# Politiques de sécurité: génération d'attaques et détection d'intrusion basées modèles

Thierry Jéron, Hervé Marchand, Vlad Rusu

Irisa – Projet Vertecs

# Nos compétences / Potestat

Modèles et techniques de vérification

(i.e. model checking, interprétation abstraite, preuve)

appliquées à :

- Génération de tests de conformité / objectif de test
- Synthèse de contrôleurs / objectif de contrôle
- Diagnostic / motifs de surveillance

**Modèles :** automates et systèmes de transition, étendus avec variables (commandes gardées)

**Techniques :** model-checking, analyses statiques, interprétation abstraite, résolution de contraintes.

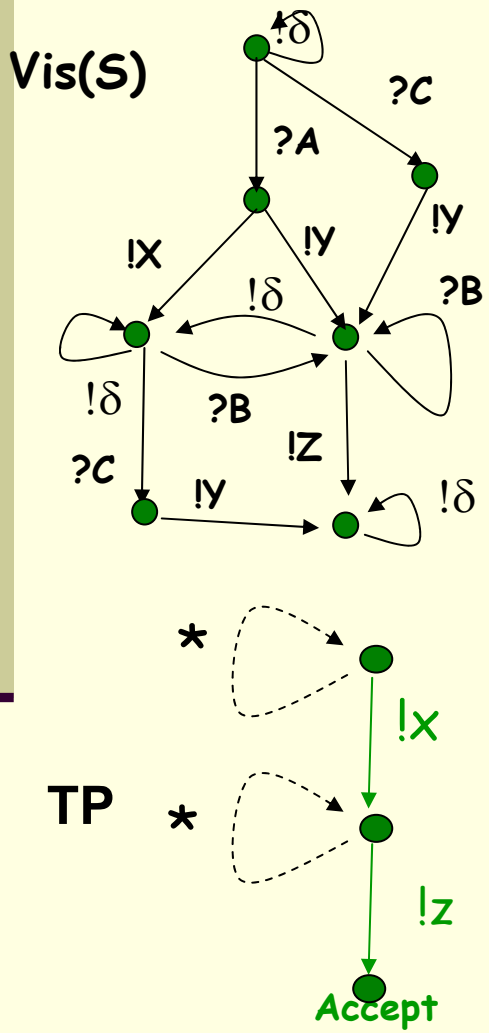
**Outils :** test (TGV, STG), contrôle (Sigali), IA (NBAC)

# Plan

- Génération d'attaques par techniques de génération de tests
  - Rappel sur la génération de tests
  - Génération d'attaques
- Détection d'intrusion par techniques de diagnostic

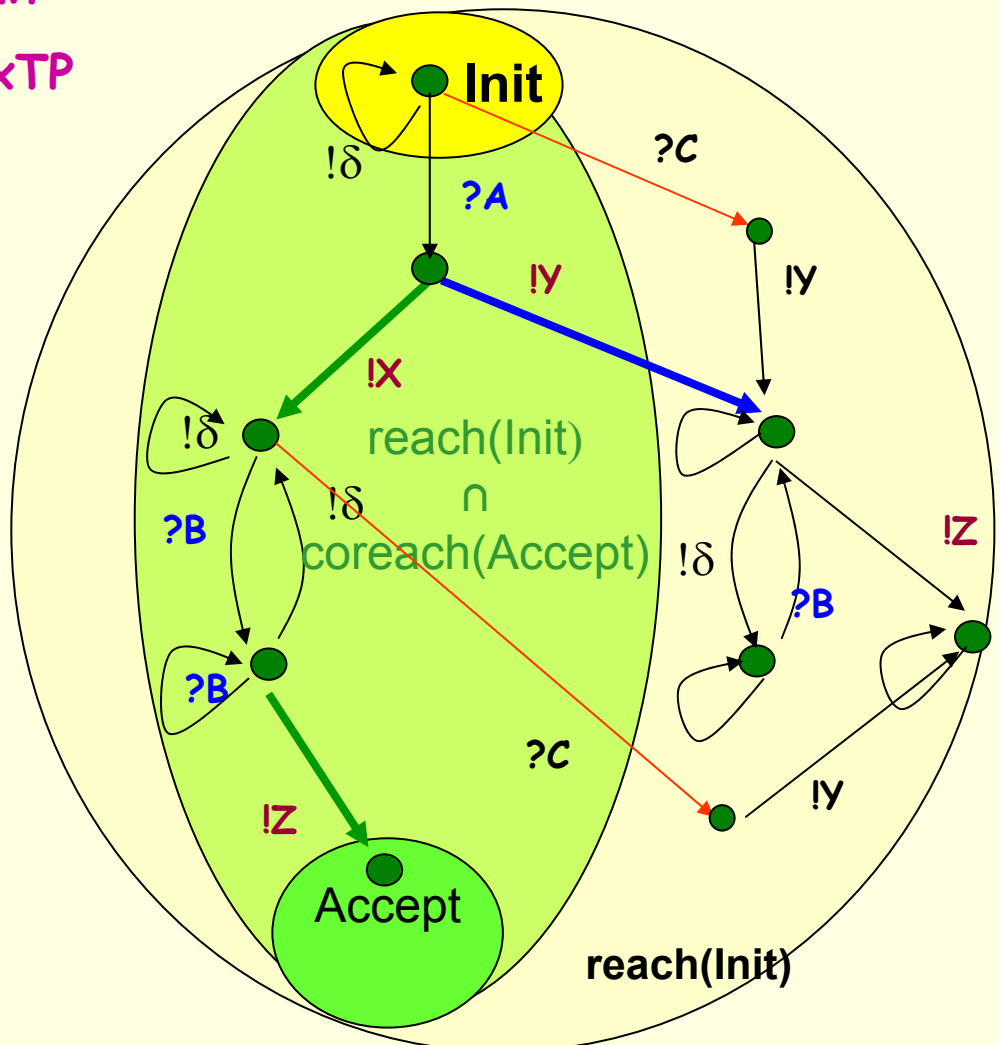
Basées sur des modèles

# Principes de la génération de tests



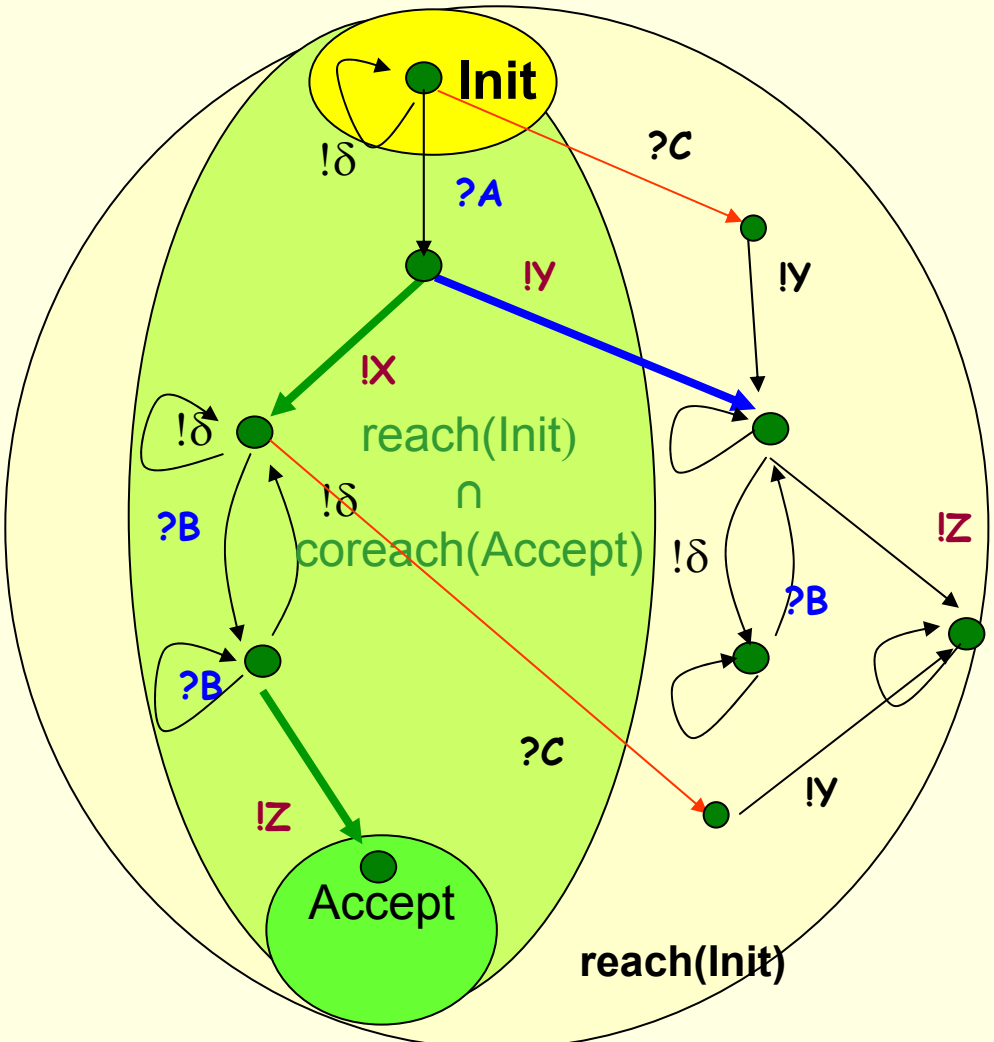

Produit

Vis(S)xTP



# Étapes de la génération

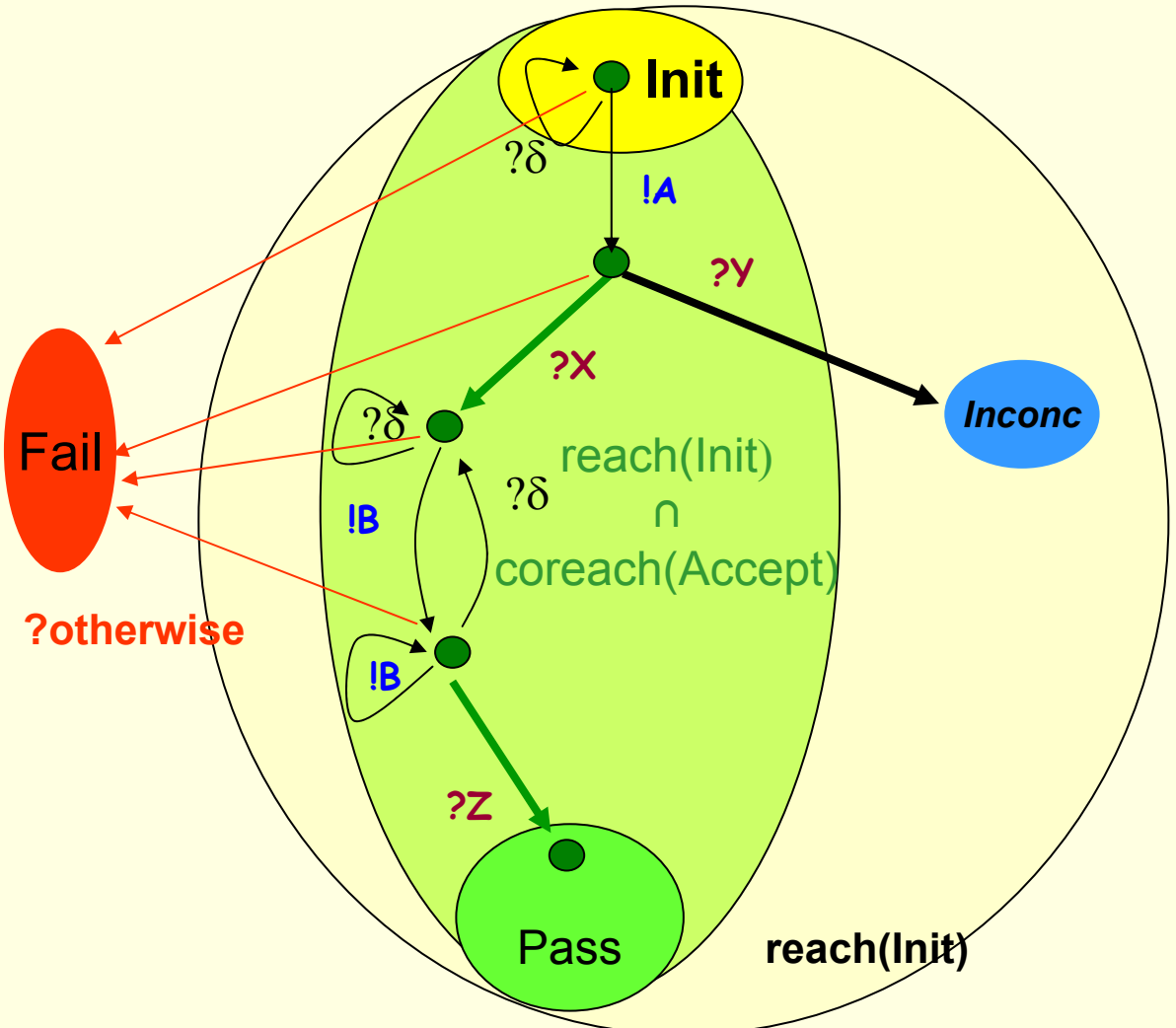
**Sélection**



# Étapes de la génération

Verdicts
Mirroir

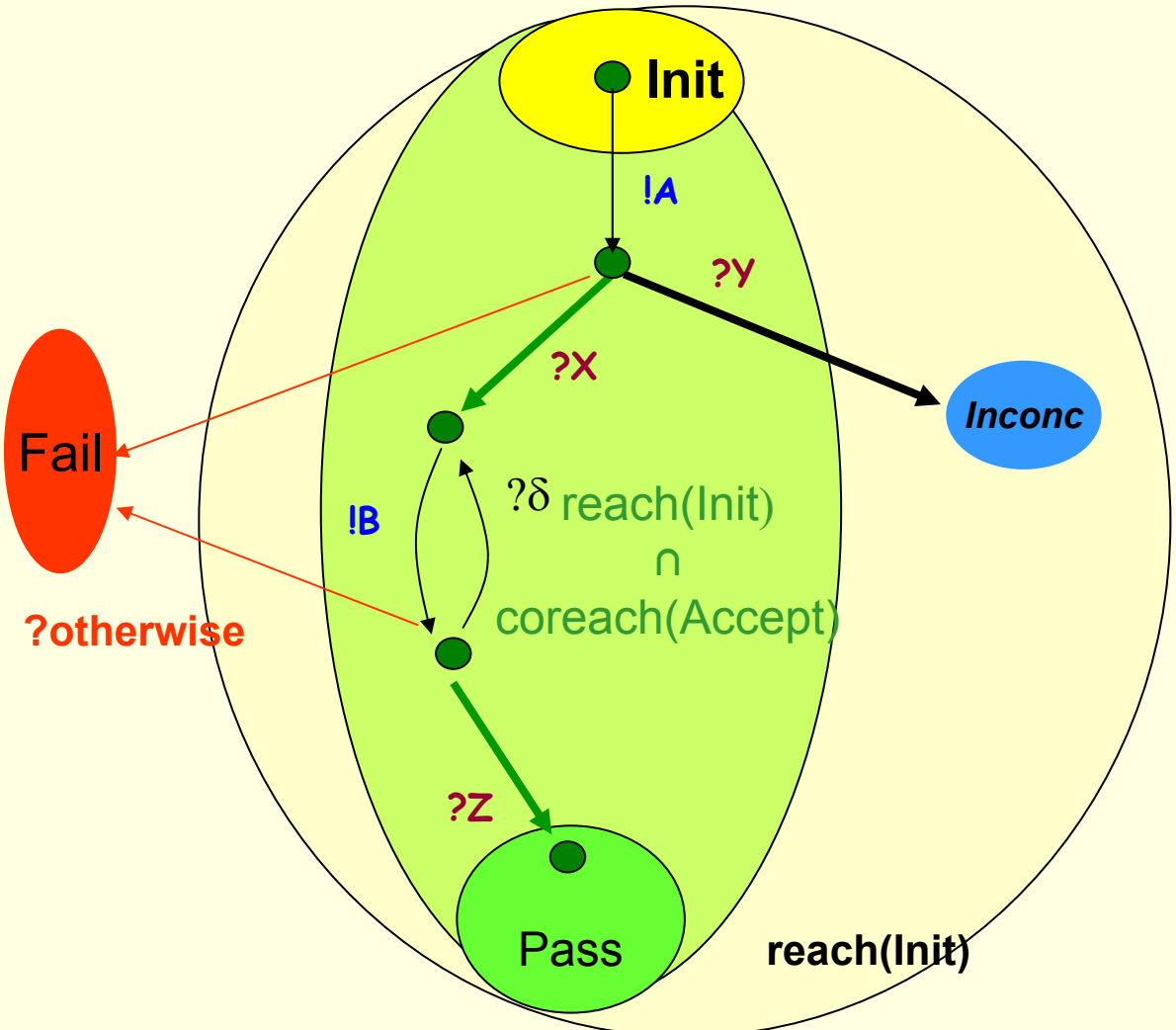
CTG



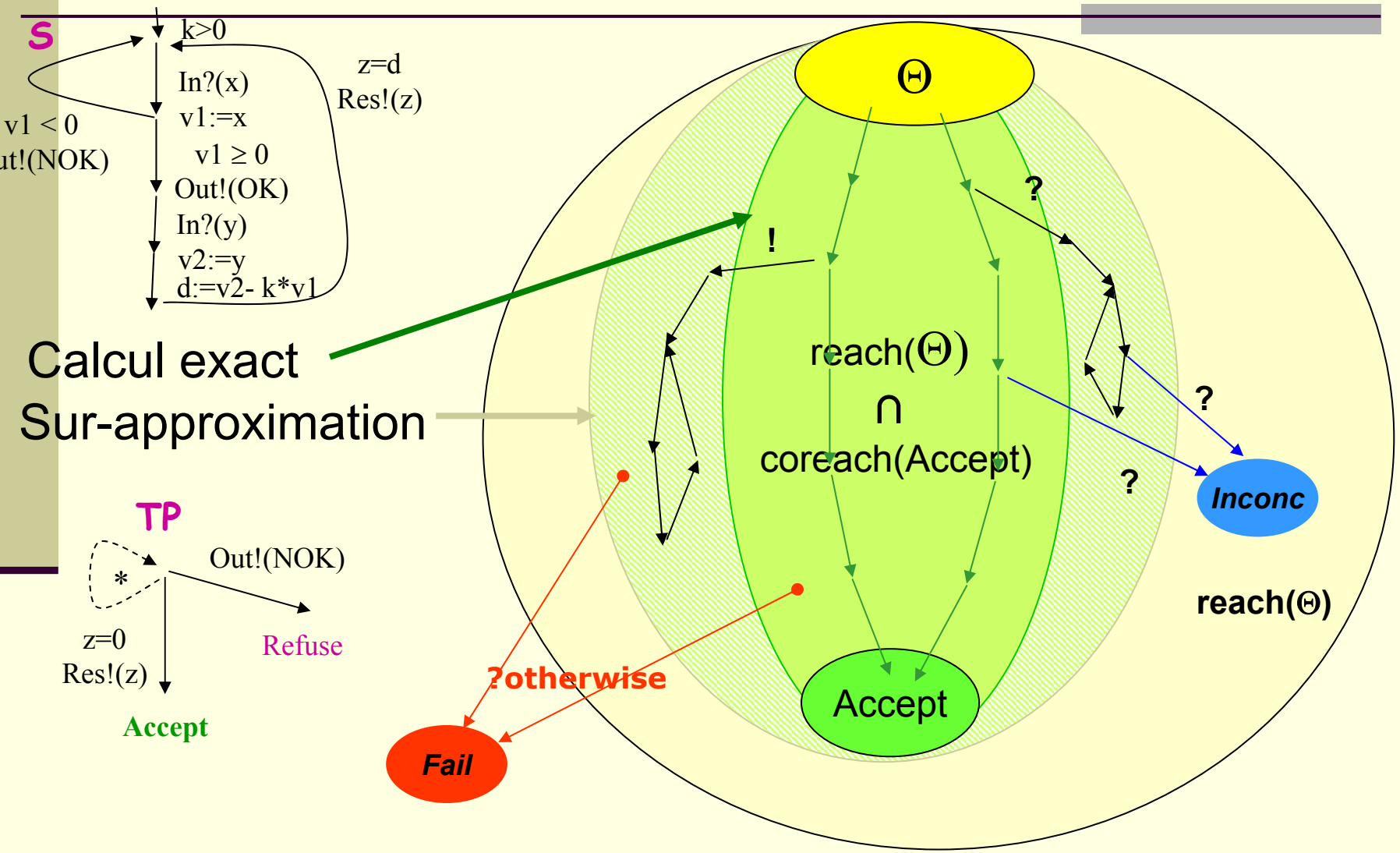
# Étapes de la génération

Conflits de contrôlabilité

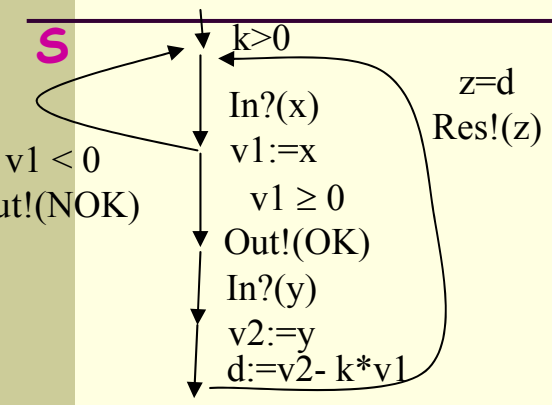
TC



# Extensions à des automates avec variables

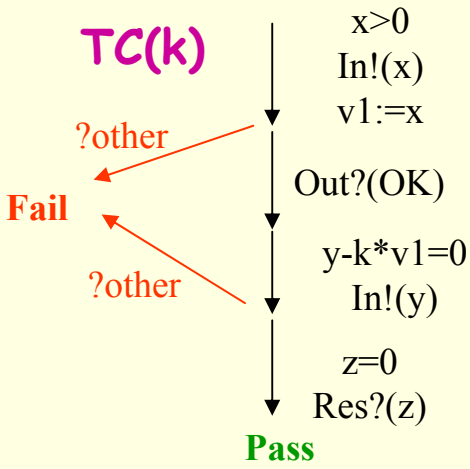


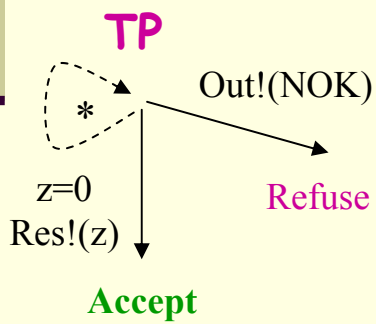
# Extensions à des automates avec variables




Calcul exact

Sur-approximation



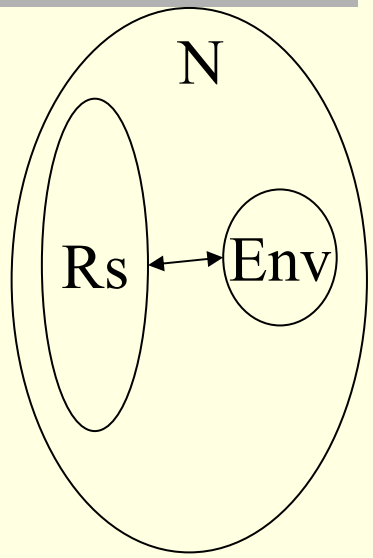
# Modèles pour la génération d'attaques

- Modèle de description du réseau (+ envt)
  automates étendus communicants avec variables

  e.g. spécification en IF (← UML)

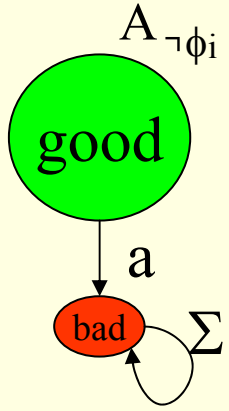
  abstraction nécessaire/réseau réel:

  $L(RN) \subseteq$ mapping $(L(N))$




- Propriétés de sécurité $\{\phi_i | i=1..n\}$ :
  propriété de sûreté (i.e. peut être violée sur une trace finie)

  → observateurs à la Schneider i.e. automates (étendus avec variables)
  tq $L(A_{\neg\phi i}, \text{Bad}) = L(\neg\phi_i)$



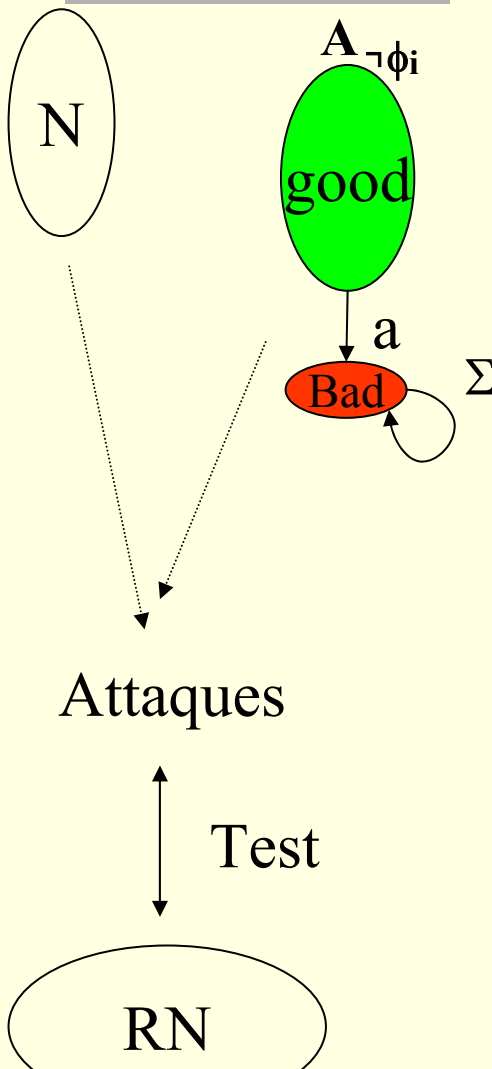
# Génération d'attaques

Depuis le modèle du réseau et la propriété de sécurité par des techniques de génération de test:

$$N, A_{\neg\phi i} \rightarrow Att$$

Pb: trouver $Att \in L(N) \cap L(A_{\neg\phi i}, Bad)$

**Principe:** amener le réseau réel RN là où il risque de violer $\phi_i$



# Calcul d'attaques par techniques similaires à la génération de test

**Hyp:** le modèle de réseau est une abstraction du réseau réel

i.e. L(RN) $\subseteq$ mapping(L(N))

Att $\leftrightarrow$ coreach(Bad) dans N x $A_{\neg\phi i}$

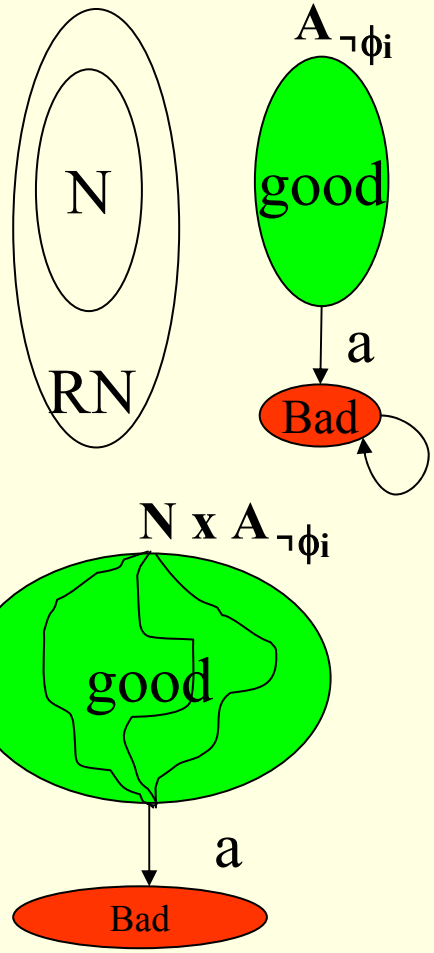
**Pb sup :** coreach(Bad) non calculable de manière exacte $\Rightarrow$ sur-approximation

$Att^{\alpha} \leftrightarrow$ $coreach^{\alpha}$(bad)

Progr. d'attaque non instancié dont certains chemins peuvent être infaisables

Chemins faisables jqa Bad: attaque ds N

(peut être fausse sur RN $\leftarrow$ abstraction)



# Extension

- Utilisation de nos travaux sur le test de violation de propriétés de sûreté.
- Hyp: $L(RN) \subseteq mapping(L(N).\Sigma^*)$
  i.e. le réseau réel peut diverger du modèle
- Génération sur $L(N).\Sigma^*$:
  - Verdicts:
    - non-conformité de RN / N
    - violation de la prop. sécurité par RN même si N satisfait la prop.
    - violation de la propriété de sécurité par N et RN

# Test d'attaques

- On-line test execution: Att || RN
  - Instanciation des entrées de l'attaque par résolution de contraintes
  - Détection de
    - non-conformité de RN/N
    - violation de $\phi_i$ par RN et/ou N

# Renforcement de politiques de sécurité

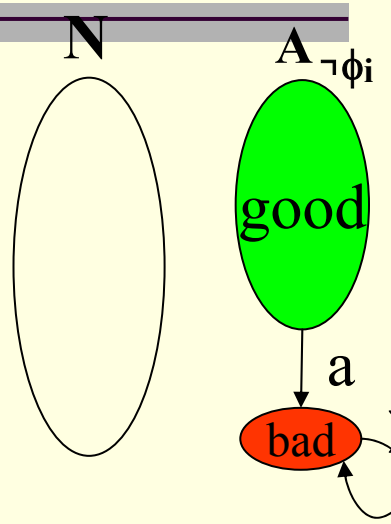
**Contrôler** le modèle de réseau de sorte qu'il satisfasse les propriétés de sécurité

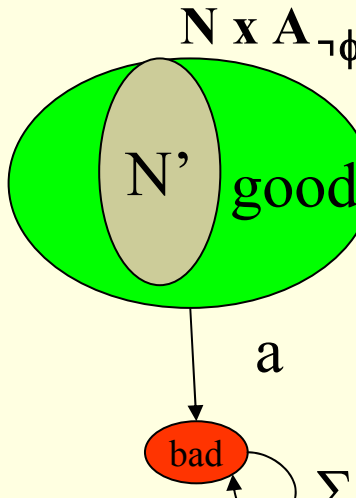
$N \models \phi_i$ ? i.e. $L(N \times A_{\neg\phi i}, Bad) = \varnothing$ ?

- vrai si Bad inaccessible dans $N \times A_{\neg\phi i}$
- sinon construire N'= N x Ctrl tq $N' \models \phi_i$ (actions de l'intrus supposées incontrôlables)

Pb: $N \models \phi_i$ est indécidable en général.
⇒ sur-approximation par Int. Abst.

- vrai sur modèle abstrait ⇒ vrai sur modèle concret
- faux sur modèle abstrait ⇒ inconnu sur modèle concret



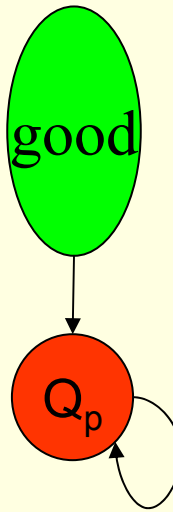
# Utilisation du diagnostic pour la détection d'intrusions

- **Monitoring / test passif / détection d'intrusion** :
  - Sondes/capteurs dans le système → observables
  - Analyse des traces observables / comportement attendu ou scenarios d'attaques (automate à la Schneider)
- **Diagnostic** [Sampath et al]
  - Sondes/capteurs → observables
  - Observation des traces observables
  - Modèle du système supposé correct
    ⇒ Le diagnostiqueur peut inférer des informations sur les comportements internes en observant ses traces observables
- Utilisation des techniques de diagnostic pour affiner les techniques de détection d'intrusion:
  - Actions internes dans la description de scenario d'attaques
  - Détection d'intrusion sans observer toutes les actions de l'attaque.

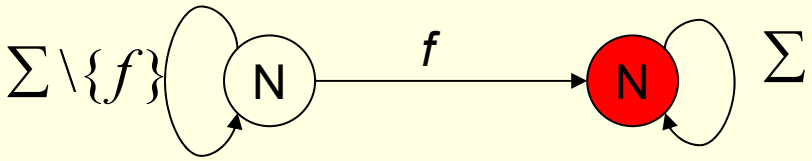
# Principe du diagnostic

- **Problématique** : Diagnostiquer l'occurrence de motifs
  - d'événements particuliers
  - de séquences d'événements, etc

- **Modèle**
  - système G=(Q, Σ, δ, $x_0$), avec $\Sigma = \Sigma_0 \bigcup \Sigma_{uo}$
  - motif de surveillance : O =($X_o$,Σ, $\delta_o$, $x_o$)
    - Complet, déterministe
    - Ensemble stable d'états finals $Q_p$

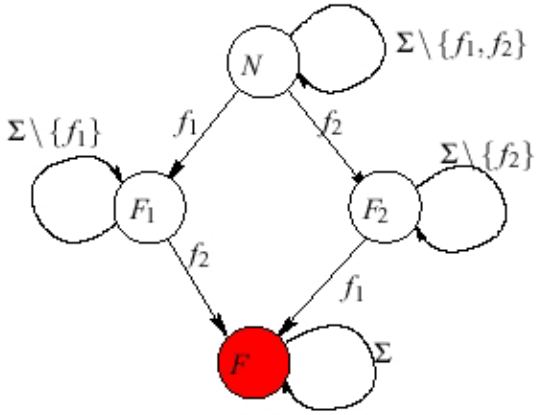



  - Exemple:

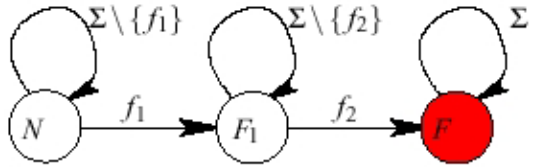



- **Séquences à surveiller** : $s \in L(G) \cap L(O, Q_P)$

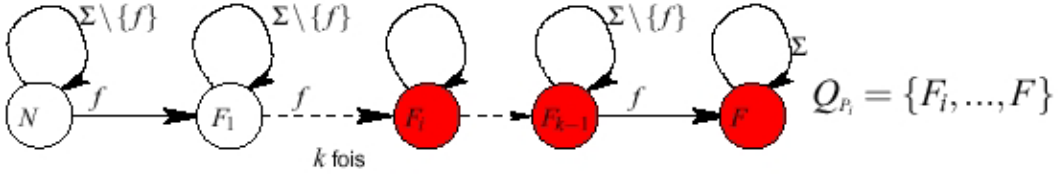
# Exemples de motifs de surveillance



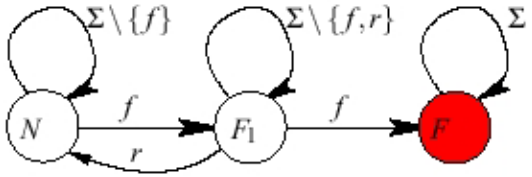

Diagnostic de $f_1$ et $f_2$

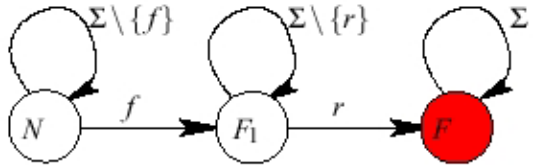



Pannes en cascade ($f_1$ suivie de $f_2$)




Occurrences multiples de la même panne ($k$ et $[1-k]$ diagnosticabilité)



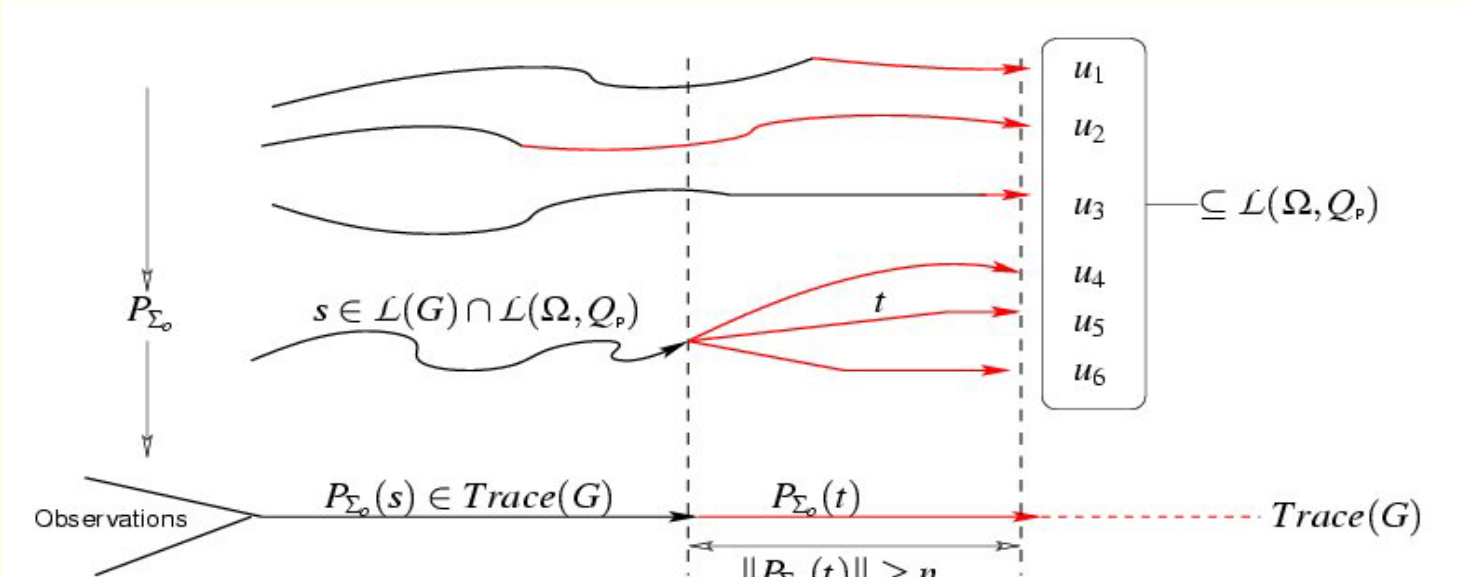




Diagnostic de pannes non permanentes

# Principe du diagnostic (II)

- **But** : déterminer si le système a effectué une séquence s acceptée par $O_P$ en $Q_P$ en observant seulement des événements de $\Sigma_o$
- Système G **diagnosticable** / O ssi

$$\exists n, \forall s \in L(G) \cap L(O, Q_P), \forall t \in L(G)/s, \left\| P_{\Sigma_o}(t) \right\| \geq n_f \Rightarrow$$
$$\forall \omega \in L(G), Trace(st) = Trace(\omega) \Rightarrow \omega \in L(O, Q_P)$$



# Diagnosticabilité

- Vérifier la diagnosticabilité d'un système par rapport à un motif de surveillance
  - Soit $G_0 = \varepsilon(G \times O) = \langle X_o, \Sigma_o, x_o, \rightarrow_o \rangle$

    ($\varepsilon$-cloture = $\tau$*a, LTS non-déterministe de même trace que G x O et sans action interne)
  - G diagnosticable / O $\Leftrightarrow$ pas de cycle indéterminé dans $G_o$ x $G_o$

    où

    état indéterminé $x=(q_1,q_2)$ si $q_1 \in QxQ_p$, $q_2 \notin QxQ_p$

    i.e. pas de cycle observable correspondant à deux cycles de G, l'un accepté, l'autre non.

# Diagnostiqueur

- Diagnostiqueur: $G_d = Det(G \times O) = \langle X, \Sigma_o, x_o, \rightarrow \rangle$
  - muni de $Diag : X \rightarrow \{P, N, IN\}$ t.q.

$$\forall x \in X,\ Diag(x) = \begin{cases} P \text{ si } x \subseteq Q \times Q_P \\ N \text{ si } x \cap Q \times Q_P = \phi \\ In \text{ sinon} \end{cases}$$

# Extensions

- Motifs de surveillance où $Q_p$ non stable

  ↔ pannes intermittentes, répétition d'attaques

- Modèles de systèmes et motifs de surveillance

  → automates étendus avec variables

  - Diagnosticabilité: indécidable mais sur-approximation diagnosticable implique diagnosticable
  - Construction syntaxique du diagnostiqueur par $\varepsilon$-cloture et déterminisation symboliques (travaux similaires en génération de tests)

# Adapation pour la détection d'intrusions

- Modèle du système: automate étendu
  Hypothèse de « correspondance » entre modèle et système réel (mapping)

- Propriété de sécurité descriptible par automate étendu : motif de surveillance

- Diagnosticabilité = capacité à détecter les violations de propriété de sécurité
  Non-diagnosabilité ⇒ - ajouter des sondes
  - contrôler les entrées

- Diagnostiqueur: IDS, observateur de violation de propriété de sécurité.