, la résolution de l'ordonnancement pourrait être exprimée sur toute la saison vinicole en prenant en compte tous les automates fenêtres temporelles et tâches pour toutes les parcelles. Or la résolution par Kronos serait impossible car le nombre d'états et transitions de l'automate résultant du produit serait trop important.
4. Enfin, l'interprétation des résultats générés par le model-checker Kronos est relativement difficile. Les résultats relatifs aux valuations d'horloges sont donnés sous forme d'inéquations alphanumériques dont l'expression ne permet pas une compréhension naturelle directement à la lecture. Nous avons dû transformer ces inéquations pour les rendre exploitables pour notre usage spécifique ou encore résoudre des systèmes linéaires par solveur de programmes linéaires. Cela s'explique par le fait que Kronos n'a pas été conçu à la base pour résoudre des problèmes d'ordonnancement.

## Outil décisionnel

Le modèle proposé permet de décider la mise en oeuvre ou non de règles phytosanitaires en fonction de la configuration d'une exploitation. La configuration correspond au nombre de parcelle, à l'évolution de ces parcelles à travers la phénologie (ie les divers stades des plants vignes) et aux ressources disponibles pour une application convenable des règles de gestion à la parcelle. Cela permet d'évaluer le risque par un exploitant avant de se lancer dans l'application effective dans ses parcelles.

Concernant la règle modélisée en section 4.1, le facteur le plus important est la prévision de la pluie. Nous avons montré comment obtenir les dates de traitements des parcelles en fonction des événements pluvieux prévus à l'avance par les services météorologiques (cf section 4.3.1). Après instantiation effective des traitements sur les parcelles d'une exploitation donnée au cours d'une saison en fonction de cette météo, il serait intéressant d'analyser les résultats avec les événements pluvieux réels afin d'obtenir des résultats concernant l'efficacité de la règle. Ainsi, on pourrait connaître non seulement l'applicabilité de celle-ci par rapport aux ressources mais également par rapport au climat qui règne dans une zone géographique donnée. A partir de là, une révision de la règle pourrait être faite, avec des tests sur plusieurs saisons, afin de l'adapter au mieux, éventuellement à l'aide d'algorithmes d'apprentissage.

Les diverses traces données par Kronos (ie option *-allpaths*) pour la résolution de l'ordonnancement permettent un choix pour l'organisation convenable au viticulteur. Précisons que ce choix pourrait très bien être l'optimal en temps puisque ce résultat fait parti des chemins résultats. Un travail logiciel est néanmoins nécessaire pour réexprimer les traces issues de Kronos d'une façon plus utilisable.

## Sous-utilisation du model-checking

Dans notre modélisation, la puissance du model-checking a été sous utilisée. En effet, nous avons seulement utilisé des requêtes d'atteignabilité alors que Kronos peut aller plus loin que cela. On pourrait utiliser le model-checking afin d'assurer par exemple l'absence de pathologies sur les parcelles sur toute la saison viticole en fonction des règles de décision et des traitements effectués.

Une telle requête pourrait être : "Pour tous les jours de la saison, mes parcelles ne sont jamais atteintes d'une maladie".

Cela demande bien sûr plus de temps de travail pour la modélisation et les tests correspondant. Nous n'avons pas pu aller jusque là car un gros travail de compréhension et de modélisation a été nécessaire pour en arriver là où nous en sommes aujourd'hui.

## Perspectives techniques et scientifiques

Nous préconisons l'implémentation d'un module permettant une analyse des résultats générés par Kronos afin de les rendre plus "lisibles" pour l'humain.

D'autres approches de résolution de problèmes d'ordonancement sont possibles pour résoudre des problèmes d'ordonancement de tailles importantes en nombre de tâches et ressources (ex : programmation linéaire, CSP, métaheuristiques...etc...). L'idée de notre approche était cependant d'essayer d'utiliser au maximum la sémantique des automates temporisés pour spécifier la totalité des contraintes liées à notre problème.

Nous pouvons continuer à enrichir le modèle actuel en prenant en compte par exemple l'existence des jours non-travaillés tels que les jours fériés ou dimanche et congés.

Il est possible de le faire en modélisant ces nouvelles contraintes par des automates temporisés simples :

- Un automate "semaine" cyclique unique interdisant la continuité du travail les dimanches, un peu à l'image des fenêtre temporelles qui oblige à ne traiter une parcelle que sur période déterminée.
- Les jours fériés n'étant pas du tout cycliques, on aurait un automate par jour férié.
- Les congés seraient des automates à construire au fur et à mesure des demandes de congés d'un travailleur.

Pour améliorer la recherche de solution de certains problèmes modélisés par les automates temporisés, nous pourrions modifier nous même le produit des automates généré par Kronos et éliminer les états que l'on qualifierait d'indésirable ou incohérent pour nous relativement à un problème particulier.

A l'instar de C. Largouët dans [Lar00], une modélisation avec automates probabilistes permettrait de mesurer la probabilité d'atteignabilité de certains sommets des automates. Cela pourrait servir par exemple à quantifier le risque pour une exploitation d'adopter ou non les règles de décision. Pour cela, un travail de réalisation d'un modèle épidémiologique simplifié, sous la forme d'automates probabilistes, serait à mener. La propagation spatiale de l'épidémie pourrait être travaillée par des automates cellulaires.

NB : Les automates cellulaires probabilistes existent, mais c'est un champ théorique (complexe) que nous n'avons pas eu le temps de discuter.

## Conclusion

A l'heure actuelle, bien que l'existence des automates temporisés date de plus de 10 ans, peu de modèles les utilisent à part ceux concernant des protocoles de communication dans les réseaux. Nous avons montré comment l'on pouvait modéliser différentes contraintes d'un problème agricole, toutes relatives au temps, par l'usage exclusif de l'expressivité et de la sémantique des automates temporisés.

Ainsi, après construction de nos modules "contraintes" sous forme d'automates temporisé, les diverses requêtes faites à l'outil de model-checking Kronos permettent de donner une première réponse au problème :

*Est-il possible de mettre en place des règles de gestion à la parcelle dans une exploitation donnée ?*

# Bibliographie

- [AAM06] Y. Abdeddaim, E. Asarin, and O. Maler. Scheduling with timed automata. *Theoretical Computer Science*, 354(2) :272–300, 2006.
- [AD94] Rajeev Alur and David L Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2) :183–235, 1994.
- [AP98] Pankaj K. Agarwal and Cecilia Magdalena Procopiuc. Exact and approximation algorithms for clustering (extended abstract). In *Symposium on Discrete Algorithms*, pages 658–667, 1998.
- [BBF<sup>+</sup>01] Béatrice Bérard, Michel Bidoit, Alain Finkel, François Laroussinie, Antoine Petit, Laure Petrucci, and Philippe Schnoebelen. *Systems and Software Verification. Model-Checking Techniques and Tools*. Springer, 2001.
- [BDCA03] Beyer, Lewerentz Dirk, Noack Claus, and Andreas. Rabbit : A tool for bdd-based verification of real-time systems. In Warren A. Hunt, Jr. and Fabio Somenzi : Proceedings of the 15th International Conference on Computer Aided Verification (CAV 2003), LNCS 2725, pages 122-125, Springer-Verlag, Berlin, 2003, ISBN 3-540-40524-0, 2003.
- [BS00] Béatrice Bérard and Luis Sierra. Comparing verification with HyTech, Kronos and Uppaal on the railroad crossing example. (LSV-00-2), January 2000.
- [CE81] Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs*, pages 52–71, 1981.
- [Cle04] M. Clerjeau. Le problème de la décision des interventions phytosanitaires en protection intégrée de la vigne. In *Innovigne et Vin*, Gruissan (F), 2004.
- [DOTY95] C. Daws, A. Olivero, S. Tripakis, and S. Yovine. The tool KRONOS. In *Hybrid Systems III : Verification and Control*, volume 1066, pages 208–219, Rutgers University, New Brunswick, NJ, USA, 22–25 October 1995. Springer.
- [Hel03] A. Helias. *Agrégation/Abstraction de modèles pour l’analyse et l’organisation de réseaux de flux : application à la gestion des effluents d’élevage à la réunion*. PhD thesis, Ecole National Supérieure Agronomique de Montpellier, 2003.
- [HHWT97] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. HYTECH : A model checker for hybrid systems. *International Journal on Software Tools for Technology Transfer*, 1(1–2) :110–122, 1997.
- [Kat98] Joost-Pieter Katoen. Concepts, algorithms and tools for model-checking. Lecture Notes, University of Twente, september 1998.
- [Lar00] C. Largouët. *Modélisation par automates temporisés pour aider à l’idenfication de l’occupation du sol*. PhD thesis, Ecole National Supérieure Agronomique de Rennes., 2000.

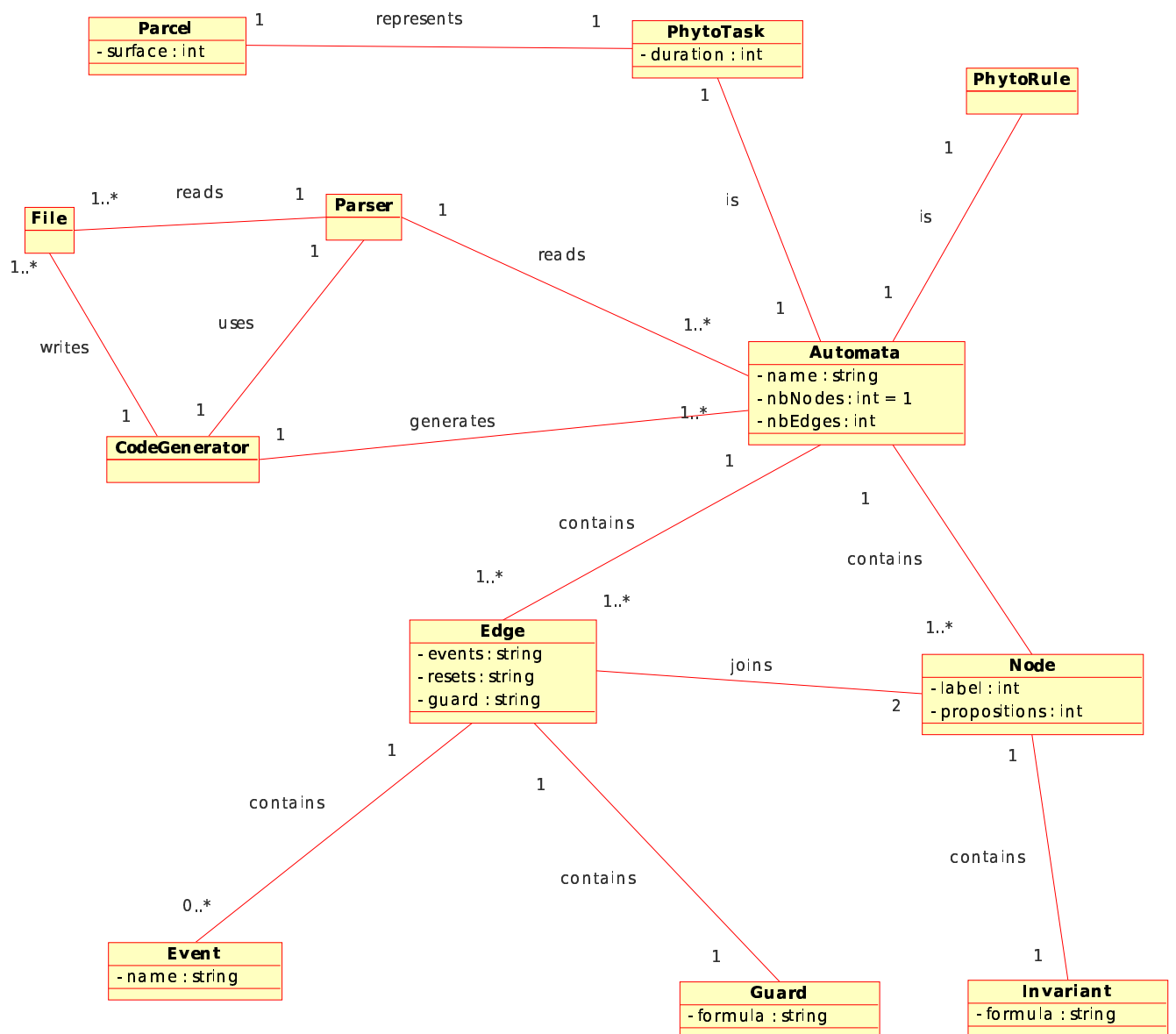
- [LPY97] Kim G. Larsen, Paul Pettersson, and Wang Yi. UPPAAL in a Nutshell. *Int. Journal on Software Tools for Technology Transfer*, 1(1–2) :134–152, October 1997.
- [Pnu85] Amir Pnueli. Linear and branching structures in the semantics and logics of reactive systems. In *ICALP*, pages 15–32, 1985.
- [Tri98] Stavros Tripakis. *L'analyse formelle des systèmes temporisés en pratique*. PhD thesis, Université Joseph Fourier de Grenoble, 1998.
- [TXJS92] T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic Model Checking for Real-Time Systems. In *7th. Symposium of Logics in Computer Science*, pages 394–406, Santa-Cruz, California, 1992. IEEE Computer Science Press.
- [YO93] S. Yovine and A. Olivero. Kronos : A tool for verifying real-time systems — user's guide and reference manual, 1993. Laboratoire VERIMAG de Grenoble, <http://cite-seer.ist.psu.edu/olivero93kronos.html>.
- [Yov93] S. Yovine. *Méthodes et outils pour la vérification symbolique de systèmes temporisés*. PhD thesis, Institut National Polytechnique de Grenoble., 1993.

## Autres références à consulter

- [AM02] Yasmina Abdeddaim and Oded Maler. Preemptive job-shop scheduling using stopwatch automata. In *Tools and Algorithms for Construction and Analysis of Systems*, pages 113–126, 2002.
- [DBL02] H. Dierks, G. Behrmann, and K. G. Larsen. Solving planning problems using real-time model checking. 2002.

# Annexe A

## Schéma UML de l'application développée



# Annexe B

## Diagramme de règle

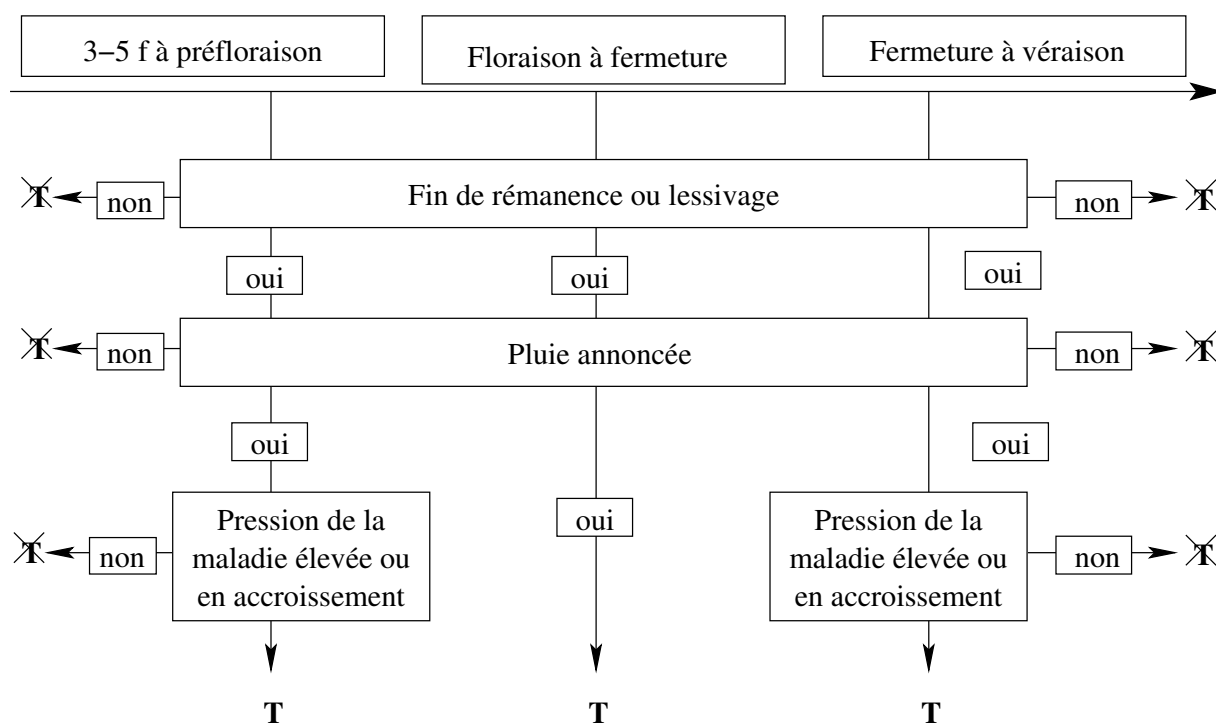


FIG. B.1 – Règle de traitement du "Mildiou"

# Annexe C

## Figures

### C.1 Automate regle modifiée après réalisation de deux traitements

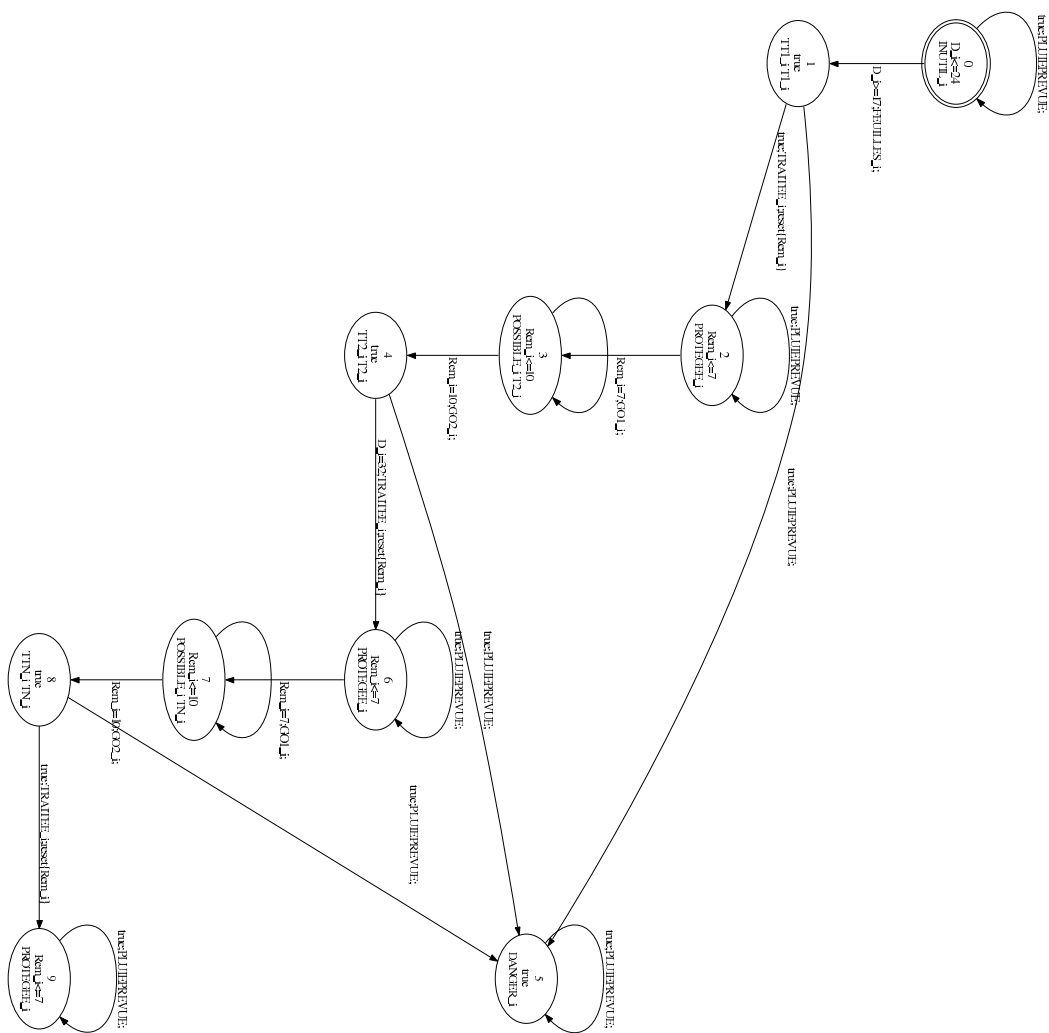


FIG. C.1 – Règle mise à jour de gestion du Mildiou

## C.2 Automate tâches

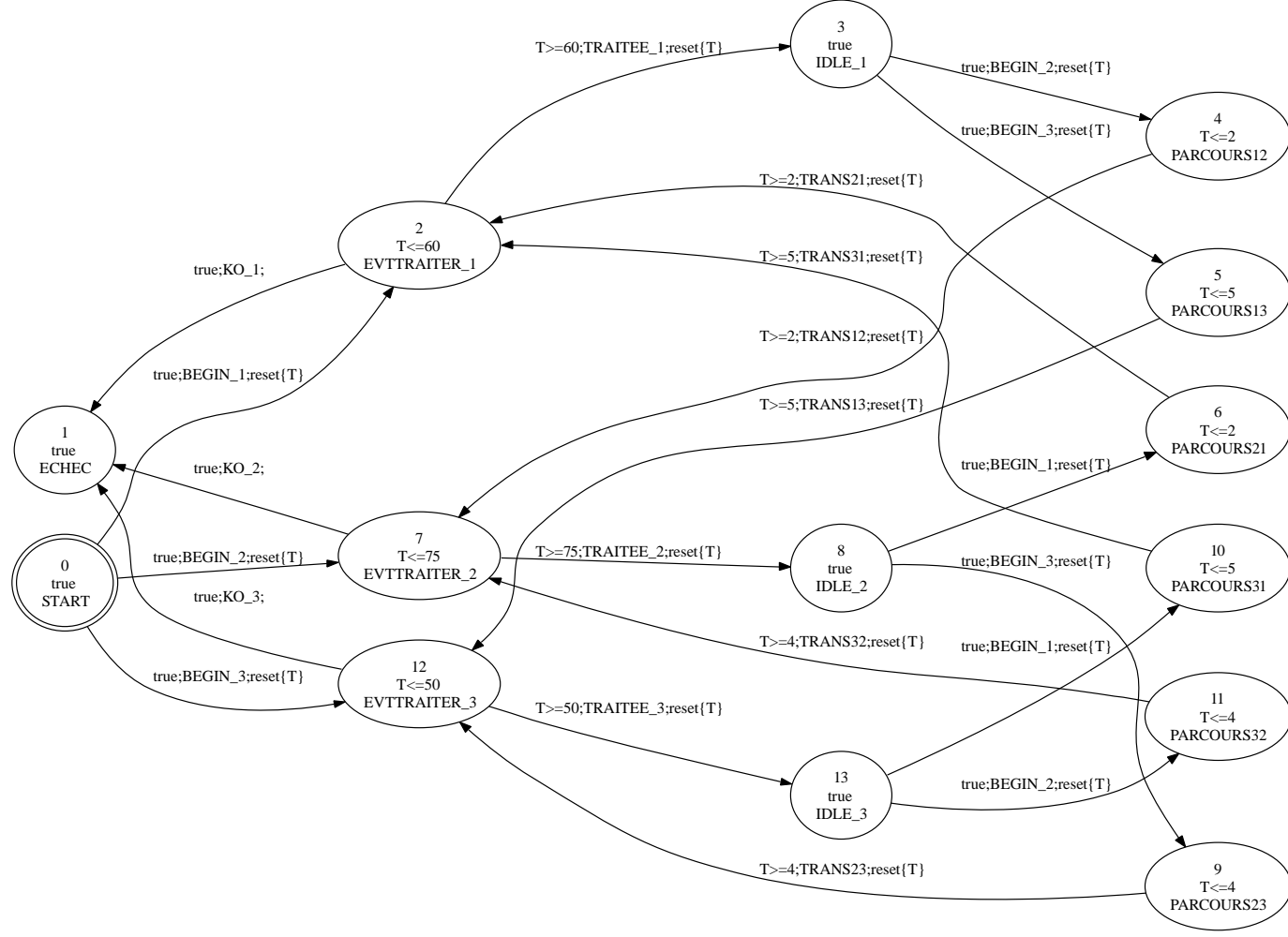


FIG. C.2 – Exemple d’automate de tâches pour 3 parcelles

# Annexe D

## Résultats Kronos

### D.1 Résultat d'une requête sur le produit "*pluiePrevue* × *regle*" :

- Trace #1 –  
< 22, 20<D and D<=24>
- Trace #2 –  
< 28, 20<D and D<=25>
- Trace #3 –  
< 34, 20<D and D<=26>
- Trace #4 –  
< 40, 20<D and D<=27>
- Trace #5 –  
< 46, 20<D>

La requête Kronos sur l'automate appelé "Res" est :

*Kronos – FULLDFS – allpaths – R "TT1" – I "D > 20 and TT1" Res.tg*

La fenêtre temporelle correspondant à ces traces est l'intervalle [20, 24] car on effectue une minimisation des bornes supérieures. La cinquième trace n'a tout simplement pas de borne supérieur et cela est équivalent à  $+\infty$

## D.2 Résultat d'une requête "on the fly" sur des automates "fenêtres temporelles" et "tâches" :

– Trace #1 –

$\langle (0, 0, 0, 0, 0, 0), X1=X2 \text{ and } X1=X3 \text{ and } X1=X4 \text{ and } X1=X5 \text{ and } X1=T \rangle \text{ — OK1 —} \langle (1, 0, 0, 0, 0, 0), X1 \leq 60 \text{ and } X1=X2 \text{ and } X1=X3 \text{ and } X1=X4 \text{ and } X1=X5 \text{ and } X1=T \rangle \text{ — BEGIN1 —} \langle (1, 0, 0, 0, 0, 2), X1 \leq 60 \text{ and } X1=X2 \text{ and } X1=X3 \text{ and } X1=X4 \text{ and } X1=X5 \text{ and } T \leq X1 \rangle \text{ — OK2 —} \langle (1, 1, 0, 0, 0, 2), X1 \leq 60 \text{ and } 60 \leq X2 \text{ and } X2 \leq X1 \text{ and } X1=X3 \text{ and } X1=X4 \text{ and } X1=X5 \text{ and } T \leq X1 \rangle \text{ — TRAITEE1 —} \langle (3, 1, 0, 0, 0, 3), 60 \leq X2 \text{ and } X2 \leq 137 \text{ and } X2 \leq X1 \text{ and } X1 \leq X3 \text{ and } X1 \leq X4 \text{ and } X1 \leq X5 \text{ and } X1=T+60 \text{ and } X3 \leq X2 \text{ and } X4 \leq X2 \text{ and } X5 \leq X2 \rangle \text{ — BEGIN2 —} \langle (3, 1, 0, 0, 0, 4), X2 \leq 137 \text{ and } T \leq 2 \text{ and } X1=X2 \text{ and } X1=X3 \text{ and } X1=X4 \text{ and } X1=X5 \text{ and } X1 \leq T+137 \text{ and } T+60 \leq X1 \rangle \text{ — TRANS12 —} \langle (3, 1, 0, 0, 0, 9), X2 \leq 137 \text{ and } T \leq 75 \text{ and } X1=X2 \text{ and } X1=X3 \text{ and } X1=X4 \text{ and } X1=X5 \text{ and } X1 \leq T+137 \text{ and } T+62 \leq X1 \rangle \text{ — OK5 —} \langle (3, 1, 0, 0, 1, 9), X2 \leq 137 \text{ and } 137 \leq X5 \text{ and } T \leq 75 \text{ and } X1=X2 \text{ and } X1=X3 \text{ and } X1=X4 \text{ and } X1=X5 \text{ and } X1 \leq T+137 \text{ and } T+62 \leq X1 \rangle \text{ — TRAITEE2 —} \langle (3, 3, 0, 0, 1, 10), 137 \leq X1 \text{ and } X5 \leq 227 \text{ and } X1=X2 \text{ and } X1=X3 \text{ and } X1=X4 \text{ and } X1=X5 \text{ and } X1=T+137 \rangle \text{ — BEGIN5 —} \langle (3, 3, 0, 0, 1, 13), X5 \leq 227 \text{ and } T \leq 5 \text{ and } X1=X2 \text{ and } X1=X3 \text{ and } X1=X4 \text{ and } X1=X5 \text{ and } X1 \leq T+227 \text{ and } T+137 \leq X1 \rangle \text{ — TRANS25 —} \langle (3, 3, 0, 0, 1, 30), X5 \leq 227 \text{ and } T \leq 85 \text{ and } X1=X2 \text{ and } X1=X3 \text{ and } X1=X4 \text{ and } X1=X5 \text{ and } X1 \leq T+227 \text{ and } T+142 \leq X1 \rangle \text{ — OK3 —} \langle (3, 3, 1, 0, 1, 30), 227 \leq X3 \text{ and } X5 \leq 227 \text{ and } T \leq 85 \text{ and } X1=X2 \text{ and } X1=X3 \text{ and } X1=X4 \text{ and } X1=X5 \text{ and } X1 \leq T+227 \text{ and } T+142 \leq X1 \rangle \text{ — TRAITEE5 —} \langle (3, 3, 1, 0, 3, 31), 227 \leq X1 \text{ and } X3 \leq 278 \text{ and } X1=X2 \text{ and } X1=X3 \text{ and } X1=X4 \text{ and } X1=X5 \text{ and } X1=T+227 \rangle \text{ — BEGIN3 —} \langle (3, 3, 1, 0, 3, 28), X3 \leq 278 \text{ and } T \leq 1 \text{ and } X1=X2 \text{ and } X1=X3 \text{ and } X1=X4 \text{ and } X1=X5 \text{ and } X1 \leq T+278 \text{ and } T+227 \leq X1 \rangle \text{ — TRANS53 —} \langle (3, 3, 1, 0, 3, 16), X3 \leq 278 \text{ and } T \leq 50 \text{ and } X1=X2 \text{ and } X1=X3 \text{ and } X1=X4 \text{ and } X1=X5 \text{ and } X1 \leq T+278 \text{ and } T+228 \leq X1 \rangle \text{ — OK4 —} \langle (3, 3, 1, 1, 3, 16), X3 \leq 278 \text{ and } 278 \leq X4 \text{ and } T \leq 50 \text{ and } X1=X2 \text{ and } X1=X3 \text{ and } X1=X4 \text{ and } X1=X5 \text{ and } X1 \leq T+278 \text{ and } T+228 \leq X1 \rangle \text{ — TRAITEE3 —} \langle (3, 3, 3, 1, 3, 17), 278 \leq X1 \text{ and } X4 \leq 359 \text{ and } X1=X2 \text{ and } X1=X3 \text{ and } X1=X4 \text{ and } X1=X5 \text{ and } X1=T+278 \rangle \text{ — BEGIN4 —} \langle (3, 3, 3, 1, 3, 18), X4 \leq 359 \text{ and } T \leq 1 \text{ and } X1=X2 \text{ and } X1=X3 \text{ and } X1=X4 \text{ and } X1=X5 \text{ and } X1 \leq T+359 \text{ and } T+278 \leq X1 \rangle \text{ — TRANS34 —} \langle (3, 3, 3, 1, 3, 23), X4 \leq 359 \text{ and } T \leq 80 \text{ and } X1=X2 \text{ and } X1=X3 \text{ and } X1=X4 \text{ and } X1=X5 \text{ and } X1 \leq T+359 \text{ and } T+279 \leq X1 \rangle \text{ — TRAITEE4 —} \langle (3, 3, 3, 3, 3, 24), 359 \leq X1 \text{ and } X1=X2 \text{ and } X1=X3 \text{ and } X1=X4 \text{ and } X1=X5 \text{ and } X1=T+359 \rangle$

La requête Kronos sur les automates fenêtres temporelles et tâches concernant 5 parcelles est :

*kronos - v - DFS - R "FINI1 and FINI2 and FINI3 and FINI4 and FINI5" F1.tgF2.tgF3.tgF4.tgF5.tgTaches.*

On obtient au final un parcours dans les états des automates correspondant. Par exemple, ( 3, 3, 3, 3, 24 ) signifie que les  $F_i$  sont tous dans leurs états respectifs numérotés 3 et que l'automate tâches est dans l'état numéroté 24. Ainsi une reconstitution du parcours des parcelles est possible. Avec une analyse minutieuse des horloges, on peut connaître les dates de début et de fin des tâches de traitement et de parcours entre parcelles.

## Résumé :

Des traitements chimiques sont nécessaires tout au long de la saison viticole pour protéger la vigne contre diverses maladies. Lors de ces traitements, les produits sont pulvérisés sur les pieds de vigne mais également sur le sol ou dans l'air, et sont donc causes possibles d'une pollution de l'environnement. Afin de réduire le nombre de traitements, des experts ont mis au point de nouvelles règles. Ces règles décident de l'opportunité de traiter une parcelle indépendamment des autres parcelles. Cependant, les viticulteurs traitent le plus souvent leur exploitation entièrement. Donc, avant de promouvoir ces nouvelles règles, il est important de vérifier si les exploitations peuvent supporter les changements dans l'organisation du travail qu'elles susciteraient. Ce mémoire de Master porte sur la modélisation de l'ordonnancement des traitements, avec ressources limitées, par le formalisme des automates temporisés. Nous montrons qu'une règle experte peut être transformée en un automate temporisé et synchronisée avec des prévisions d'événements climatiques pour trouver les intervalles de temps des prochains traitements sur les parcelles. Nous avons utilisé la même technique dans la modélisation des ressources et des contraintes temporelles pour trouver des ordonnancements réalisables prenant en compte les parcours entre parcelles. Pour cela, nous avons appliqué des requêtes de logique temporelle sur les automates à l'aide du logiciel de model-checking Kronos. La recherche sera poursuivie avec deux préoccupations: traiter le problème combinatoire issu de l'analyse simultanée d'un grand nombre de parcelles, par des méta-heuristiques, et vérifier l'applicabilité d'un ensemble d'autres règles expertes.



Direction générale  
Parc de Tourvoie  
BP 44, 92163 Antony cedex  
Tél. 01 40 96 61 21 – Fax 01 40 96 62 95  
Web : <http://www.cemagref.fr>