Institut
Mines-Télécom

Système à transitions probabilistes et vérification

Structure du cours

- Motivations
- Quelques éléments de contexte : les bases en probabilités
  - Vocabulaire clés, notations importantes
  - Théorèmes et principes requis
- Modèle de chaine de Markov à Temps Discret
- Problèmes de vérification vs quantification
  - Solution 1 : extension de CTL => PTCL
  - Solution 2 : Récompense et intervalle de confiance
- Produit de Chaines de Markov
  - Produit de CTMD endogène (i.e. reste une CMTD)
  - non déterminisme & probabilité (intuition du PMD)

Motivations

Evaluer des objectifs de fiabilité / disponibilité

- Fiabilité : capacité du système à produire un résultat correct
- Pb : incertitude sur l'occurrence de l'activation des fautes et l'apparition de défaillances
- Idée : modéliser l'activation comme un phénomène aléatoire mais quantifiable (e.g. probabilité).
- Objectif :
  - Quels formalismes peuvent être utiles ?
  - Quelles sont les analyses faisables sur de tels modèles ?

- Un petit exemple peut être ?

Vous n'avez pas confiance dans votre régulateur ? Dupliquez le….

- **Pb : vous souhaitez intégrer un composant de détection de panneaux de signalisation dans un drone terrestre.**
  - **Vous ne faites pas trop confiance aux solutions clés en main mais vous ne pouvez pas développer votre solution**
  - **Vous pouvez en acheter plusieurs … de différentes sources/ utilisant différentes technologies**
- **Solution : embarquez N versions différentes de la même fonction et fusionnez les résultats…**
- **Idée : modéliser l'exécution et l'aléa du système et se munir d'une logique vérifiable pour exprimer le choix de la réponse**

Autres application des modèles probabilistes

- Algorithmes reposant sur un choix aléatoire :
  - Synchronisation
  - Diffusion par commérage
  - Sécurité (aléa de génération de défis)
- Intérêt des logiques temporelles (permet d'avoir des expression clés en main pour analyser des propriétés d'intérêt pour la SdF)
- Il faut pouvoir quantifier la vraisemblance des chemins d'exécution les uns par rapports aux autres.

Quelques bases en probabilités …

Kit de survie pour les probabilité

- **Espace de probabilité : $(\Omega, \mathcal{F}, P)$**
  - **Ω ensemble des « cas possibles »**
  - **$\mathcal{F}$ ensemble de parties de Ω permettant de définir une mesure (surtout utile si Ω est infini non dénombrable)**
  - **P mesure dans [0,1] définie sur $\mathcal{F}$**
    cette fonction doit par extension permettre de « mesurer » toute partie de Ω)
- **Evénement = sous ensemble de Ω => P(E)= mesure de E.**
- **Attention on manipule rarement Ω … on utilise plutôt le notion de variable aléatoire**

Kit de survie pour les probabilité

- **Variable aléatoire X (le truc réellement manipulé)**
  Fonction de Ω vers le E domaine mesurable de la variable.
  $A \subseteq E, P_X(B) = P(X^{-1}(A)) = P(X \in A)$
- **En X n'a pas 1 valeur dans E mais un ensemble de valeur de E aura une probabilité d'être la valeur retournée si on consulte X (attention, on part du principe que cette valeur n'est pas consultable => on fait donc des hypothèses dessus)**
- **Correlations : 2 variables aléatoires X1,X2 booléennes t.q.**
  $P(X1 \in \{true\}) = 0.4, P(X2 \in \{true\}) = 0.4$ **=>**
  $P(X1,X2 \in \{(true, true)\}) \in [0,0.4]$
  tout dépend de la mesure de $X1^{-1}(\{true\}) \cap X2^{-1}(\{true\})$

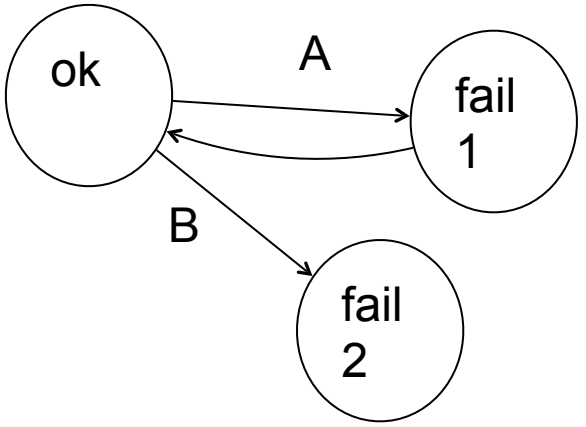
Indépendance, probabilités conditionnelles et échantillonnage

- **2 événements de Ω = 2 sous ensembles**
  Questions : est ce que le fait d'appartenir au sous ensemble A change la probabilité d'être aussi dans B …
  ~ à mesure de $(A \cap B)$ dans A proportionnellement identique à mesure de B dans Ω == indépendance …
- **Probabilités conditionnelles** $P(\cdot \mid \cdot)$:
  $P(B \mid A) \cdot P(A) = P(A \cap B)$
- **Independence v2:** $P(B) \cdot P(A) = P(A \cap B)$
- **Evénements disjoints** : $P(A \cup B) = P(A) + P(B)$

De l’automate fini à la chaîne de Markov

Rappel automate fini étiqueté.

- **Q : états**
- **Σ : étiquette des transitions**
- **Δ : transitions (q,q',a) dans $Q \times Q \times \Sigma \cup \{\epsilon\}$**
  Σ est l'ensemble des étiquette de transition et ϵ modélise l'absence d'étiquette.
- **I : ensemble des état initiaux.**
- **AP : ensemble de variables propositionnelles**
- **LQ : étiquetage d'état de Q dans Partie(AP).**
  (comme pour les structures de Kripke)



Chaine de Markov à temps discret

Machine à état fini + probabilités sur transitions et état initial

CMTD sur D = séquence de variables aléatoire (Xi),i entier tel que

- **Chaque Xi est une variable aléatoire dans D**
- **Xi représente l'état du système à la date discrète i**
- **$P(X_i=v_i \mid X_{i-1}=v_{i-1}, X_{i-2}=v_{i-2},\ldots X_0=v_0)= P(X_i=v_i \mid X_{i-1} =v_{i-1})$ est une constante (i.e. ne change pas en fonction de i — du temps) == chaîne homogène**
- **Si |D|=n, nous prendrons à partir de maintenant D= {1,…,n}**

Matrice de transition

- **Distribution de probabilité pour un état « incertain » X,** $v = (v_1, \ldots, v_n)$**, t.q.** $\sum_{1 \leq i \leq n} v_i = 1$**, et** $P(X = i) = v[i]$
- **Matrice de transition =** $(M_{i,j})_{1 \leq i \leq n, 1 \leq j \leq n}$**, tq pour tout k,** $P(X_{k+1} = j \mid X_k = i) = M_{i,j}$
  - **Propriété 1 : si v distribution de probabilité pour l'état** X_k**, alors** ${}^t v \,.\, M$ **représente la distribution de probabilité pour l'état aléatoire** X_{k+1}
  - **Propriété 2 :** M^k**, matrice des probabilités pour k transitions consécutives dans la chaine de Markov (k=0, cas trivial: I)**
- **Caractérisation d'une chaine :** (M, v_0) **matrice + distribution initiale** v_0

Modélisation de séquence d'exécution par CMTD

- Idée rassembler l'automat fini et la chaine de Markov
  - Surcharge des états et des transitions par les deux étiquetages.
  - Adaptation de la notion d'état initial
- Exécution = chemin = séquence d'états dans le graphe de la chaîne de Markov
- Chaque préfix fini d'un chemin aura une probabilité = probabilité de franchir la séquence fini de transitions indiquée à partir de l'état initial (aléatoire, rappel).

Modélisation de systèmes par chaine de Markov discrète

- **Q : ensemble fini d'états préférablement = {1… n},**
- $(M_{i,j})_{1\leq i\leq n, 1\leq j\leq n}$ **: matrice de probabilité** $(M_{i,j})_{1\leq i\leq n, 1\leq j\leq n}$ **telle que** $M_{i,j}$ **est la probabilité de passer dans l'état j à partir de l'état i,**
- Δ **: partie de QxQ telle que si** $M_{i,j} \neq 0$**, alors (i,j) dans** Δ**,**
- v_0 **: distribution de probabilité pour l'état initial,**
- **AP : un ensemble de variable propositionnelle,**
- **LQ : une fonction qui associe à chaque état s de Q l'ensemble des variables vrai dans l'état s,**

(Au départ nous considèrerons qu'il n'y a pas de symboles attachés aux transitions)

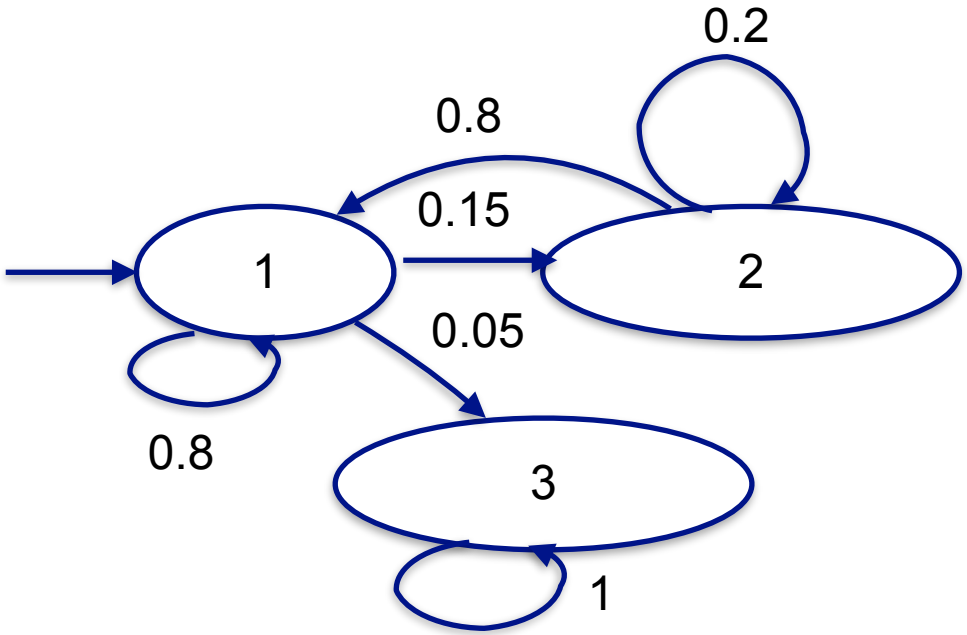
- A partir de maintenant une chaine de Markov discrete sera notée CM et correspondra à cette définition, et les notations ci-dessus seront utilisées pour représenter les attributs correspondant de la chaine : CM.v0 (ou v0 lorsque il n'y a pas d'ambiguïté possible)

Un petit exemple lié à la fiabilité

Variables booléens (Ok, Failed1, Failed2)

Etats et probabilités des transitions

D	LQ	(j,P(Succ=j)
1	{OK}	(1,0.8) (2,0.15) (3,0.5)
2	{Failed1}	(1,0.8) (2,0.2)
3	{Failed2}	(3,1)



V0=(1,0,0)

$$\begin{pmatrix} 0.8 & 0.15 & 0.05 \\ 0.8 & 0.2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

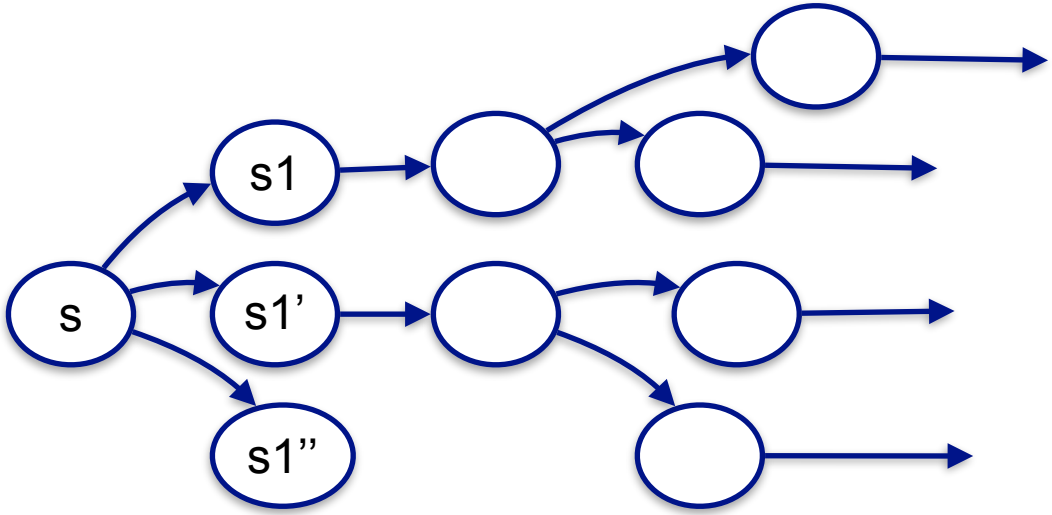
Matrice de transition

Chemins d'exécution (formalisation)

- **Chaine CM=(Q,M, Δ,v0, AP, Lap, L$\sum$),**
  Q états, I états initiaux, Δ transitions, Lp étiquetage de probabilités, Lap étiquetage de formules d'état, et L$\sum$ étiquetage d'action des transitions
- **Chemin dans CM depuis s = chemin w dans T depuis s = une séquence d'états de Q (i.e. {1,…,n}) finie ou non :**
  $w = s\,.\,s1.s2.\cdots\,.\,sk\cdots$
- **On notera $\{1,..,n\}^{\omega} = Q^{\omega}$ l'ensemble des chemins infinis, et path(w,s,CM) est vrai si w est un chemin partant de s dans le système à transition sous-jacent à CM.**
- $Path^{\omega}_{CM}(s) = \{w \,|\, w \in \{1,..,n\}^{\omega}, path(w,s,CM)\}$ **est l'ensemble des exécution de CM à partir de s**
- $Path^{*}_{M}(s) = \{p \,|\, p \in \{1,..,n\}^{*}, path(p,s,M)\}$ **ensemble des chemins finis.**

Chemins d'exécution, un espace probabiliste (formalisation)

- **Cylindre du prefix pref,** $pref = s\,.\,s1.s2.\cdots.\,sk,$
  $Cyl_{CM}(pref) = \{pref\,.\,p \,|\, p \in Q^{\omega}\}$
- **On état la définition des cylindres au cas ou pref est infini :**
  $Cyl_{M}(pref) = \{pref\}$
- **Représentation de** $Path_M(s)$ **sous forme d'arbre (prefix communs fusionnés)**



- **Mesurer de probabilité sur un ensemble de chemin = somme dénombrable de la probabilité d'un Cylindre**

Mesurer un cylindre : cas pref fini

- **Rappel $X_{i,i\geq 0}$, désigne la séquence d'états aléatoire de CM.**
- **Mesurer la probabilité d'un cylindre =**
  - $P(X_{i,i\geq 0} \in Cyl_{CM}(pref))$
  - **Equivalent à $P(A \wedge B)$, A et B respectivement $X_{i,0\leq i<|pref|} = pref$, et $X_{j\geq|perf|} \in Q^{\omega}$**
  - $P(A \wedge B) = P(B \mid A) \,.\, P(A)$, **or** $\forall j, P((X_{i,i\geq j}) \in Q^{\omega}) = 1$ **(car X est forcément dans Q) donc** $P(B \mid A) \,.\, P(A) = P(A)$
- **Mesure de** $X_{i,0\leq i<|pref|} = pref$: $pref = s \,.\, s_1 \,.\, \cdots \,.\, s_k$,
  $P(Cyl_M(pref) = P(X_0 = s \wedge X_1 = s1 \wedge \cdots \wedge X_k = s_k)$ **=**
  $P(X_0 = s \wedge X_1 = s_1 \wedge \cdots \wedge X_{k-1} = s_{k-1}) \,.\, P(X_k = s_k \mid X_{k-1} = s_{k-1})$ **=**
  $P(X_0 = s) \,.\, M_{s,s1} \,.\, M_{s1,s2} \,.\, \cdots \,.\, M_{s_{k-1},s_k}$
- **Nous avons une mesure de probabilité sur $(Path^{\omega}_M(s), \Sigma_{Cyl()})$, [KSK76] pour les détails :)… cas pref infini plus dur, ce qu'il faut retenir = construction correct**

Hum et en pratique …

- **Que vaut** $P(Cyl_{CM}(ok\,.\,ok\,.\,ok))$**?**

- **Que vaut** $P(Cyl_{CM}(ok\,.\,failed1.ok\,.\,failed1.failed2))$**?**

- **Existe-t-il un pref infini dans notre exemple tel que** $P(Cyl_{CM}(perf)) = 0$**?** $P(Cyl_{CM}(perf)) \neq 0$ **?**

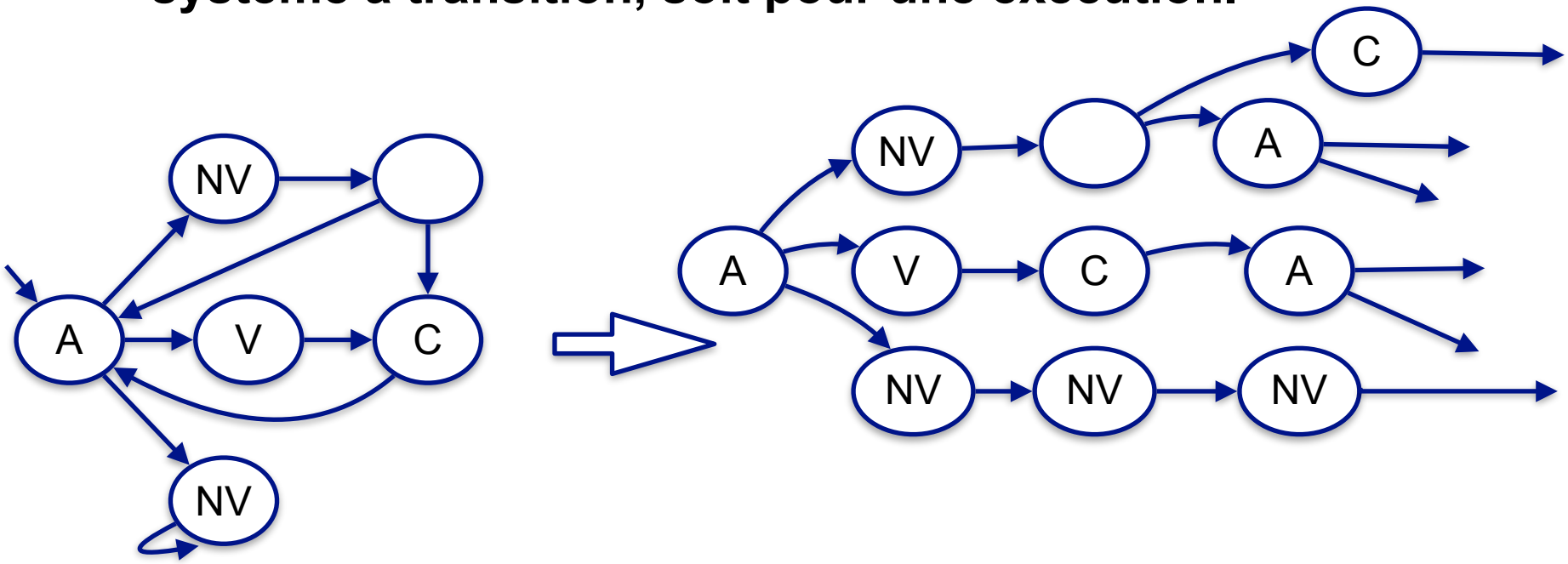
Problèmes de vérification vs quantification

Requis si SE301b : https://se206.wp.imt.fr/files/2020/01/se206-2020-hardware.pdf

Tr 16 and follow up …

Comprendre les formules logiques

- 1 formule = 1 contraintes sur l'exécution (identifie les exécutions rendant la formule vrai)
- La formule est « évaluée » soit pour un état d'un système à transition, soit pour une exécution.



Système à transition sous-jacent

- **Soit une chaine de Markov CM sont automate sous jacent est : (CM.Q, CM.Δ, $\{i \mid v_0[j] > 0\}$, CM.AP, CM.LQ)**
  - **Etats : CM.Q: {1,…,n}**
  - **Transitions : CM.Δ, : {(i,j,$M_{i,j}$) / $M_{i,j}$>0}**
  - **Etats initiaux : $\{i \mid v_0[j] > 0\}$, CM.$\sum$, CM.AP, CM.LQ, CM.L$\sum$)**
  - **Variables propositionnelles et étiquetage des états = idem CM**
- **Remarques:**
  - **Le système à transition ne contient pas d'état sans transition sortante = aucune impasse.**
  - **La somme de l'ensemble des valeurs sur les transitions sortantes d'un état vaut 1.**
  - **Une seule transition par couple d'état.**

- **On peut verifier des propriétés CTL sur le système à transition sous jacent à une chaine de Markov.**

De CTL à PTCL

- **CTL => définit un sous ensemble du cylindre démarrant à l'état initial ….**
- **Question légitime : quelle serait la probabilité à partir d'un état de ne franchir que les transitions qui permettent de satisfaire une formule CTL donnée (≠ probabilité que la formule soit vrai : i.e. soit 0 ou 1).**
- **Syntax**
  - $\Phi :: true \,|\, false \,|\, a, a \in AP \,|\, \Phi \wedge \Phi \,|\, \neg\Phi \,|\, P < cstr > \Psi$
  - $< cstr > :: < c \,|\, > c \,|\, \leq c \,|\, \geq c, c \in \mathbb{N}$
  - $\Psi :: X\Phi \,|\, \Phi \cup \Phi \; (opt\,. \,|\, \Phi \cup^{\leq c} \Phi)$
- **Une seule quantification sur les chemins P…**

Que veut dire l'opérateur P : intuition

- **Notons** $[[\Phi]]_{CM}(s) = \{w \in Path(CM, s) \mid w \vDash \Phi\}$
  - **Cet ensemble ne contient que les chemins pour lesquels** Φ **est satisfaite pour** w
  - **Cet ensemble peut se définir comme une somme dénombrable de cylindre.**
- **Pour éviter toute ambiguïté la probabilité sera noté Pr désormais :**

  Théoriquement $Pr(X_{i,i\geq 0} \in [[\Phi]]_{CM}(s)))$ **est définie et caractérise la probabilité qu'une exécution de notre chaine satisfasse la propriété.**
- $P_{Cstr}(\Phi)$ **contraint la valeur de** $Pr(X_{i,i\geq 0} \in [[\Phi]]_{CM}(s)))$

Sémantique de PCTL

- **Sémantique des formules sur les états :**
  - **État s pour chaine induite de (M, s) avec Lap étiquetage de formules.**
  - **Pour tout s,** $s \models true$ **est vrai et** $s \models false$ **est faux**
  - $s \models a$**, si a est dans LQ(s).**
  - $s \models \Phi 1 \wedge \Phi 2$, $s \models \Phi 1$ **et** $s \models \Phi 2$
  - $s \models \neg\Phi$, $s \models \Phi$ **est faux.**
- **Sémantique sur les chemins** $w = s_0 . s_1 . \cdots . s_k . \cdots$**, tq on note** $w_i = s_i . s_{i+1} . \cdots . s_k . \cdots$ **(e.g.** $\forall i > 0, w = s_0 \ldots s_{i-1} . w_i$ **et** $w = w_0$**)**
  - $w \models X\Phi$**, si** $w_1 \models \Phi$
  - $w \models \Phi 1 U^{\leq k} \Phi 2$**, si il existe 0≤j ≤k t.q.** $w_j \models \Phi 2$**, et pour tout i <j,** $w_i \models \Phi 1$
  - $w \models \Phi 1 U \Phi 2$**, si il existe j ≥0 t.q.** $w_j \models \Phi 2$**, et pour tout i <j,** $w_i \models \Phi 1$

Sémantique opérateur P

- **Satisfaction de P :**
  $s \models P_{\sim C}(\Phi)$ **s.s.i.** $Pr(X_{i,i\geq 0} \in [[\Phi]]_{CM}(s))) \sim C$

- **Par exemple** $P_{>5}(ok \cup failed2)$ **est vrai pour l'état ok car contient au moins** $Cyl_{CM}(ok\,.\,ok\,.\,failed2^{\omega})$ **dont la mesure est 0.8 (rappel état initial = ok avec probabilité 1)**

- **Remarque contre intuitive :**
  $s \models P_{=1}(\Phi)$ **n'implique pas que** Φ **est vraie à partir de s pour le système à transition sous-jacent**
  pourquoi ? Un exemple sur notre chaine à 3 états ?

Approche de vérification reposant sur les récompenses

- **Etiqueter les transitions et variables par un scalaire (identifiant un coût ou un gain) appelé recompense.**
- **Contraindre pour la valeur moyenne de la récompense de l'état occupé à la fin d'un chemin fini**

$$ExpectedReward(X_k) = \sum_{s \in Q} P(X_k = s) \,.\, Reward(s))$$

- **Contraindre pour un chemin la valeur moyenne accumulée**

$$CumulativeReward(w) = P(w) \,.\, \sum_{\sigma \in w} Reward(\sigma)$$

- **Exemple : Récompense de 1 pour ok, 0 pour les autres états**
  - **Récompense moyenne pour date k = disponibilité à la date k**
  - **Récompense cumulative = disponibilité moyenne sur un intervalle**
  - **Possibilité de calcul [min max] ou probabilité d'un intervalle**

Pour aller plus loin sur les récompenses

Sur le site de PRISM, un cours dédié au sujet :

https://www.prismmodelchecker.org/lectures/pmc/07-costs%20rewards.pdf

Produit de chaine de Markov

non déterminisme vs probabilité

Rappel sur produit libre de deux automates

- **2 automates** $A_1 = (Q_1, \Delta_1, I_1, AP_1, LQ_1, \Sigma_1, L\Sigma_1)$, $A_2 = (Q_2, \Delta_2, I_2, AP_2, LQ_2, \Sigma_2, L\Sigma_2)$
- **on appellera respectivement** $\Delta_{1,2}(i)$ **l'ensemble des n-uplets :** $((q_1, q_2), (q'_1, q'_2))$ **tels que** $(q_i, q'_i) \in \Delta_i$ **et** $q_{\bar{i}} = q'_{\bar{i}}$
- **Produit libre A1xA2 : un automate tel que son espace d'état est Q1xQ2, ses transitions sont dans** $\Delta_{1,2} = \Delta_{1,2}(1) \cup \Delta_{1,2}(2)$,
- **Les étiquettes sont fixées comme suit chaque élément de** $\Delta_{1,2}(i)$ **est étiqueté par le résultat de** $L\Sigma_i$ **appliqué à** (q_i, q'_i)

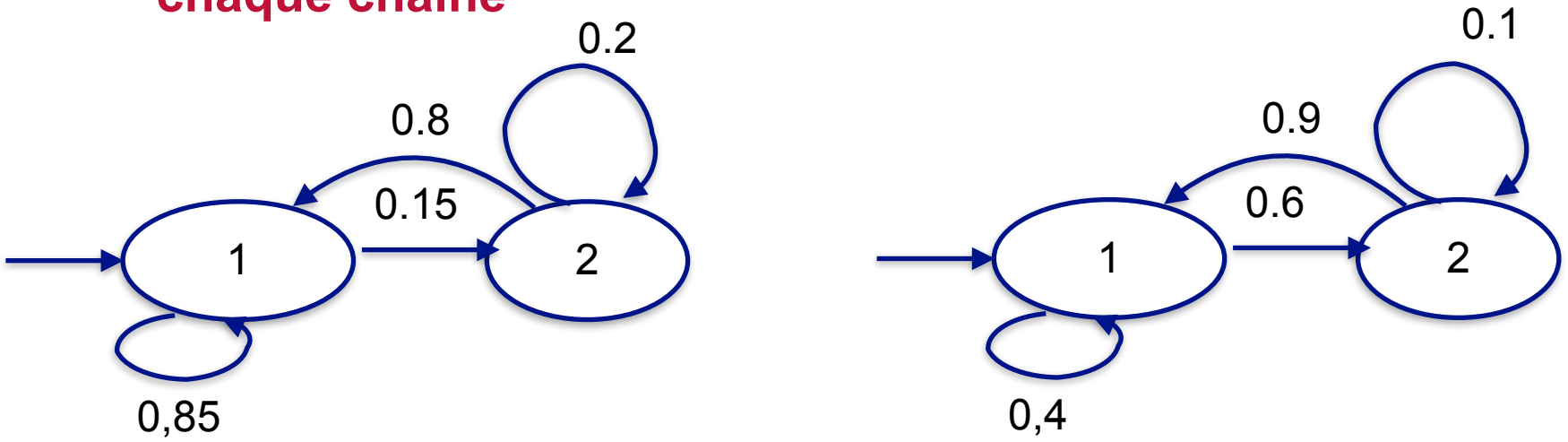
- **Idée : peut on faire le produit de deux chaines de Markov et adapter la règle ci-dessus pour définir les probabilité associées aux transitions …… pas si simple**

Produit synchronisé de deux automates

- **2 automate** $A_1 = (Q_1, \Delta_1, I_1, AP_1, LQ_1, \Sigma_1, L\Sigma_1)$, $A_2 = (Q_2, \Delta_2, I_2, AP_2, LQ_2, \Sigma_2, L\Sigma_2)$
- **Nous noterons** $\Delta_{1/2}$ **le plus grand sous ensemble de des transitions de** A_1 **telle que pour** $L\Sigma_1(\Delta_{1/2}) \cap \Sigma_2 = \varnothing$ **(transition dites locales,** $\Delta_{2/1}$ **est l'équivalent pour** A_2**)**
- $Self_i = \{(q, q) \mid \in Q_i\}$
- **Automate produit**
  - **Etats** $Q_1 \times Q_2$
  - **Transitions** $\Delta_A \cup \Delta_B \cup \Delta_C$
  - $\Delta_A = \{((q_1, q_2), (q_1', q_2'), a) \mid (q_1, q_1', a) \in \Delta_{1/2} \wedge (q_2 = q_2')\}$
    $\Delta_B = \{((q_1, q_2), (q_1', q_2'), a) \mid (q_2, q_2', a) \in \Delta_{2/1} \wedge (q_1 = q_1')\}$
    $\Delta_C = \{((q_1, q_2), (q_1', q_2'), a) \mid ((q_1, q_1', a) \in \Delta_1 \wedge (q_2, q_2', a) \in \Delta_2)$

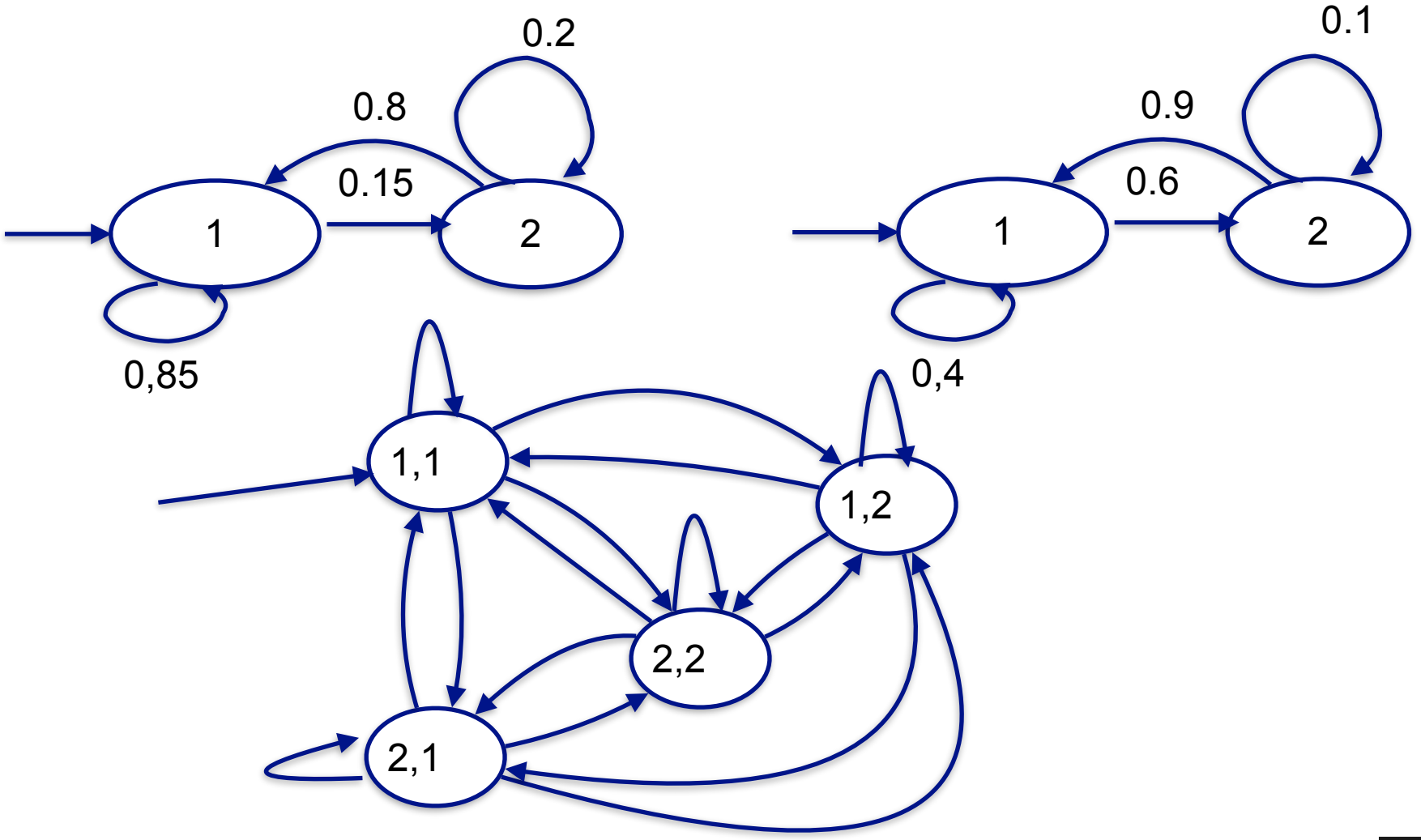
Intuition du produit de chaines de Markov synchronisé

- Idée : on ne va pas utiliser les probabilité comme étiquette, on va considérer que chaque transition est étiqueté τ.
- Le produit de 2 chaines = franchir une transition sur chaque chaine



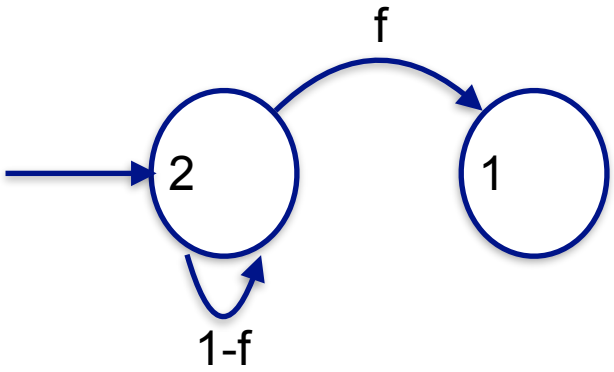
- Quelles probabilité ? Pour $((i_1, i_2), (j_1, j_2), \tau)$, $M_{i_1,j_1} . M_{i_2,j_2}$

Résultat du produit sur l'exemple



Exemple d'usage pragmatique

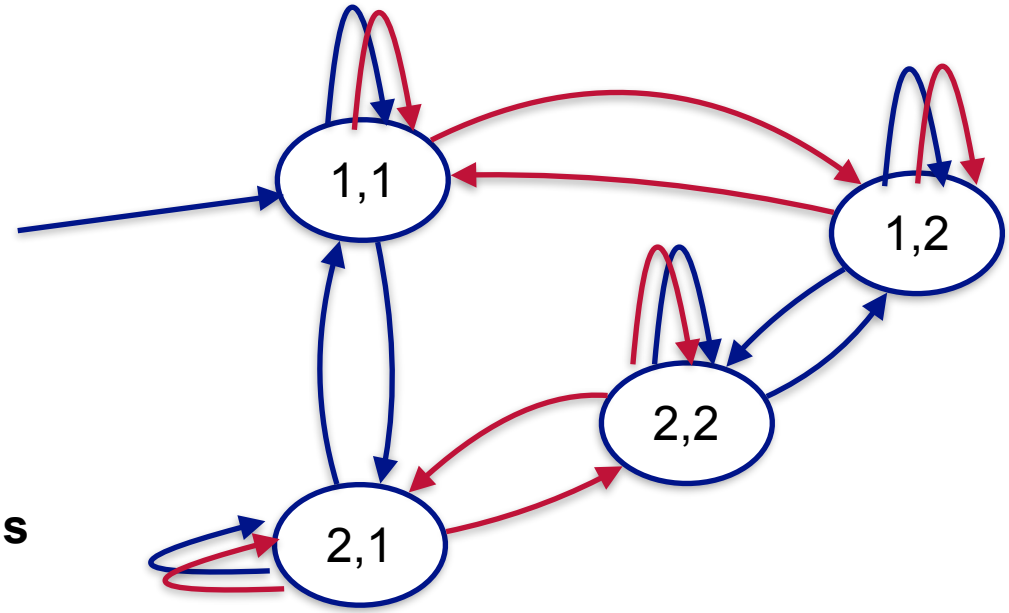
- **Modélisation d'une réplique avec probabilité de défaillance f distribution initial (0,1)**



- **Modèle réplication active Produit de 3 copies de cette chaine => état fiables (2,2,1) (2,2,2) (1,2,2) …**

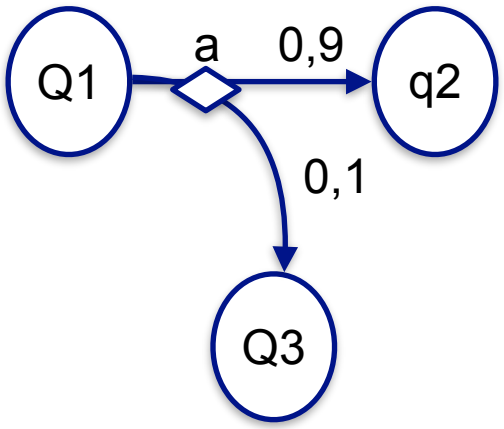
Modèle alternatif

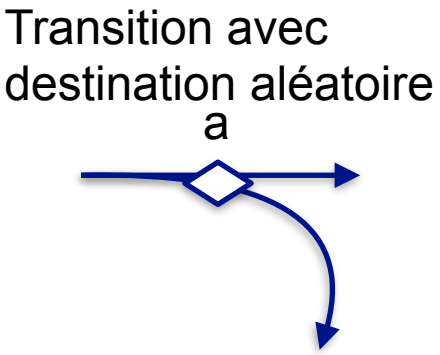
- Et si le produit synchronisé sur une partie des transitions => 1 transition ≠ 1 transition sur chaque chaine
- L'ensemble des chemins ~ chemins du produit synchronisé des systèmes à transition sous jacents.
- Pb : le choix de quel « type » de transition doit être franchie = pas défini
- probabilité(1,1) une infinité de valeurs possibles si chemin
  - bleu*
  - rouge*
  - {blue, rouge}*
- Mélange non-déterminisme + probabilités => motivation des Processus Markoviens



Processus de décision markovien

- **PDM= (Q, Init, Step,L) / Q =états, Init=état initial, L étiquetage de Q par ensemble de variable propositionnelles, et Step = « transitions »**
- **Intuition : chaque action => l'état de destination est une variable aléatoire à valeur dans l'ensemble des états**





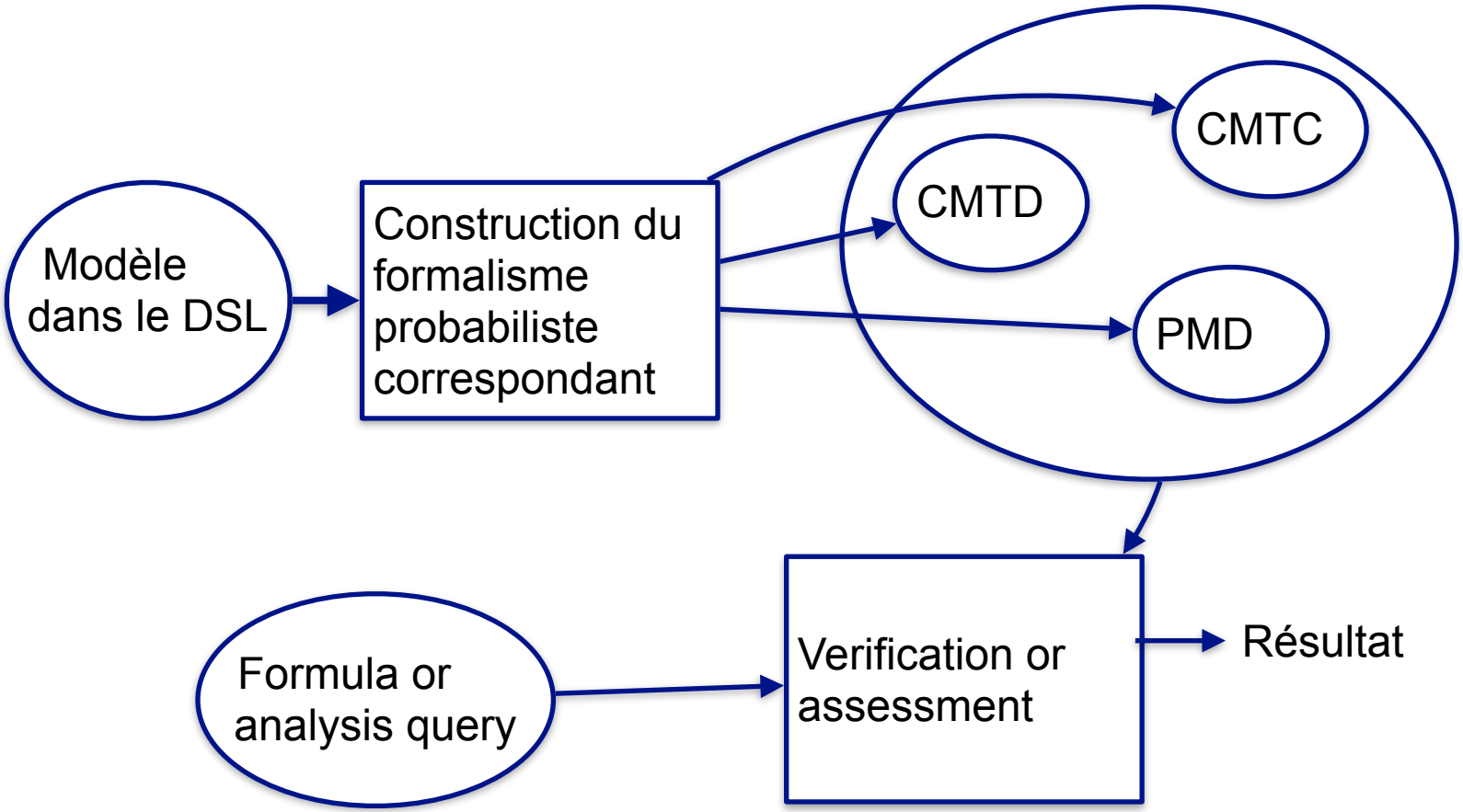
- **Chaque « transition » =** $(a, Dist)\ s.t.\ a \in \Sigma$ **et Dist est une distribution de probabilité sur Q**
- **Il faut connaitre la séquence de « transitions » franchies (donc de symboles pour déterminer la probabilité d'un état !**

Petit tour rapide de PRISM

PRISM : https://www.prismmodelchecker.org/

- Outil pour la vérification de PCTL, PTCL*, calcul de moyennes, variances et intervalles min-max pour
  - DTMC
  - CTMC
  - PMD
  - ….
- Un langage dédié pour décrire les systèmes (textuel, modulaire, concis …)
- Un moteur de vérification qui supporte
  - Les analyses exactes (symboliques/ paramétriques)
  - Les analyses par simulation (Monte Carlo ou autre)

Logique de l'outil



DSL de spécification PRISM

- Idée : permettre de choisir le formalisme de manière non ambigu entre CMTD (mot clé : dtmc), ou chaine continue ou processus de décision Markovien.
- Syntax unifiée (unique ?) pour décrire les transitions du modèle
- Notion de module (module) (utile pour la composition parallèle)
- Notion de variable : définit l'espace d'état des processus stochastiques
- Notion de constante (const) : définit des paramètres constant lors de l'analyse mais modifiables entre 2 analyses

Les points clés de la syntaxe

dtmc → Type du modèle 1 fois par fichier

module permanentFail — Unité composition \\,

x : [0..1] init 1; — Espace d'état + état initial (e.g. une Distrib particulière)

[] x=1 -> 0.9: (x'=1) +0.1: (x'=0); — {transitions}

[] x=0 -> 1: (x'=0);

endmodule

Spécification de module supplémentaire

- 2 choix :
  - vous donnez une définition complète de module
  - Vous recopiez un module existant puis réalisez des substitutions sur les noms
    - De variables
    - D'événements de synchro
    - De constantes
    - De module …
- Comment marche la copie/substitution
  - Module new =old [subst1,… substn] endmodule
  - subst1 : oldname=newname

Copie / substitution en pratique

```
dtmc
module permanentFail1
x : [0..1] init 1;
[] x=1 -> 0.9: (x'=1) +0.1: (x'=0);
[] x=0 -> 1: (x'=0);
endmodule

module permanentFail2
y : [0..1] init 1;
[] y=1 -> 0.9: (y'=1) +0.1: (y'=0);
[] y=0 -> 1: (y'=0);
endmodule
```

```
dtmc
module permanentFail1
x : [0..1] init 1;
[] x=1 -> 0.9: (x'=1) +0.1: (x'=0);
[] x=0 -> 1: (x'=0);
endmodule

module permanentFail2
[x=y]
Endmodule
```

Questions ?

Références

- Inspiration pour ce cours pris dans les cours sur le site de PRISM : (et surtout celui ci) https://www.prismmodelchecker.org/lectures/pmc/04-prob%20logics.pdf
- [KSK76] : J. Kemeny, J. Snell, and A. Knapp. Denumerable Markov Chains. Springer-Verlag, 2nd edition, 1976.