Synthèse de moniteurs de références

Hervé Grall

Équipe OBASCO (EMN/INRIA – LINA)
École des mines de Nantes

8-9 juin 2006
Luchon

Plan

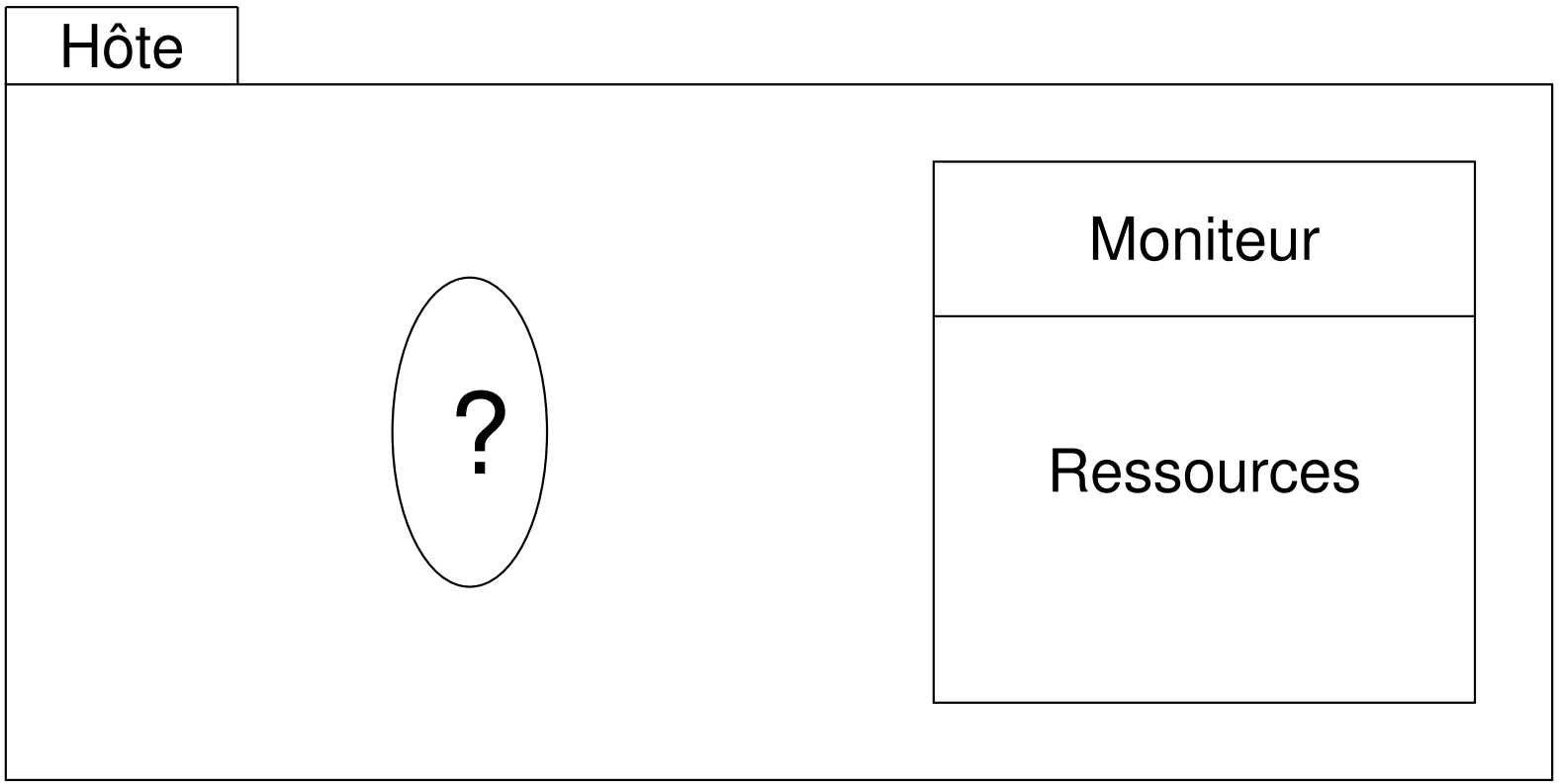
Plan

Des contrôleurs de sécurité

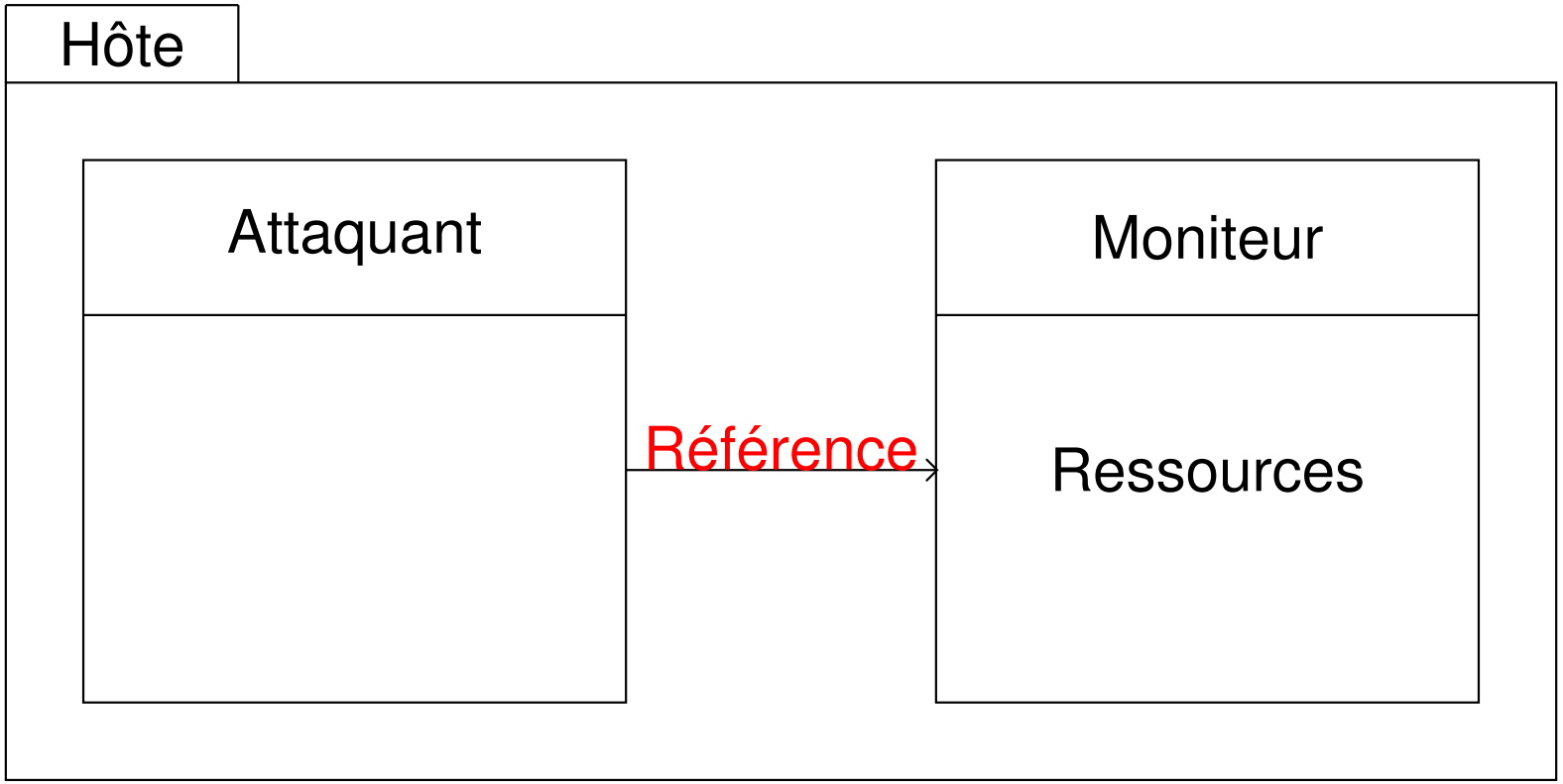
- Technique efficace pour assurer la protection d'un système
- Modèle simple
  - programmes exécutables
  - ressources auxquelles les programmes en cours d'exécution peuvent faire référence, pour y accéder
  - moniteur de références validant toute référence faite à une ressource par un programme, en accord avec les règles de sécurité

  $\longrightarrow$ Modélisation de nombreuses situations (Ex : programme client appelant une bibliothèque)
- Trois conditions
  - être résistant aux intrusions $\longrightarrow$ composant protégé
  - être systématiquement appelé $\longrightarrow$ pas de canal caché
  - être suffisamment simple pour être analysé et testé $\longrightarrow$ composant de confiance

Modèle schématique



Modèle schématique



Propriétés de sécurité garanties

- Historiquement : contrôle des accès
- Plus généralement : propriétés de sûreté (cf Schneider)
  - Moniteur de références : tout ou rien

Validation	Action
référence validée	exécution puis continuation
référence invalidée	arrêt définitif

  - Ensemble des suites de références (après contrôle) : fermé par préfixe
- Question : au delà de la sûreté ?

Plan

Contrôler des suites de références

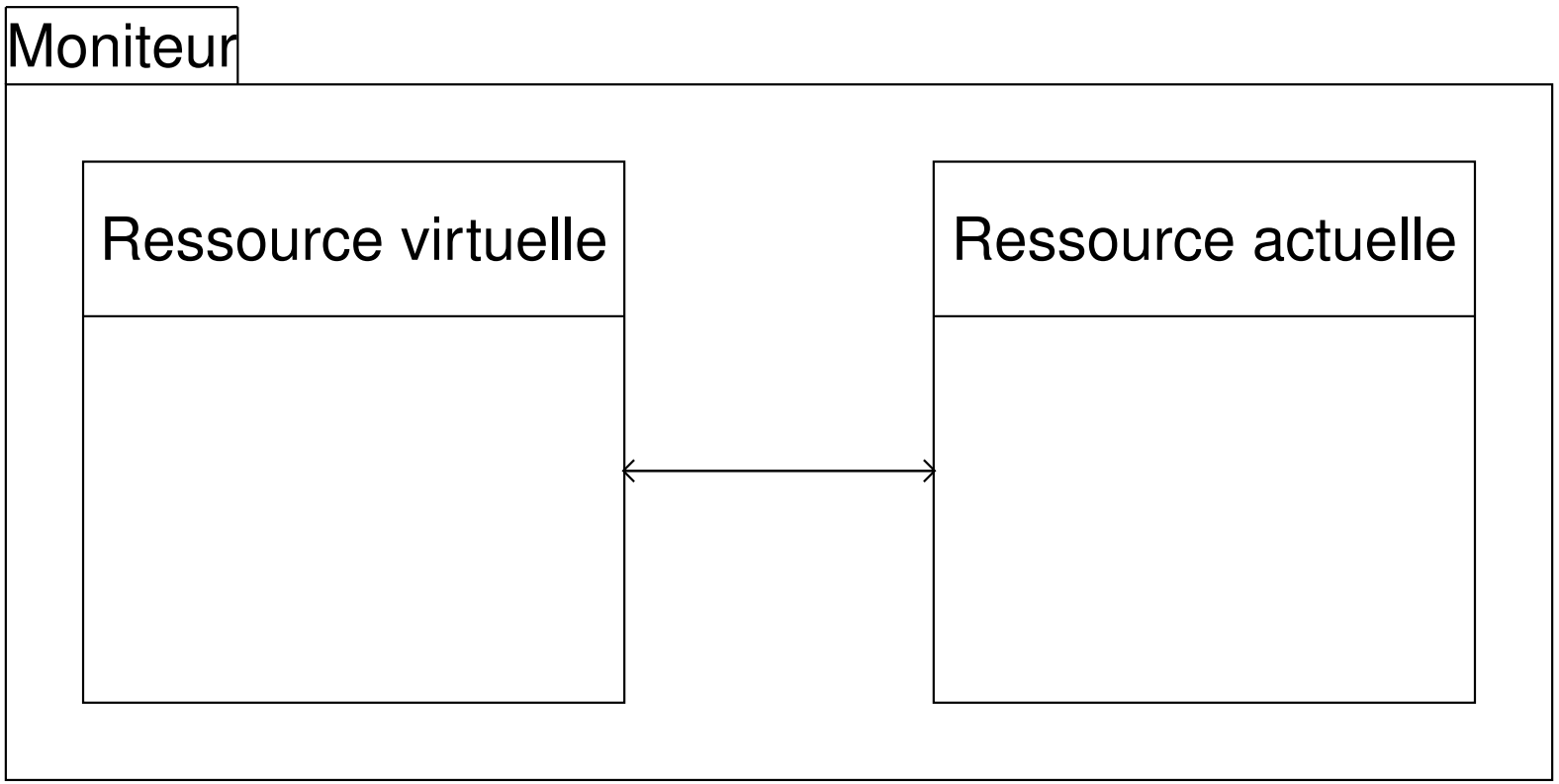
- Moniteur classique : contrôle appliqué à chaque référence, prise individuellement
- Extension : contrôle de la suite de références
  - Moniteur $\stackrel{def}{=}$ transducteur

    suite de références en entrée (requêtes) $\mapsto$ suite de références en sortie (réalisations)
  - Généralisation de travaux récents (cf “Edit Automata”, Walker et al.)

Ressources virtualisables

- Extension intéressante pour une classe particulière de ressources :
  les ressources virtualisables
- Ressource virtuelle / Ressource actuelle
  - Instances : spooling (“simultaneous peripheral operations on-line”) $\longrightarrow$ mémoire tampon, cache
  - Exemples : imprimante, ressource Web, mémoire partagée, fichier

Ressources virtualisables – Schéma



Plan

Plan

Un service simple

Le système : programme client ouvrant un service, l'utilisant et le fermant.
Références possibles :

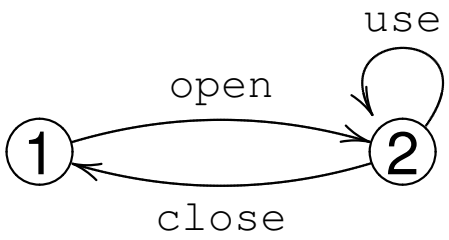
```
open
use
close
```

Interprétation

- `open` : ouverture du service, avec verrouillage de la ressource associée au service
- `use` : utilisation de la ressource
- `close` : fermeture du service, avec libération de la ressource

Le protocole d'accès au service

Protocole ouverture–usage–fermeture



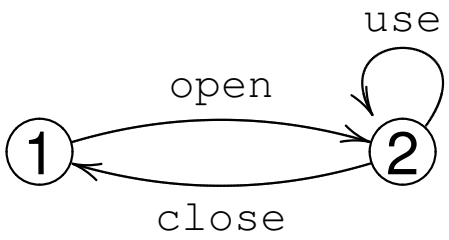
Suites finies reconnues :

(open.use*.close)*

(open.use*.close)*.open.use*

Le protocole d'accès au service

Protocole ouverture–usage–fermeture



Suites infinies reconnues :

$(\texttt{open.use}^*\texttt{.close})^\omega$

$(\texttt{open.use}^*\texttt{.close})^*\texttt{.open.use}^\omega$

Premier problème

On voudrait vérifier la propriété suivante :

- si l'exécution se termine, alors le service est toujours fermé après ouverture :

$$(\texttt{open.use}^*.\texttt{close})^*$$

Interprétation en termes de disponibilité : la fermeture rend disponible la ressource à d'autres clients, régulièrement puis finalement.

Second problème

On voudrait vérifier la propriété suivante :

- si l'exécution ne se termine pas, alors le service est toujours fermé après ouverture :

$$(\texttt{open.use}^*\texttt{.close})^\omega$$

Interprétation en terme de disponibilité : la fermeture rend disponible la ressource à d'autres systèmes clients, régulièrement.

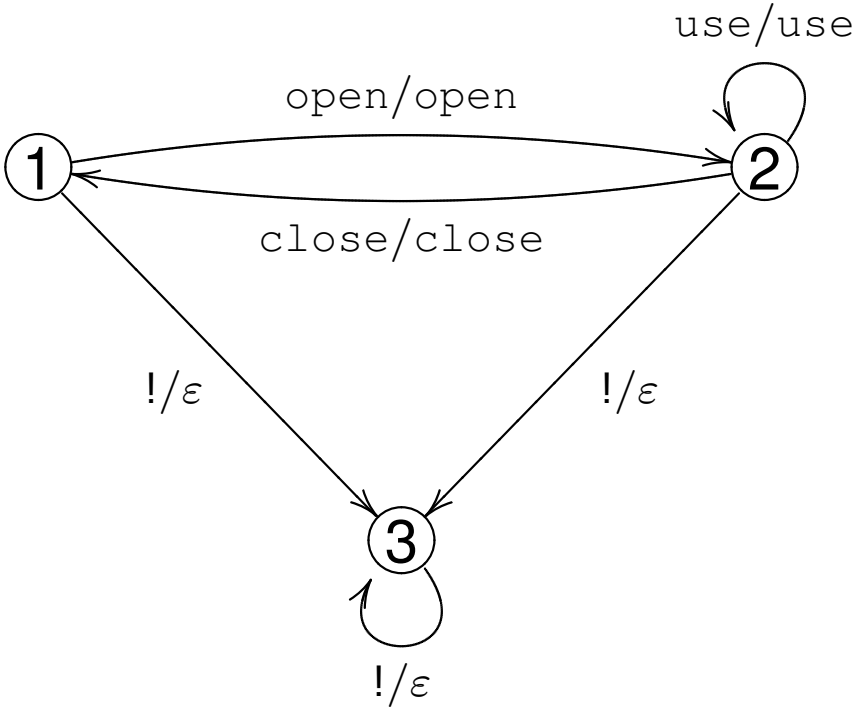
Le modèle usuel

Moniteur à la Schneider : déduit du protocole
Deux comportements possibles :

- il accepte la référence en entrée et l'exécute,
- il refuse la référence en entrée et termine le programme.

Représentation du moniteur par un transducteur

Le modèle usuel



Limitations du modèle usuel

Un moniteur de références à la Schneider garantit seulement des propriétés de sûreté.
→ Aucun des deux problèmes précédents ne peut être résolu.

Limitations du modèle usuel

Un moniteur de références à la Schneider garantit seulement des propriétés de sûreté.
$\longrightarrow$ Aucun des deux problèmes précédents ne peut être résolu.

- Premier problème
  Si l'exécution se termine, alors le service est toujours fermé après ouverture :

$$(\texttt{open}.\texttt{use}^*.\texttt{close})^*$$

  $\longrightarrow$ Pas une sûreté finitaire (non fermée par préfixe)
  $\longrightarrow$ Composante de vivacité

Limitations du modèle usuel

Un moniteur de références à la Schneider garantit seulement des propriétés de sûreté.
$\longrightarrow$ Aucun des deux problèmes précédents ne peut être résolu.

- Second problème
  Si l'exécution ne se termine pas, alors le service est toujours fermé après ouverture :

$$(\texttt{open.use}^*\texttt{.close})^\omega$$

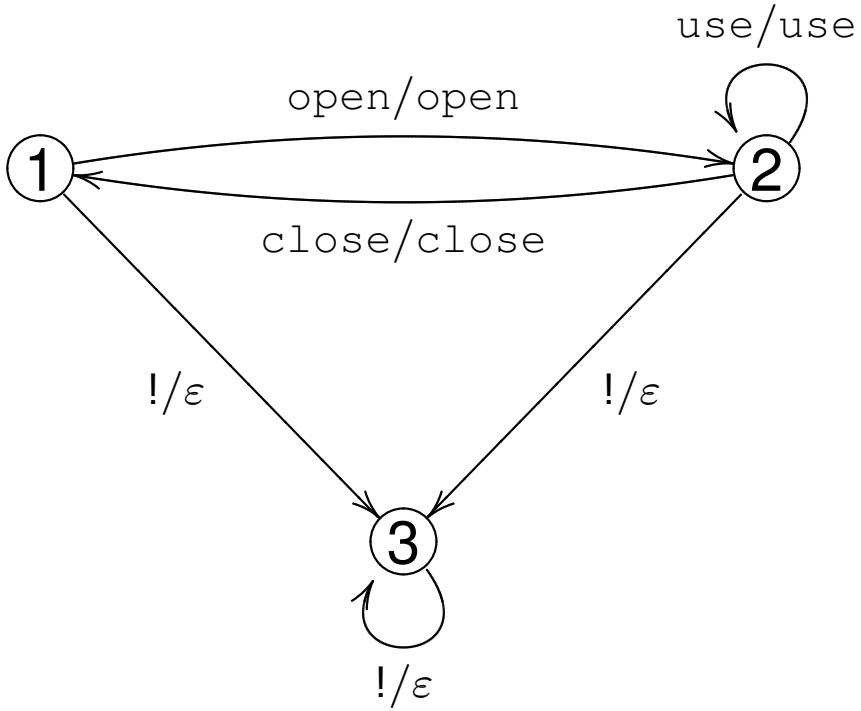
$\longrightarrow$ Pas une sûreté infinitaire

$$\texttt{open.use}^n \in \text{préfixes}, \texttt{open.use}^\omega \notin \text{propriété}$$

$\longrightarrow$ Composante de vivacité

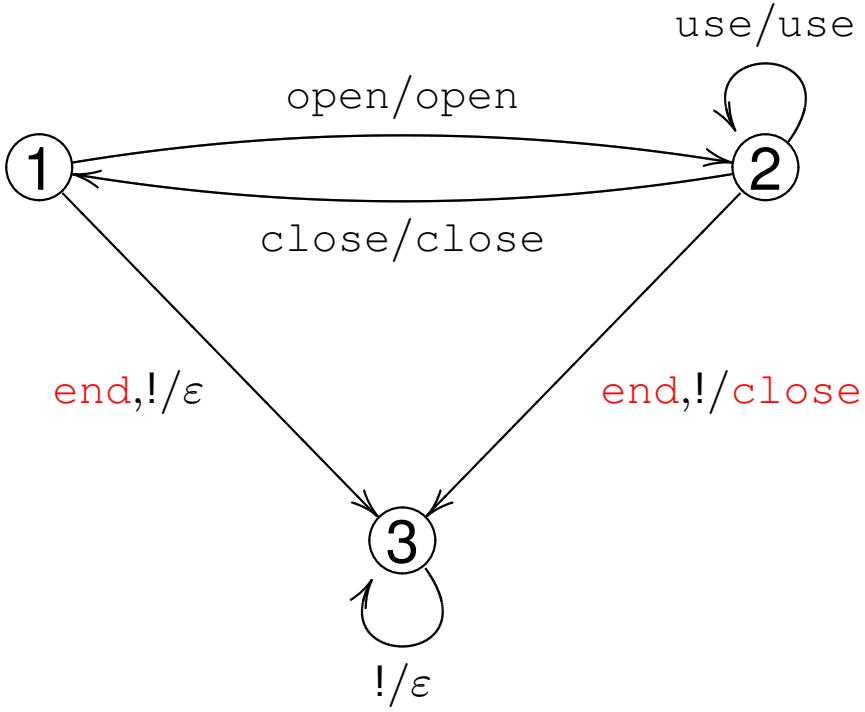
Résolution du premier problème

Si l'exécution se termine, alors le service est toujours fermé après ouverture : $(\texttt{open.use}^*\texttt{.close})^*$



Résolution du premier problème

Si l'exécution se termine, alors le service est toujours fermé après ouverture : $(\texttt{open.use}^*\texttt{.close})^*$



Résolution du premier problème

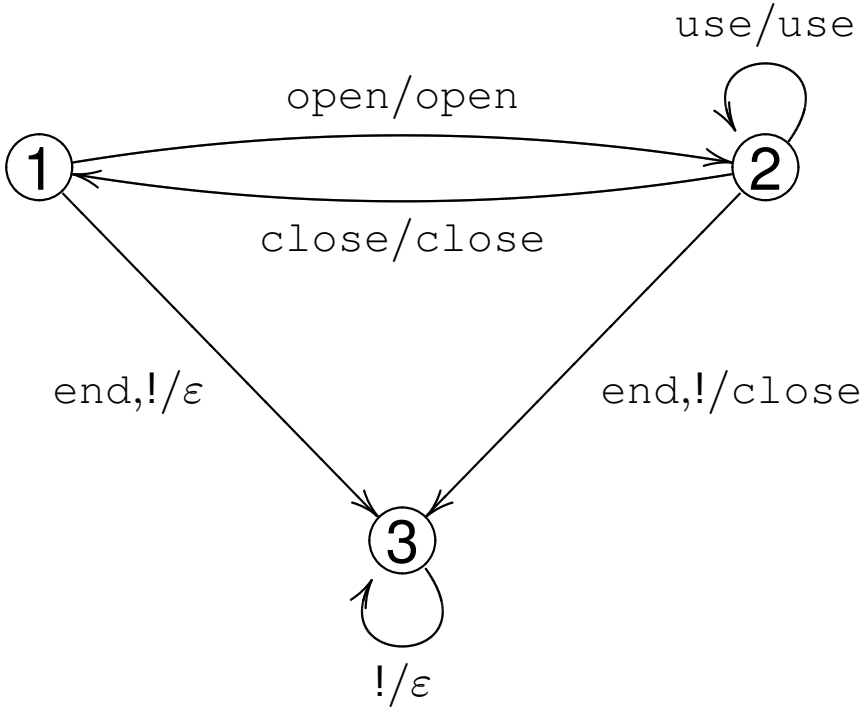
On ajoute un évènement marquant l'arrêt du programme (évènement `end`) :

- lorsque l'exécution se termine, l'évènement `end` est transmis au moniteur ;
- le moniteur peut alors compléter la suite pour produire en sortie une suite acceptable.
  $\longrightarrow$ Correction

$\longrightarrow$ Le moniteur garantit les propriétés :
$(\texttt{open.use}^*\texttt{.close})^*$ (du premier problème)
$(\texttt{open.use}^*\texttt{.close})^\omega +$
$(\texttt{open.use}^*\texttt{.close})^*\texttt{.open.use}^\omega$ (sûreté infinitaire)

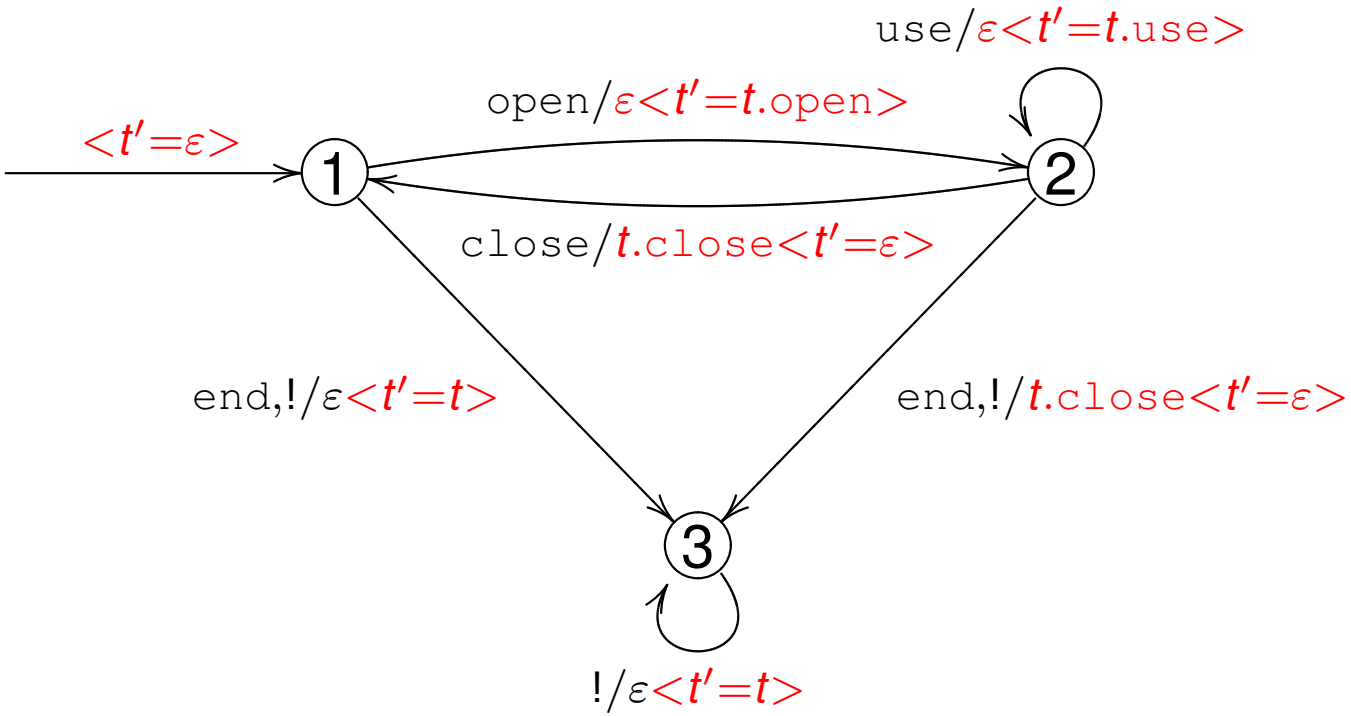
Résolution du second problème

Si l'exécution ne se termine pas, alors le service est toujours fermé après ouverture : $(\texttt{open.use}^*\texttt{.close})^\omega$



Résolution du second problème

Si l'exécution ne se termine pas, alors le service est toujours fermé après ouverture : $(\texttt{open.use}^*\texttt{.close})^\omega$



Résolution du second problème

On ajoute un mécanisme de transaction (représenté par les affectations de la variable t) :

- la transaction t est initialement vide ;
- à chaque demande d'ouverture du service, une nouvelle transaction commence ;
- elle se termine à la demande de fermeture du service ou en cas d'arrêt du programme : elle est alors exécutée puis réinitialisée.

$\rightarrow$ Le moniteur garantit les propriétés :
$(\texttt{open.use}^*\texttt{.close})^*$ (du premier problème)
$(\texttt{open.use}^*\texttt{.close})^\omega$ (du second problème)

Plan

Utilisation d'un service et arrêt

Le système : programme client utilisant un service, et terminant suivant deux procédures
Références possibles :

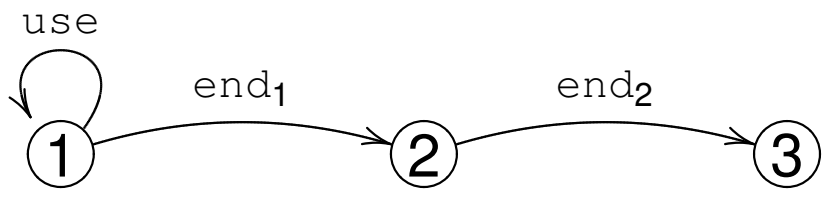
$$\texttt{use}$$
$$\texttt{end}_1$$
$$\texttt{end}_2$$

Interprétation

- `use` : utilisation de la ressource
- $\texttt{end}_1$: arrêt suivant la procédure courte, supposée non satisfaisante
- $\texttt{end}_1.\texttt{end}_2$: arrêt suivant la procédure longue, supposée satisfaisante

Le protocole d'utilisation du service

Protocole : utilisation et éventuellement arrêt



Suites finies reconnues :

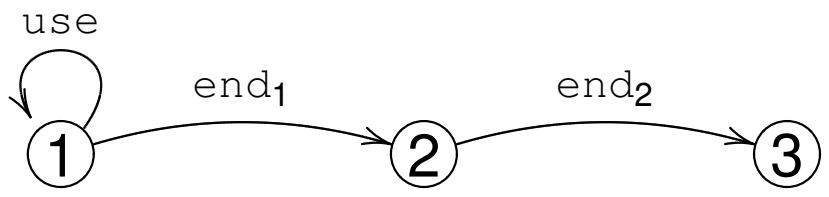
$\texttt{use}^*$

$\texttt{use}^*.\texttt{end}_1$

$\texttt{use}^*.\texttt{end}_1.\texttt{end}_2$

Le protocole d'utilisation du service

Protocole : utilisation et éventuellement arrêt



Suites infinies reconnues :

$\texttt{use}^\omega$

Problème

On voudrait vérifier la propriété suivante :

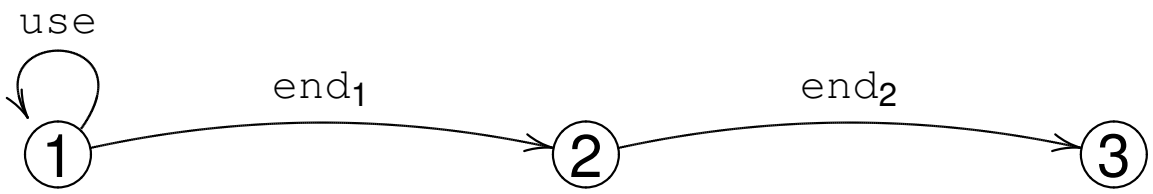
- si l'exécution se termine, alors elle se termine suivant la procédure longue :

$$\mathtt{use}^*.\mathtt{end}_1.\mathtt{end}_2 + \mathtt{use}^\omega$$

Interprétation en termes de disponibilité : la procédure longue d'arrêt libère les ressources contrairement à la procédure courte.

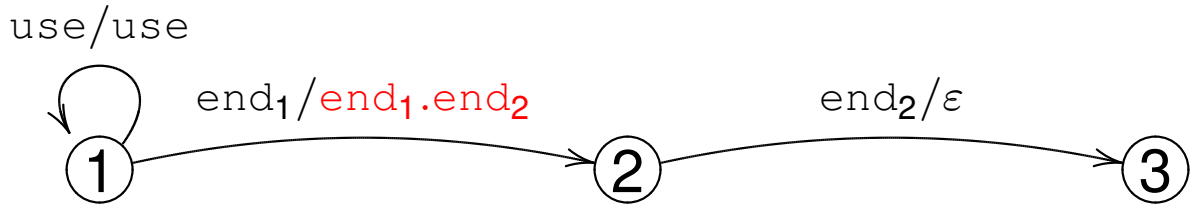
Résolution du problème

Si l'exécution se termine, alors elle se termine suivant la procédure longue : $\texttt{use}^*.\texttt{end}_1.\texttt{end}_2 + \texttt{use}^\omega$



Résolution du problème

Si l'exécution se termine, alors elle se termine suivant la procédure longue : $\mathtt{use}^*.\mathtt{end}_1.\mathtt{end}_2 + \mathtt{use}^\omega$



Résolution du problème

On réalise une anticipation :

- lorsque l'exécution se termine suivant la procédure courte, on anticipe un arrêt suivant la procédure longue.

$\rightarrow$ Le moniteur garantit la propriété :

$\mathtt{use}^*.\mathtt{end}_1.\mathtt{end}_2 + \mathtt{use}^\omega$

Plan

Le modèle

- Moniteur de références : fonction croissante et continue θ

  suite de références en entrée $\mapsto$ suite de références en sortie
- Propriété P (ensemble de suites) à garantir
- Propriété S définissant les points d'arrêt

  $\rightarrow$ domaine de θ : $\text{préfixes}(S) \cup S$
- Deux contraintes
  - CORRECTION : le moniteur garantit aux points d'arrêt la propriété P.

    $$\forall\, x \in S\,.\ \theta(x) \in P$$
  - NEUTRALITÉ : le moniteur accepte sans la modifier une suite appartenant à la propriété P.

    $$\forall\, x \in P\,.\ \theta(x) = x$$

Le problème de la synthèse

- Déterminer un critère $\mathcal{C}(S, P)$ tel que
  $\mathcal{C}(S, P)$ est vérifié
  si et seulement si
  il existe un moniteur de références correct et neutre pour P et S.
- Synthèse effective en utilisant un mélange de transactions, d'anticipations et de corrections

Aperçu du critère et de la synthèse

- Une condition locale relative aux points d'arrêt
- Plusieurs conditions pour les point-limites

$\in P_\omega$		$\notin P_\omega$	
$\in \text{Lim } P_*$	$\notin \text{Lim } P_*$	$\in \text{Lim préfixes}(P)$	$\notin \text{Lim préfixes}(P)$
cond.	cond. anticipation	cond. transaction	- correction