Ainsi, grâce au théorème 47,

> ϕ est satisfaisable ssi l'ensemble des abrres à branchement au plus d reconnus par l'automate $\mathcal{A}_\phi$ est non vide.

Il en résulte que la satisfaisabilité se ramène, en temps linéaire, au vide d'un automate alternant d'arbres faible sur un domaine d'arbres de branchements $D = [1..d]$.

On procède alors de manière analogue à LTL :

Lemme 50 *Pour tout automate alternant d'arbres $\mathcal{A}$ avec condition de Büchi, on peut calculer en temps déterministe exponentiel un automate de Büchi d'arbres $\mathcal{A}'$ tel que $L(\mathcal{A}) = L(\mathcal{A}')$.*

Preuve: Elle est analogue à la preuve du lemme 16.

Remarquons que ce lemme n'utilise pas la faiblesse de l'automate.

Exercice 42
Si l'automate d'arbres est faible, l'automate déterministe obtenu par le lemme 50 est il toujours faible ?

Exercice 43
Montrer que $A\Diamond\Box P$ n'est logiquement équivalente à aucune formule de CTL. (On pourra utiliser l'exercice 30).

Lemme 51 *Le vide des automates de Büchi d'arbres à branchement borné est PTIME-complet.*

Preuve: Comme dans la preuve du théorème 40, on ramène en temps plolynomial le problème du non-vide au problème du non-vide vide pour les automates alternants de mots sur un alphabet à une lettre, avec condition d'acceptance de Büchi. Ce problème se ramène à son tour à déterminer s'il existe une stratégie gagnante pour le joueur 1 dans un graphe fini G, avec condition de victoire de Büchi.

Pour résoudre ce dernier problème, on calcule le plus grand sous-ensemble S de Q_f tel que de toute configuration de S on peut atteindre S dans le graphe G. Ce plus grand point fixe est calculé par raffinements successifs :

1. Initialement, $S_0 = \emptyset$ et $S_1 = Q_f$
2. Tant que $S_1 \neq S_0$
   (a) Remplacer S_0 par S_1
   (b) calculer le sous-ensemble S_1 de S_0 des configurations à partir desquelles on peut atteindre S_0 dans G
3. S est la valeur commune de S_0 et S_1

Le joueur 1 a une stratégie gagnante si et seulement si S est accessible dans G à partir de l'état initial.

La boucle est parcourue au plus $|Q_f|$ fois et chaque étape utilise au plus $O(|Q_f|)$ calculs d'accessibilité dans un graphe ET/OU, ce qui se fait en temps linéaire d'après le théoreme 32. On obtient ainsi en fin de compte un algorithme cubique (on peut faire mieux en utilisant une méthode un peu moins naïve). $\Box$

Pour une autre preuve de ce lemme ainsi que des références, voir [Tho90].

Theorème 52 *La satisfaisabilité de formules de CTL est un problème DEXPTIME-complet.*

Preuve: Concernant l'appartenance à DEXPTIME, il suffit d'enchainer les résultats qui précèdent : lemme 49, théorème 47, lemme 50, lemme 51. Noter que le branchement borné par $O(|\phi|)$ est utilisé pour construire le graphe fini dans la preuve du lemme 51.

Pour montrer la difficulté pour DEXPTIME, il suffit, étant donnée une machine de Turing alternante M travaillant en espace polynômial, et un mot w, de calculer (en temps polynômial) une formule $\phi_{M,w}$ de CTL qui est satisfaisable si et seulement si M accepte w. La construction est semblable à celle du théorème 20. □

10.4 Principes du model-checking en CTL

Etant donnée une structure de Kripke finie $\mathcal{K}$, on lui associe son dépliage $T_{\mathcal{K}}$ dont les nœuds, au lieu d'être étiquetés par des positions de $\mathcal{K}$, sont étiquetés par les propositions qui sont vraies à ces positions (on remplace s par $f(s)$). Grâce au théorème 47, on obtient l'équivalence :

$\mathcal{K} \models \phi$ si et seulement si $T_{\mathcal{K}} \in L(\mathcal{A})$

Autrement dit, le model-checking se ramène à tester l'appartenance d'un arbre infini à un langage défini par un automate d'arbres alternant faible. Ceci doit être mis en parallèle avec la vérification de modèles pour les formules de LTL : il s'agissait alors de tester l'inclusion de l'ensemble des executions de $\mathcal{K}$ dans le langage défini par $\mathcal{A}_\phi$. Or l'appartenance est un problème a priori plus simple que l'inclusion. Et, en effet, si le model-checking de LTL est PSPACE-complet, le model-checking de CTL est dans PTIME, comme nous allons le voir.

Pour tester l'appartenance de $T_{\mathcal{K}}$ à L, on procède de la manière suivante :

1. construire un automate qui accepte un et un seul arbre : $T_{\mathcal{K}}$
2. construire un automate qui accepte l'intersection de l'automate précédent et de l'automate $\mathcal{A}_\phi$.
3. Tester le vide de l'automate intersection.

Remarquons que l'automate intersection accepte 0 ou 1 arbre : il n'accepte aucun arbre si $\mathcal{K}$ ne satisfait pas ϕ, et il en accepte exactement un sinon.

Le calcul de l'intersection de deux automates est peu complexe (complexité= produit des complexités de chacun des deux automates). Pour obtenir le résultat voulu, il suffit donc de savoir effectuer la dernière opération (test du vide) en temps linéaire. Arrivé à ce point, nous utiliserons trois propriétés de l'automate intersection :

- c'est un automate faible (ce qu'il faudra prouver)
- L'alphabet peut être ignoré, puisqu'on ne s'intéresse qu'à l'existence d'un arbre accepté, sans s'occuper de l'étiquetage des nœuds
- Lorsque l'alphabet est réduit à une lettre, automates d'arbres alternants et automates de mots alternants sont équivalents : les copies de l'arbres envoyées aux états ne sont pas pertinentes. On peut donc considérer qu'il s'agit d'un automate de mots.

Et nous avons vu (corollaire 34) que ce dernier problème peut être résolu en temps linéaire.

10.5 L'automate intersection

Soit $\mathcal{K} = (S, R, s^0, f, \mathcal{P})$ une structure de Kripke, d son branchement maximal, ϕ une formule de CTL sur les variables propositionnelles $\mathcal{P}$. On définit comme suit l'automate alternant $\mathcal{A}_{\mathcal{K},\phi}$.

Soit $\mathcal{A}_\phi = (2^{\mathcal{P}}, Q, \delta_\phi, q_0, Q_f)$ l'automate du théorème 47. $\mathcal{A}_{\mathcal{K},\phi}$ est un automate alternant de mots sur un alphabet à une lettre (lettre désormais omise) dont les états sont $Q \times S$, l'état initial est $< q_0, s^0 >$, la relation de transition $\delta_{\mathcal{K},\phi}$ est définie par :

$$\delta_{\mathcal{K},\phi}(< q, s >) = \delta_\phi(q, f(s))\{(A, q_i) \mapsto \bigwedge_{s \to s'} < q_i, s' >,\ (E, q_i) \mapsto \bigvee_{s \to s'} < q_i, s' >\}$$

La condition d'acceptance de $\mathcal{A}_{\mathcal{K},\phi}$ est une condition de parité telle que l'automate soit faible ; le rang est défini par :

$$r(< \phi, s >) = \begin{cases} 2 \times d(\phi) \text{ si } \phi = \neg(E\phi_1 \,\mathcal{U} \phi_2) \text{ ou } \neg(A\phi_1 \,\mathcal{U} \phi_2) \\ 2 \times d(\phi) + 1 \text{ Sinon} \end{cases}$$

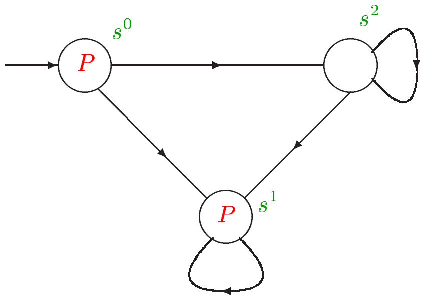

FIG. 23 – Une structure de Kripke

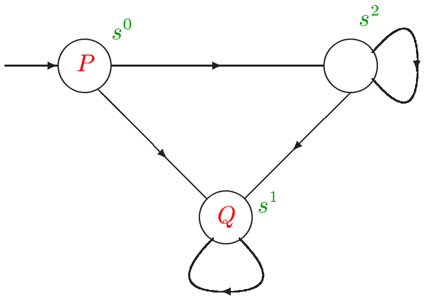

FIG. 24 – Une structure de Kripke

où $d(\phi)$ est le nombre d'imbrications de négations de ϕ. (On peut aussi prendre la profondeur ou la taille de ϕ). La faiblesse de l'automate est assurée par la faiblesse de $\mathcal{A}_\phi$.

Exemple 53 Reprenons l'exemple 48, avec la structure de Kripke donnée dans la figure 23.

Les transitions (pertinentes) de l'automate intersection sont données par :

δ	
$< \phi, s^0 >$	$(< P, s^1 > \vee < P, s^2 >) \wedge (< \phi, s^1 > \vee < \phi, s^2 >)$
$< \phi, s^1 >$	$< P, s^1 > \wedge < \phi, s^1 >$
$< \phi, s^2 >$	$< P, s^1 > \vee < P, s^2 >$
$< P, s^1 >$	**true**
$< P, s^2 >$	**false**

Exercice 44
Calculer l'automate intersection en considérant las tructure de Kripke donnée figure 24 et la formule de l'exemple 40.

Theoreme 54 *$\mathcal{K} \models \phi$ si et seulement si $L(\mathcal{A}_{\mathcal{K},\phi}) \neq \emptyset$.*

Preuve: Supposons tout d'abord que $T_{\mathcal{K}} \in L(\mathcal{A}_\phi)$. Il existe donc un calcul réussi T_r de l'automate sur $T_{\mathcal{K}}$. On construit un calcul ρ de $\mathcal{A}_{\mathcal{K},\phi}$ sur 1^ω (1 est la seule lettre de l'alphabet) de la manière suivante : ρ a même ensemble de positions que T_r et, si $T_r(p) = (q, w)$, alors $\rho(p) =< q, s_w >$, s_w étant défini par récurrence sur la longueur de w par : $s_\epsilon = s^0$, $s_{w \cdot i}$ est le $i$ème successeur de s_w dans $\mathcal{K}$.

Comme T_r est un calcul sur $T_{\mathcal{K}}$, si $T_r(p) = (q, w)$, $T_{\mathcal{K}}(w) = a (= f(s_w))$, alors il existe un ensemble $S = \{(A, q_1), \ldots, (A, q_n), (E, q'_1), \ldots, (E, q'_m)\}$ tel que $S \models \delta_\phi(q, a)$. L'ensemble S' obtenu en remplaçant chaque (A, q_i) par $< q_i, s_1 >, \ldots, < q_i, s_{d(s)} >$ et chaque (E, q_i) par un $< q_i, s_{j_i} >$ tel que $T_r(p \cdot l) = (q_i, w \cdot j_i)$, satisfait $\delta_{\mathcal{K},\phi}(< q, s_w >)$ par définition. ρ est donc un calcul de $\mathcal{A}_{\mathcal{K},\phi}$ sur 1^ω. De plus, comme T_r est un calcul réussi (et que $\mathcal{A}_\phi$ est faible), pour tout chemin infini γ de T_r, il existe une formule ψ_γ de la forme $\neg(A\phi_1 \mathcal{U} \phi_2)$ ou $\neg(E\phi_1 \mathcal{U} \phi_2)$ telle que, pour n suffisamment grand, $\gamma(n) = (\psi_\gamma, w_n)$. Il en résulte que, pour tout chemin infini γ' de ρ, pour n suffisamment grand, $r(\gamma'(n)) = 2 \times d(\psi_\gamma)$ est pair. Donc ρ est un calcul réussi, et donc $L(\mathcal{A}_{\mathcal{K},\phi}) \neq \emptyset$.

Réciproquement, supposons que $L(\mathcal{A}_{\mathcal{K},\phi})$ est non vide : soit ρ un calcul réussi. On construit un calcul T_r sur $T_{\mathcal{K}}$ comme suit. Les positions de T_r sont les mêmes que celles de ρ. $T_r(\epsilon) = (q_0, \epsilon)$ et, si $\rho(p \cdot i) =< q, s >$ et $T_r(p) = (q', w)$, alors $T_r(p \cdot i) = (q, w \cdot j)$ si s est le $j$ème successeur de s_w dans $\mathcal{K}$.

Pour toute position p de T_r, si $T_r(p) = (q, w)$, alors, par récurrence sur $|p|$, w est une position de $T_{\mathcal{K}}$. Soit $a = T_{\mathcal{K}}(w) = f(s_w)$. Par définition, $\rho(p) =< q, s_w >$ et, si $p \cdot 1, \ldots p \cdot n$ sont les successeurs de la position p dans ρ et $\rho(p \cdot i) =< q_i, s_{w \cdot j_i} >$ avec $j_i \in \{1..k\}$ où k est le nombre de successeurs de s_w, alors $\{< q_1, s_{w \cdot j_1} >, \ldots, < q_n, s_{w \cdot j_n} >\}$ satisfait $\delta_{\mathcal{K},\phi}(< q, s_w >)$ (par définition d'un calcul). Maintenant, par définition de $\delta_{\mathcal{K},\phi}$,

$$\begin{array}{rcl} S = & & \{(A, q) \mid \forall i \in \{1, ...k\}, \exists j_i \in \{1, ..., n\}, q = q_{i_j}\} \\ & \cup & \{(E, q) \mid \exists i \in \{1, ..., k\}, \exists j_i \in \{1, ..., n\}, q = q_{i_j}\} \end{array}$$

satisfait $\delta_\phi(q, a, k)$, ce qui montre que T_r est un calcul. Il est réussi car ρ est réussi. □

10.6 Conséquence pour la vérification en CTL

Theorème 55 *Etant donnés $\mathcal{K}$ et $\phi \in CTL$, on peut décider en temps $O(|\mathcal{K}| \times |\phi|)$ si $\mathcal{K} \models \phi$.*

Preuve: $\mathcal{K} \models \phi$ se ramène au vide de $\mathcal{A}_{\mathcal{K},\phi}$ qui est construit en temps $O(|\mathcal{K}| \times |\phi|)$. Cet automate est un automate alternant à parité, faible, sur un alphabet à une lettre. Donc, d'après le corollaire 34, le vide peut en être décidé en temps linéaire, d'où un algorithme de vérification linéaire à la fois en la taille de la spécification et en la taille du modèle. □

Exemple 56 Poursuivons l'exemple 53 ; on obtient la fonction de transition puis sa simplification après marquage :

δ	
q_0	$q_{0,1} \wedge q_{0,2}$
$q_{0,1}$	$q_4 \vee q_5$
$q_{0,2}$	$q_2 \vee q_3$
q_2	$q_2 \wedge q_4$
q_3	$q_4 \vee q_5$
q_4	**true**
q_5	**false**

q_0	**true**
$q_{0,1}$	**true**
$q_{0,2}$	**true**
q_2	q_2
q_3	**true**
q_4	**true**
q_5	**false**

L'état initial q_0 étant déjà marqué, on peut conclure sans aller plus avant. On peut aussi remplacer q_2 par **true** puisqu'il est final.

Exercice 45
Poursuivre l'exercice 44 en montrant comment l'algorithme de décision de la satisfaction fonctionne sur cet exemple.