

Cinquième partie

TLA⁺ – la logique



Objectifs

- Exprimer des propriétés vérifiées par une spécification TLA⁺
- Exprimer l'équité garantissant la progression
- Démontrer des liens entre spécifications (équivalence, raffinage)
- Pouvoir vérifier ces propriétés de manière mécanisée (automatiquement – vérification de modèles –, ou manuellement – preuve axiomatique –)



Plan

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



La logique TLA⁺

Expressions logiques

Expressions de LTL avec \square , \diamond , \rightsquigarrow (leads-to) et variables primées
+ quantificateurs \forall, \exists .

Pas de \mathcal{U} , ni de \mathcal{W} , mais :

$$\square(p \Rightarrow (p\mathcal{W}q)) = \square(p \Rightarrow (p' \vee q))$$

$$\square(p \Rightarrow (p\mathcal{U}q)) = \square(p \Rightarrow (p' \vee q)) \wedge \square(p \Rightarrow \diamond q)$$

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

- $WF_e(\mathcal{A}) \triangleq \Box\Diamond\neg(\text{ENABLED } \langle\mathcal{A}\rangle_e) \vee \Box\Diamond\langle\mathcal{A}\rangle_e$
si \mathcal{A} est constamment déclenchable, elle sera déclenchée.
- $SF_e(\mathcal{A}) \triangleq \Diamond\Box\neg(\text{ENABLED } \langle\mathcal{A}\rangle_e) \vee \Box\Diamond\langle\mathcal{A}\rangle_e$
si \mathcal{A} est infiniment souvent déclenchable, elle sera déclenchée.

Forme d'une spécification TLA⁺

En général, une spécification TLA⁺ est une conjonction

$$\mathcal{I} \wedge \square[\mathcal{N}]_v \wedge \mathcal{E}$$

- \mathcal{I} = prédicat d'état décrivant les états initiaux
- \mathcal{N} = disjonction d'actions $\mathcal{A}_1 \vee \mathcal{A}_2 \vee \mathcal{A}_3 \vee \dots$
- \mathcal{E} = conjonction de contraintes d'équité portant sur les actions : $WF_v(\mathcal{A}_1) \wedge SF_v(\mathcal{A}_3) \wedge \dots$

Raffinage de spécification

Raffinage simple

Une spécification (concrète) P_c raffine une spécification (abstraite) P_a si $P_c \Rightarrow P_a$: tout ce que fait P_c est possible dans P_a .

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



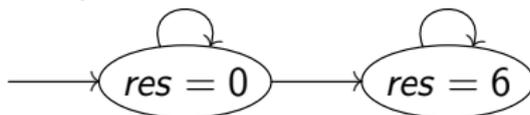
Raffinage – exemple

Somme abstraite

MODULE *somme1*EXTENDS *Naturals*CONSTANT *N*VARIABLE *res* $TypeOK \triangleq res \in Nat$ $Init \triangleq res = 0$ $Next \triangleq res' = ((N + 1) * N) \div 2$ $Spec \triangleq Init \wedge \square [Next]_{res} \wedge WF_{res}(Next)$

Raffinage – exemple

Grphe des exécutions pour $N = 3$



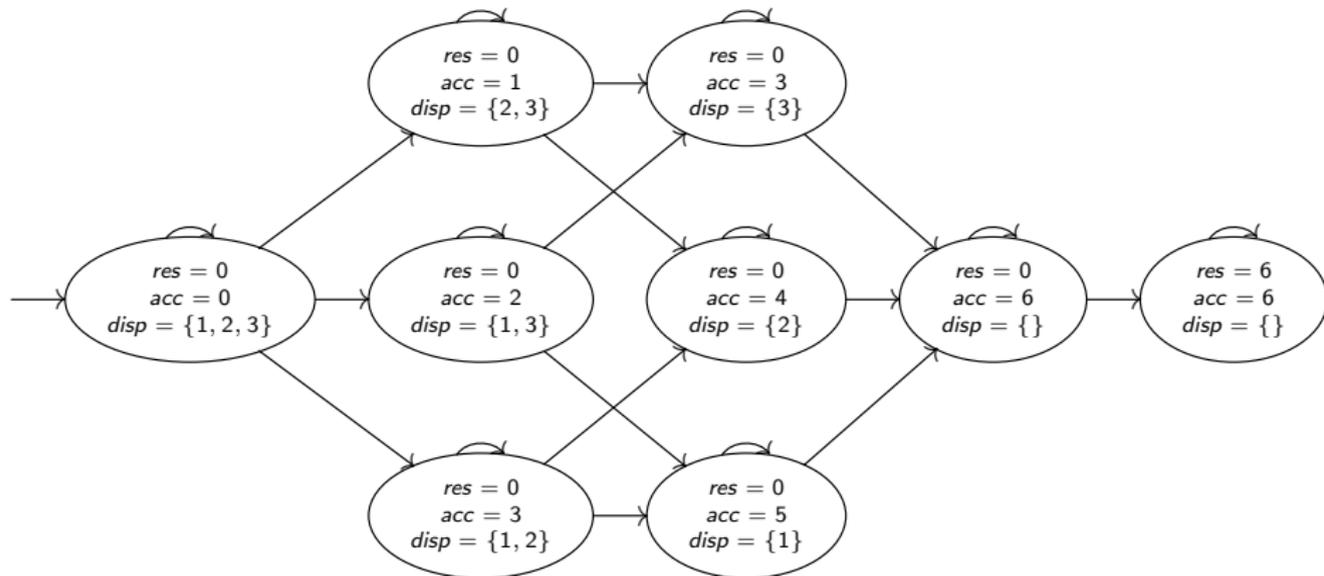
Raffinage – exemple

Somme plus concrète

MODULE *somme2*EXTENDS *Naturals*CONSTANT *N*VARIABLE *res, acc, disp* $TypeOK \triangleq res \in Nat \wedge acc \in Nat \wedge disp \in SUBSET\ 1..N$ $Init \triangleq res = 0 \wedge acc = 0 \wedge disp = 1..N$ $Next \triangleq \forall \exists i \in disp : acc' = acc + i \wedge disp' = disp \setminus \{i\}$
 $\wedge UNCHANGED\ res$ $Spec \triangleq Init \wedge \square[Next]_{\langle res, disp, acc \rangle} \wedge WF_{\langle res, disp, acc \rangle}(Next)$

Raffinage – exemple

Graphe des exécutions pour $N = 3$



Décomposition : introduction de transitions intermédiaires.

Handwritten signature

Raffinage – exemple

Somme2 raffine somme1

```
MODULE somme2_raffine_somme1  
EXTENDS somme2  
Orig  $\triangleq$  INSTANCE somme1  
Raffinement  $\triangleq$  Orig!Spec  
THEOREM Spec  $\Rightarrow$  Orig!Spec
```

Équivalent à $\textit{somme2!Spec} \Rightarrow \textit{somme1!Spec}$, où les variables homonymes (*res*) sont fusionnées.

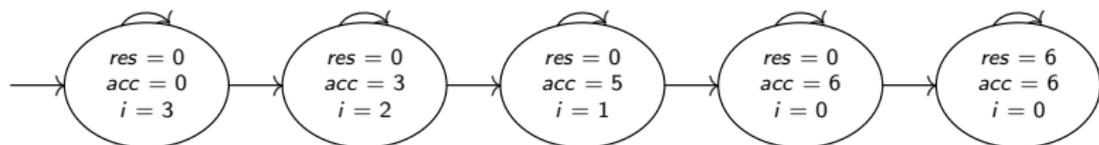
Raffinage – exemple

Somme concrète

MODULE *somme3*EXTENDS *Naturals*CONSTANT *N*VARIABLE *res, acc, i* $TypeOK \triangleq res \in Nat \wedge acc \in Nat \wedge i \in 1..N$ $Init \triangleq res = 0 \wedge acc = 0 \wedge i = N$ $Next \triangleq \vee i > 0 \wedge acc' = acc + i \wedge i' = i - 1 \wedge UNCHANGED\ res$
 $\vee i = 0 \wedge res' = acc \wedge UNCHANGED\ \langle i, acc \rangle$ $Spec \triangleq Init \wedge \square[Next]_{\langle res, i, acc \rangle} \wedge WF_{\langle res, i, acc \rangle}(Next)$

Raffinage – exemple

Grphe 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
    
```

Équivalent à $\textit{somme3!Spec} \Rightarrow \textit{somme2!Spec}$, où les variables homonymes (*res* et *acc*) sont fusionnées, et la variable $\textit{somme2!disp}$ évolue comme $1..\textit{somme3!i}$.

Plan

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



Règles de preuve – simple temporal logic

$$\frac{F \text{ prouvable en } \text{logique propositionnelle}}{\Box F} \text{STL1}$$

$$\frac{F \Rightarrow G}{\Box F \Rightarrow \Box G} \text{STL4}$$

$$\overline{\Box F \Rightarrow F} \text{STL2}$$

$$\overline{\Box \Box F = \Box F} \text{STL3}$$

$$\overline{\Box(F \wedge G) = (\Box F) \wedge (\Box G)} \text{STL5}$$

$$\overline{\Diamond \Box F \wedge \Diamond \Box G = \Diamond \Box(F \wedge G)} \text{STL6}$$

Règles de preuve – TLA⁺ invariant

$$\frac{P \wedge (v' = v) \Rightarrow P'}{\Box P = (P \wedge \Box [P \Rightarrow P']_v)}_{\text{TLA1}} \quad \left| \quad \frac{P \wedge [A]_{v_1} \Rightarrow Q \wedge [B]_{v_2}}{\Box P \wedge \Box [A]_{v_1} \Rightarrow \Box Q \wedge \Box [B]_{v_2}}_{\text{TLA2}}$$

TLA1 : principe d'induction pour prouver $\Box P$ à partir de l'état initial et de la conservation de P par bégaiement. TLA2 : le raffinement de spécifications se ramène au raffinement des actions.

$$\frac{I \wedge [N]_v \Rightarrow I'}{I \wedge \Box [N]_v \Rightarrow \Box I}_{\text{INV1}} \quad \left| \quad \frac{}{\Box I \Rightarrow (\Box [N]_v = \Box [N \wedge I \wedge I']_v)}_{\text{INV2}}$$

INV1 : preuve par induction d'un invariant

Hypothèse : I est préservé par \mathcal{N} et le bégaiement.

Conclusion : si I est initialement vrai et toute transition vérifie \mathcal{N} ou bégaiement ($\Box [N]$), alors I est un invariant ($\Box I$).

INV2 : injection d'un invariant dans la spécification.

Règles de preuve – TLA⁺ vivacité avec équité faible

$$\begin{array}{l}
 P \wedge [\mathcal{N}]_v \Rightarrow (P' \vee Q') \\
 P \wedge \langle \mathcal{N} \wedge \mathcal{A} \rangle_v \Rightarrow Q' \\
 P \Rightarrow \text{ENABLED } \langle \mathcal{A} \rangle_v \\
 \hline
 \square[\mathcal{N}]_v \wedge \text{WF}_v(\mathcal{A}) \Rightarrow (P \sim Q) \text{WF1}
 \end{array}$$

Hypothèses :

- ① si on a P , en faisant une transition quelconque ($[\mathcal{N}]$), on conserve P ou on établit Q ;
- ② Il y a une action \mathcal{A} qui établit Q ;
- ③ Quand P est vrai, l'action \mathcal{A} est faisable.

Alors, sous contrainte d'équité faible sur \mathcal{A} , si P est vrai, \mathcal{A} doit finir par avoir lieu (car P reste constamment vrai au moins jusqu'à établir Q et P garantit que \mathcal{A} est faisable), et donc Q finira par être vrai.

Règles de preuve – TLA⁺ vivacité avec équité forte

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

Hypothèses :

- ① si on a P , en faisant une transition quelconque ($[\mathcal{N}]$), on conserve P ou on établit Q ;
- ② Il y a une action \mathcal{A} qui établit Q ;
- ③ Si P est constamment vrai, l'action \mathcal{A} finira par être faisable.

Alors, sous contrainte d'équité forte sur \mathcal{A} , si P est vrai, \mathcal{A} doit finir par avoir lieu (car P reste constamment vrai au moins jusqu'à établir Q et P garantit que \mathcal{A} sera faisable, donc est infiniment souvent faisable), et donc Q finira par être vrai.

($\Box F$ est un invariant qui facilite en général la preuve du 3)

Règles de preuve dérivées

$$\frac{\Box(P \Rightarrow \Box P) \wedge \Diamond P}{\Diamond \Box P} \text{LDSTBL}$$

$\Box(P \Rightarrow \Box P)$ signifie que P est stable : une fois vrai, il le reste. Combiné avec $\Diamond P$ (un jour P sera vrai), on obtient que P est finalement toujours vrai.

$$\frac{P \rightsquigarrow Q \wedge Q \rightsquigarrow R}{P \rightsquigarrow R} \text{TRANS}$$

Transitivité du \rightsquigarrow .



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 $\Diamond P, P \rightsquigarrow Q \dots$: 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 $O(|S| \times 2^{|F|})$.



Vérificateur TLC

Le vérificateur de modèles TLC sait vérifier :

- les spécifications avec des actions gardées dont l'ordre des variables primées est directement évaluable
($x' = 3 \wedge y' = x' + 1$ est ok, $y' = x' + 1 \wedge x' = 3$ est KO) ;
- (efficacement) les invariants sans variables primées : $\Box P$ où P est un prédicat d'état ;
- les formules de sûreté pure avec variables primées et bêgatement : $\Box[P]_v$ où P est un prédicat de transition ;
- $P \rightsquigarrow Q$ 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 exemple.



Conclusion

- Propriétés de sûreté et de vivacité : LTL (logique temporelle linéaire)
- Équité pour isoler les contraintes de progression
- Vérification mécanisée (par modèle ou par preuve axiomatique)

