Travail d’étude
La Logique Temporelle

M’SAKNI CHARFEDDINE
Licence Informatique

18 juin 2004

LOGOS

LTL

LOGIC&UE

CTL

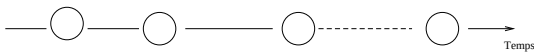
1 Introduction

La logique temporelle est une extension de la logique conventionnelle (propositionnelle), elle intègre de nouveaux opérateurs qui expriment la notion du temps. En effet, par exemple, il n'est pas possible d'exprimer dans la logique classique une assertion liée au comportement d'un programme telle que "après exécution d'une instruction *i*, le système se bloque". Dans cette assertion, les actions s'exécutent suivant un axe de temps : à l'instant *t*, exécution de l'instruction *i*, et à *t + 1* blocage du système. Il faut donc une logique qui modélise les expressions du passé et du futur [1, 8, 9].
On distingue principalement deux classes de logiques :

- Logique du temps linéaire comme *LTL* permettant d'exprimer des propriétés portant sur des chemins individuels (issus de l'état initial) du programme.
- Logique du temps arborescent comme *CTL* permettant d'exprimer des propriétés portant sur les arbres d'exécution (issus de l'état initial) du programme.

2 Logique LTL

La logique temporelle linéaire ou *LTL* « Linear Temporal Logic *(en aglais)*» permet de représenter le comportement des systèmes réactifs au moyen des propriétés qui décrivent le système pour lesquels le temps se déroule linérairement [1, 8, 9]. En clair, on spécifie le comportement attendu d'un système, en spécifiant l'unique futur possible tel que dans *fig1* où une séquence d'actions qui se suivent :



fi g 1 :Le déroulement temporel d'un système dans une logique linéaire[1].

2.1 Syntaxe

Les formules de la logique temporelle linéaire *LTL* sont définies par la grammaire suivante [1, 8, 9] :

– Les propositions atomiques :

$$\Phi, \Psi := p \mid q \mid \ldots \mid \text{true} \mid \text{false} \qquad \Phi, \Psi \in LTL$$

– Les connecteurs booléens :
Soit $\Phi, \Psi \in LTL$, toutes formules de la forme

$$\neg\,\Phi \mid \Phi \wedge \Psi \mid \Phi \vee \Psi \mid \Phi \Rightarrow \Psi \mid \Phi \Leftrightarrow \Psi$$

appartiennent à *LTL*.

– Les connecteurs temporels :
Soit $\Phi, \Psi \in LTL$, toutes formules de la forme

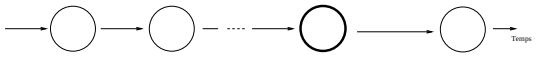
$$\boldsymbol{G}\,\Phi \mid \boldsymbol{F}\,\Phi \mid \Phi\,\boldsymbol{U}\,\Psi \mid \boldsymbol{X}\,\Phi$$

appartiennent à *LTL*.

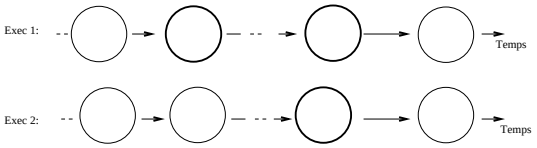
2.2 Sémantique

Notons que, dans les dessins [2] qui suivent, les ronds renforcés noirs expriment que Φ est vrai. Ainsi, on définit les opérateurs temporels comme suit :

1. $\boldsymbol{X}$ (next) $\boldsymbol{X}\,\Phi$: "Φ est vérifié à l'état suivant" :

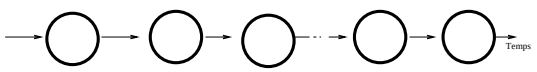


2. $\boldsymbol{F}$ (eventually) $\boldsymbol{F}\,\Phi$: "il existe un état pour lequel Φ est vrai" :



$$exec1, exec2 \models \boldsymbol{F}\Phi$$

3. $\boldsymbol{G}$ (always) $\boldsymbol{G}\,\Phi$: "Φ est toujours vérifié dans le futur" :

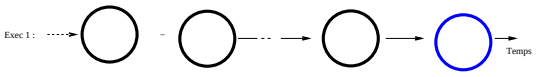


4. ***GF***p : "p est vérifié une infinité de fois dans le futur".

5. ***FG***p : "p est toujours vérifié à partir d'un certain état".

   Dans ce cas le rond renforcé bleu exprime que Ψ est vrai.

6. ***U*** (until) Φ ***U*** Ψ : Φ est vérifié jusu'à ce que Ψ le soit :



2.3 Exemple

Pour illustrer l'utilisation de la logique temporelle linéaire dans la formalisation des problèmes pratiques, nous donnons l'exemple du publiphone qui consiste à modéliser les actions qu'un utilisteur exécute pendant l'opération de tentative de téléphonner jusqu'à la fin de la communication.
Donnons, d'abord, la procédure téléphoner :

- L'utilisateur doit décrocher le téléphone.
- Le système affiche insérer une carte.
- L'utilisateur insère une carte.
- Le système affiche le nombre d'unités restantes.
- L'utilisateur compose son numéros de téléphone avec la possibilité de le corriger pour obtenir le bon numéros dans un délais de 2 secondes.
- L'utilisateur communique avec son correspondant, tant qu'il lui reste des unités.
- Si la carte est épuisée, l'utilisateur a 10 secondes pour la changer.
- Lorsque la communication est terminée, l'utilisateur raccroche le téléphone.
- Le système affiche retirer la carte.
- L'utilisateur retire sa carte.

Ensuite, formalisons les propositions atomiques :

- u_décrocher : l'utilisateur décroche le téléphone.
- sys_affiche(message) : le système affiche le message.

– u_insérer_carte : l'utilisateur insère une carte.
– u_compser_no : l'utilisateur compose le numéros.
– u_correction_no : l'utilisateur corrige le numéros.
– numéros_correct : le numéros est correct.
– u_communiquer : l'utilisateur communique avec son correspondant.
– u_changer_carte_vide : l'utilisateur change sa carte qui est épuisée.
– u_raccrocher : l'utilisateur raccroche le téléphone.
– u_retirer_carte : l'utilisateur retire sa carte.
– ltl_téléphoner : établit la clause de l'activité téléphoner.

Enfin, on obtient la formule [1] :

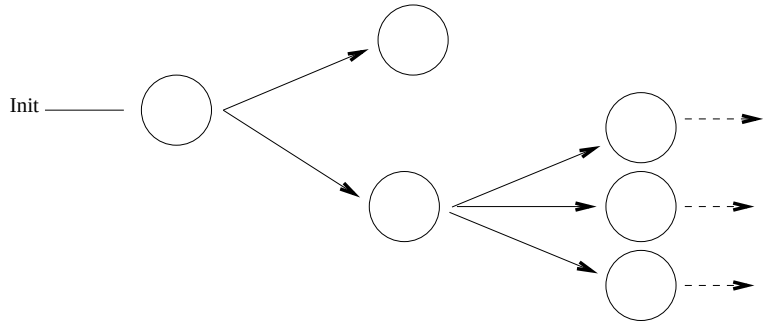
$$u\_decrocher \wedge \boldsymbol{X}\, sys\_affiche(\textit{Insérer carte}) \wedge$$
$$\boldsymbol{XX}\,[(u\_insrer\_carte \wedge \boldsymbol{X}\, sys\_affiche(nb\_units\_restantes)) \wedge$$
$$(\boldsymbol{X}\, u\_composer\_no \vee \boldsymbol{F}\,(u\_correction\_no \boldsymbol{U}\, numros\_correct)) \wedge$$
$$((\boldsymbol{X}\, u\_communiquer \vee \boldsymbol{F}\,(u\_changer\_carte\_vide\, \boldsymbol{U}\, unites(carte) > minimum))$$
$$\boldsymbol{U}\,(u\_raccrocher \wedge u\_retirer\_carte))] \quad \Rightarrow ltl\_telephoner$$

Pour téléphoner, l'utilisateur doit décrocher le téléphone, puis, dans l'état suivant ***(opérateur X)***, il doit insérer sa carte. Ensuite, il compose son numéros avec la possibilité ***(opérateur F)*** de le corriger. Si le numéros est correct, l'utilisateur pourra, alors communiquer avec la possibilité de changer sa carte. La communication dure jusqu'à ce qu'il raccroche le téléphone et retire sa carte.

On a pu exprimer la notion de boucle à l'aide de l'opérateur ***U***. Par exemple, on corrige le numéros de téléphone un nombre infini de fois jusqu'à obtenir le bon numéros. En plus, l'opérateur ***X*** précise le séquencement des évènements : l'utilisateur compose son numéros juste après que le système affiche le nombre d'unités restantes.

3 La logique du temps arborescent

La logique du temps arborescent ou *CTL* «Computation Tree Logic *(en aglais)*» permet de considérer plusieurs futurs possibles à partir d'un état du système plus-tôt que d'avoir une vue linéaire du système considéré [1, 3].



fi g 2 :Le déroulement temporel d'un système dans une logique arborescente[1].

3.1 Syntaxe

Les formules de la logique temporelle linéaire *CTL* sont définies par la grammaire suivante [1, 3] :

– Les propositions atomiques :

$$\Phi, \Psi := p \mid q \mid \ldots \mid \text{true} \mid \text{false} \qquad \Phi, \Psi \in CTL$$

– Les connecteurs booléens :
Soit $\Phi, \Psi \in LTL$, toutes formules de la forme

$$\neg\,\Phi \mid \Phi \wedge \Psi \mid \Phi \vee \Psi \mid \Phi \Rightarrow \Psi \mid \Phi \Leftrightarrow \Psi$$

appartiennent à *CTL*.

– Les connecteurs temporels :
Soit $\Phi, \Psi \in LTL$, toutes formules de la forme

$$\boldsymbol{EF}\,\Phi \mid \boldsymbol{EG}\,\Phi \mid \boldsymbol{E}\,\Phi\boldsymbol{U}\,\Psi \mid \boldsymbol{EX}\,\Phi \mid$$
$$\boldsymbol{AF}\,\Phi \mid \boldsymbol{AG}\,\Phi \mid \boldsymbol{A}\,\Phi\boldsymbol{U}\,\Psi \mid \boldsymbol{AX}\,\Phi$$

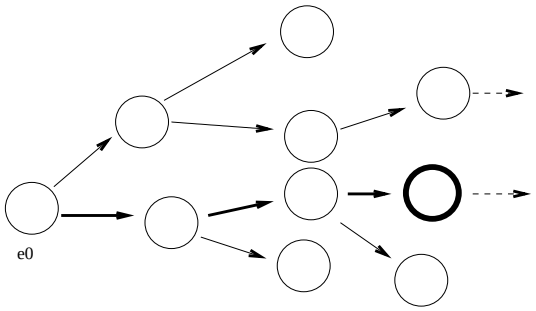
appartiennent à *CTL*.

E, A : quantifications sur toutes les exécutions possibles à partir de l'état courant.

3.2 Sémantique

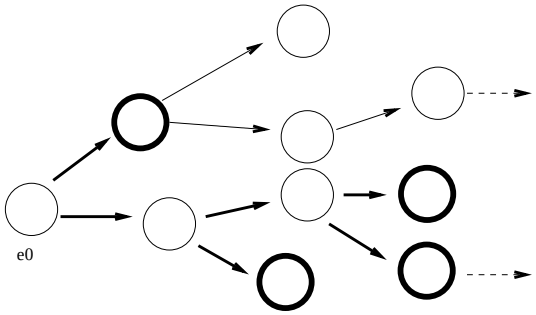
Notons que, dans les dessins[2, 3] qui suivent, les ronds renforcés noirs expriment que Φ est vrai et ceux en bleu exriment que Ψ est vrai. Ainsi, on définit les opérateurs temporels comme suit :

1. ***EF*** Φ : « il est possible d'atteindre un état où Φ est vérifié » ou « il existe une exécution ***(opérateur E)*** conduisant à un état où Φ est vérifié ***(opérateur F)***».



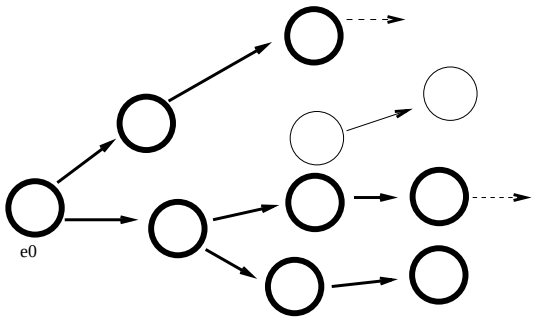
$e0 \models \boldsymbol{EF}\ \Phi$

2. ***AF*** Φ : «Φ est vérifié dans le futur» ou « pour toute exécution ***(opérateur A)***, il existe un état où Φ est vérifié ***(opérateur F)***».



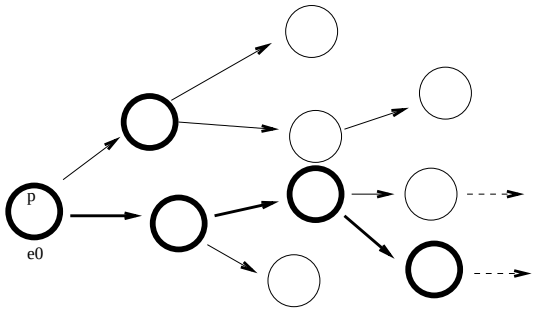
$e0 \models \boldsymbol{AF}\ \Phi$

3. ***AG*** Φ : «Φ est vérifié pour tout état atteignable» ou « pour toute exécution ***(opérateur A)***, Φ est toujours vérifié ***(opérateur G)***» :



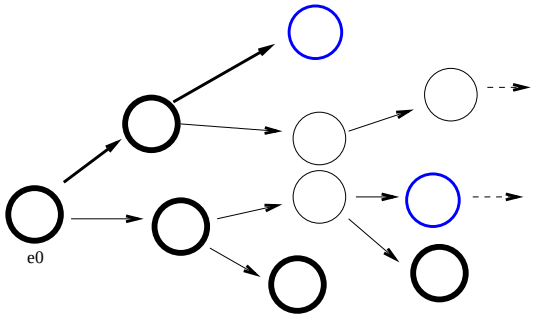
e0 $\models$ ***AG*** Φ

4. ***EG*** Φ : «il existe une exécution ***(opérateur E)*** où Φ est toujours vérifié ***(opérateur G)***» :



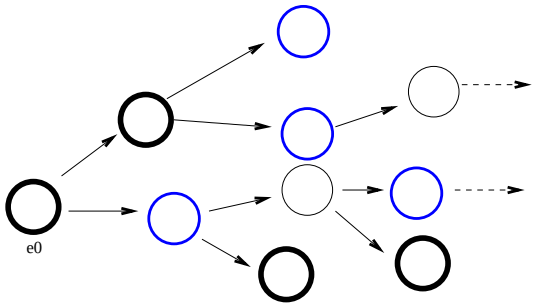
e0 $\models$ ***EG*** Φ

5. ***E*** Φ ***U*** Ψ : «il existe une exécution ***(opérateur E)*** durant laquelle Φ est vérifié jusqu'à ce que Ψ le soit *(*Φ ***U*** Ψ *)*» :



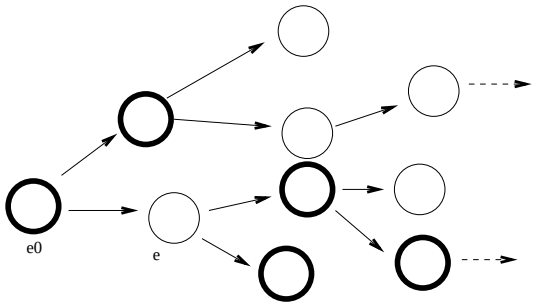
e0 $\models$ ***E*** Φ ***U*** Ψ

6. ***A*** Φ ***U*** Ψ : «pour toute exécution ***(opérateur A)*** Φ est vérifié jusqu'à ce que Ψ le soit *(*Φ ***U*** Ψ *)*» :



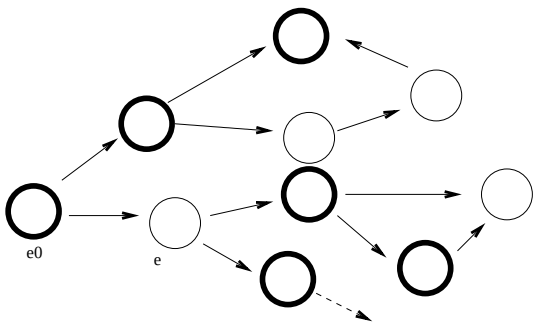
e0 $\models$ ***A*** Φ ***U*** Ψ

7. ***EX*** Φ : «il existe une exécution ***(opérateur E)*** dont le prochain état satisfait Φ».
8. ***AX*** Φ : «tous les états immédiatement successeur satisfont Φ» :



e0 $\models$ ***EX*** Φ
e $\models$ ***AX*** Φ

9. ***AG EF*** Φ : «A partir de n'importe quel état atteignable, il est possible d'atteindre un état satisfaisant Φ» :



e0 $\models$ ***AG EF*** Φ

3.3 Exemple :

On reprend l'exemple du publiphone pour illustrer comment on modélise le problème dans la logique *CTL* [2].

La clause ltl_téléponer sera ctl_téléphoner. La formalisation en *CTL* donne [1] :

$$u\_decrocher \wedge u\_inserer\_carte \wedge$$
$$\boldsymbol{AX}\,[(u\_composer\_no \wedge \boldsymbol{EF}\,u\_reprise\_sur\_erreur) \wedge$$
$$\boldsymbol{AX}\,((u\_communiquer \wedge \boldsymbol{EF}\,changer\_carte\_vide)\,\boldsymbol{U}$$
$$(u\_raccrocher \wedge u\_retirer\_carte))] \quad \Rightarrow ctl\_telephoner$$

Pour téléphoner, l'utilisateur doit décrocher le téléphone et insérer une carte (l'ordre de ces deux actions n'intervient pas). Puis, dans tous les états possibles ***(opérateur AX)***, il composera le numéros, et pour terminer il raccrochera le téléphone et retirera sa carte.

Pour composer le numéros, l'utilisateur pourra ***(opérateur E)*** se ramener à le corriger ***(opérateur F)***. Dans tous les états possibles, l'utilisateur communique au téléphonne jusqu'à ***(opérateur U)*** ce qu'il raccroche et retire sa carte avec une possibilité de l'échanger ***(opérateur EF)***. ***EF*** se traduit dans notre cas par eventuellement au lieu de finallement pour "eventually".

On remarque que cette version n'est pas aussi précise que la précedente. De plus, on donne des informations de plus en plus floues. Remarquons, par exemple, l'abscence de l'indicateur de temps qui précise la durée pour changer une carte téléphonique.

4 Exemple de propriétés

La logique temporelle est utilisée pour exprimer des propriétés à vérifier, notamment la spécifications comportementales, caractérisant le comportement d'un programme. De façon à les utiliser pour faire de la vérification ; par exemple, utilisation des outils comme SMV, PVS, SPIN et autres qui utilisent des expressions de la logique temporelle pour examiner si celle ci vérifie un programme ou non.

Donnons à présent des exemples très simples de propriétés exprimées en Logique Temporelle :

- evt1 et evt2 arrivent ou n'arrivent pas simultanément :
$$\boldsymbol{G}(\text{evt1} \Leftrightarrow \text{evt2})$$
- evt2 n'arrive pas avant evt1 :
$$\boldsymbol{G}(\neg\ (\text{evt2})\ \boldsymbol{U}\ \text{evt1})$$
- Si evt2 alors evt1 doit arriver 3 cycles après :
$$\boldsymbol{G}(\text{evt1} \Rightarrow \boldsymbol{XXX}\ \text{evt2})$$
- Une ressource est accordée à au plus un processus demandeur à la fois :
$$\boldsymbol{G}(\mathrm{a}_i \wedge a_j)_{i \neq j}$$
- Propriétés de sûreté :
  Soit P_i et P_j deux processus et CS_ila proposition atomique dénotant que le processus P_i est en section critique, la formule
$$\boldsymbol{G}(\neg(\ CS_i \wedge CS_j))$$
  caractérise l'exclusion mutuelle de P_i et P_j.

- Propriété de vivacité :
  Soit try_i la proposition exprimant la demande d'accès à la section critiuqe. La formule suivante,
$$try_i \Rightarrow \boldsymbol{F}\ CS_i$$
  exprime qu'un processus demandant l'accès à la section critique finira par l'obtenir.

5 Conclusion

On vient de voir deux types de logiques temporelles : *LTL* et *CTL*.Toute fois, il existe d'autres logiques temporelles comme *TLA*, *CTL**, *PLT*…. L'aspect important de la logique temporelle est son aspect expressif, c'est à dire sa capacité de décrire différentes classes de propriétés. Évidement la logique *CTL* est plus puissante que la *LTL* puisque cette dernière étend la *CTL* [1, 3].

Néanmoins, la logique linéaire est utilisable et bien simple pour exprimer des comportements (actions) qui ne nécessitent pas des branchements. Ainsi, nous remarquons dans l'exemple 2.3 et 3.3 que la formule de l'assertion est plus com-

pacte et des fois même plus lisible que celle exprimée en *LTL*. Cependant, ”le but de la logique temporelle est de proposer des formalismes de haut niveau permettant d’analyser de façon simple une classe déterminée de propriétés”[3].

Il serait intéssant, que je m’attaque à un problème pratique, par exemple spécification d’un problème inforamatique en modélisant un programme et/ou une propriété à vérifié pour les utiliser, toucher de plus prés leurs intérêts et voir leurs limites.

Références

[1] membres.lycos.fr/interaction/ Theses/Brun1998/Brun1998.pdf

[2] www.infres.enst.fr/ najm/ips/session-2004/Cours_Logique_temporelle_et_Model_checking.pdf

[3] Chapitre 8 : La logique du temps arborescent *(article)*

[4] http ://philologos.free.fr/logique.html

[5] http ://www.relst.uiuc.edu/durkheim/Texts/1884a/38.html

[6] Encyclopédie Hachette 2002

[7] http ://www.univ-nancy2.fr/ENSGT/PHILO/walter/lfah.html

[8] http ://wwwsi.supelec.fr/gvn/LogiqueTemporelle.pdf

[9] L'outil de vérifi cation SPIN Auteur :Eric Vachon

[10] http ://www.loria.fr/ bonfante/modelchecking/poly004.html