Cinquième partie

TLA$^+$– la logique

Plan

1. LTL et TLA+
   - Logique TLA+
   - Raffinage
2. Preuve axiomatique
3. Vérification de modèles

La logique TLA$^+$

Expressions logiques

Expressions de LTL avec $\Box$, $\Diamond$, $\leadsto$ (leads-to) et variables primées + quantificateurs $\forall, \exists$.

Pas de $\mathcal{U}$ mais :

$$\begin{array}{lcl}
\Box(p \Rightarrow (p \mathcal{W} q)) & = & \Box(p \Rightarrow (p' \vee q)) \\
\Box(p \Rightarrow (p \mathcal{U} q)) & = & \Box(p \Rightarrow (p' \vee q)) \wedge \Box(p \Rightarrow \Diamond q)
\end{array}$$

Équité / *Fairness*

ENABLED

ENABLED $\mathcal{A}$ est la fonction d'état qui est vraie dans l'état s ssi il existe un état t accessible depuis s par l'action $\mathcal{A}$.

Weak/Strong Fairness

- $\mathrm{WF}_e(\mathcal{A}) \triangleq \Box\Diamond\neg(\textsc{enabled}\ \langle\mathcal{A}\rangle_e) \vee \Box\Diamond\langle\mathcal{A}\rangle_e$
  si $\mathcal{A}$ est constamment déclenchable, elle sera déclenchée.
- $\mathrm{SF}_e(\mathcal{A}) \triangleq \Diamond\Box\neg(\textsc{enabled}\ \langle\mathcal{A}\rangle_e) \vee \Box\Diamond\langle\mathcal{A}\rangle_e$
  si $\mathcal{A}$ est infiniment souvent déclenchable, elle sera déclenchée.

Forme d'un programme TLA+

En général, une spécification TLA+ est une conjonction

$$\mathcal{I} \land \Box[\mathcal{N}]_v \land \mathcal{E}$$

- $\mathcal{I}$ = prédicat d'état décrivant les états initiaux
- $\mathcal{N}$ = disjonction d'actions $\mathcal{A}_1 \lor \mathcal{A}_2 \lor \mathcal{A}_3 \lor \ldots$
- $\mathcal{E}$ = conjonction de contraintes d'équité portant sur les actions : $\mathrm{WF}_v(A_1) \land \mathrm{SF}_v(A_3) \land \ldots$

Raffinage de programme

Raffinage simple

Un programme (concret) Pc raffine un programme (abstrait) Pa si $Pc \Rightarrow Pa$: tout ce que fait Pc est possible dans Pa.

Cela signifie que si $Pa \models P$ pour une propriété LTL quelconque, alors $Pc \models P$.

Raffinage – exemple

Somme abstraite

MODULE *somme1*

EXTENDS *Naturals*
CONSTANT *N*
VARIABLE *res*

$$TypeInvariant \triangleq res \in Nat$$

$$\begin{aligned} Init &\triangleq res = 0 \\ Next &\triangleq res' = ((N+1) * N) \div 2 \\ Spec &\triangleq Init \land \Box[Next]_{res} \land \mathrm{WF}_{res}(Next) \end{aligned}$$

Raffinage – exemple

Graphe des exécutions pour $N = 3$

$$\boxed{res = 0} \longrightarrow \boxed{res = 6}$$

Raffinage – exemple

Somme plus concrète

MODULE *somme2*

EXTENDS *Naturals*

CONSTANT *N*

VARIABLE *res*, *acc*, *disp*

$$TypeInvariant \triangleq res \in Nat \land acc \in Nat \land disp \in \text{SUBSET } 1 \mathrel{..} N$$

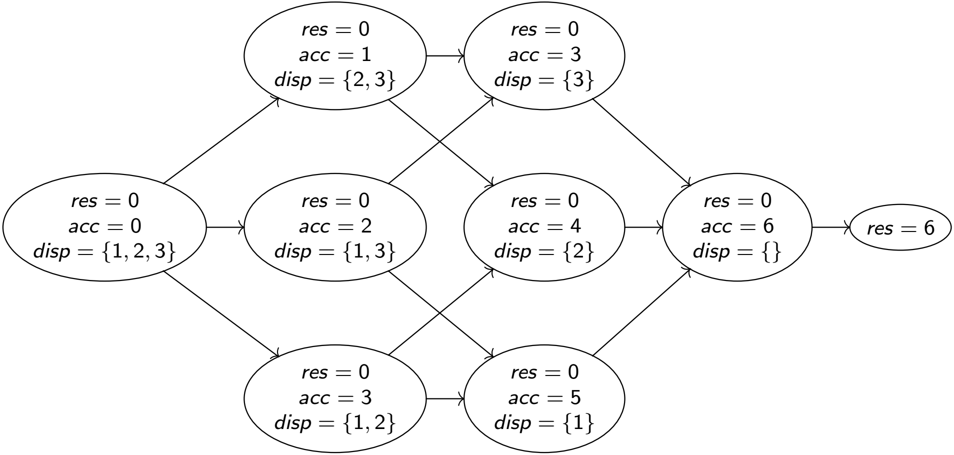
$$Init \triangleq res = 0 \land acc = 0 \land disp = 1 \mathrel{..} N$$

$$\begin{aligned} Next \triangleq\ & \lor \exists i \in disp : acc' = acc + i \land disp' = disp \setminus \{i\} \\ & \qquad\qquad \land \text{UNCHANGED } res \\ & \lor disp = \{\} \land res' = acc \land \text{UNCHANGED } \langle disp, acc \rangle \end{aligned}$$

$$Spec \triangleq Init \land \Box[Next]_{\langle res, disp, acc \rangle} \land \mathrm{WF}_{\langle res, disp, acc \rangle}(Next)$$

Raffinage – exemple

Graphe des exécutions pour N = 3



Décomposition : introduction de transitions intermédiaires.

Raffinage – exemple

Somme2 raffine somme1

$$\text{MODULE } \mathit{somme2\_raffine\_somme1}$$

$\text{EXTENDS } \mathit{somme2}$

$\mathit{Orig} \triangleq \text{INSTANCE } \mathit{somme1}$

$\mathit{Raffinement} \triangleq \mathit{Orig}!\mathit{Spec}$

$\text{THEOREM } \mathit{Spec} \Rightarrow \mathit{Orig}!\mathit{Spec}$

Raffinage – exemple

Somme concrète

MODULE *somme3*

$\textsc{extends}\ Naturals$

$\textsc{constant}\ N$

$\textsc{variable}\ res, acc, i$

$$TypeInvariant \triangleq res \in Nat \land acc \in Nat \land i \in 1 \mathrel{..} N$$

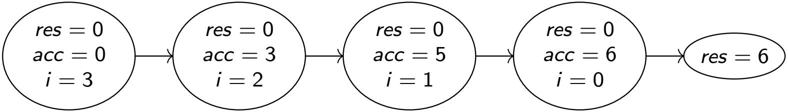
$$Init \triangleq res = 0 \land acc = 0 \land i = N$$

$$\begin{aligned} Next \triangleq\ & \lor\ i > 0 \land acc' = acc + i \land i' = i - 1 \land \textsc{unchanged}\ res \\ & \lor\ i = 0 \land res' = acc \land \textsc{unchanged}\ \langle i, acc \rangle \end{aligned}$$

$$Spec \triangleq Init \land \Box[Next]_{\langle res, i, acc \rangle} \land \mathrm{WF}_{\langle res, i, acc \rangle}(Next)$$

Raffinage – exemple

Graphe des exécutions pour N = 3



Réduction du non déterminisme + changement de représentation (raffinement de données) $disp = 1..i$

Raffinage – exemple

Somme3 raffine somme2

MODULE *somme3_raffine_somme2*

EXTENDS *somme3*

$dispMapping \triangleq 1 \,..\, i$

$Orig \triangleq$ INSTANCE *somme2* WITH $disp \leftarrow dispMapping$

$Raffinement \triangleq Orig!Spec$

THEOREM $Spec \Rightarrow Orig!Spec$

Plan

1. LTL et TLA$^+$
   - Logique TLA$^+$
   - Raffinage
2. Preuve axiomatique
3. Vérification de modèles

Règles de preuve – simple temporal logic

$$\frac{F \text{ prouvable en logique propositionnelle}}{\Box F}\,\mathrm{STL1} \qquad \frac{F \Rightarrow G}{\Box F \Rightarrow \Box G}\,\mathrm{STL4}$$

$$\frac{}{\Box F \Rightarrow F}\,\mathrm{STL2} \qquad \frac{}{\Box\Box F = \Box F}\,\mathrm{STL3}$$

$$\frac{}{\Box(F \wedge G) = (\Box F) \wedge (\Box G)}\,\mathrm{STL5} \qquad \frac{}{\Diamond\Box F \wedge \Diamond\Box G = \Diamond\Box(F \wedge G)}\,\mathrm{STL6}$$

Règles de preuve – TLA$^+$ invariant

$$\frac{P \wedge (v' = v) \Rightarrow P'}{\Box P = P \wedge \Box[P \Rightarrow P']_v}\,\mathrm{TLA1} \qquad \frac{P \wedge [\mathcal{A}]_{v_1} \Rightarrow Q \wedge [\mathcal{B}]_{v_2}}{\Box P \wedge \Box[\mathcal{A}]_{v_1} \Rightarrow \Box Q \wedge \Box[\mathcal{B}]_{v_2}}\,\mathrm{TLA2}$$

$$\frac{I \wedge [\mathcal{N}]_v \Rightarrow I'}{I \wedge \Box[\mathcal{N}]_v \Rightarrow \Box I}\,\mathrm{INV1} \qquad \frac{}{\Box I \Rightarrow (\Box[\mathcal{N}]_v = \Box[\mathcal{N} \wedge I \wedge I']_v)}\,\mathrm{INV2}$$

Règles de preuve – TLA$^+$ vivacité

$$\frac{\begin{array}{c} P \wedge [\mathcal{N}]_v \Rightarrow (P' \vee Q') \\ P \wedge \langle \mathcal{N} \wedge \mathcal{A} \rangle_v \Rightarrow Q' \\ P \Rightarrow \text{ENABLED } \langle \mathcal{A} \rangle_v \end{array}}{\Box[\mathcal{N}]_v \wedge WF_v(\mathcal{A}) \Rightarrow (P \leadsto Q)} \text{WF1}$$

$$\frac{\begin{array}{c} P \wedge [\mathcal{N}]_v \Rightarrow (P' \vee Q') \\ P \wedge \langle \mathcal{N} \wedge \mathcal{A} \rangle_v \Rightarrow Q' \\ \Box P \wedge \Box[\mathcal{N}]_v \wedge \Box F \Rightarrow \Diamond \text{ENABLED } \langle \mathcal{A} \rangle_v \end{array}}{\Box[\mathcal{N}]_v \wedge SF_v(\mathcal{A}) \wedge \Box F \Rightarrow (P \leadsto Q)} \text{SF1}$$

Règles de preuve dérivées

$$\frac{\Box(P \Rightarrow \Box P) \land \Diamond P}{\Diamond\Box P}\text{LDSTBL} \qquad \frac{P \leadsto Q \land Q \leadsto R}{P \leadsto R}\text{TRANS}$$

$$\frac{\forall m \in W : (P(m) \leadsto Q)}{(\exists m \in W : P(m)) \leadsto Q}\text{INFDIJ}$$

Plan

1. LTL et TLA$^+$
   - Logique TLA$^+$
   - Raffinage
2. Preuve axiomatique
3. Vérification de modèles

Vérification de modèles

Principe

Construire le graphe des exécutions et étudier la propriété.

- $\Box P$, où P est un prédicat d'état (sans variable primée) : au fur et à mesure de la construction des états.
- $\Box P(v, v')$, où $P(v, v')$ est un prédicat de transition (prédicat non temporel avec variables primées et non primées) : au fur et à mesure du calcul des transitions.
- Vivacité $\Diamond P$, $P \leadsto Q\ldots$: une fois le graphe construit, chercher un cycle qui respecte les contraintes d'équité et qui invalide la propriété.

Uniquement sur des modèles finis, et, pratiquement, de petites tailles.

Complexité

Soit $|S|$ le nombre d'états d'un système $\mathcal{S} = \langle S, I, R \rangle$ et $|F|$ la taille (le nombre d'opérateurs temporels) d'une formule LTL F.
La complexité en temps (et espace) pour vérifier $\mathcal{S} \models F$ est $O(|S| \times 2^{|F|})$.

Le principe est de parcourir l'espace d'états en marche avant, i.e. à partir des états initiaux du système, on visite les états successeurs par R. On arrête le parcours lorsque la formule LTL courante est atomique (i.e. de la forme s_i ou $\neg s_i$) en comparant simplement avec l'état courant, ou bien lorsqu'on retombe sur un état déjà visité.

Vérificateur TLC

Le vérificateur de modèles TLC sait vérifier :

- les programmes avec des actions gardées ;
- (efficacement) les invariants sans variables primées : $\Box P$ où P est prédicat d'état ;
- les formules de sûreté pure avec variables primées : $\Box P$ où P est prédicat de transition ;
- $P \leadsto Q$ où P et Q sont des prédicats d'état (*sans* variables primées) ;
- les formules combinant $\Box, \Diamond$ *sans* variables primées.

Note : l'espace d'états du système et des formules doit être fini : toute quantification bornée par ex.