Méthodes Formelles pour la Conception de Logiciels, Cours M2 Mops : Partie II

S. Cerrito

2010-2011

Table des matières

Partie n^o 1
Vérification, Logique Temporelle et Déduction

1.1 Préliminaires

Cours de MOPS connexes :

- PRFM
- TETA
- COTE
- ANST

URL avec le support de ce cours :

`www.ibisc.univ-evry.fr/~serena}`

(suivre le lien Current Teaching).

1.2 Problèmatique

Vérification (formelle) de propriétés de systèmes informatiques

S : programme ou système dont l'exécution ne termine pas (par ex. système d'exploitation). Deux approches possibles à la vérification de S :

1. Approche fondée sur la déduction : S est décrit (spécifié) par une formule $\mathcal{S}$ d'une logique, les propriétés souhaitées pour S sont exprimées par une formule P de cette logique. On cherche une déduction de $\mathcal{S} \rightarrow P$.
2. Approche fondée sur les modèles (*model checking*) : S est vu comme une interprétation $\mathcal{I}_S$ d'une logique, les propriétés sont exprimées par une formule P de cette logique. On teste si P est vraie pour $\mathcal{I}_S$.

Logique utilisé dans ce cours : la **Logique Temporelle Linéaire** (**LTL**).
Le cours est centré surtout sur la première approche, mais donne aussi des bases pour comprendre les principes du logiciel SPIN qui fait du model checking avec LTL.

Un exemple de logiciel de vérification fondé sur LTL : SPIN

- Outil de vérification automatique de logiciels distribués.
- Développé à partir de 1980 aux Bell Labs (organisme de recherche actuellement de Alcatel-Lucent et, avant, de : American Telephone & Telegraph Company (AT&T)). Logiciel libre depuis 1991.
- URL : `http ://spinroot.com/spin/whatispin.html`
- System Software Award en 2001 (ACM).
- Utilisé pour détecter des erreurs de conception pour des systèmes d'exploitation, entre outre.
- Utilisé pour des descriptions de haut niveau et aussi pour du code détaillé (en téléphonie, par ex.).
- Il teste la cohérence logique d'une spécification, détecte les *deadlocks*, etc.
- Il peut travailler *on-the-fly*, sans besoin de la construction préliminaire du graphe d'états global.

1.3 La logique Temporelle LTL :

Linear Temporal Logic

Un élément de la famille des logiques dites modales.

Digression au tableau sur les logiques modales.

Deux autre versions de logique temporelles sont aussi souvent utilisées en vérification automatique : CTL et CTL*.

1.3.1 LTL : Syntaxe et Sémantique

Principes :

- Fondée sur une modélisation discrète et linéaire du temps.
- Conçue par Z. Manna et A. Pnueli (1983).
- Ajoute aux connecteurs booléens $\neg, \vee, \wedge$ les opérateurs $\Diamond$, $\Box$, $\mathcal{U}$, $\bigcirc$.
- Ajoute une composante "dynamique" (changement d'état) à la logique classique.
- Modélise des programmes/systèmes qui ne terminent pas (les systèmes d'exploitation, par ex.).

Syntaxe de LTL

– P : n'importe quelle variable propositionnelle (booléenne).
Déf. des formules de LTL (F) :

$$F = True \mid P \mid \neg F \mid F \wedge F \mid F \vee F \mid \Box F \mid \Diamond F \mid F \mathcal{U} F \mid \bigcirc F$$

– $F_1 \rightarrow F_2 =_{DEF} \neg F_1 \vee F_2$.
– Un atome est soit $True$ soit une variable prop. Un littéral est soit un atome soit une expression $\neg A$ où A est un atome.
– Une lecture intuitive :
$\Box F$: Maintenant et toujours dans le future F
$\Diamond F$: Soit maintenant soit à au moins un instant du future F
$F_1 \mathcal{U} F_2$: F_1 *until* F_2
$\bigcirc F$: F est vraie à l'état (ou "instant") suivant.

Sémantique Formelle de LTL

Besoin d'étendre les notions d'*interprétation*, *vérité* et *modèle* de la logique (booléenne) classique.

Notion de cadre (*frame*) $\Rightarrow$ modélisation de l'ensemble d'états d'un processus.

Dans la sémantique des logiques modales, un cadre est un ensemble muni d'une ou plusieurs relations.
Logiques modales différentes $\Rightarrow$ contraintes différentes sur les cadres.

Sémantique Formelle de LTL, suite.

Idée : une interprétation de LTL est une suite infinie à droite (mais "finie à gauche") d'interprétations de la logique booléenne classique.

Un cadre de LTL :

- C'est le couple $\langle \mathbb{N}, \leq \rangle$ (ou une structure isomorphe). Les éléments de $\mathbb{N}$ sont dits états et 0 est dit *état initial*.
- Façon différente mais équivalente de présenter la notion de cadre pour LTL : Un cadre est un triplet $\langle E, R, R_{ct} \rangle$, où $\langle E, R_{ct} \rangle$ est une structure isomorphe à $\langle \mathbb{N}, \leq \rangle$, qque soit $e \in E$ $\exists! e'$ tq eRe', et, enfin, R_{ct} est la fermeture réflexive et transitive de R.
  Intuition : R est la relation "successeur" sur E et R_{ct} sur E est analogue à $\leq$ sur $\mathbb{N}$. Les éléments de E sont dits états et l'état e_0 de E qui est le minimum par rapport à R_{ct} est dit *état initial*.

Dans la suite, on utilisera la première présentation de la notion de cadre.

Question : Pourquoi donner aussi la seconde description ?

Sémantique Formelle de LTL, suite

- Une Interprétation d'une formule de LTL dont l'ensemble des variables propositionnelles est $Prop$: une fonction $\mathcal{I} : \mathbb{N} \rightarrow 2^{Prop}$
  ($p \in \mathcal{I}(i)$: p est vraie à l'état i).
  Modélise **une** exécution d'un programme qui démarre à l'état 0.
  Exemple simple au tableau.
- $F=$ formule quelconque de LTL. Notation générale pour F est vraie à l'état i par rapport à l'interprétation $\mathcal{I}$:

$$\mathcal{I}, i \models F$$

Définition formelle de $\models$

$\mathcal{I}$ = interprétation, i = etat.

- Si At est un atome, $\mathcal{I}, i \models At$ ssi soit At est $True$ soit $At \in \mathcal{I}(i)$.
- $\mathcal{I}, i \models \neg F_1$ ssi $\mathcal{I}, i \not\models F_1$.
- $\mathcal{I}, i \models F_1 \wedge F_2$ ssi $\mathcal{I}, i \models F_1$ et $\mathcal{I}, i \models F_2$.
- $\mathcal{I}, i \models F_1 \vee F_2$ ssi $\mathcal{I}, i \models F_1$ ou $\mathcal{I}, i \models F_2$ (ou non-exclusif).
- $\mathcal{I}, i \models \Box F_1$ ssi quelque soit j tel que $i \leq j$ on a $\mathcal{I}, j \models F_1$.
- $\mathcal{I}, i \models \Diamond F_1$ ssi $\exists\, j$ tel que $i \leq j$ et $\mathcal{I}, j \models F_1$.
- $\mathcal{I}, i \models \bigcirc F_1$ ssi $\mathcal{I}, i+1 \models F_1$.
- $\mathcal{I}, i \models F_1 \mathcal{U} F_2$ ssi $\exists\, j$ tel que : $i \leq j$, et $\mathcal{I}, j \models F_2$ et, quelque soit n in $i, ..., j-1$, on a $\mathcal{I}, n \models F_1$.

Dessin au tableau pour $\mathcal{I}, i \models F_1 \mathcal{U} F_2$

F est vraie par rapport à une interprétation $\mathcal{I}$ ssi $\mathcal{I}, 0 \models F$. On dit aussi que $\mathcal{I}$ est un modèle de F.

F est valide si toute interprétation de F est un modèle de F ; elle est satisfiable s'il existe au moins une $\mathcal{I}$ qui est un modèle de F.

F_2 est une conséquence logique de F_1 ssi, qque soit l'interprétation $\mathcal{I}$ de F_1 et quelque soit i, si $\mathcal{I}, i \models F_1$ alors $\mathcal{I}, i \models F_2$. Notation : $F_1 \models F_2$.

F_1 est logiquement équivalente à F_2, ce que l'on note $F_1 \equiv F_2$, ssi $F_1 \models F_2$ et $F_2 \models F1$.

Exemples d'évaluation de formules de LTL sur des interprétations : au tableau.

Un exemples de propriétés de systèmes exprimées en LTL

Exemple d'un feu qui règle la circulation.

Variables booléennes : vert, rouge, jaune (abrégées en : v, r et j)
Ordre du changement de couleur : $v \rightsquigarrow j \rightsquigarrow r \rightsquigarrow v$
Hyp. Le feu continue toujours à travailler.

1. A tout moment, le feu a exactement une des 3 couleurs :

$$\Box(\neg(v \wedge j) \wedge \neg(r \wedge j) \wedge \neg(r \wedge v) \wedge (v \vee j \vee r)))$$

2. Si le feu est dans un état où la couleur est verte, cette couleur persiste jusqu'à quand on passe à jaune :

$$\Box(v \rightarrow (v \mathcal{U} j))$$

3. Description générale de l'ordre du changement de couleurs du feu :

$$\Box((v \mathcal{U} j) \vee (j \mathcal{U} r) \vee (r \mathcal{U} v))$$

1.3.2 Déduction automatique pour LTL : méthode des "Tableaux"

Utilité ?

- $P1, P2$: formules de LTL qui expriment des propriétés d'un système S.
  Question Q : Si $P1$ est vraie pour S, est-il vrai que $P2$ est forcement vraie ?
- Une reformulation Q' de Q : $P1 \rightarrow P2$ est valide ?
- Déduction automatique : possibilité d'utiliser un *programme* qui donne la réponse à Q'.

Préliminaires sur les *tableaux* (peu importe la logique)

- Des systèmes de preuve bien adaptés à la déduction automatique.
- Systèmes de réfutation.
- **Intuition** : Pour prouver F, on essaie systématiquement de construire un modèle de $\neg F$. Si on échoue alors F est valide, sinon F n'est pas valide, car on a trouvé une interprétation (situation) contre-exemple.

Cas simple de la logique booléenne

- Tableau de racine E = arbre binaire tq :
  - Noeuds = ensembles de formules
  - Racine = ensemble E
  - Fils d'un noeud engendrés en lui appliquant une règle d'expansion.
  - Si formules en forme normales de négation et si $\neg, \wedge, \vee$ comme seuls connecteurs, alors les *règles d'expansion sont* :

$\frac{E, A \wedge B}{E, A, B}\,(\alpha)$	$\frac{E, A \vee B}{E, A \quad \mid \quad E, B}\,(\beta)$

  E=ensemble de formules. A et B = formules. Virgule = union ensembliste. La règle β provoque un branchement.

- N.B. : f.n.n. : $\neg$ s'applique seulement à des formules atomiques ; toujours possible de réécrire en f.n.n.

- Un noeud est clos (ou contradictoire) s'il contient autant p que $\neg p$ pour quelque variable propositionnelle p, ou bien il contient $\neg True$.
- On applique une règle d'expansion à un noeud seulement si il n'est pas contradictoire.
- Tout tableau est forcement fini (pourquoi ?).

Cas simple de la logique booléenne, suite.

– $\mathcal{T}$ = un tableau de racine E. Algorithme de "élagage" de $\mathcal{T}$:
Appliquer itérativement les règles suivantes, dans l'ordre, jusqu'a quand le tableaux reste stable :

1. Effacer tout sommet clos (c'est forcement une feuille...)
2. Effacer tout sommet dont on a effacé tous les fils.

– Résultat de l'algorithme : un arbre $\mathcal{T}'$.
Le branches de $\mathcal{T}$ effacées : branches dites *closes*= "modèles impossibles".
Celles qui restent : branches dites *ouvertes*.
Si $\mathcal{T}'= \emptyset$, alors $\mathcal{T}$ est une réfutation. de E.
Sinon, branche ouverte = représentation d'une classe de modèles de E.

Exemple : réfutation de $F = ((\neg p \vee q) \wedge p) \wedge \neg q$.

On écrit F à la place de $\{F\}$

$$\dfrac{\dfrac{\dfrac{((\neg p \vee q) \wedge p) \wedge \neg q}{(\neg p \vee q) \wedge p, \neg q}}{\neg p \vee q, p, \neg q}}{\neg p, p, \neg q \qquad\qquad q, p, \neg q}$$

Les deux branches sont closes !

Exemple : tableau pour

$$F = ((p \vee (q \vee r))) \wedge \neg p$$

$$\dfrac{\dfrac{\dfrac{((p \vee (q \vee r))) \wedge \neg p}{p \vee (q \vee r), \neg p}}{\begin{array}{ll} p, \neg p & \dfrac{q \vee r, \neg p}{\begin{array}{ll} q, \neg p & r, \neg p \end{array}} \\ close! & \end{array}}}$$

Deux branches ouvertes, décrivant 2 classes de modèles :

1. Ceux où q est vrai, p est faux, r peu importe
2. Ceux où r est vrai, p est faux, q peu importe

Tableaux de P. Wolper (1985) pour LTL

Ici : formules en f.n.n. Pour récrire toute formule de LTL en f.n.n. on exploite :

$\neg\Diamond A \equiv \Box\neg A$	$\neg\Box A \equiv \Diamond\neg A$
$\neg\bigcirc A \equiv \bigcirc\neg A$	$\neg(A\mathcal{U}B) \equiv \Box\neg B \vee (\neg B\mathcal{U}(\neg A \wedge \neg B))$

Un tableau pour LTL = un graphe orienté avec une racine, dont les noeuds sont des ensembles de formules de LTL en f.n.n. et les fils d'un noeud sont engendrés en appliquant une règle d'expansion.

– **Règles d'expansion** :

$\frac{E,A\wedge B}{E,(A\wedge B)^*,A,B}(\alpha)$	$\frac{E,A\vee B}{E,(A\vee B)^*,A \mid E,(A\vee B)^*,B}(\beta)$
$\frac{E,\Diamond A}{E,(\Diamond A)^*,A \mid E,(\Diamond A)^*,\bigcirc\Diamond A}(\Diamond)$	$\frac{E,\Box A}{E,(\Box A)^*,A,\bigcirc\Box A}(\Box)$
$\frac{E,A\mathcal{U}B}{E,(A\mathcal{U}B)^*,B \mid E,(A\mathcal{U}B)^*,A,\bigcirc A\mathcal{U}B}(\mathcal{U})$	$\frac{L,\bigcirc A_1,\cdots\bigcirc A_n,B_1*,\cdots,B_k^*}{A_1,\cdots,A_n}(\bigcirc)$

Dans la règle pour $\bigcirc$, L = ensemble de littéraux. Si n=0, écrire juste $True$ dans l'expansion. * est un **marquage** d'une formule. Un noeud auquel on applique la règle $\bigcirc$ est dit état.

– Les règles d'expansion sont fondées sur les équivalences :

$$\Diamond A \equiv A \vee \bigcirc\Diamond A \qquad \Box A \equiv A \wedge \bigcirc\Box A \qquad A\mathcal{U}B \equiv B \vee (A \wedge \bigcirc(A\mathcal{U}B))$$

– Comment on applique les règles ?
  – On applique une règle à un noeud seulement si ce noeud n'est pas clos (même définition qu'avant).
  – Une formule F est traitée seulement si elle n'est pas marquée avec $*$.
  – En développant F, on crée un nouveau noeud avec une expansion Ex de F seulement si un noeud avec même étiquette Ex n'existe pas déjà. (Deux étiquettes sont identiques si elles contiennent exactement les mêmes formules, avec les mêmes marquages.)
    Sinon, on crée un arc vers ce vieux noeud.
  – Appliquer au noeud $True$ la règle $\bigcirc$ une seule fois !

Un tableau de racine $\{A\}$ est fini car le nombre d'ensembles de sous-formules distincts pouvant etiquetter un noeud est fini (pourquoi ?).

Exemple 1

Construction d'un tableau de racine $\Box p \wedge \Diamond \neg p$

Comment tester la satisfiabilité de l'ensemble de formules à la racine ?

Procédure d'élagage pour les tableaux pour LTL

Répeter les instructions suivantes (dans l'ordre) jusqu'a quand le tableau reste stable :

1. Effacer tout sommet clos (c'est forcement une feuille).
2. Effacer tout sommet dont on a effacé tous les fils.
3. Effacer tout sommet s ayant comme élément une formule existentielle (*eventuality*) - $\Diamond F_2$ ou $F_1 \mathcal{U} F_2$ - telle qu'il n'existe pas de chemin, dans le graphe, conduisant de s à un sommet s' qui contient F_2.
   N.B. Une *eventuality* peut être marquée par *.

Si le tableau résultat est $\emptyset$, alors la racine est insatisfiable. Sinon, les chemins qui restent peuvent donner des modèles de la racine

Exemple 1, Suite

Élagage du tableau de racine

$$\Box p \wedge \Diamond \neg p$$

Que peut-on conclure ?

Exemple 2

Construction et élagage d'un tableau de racine

$$\Box\Diamond p$$

Que peut-on conclure ?

Toute règle du calcul sauf ($\bigcirc$) est dite *statique*. La règle ($\bigcirc$) est dite *dynamique*.

Quel est le sens intuitif de cette terminologie ?

Si on déplie le graphe (avant l'élagage) on obtient un arbre pouvant être infini, et t.q. :

- une branche qui sera éliminée : une branche qui ne décrit pas un modèle de la racine.

- une branche qui ne sera pas éliminée : une branche ouverte, qui peut décrire un modèle (ou plus) de la racine. Comment construire un tel modèle ?

1. Ne garder que les "états" (comme définis dans la diapositive avec les règles d'expansion). Ce seront les états de l'interprétation.
2. Pour chaque état m, déclarer vrais seulement les littéraux l tels que $l \in m$. N.B : description partielle d'une inteprétation, car, pour une p donnée, il se peut que ni $p \in m$ ni $\neg p \in m$.

Suite des exemples

Dépliage et étude des modèles possibles pour les deux exemples de tableaux de racine, resp. $\Box p \wedge \Diamond \neg p$ et $\Box \Diamond p$.

Est-il vrai que l'on obtient toujours un modèle à partir de toute branche qui ne sera pas éliminée par la procédure d'élagage du tableau ? Si oui, pourquoi ? Sinon, pourquoi ?

En fait :

Soit E une *eventuality* de la forme $\Diamond F_2$ ou $F_1 \mathcal{U} F_2$. Les formules E, F_1 et F_2 peuvent être marquées.

$$f_E = \{noeud\ s\ du\ tableau \mid E \notin s\ ou\ F_2 \in s\}$$

Déf. Une branche satisfait E ssi elle contient des noeuds de f_E infiniment souvent.

Soit $\mathcal{B}$ une branche infinie qui ne sera pas éliminée. Si $\mathcal{B}$ satisfait toute *eventuality* qui est une sous-formule de la racine alors toute interprétation décrite (partiellement) par $\mathcal{B}$ est un modèle (de la racine).

Comment tester si une branche a cette propriété, et décrit donc un modèle ? Voir après : “Automates de Büchi et Tableaux’

La construction et l'élagage des tableaux donnent un algorithme de décision pour le problème de la satisfiabilité (resp. validité) d'une formule F de LTL :

Thm (Correction du système de Wolper par rapport à l'insatisfiabilité) :
Si un tableau de racine F est "clos", au sens que, après élagage, on reste avec le tableau vide, alors F est insatisfiable.

Thm (Complétude du système de Wolper par rapport à l'insatisfiabilité) :
Si F est insatisfiable alors tout tableau de racine A est "clos" (c.à.d. : après élagage, on reste avec le tableau vide).

Complexité en temps et en espace de cet algorithme qui teste la satisfiabilité de F (donc aussi la validité de $\neg F$) : exponentielle par rapport à $\mid F \mid$, où $\mid F \mid$ est la taille de F.

NB : La satisfiabilité de PTL est un problème $PSPACE$-complet.