Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Centre Universitaire de Mila
Laboratoire MISC, Université Constantine 2–Abdelhamid Mehri

Institut des Sciences et de la Technologie, Département: MI

Master 1 : STIC

Module: Ingénierie Des Logiciels (IDL)

Chapitre 04: Méthodes Formelles et Ses Applications

Dr MEGHZILI Said
Département MI
s.meghzili@univ-mila.dz

Sommaire

1. Introduction: bugs & erreurs des logiciels

Système critique: c'est un système dont lequel les erreurs peuvent conduire a une catastrophe humaine ou matérielle.

- **Exemple des *Systèmes critiques***

commandes de vol

arrêt d'urgence

contrôle de vitesse

automatisme intégral

1. *Introduction:* explosion d'Ariane 5 (ASE)

- **Exemple:** ***Agence spatiale européenne***

L'explosion d'Ariane 5 (après 37 s), le 4 juin 1996, qui a coûté un demi milliard de dollars

- **Cause : une faute logicielle d'une composante dont le fonctionnement n'était pas indispensable durant le vol.**

- **Il contenait une conversion d'un flottant sur 64 bits en un entier signé sur 16 bits**
- **Pour Ariane 5, la valeur du flottant dépassait la valeur maximale pouvant être convertie**

⇒ **Nécessité de « vérifier » certains logiciels/systèmes:** ***Méthodes Formelles:*** **Test, Démonstrateur de théorèmes, Model-checking**

Sommaire

2.Modèles Formels

- ***Classification des Modèles Formels:***

1) Algébriques

- CCS (Milner:79)
- CSP (Hoar:85)
- ACP (Bergstra:85)
- LOTOS (Bolognesi et al : 87)
- Pi-calculus (Milner:92)
- RT-LOTOS (Courtiat et al :93)
- ET-LOTOS (Leduc et al : 93)
- D-LOTOS (Saidouni et al : 2003)

2)Langages

- Estelle
- Maude (Logique de réécriture)
- Mobile Maude
- Real Time Maude

3)Réseaux de Petri

- Places/Transitions
- Prédicat
- Colorés
- Algébriques
- Bien formés
- Temporel
- Temporisé
- Temporisés stochastiques
- .....

2.Modèles Formels: Réseaux de Petri

- Les Rdp sont un des outils très utilisés pour la spécification et la vérification des systèmes
- Introduits initialement par : C.A PETRI (1962)
- Une représentation graphique facilement compréhensible par les utilisateurs.
- Une fondation mathématique solide permettant l'application des techniques de vérification sur les systèmes modélisés avant leur déploiement

Réseaux de Petri : 1. Définitions

- Un réseau de pétri R est un ***quadruplet*** {***P***, ***T***, ***Pre***, ***Post***}, avec :
  - ***P*** = {p1, p2,…, pn} un ensemble de places (les états du système)
  - ***T*** = {t1,…, tn}, un ensemble de transitions, avec P∩T = Ø
  - ***Pré***: $P_X T$ -> N est la fonction d'incidence avant
  - ***Post*** : $P_X T$ -> N est la fonction d'incidence arrière
- ***Pre(pi, tj)*** représente le poids de l'arc allant de **pi** vers **tj**
- ***Post(pi, tj)*** représente le poids de l'arc allant de **tj** vers **pi**
- Marquage initial

*Réseaux de Petri : 2.*Représentation des RDP

Deux représentations : ***Graphique & matricielle***

- Représentation ***Graphique:*** Un graphe biparti orienté et valué:

❖Places: cercles

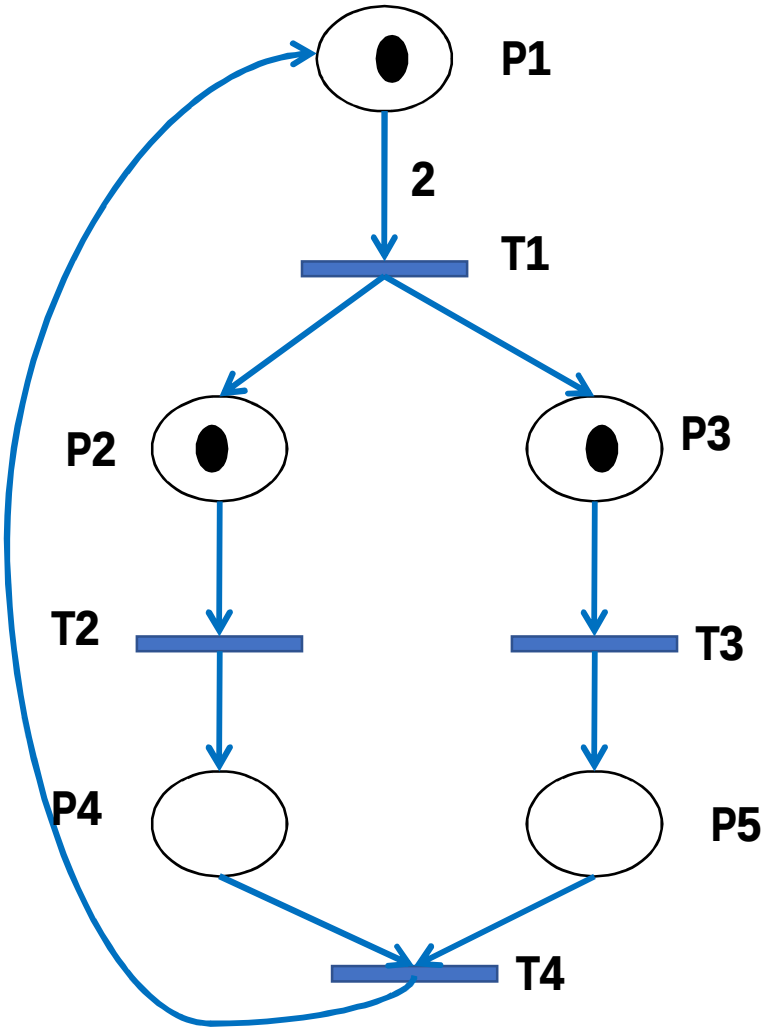
❖Transitions: bars or rectangles

❖Arcs: flèches étiquetées par des valeurs

❖Jetons: points noirs (black dots)

Réseaux de Petri :2. Représentation des RDP

Représentation *Graphique*



Représentation : *Matricielle*

La matrice **pré:**

	T1	T2	T3	T4
P1	2	0	0	0
P2	0	1	0	0
P3	0	0	1	0
P4	0	0	0	1
P5	0	0	0	1

La matrice **post:**

	T1	T2	T3	T4
P1	0	0	0	1
P2	1	0	0	0
P3	1	0	0	0
P4	0	1	0	0
P5	0	0	1	0

	M0
P1	1
P2	1
P3	1
P4	0
P5	0

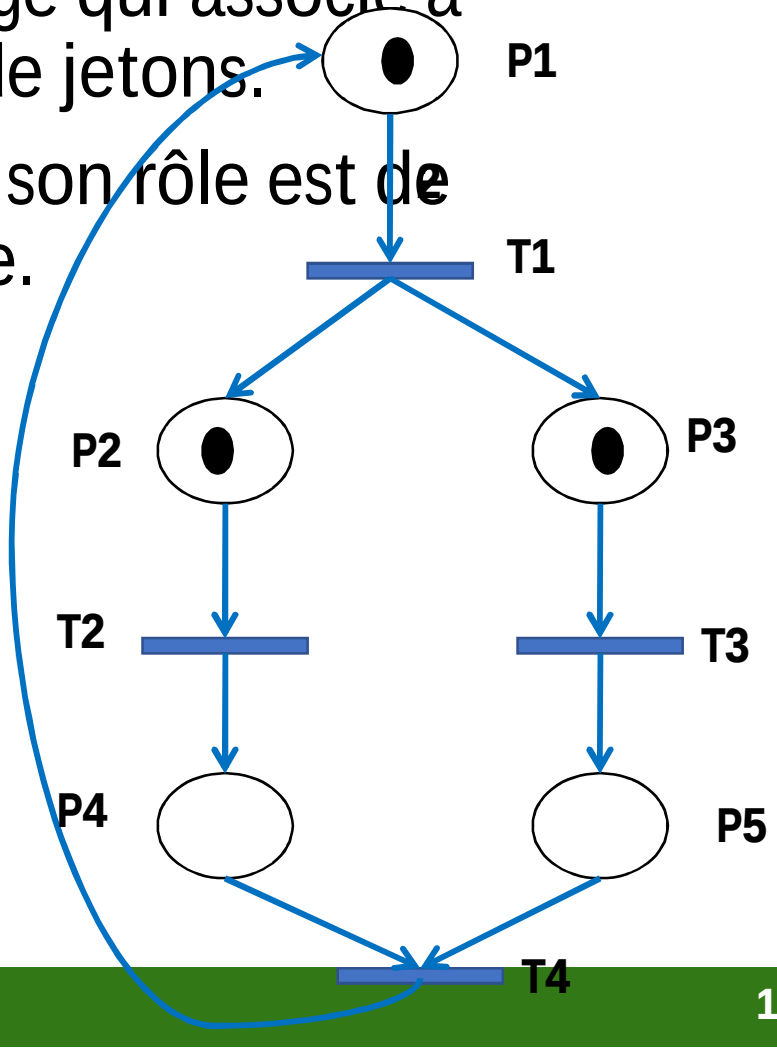
:Le marquage initial

Réseaux de Petri : 3. Marquage

- Un **réseau de pétri marqué** est définie par le couple {**R**, **M**}, où R est un réseau de pétri, et **M** : **P-> N** est une application appelé marquage qui associe à chaque place de **R** un nombre de jetons.
- Le marquage initial est noté $\mathbf{m_0}$ son rôle est de spécifier l'état initial du système.

- Exemple Rdp:

$\mathbf{m_0(1, 1, 1 , 0, 0)}$



Réseaux de Petri : 4. **Franchissement des Transitions**

- Pour une transition **t**, on dénote **•t** (resp. **t•**) l'ensemble de places d'entrée (resp. de sortie) de la transition **t**.

- Une transition est **franchissable** ("is enabled") si et seulement s'il y a assez de jetons dans chaque place en entrée

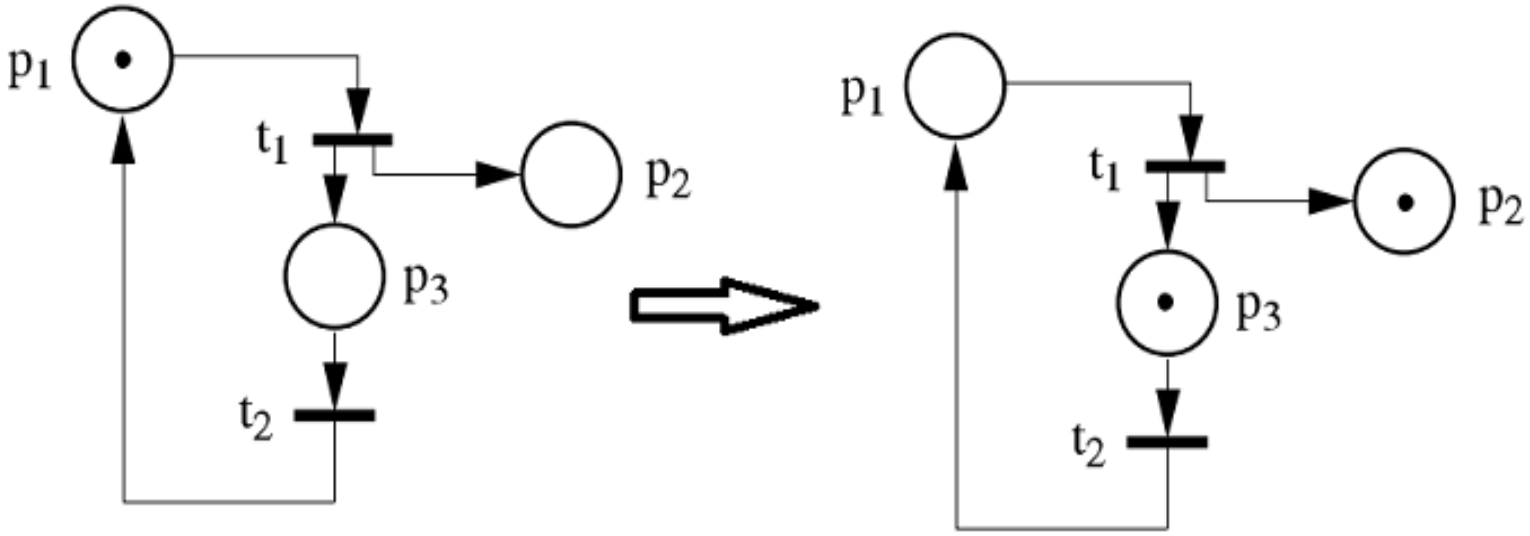
$$\forall p \in \bullet t : M(p) \geq pré(p,t)$$

- Notation **M** [**t** > : **t** est **franchissable** dans le marquage **M**

Réseaux de Petri : 4. **Franchissement des Transitions**

- Le **franchissement** d'une transition modifie le marquage en **consommant** des jetons dans les places d'entrée et en **produisant** des jetons dans les places de sortie.
- Si une transition **t** est sensibilisée pour un marquage **M, t** peut être franchie à partir de **M**, produisant un nouveau marquage **M'**, dénoté **M –(t)-> M',** où

$$\forall p \in P,\ \mathbf{M'(p) = M(p) - pré(p, t) + post(p, t)}$$



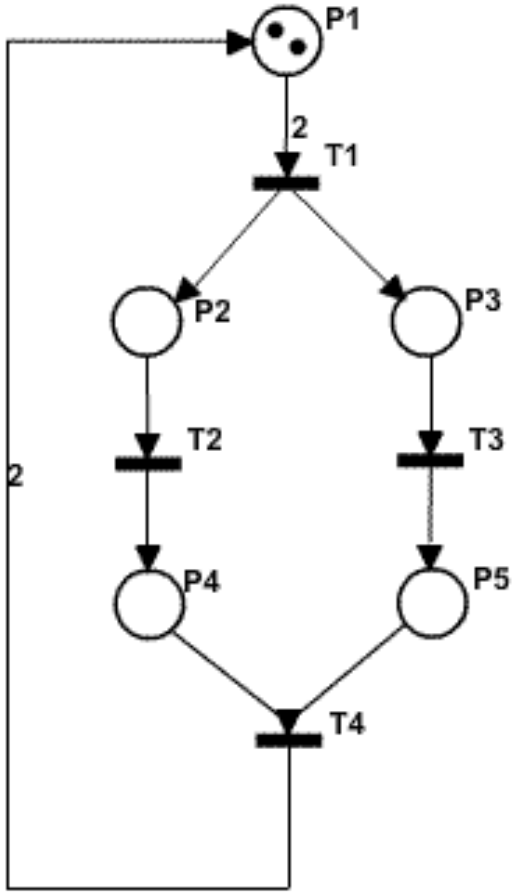
RdP: 5.Graphe de marquage (graphe d'accessibilité)

- **A(R,M0)** est l'ensemble des marquages accessibles

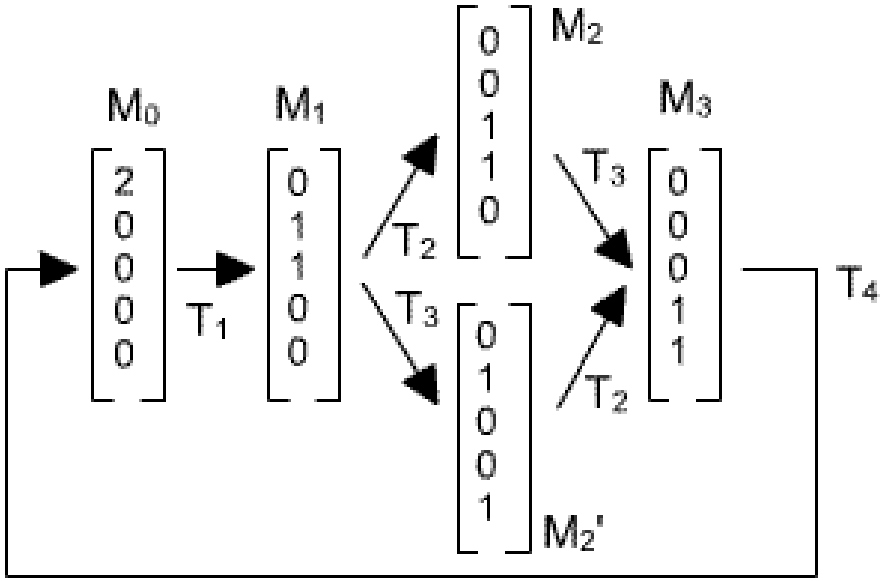
- Le graphe d'accessibilité **GA(R,M0)** de ce réseau à partir du marquage initial **M0** est un graphe dirigé et étiqueté dont les sommets sont les éléments de **A(R,M0).**

- Un arc relie deux marquages **Mi** et **Mj** si et ssi il existe une transition **t** de **R** tel que **Mi –(t)-> Mj**.

RdP: Exemple de Graphe de marquage



Graphe d'accessibilité



RdP: 6.Propriétés des Réseaux de Petri

- **Accessibilité (reachability)** :

un marquage **m** est accessible (à partir de $\mathbf{m_0}$) s'il existe une séquence de transitions **t1**,**t2**,……,**tn** qui conduit de **m0** à **m**.

C à d :$\mathbf{m_0\ [t_1> m_1\ [t_2> m_2 \ldots\ldots m_n\ [t_n> m}$

- **Deadlock-free** :

si chaque marquage accessible permet au moins une transition.

RdP: 6.Propriétés des Réseaux de Petri

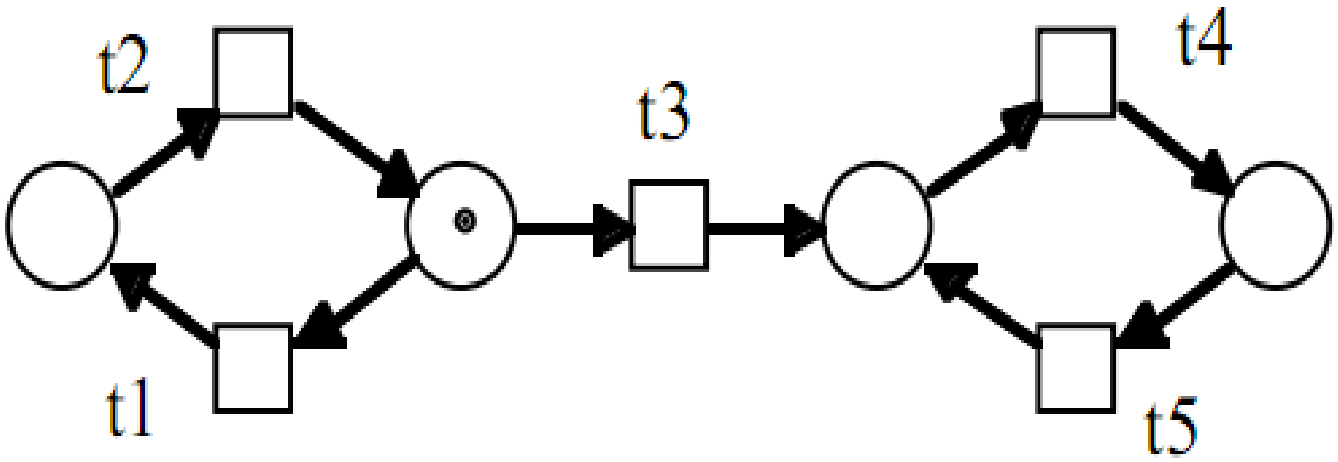
- Une transition **t** d'un réseau de Petri (**R,M0)** est **quasi-vivante :**

  ssi ∃ **M** accessible à partir de **M0**, tel que **M [t >**

- Une transition **t** d'un réseau de Petri **(R,M0)** est **vivante:** ssi pour chaque marquage accessible **M'**, il existe un marquage **M''**accessible à partir de **M'** où **t** est sensibilisée.

- Un réseau de Petri **(R,M0)** est **vivant** (resp. **quasi-vivant**) si et ssi toutes ses transitions sont **vivantes** (resp. **quasi-vivantes**)

RdP: 6.Propriétés des Réseaux



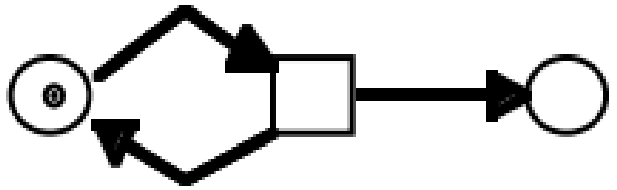
un RdP qui est **deadlock-free** mais **non vivant**

RdP: 6.Propriétés des Réseaux

- **Borné (bounded)** :

  (débordement) un RdP est dit **K-borné** (k-bounded) si le nombre de jetons dans chaque place **p** est toujours inferieur ou égale à **k** (k appartient à N^+) pour chaque marquage accessible à partir de **m0**.

Exemple un RdP non borné (unbounded



- **Safe (1-safe)** :

un Rdp est **safe** s'il est **1-borné**.

- **Réversible (reversible)** :

si le marquage initial **m0** est accessible depuis n'importe quel autre marquage.

Sommaire

3. Intégration des Méthodes Formelles avec l'IDM

- **Recouvre plusieurs propriétés: la consistance, la sûreté, l'atteignabilité , la vivacité, …**

- **Différentes techniques sont utilisées:**
  - La simulation
  - Les tests
  - Les méthodes formelles

- **Toutes ces techniques s'appuient sur l'utilisation de représentations formelles du système (à base de logique, d'automates, de Réseaux de Petri, ...)**

- **Dans le cas de systèmes complexes, ces tâches deviennent impossibles à réaliser sur un modèle préexistant**

  ➢ La tendance actuelle vise à vérifier ces propriétés par construction de plusieurs modèles adaptés

3. Intégration des Méthodes Formelles avec l'IDM (Suite)

- Les méthodes formelles (MFs) sont des techniques basées sur les mathématiques permettant à la fois de ***modéliser*** le système et de ***vérifier*** les propriétés attendues

- Les méthodes formelles reposent sur l'utilisation de langages formels

- Un langage formel est un langage doté d'une sémantique mathématique rigoureuse

- Principaux obstacles d'utilisation des MFs dans les activités de développement des systèmes sont liés à :

  - La difficulté réelle de manipuler les concepts théoriques et les méthodes d'analyse associées

  - La difficulté pour les développeurs d'exprimer les propriétés du système d'une façon aisée

3. Intégration des Méthodes Formelles avec l'IDM (Suite)

- **La vérification des modèles comptent aujourd'hui parmi les enjeux les plus importants de l'IDM**

- **L'IDM joue un rôle essentiel dans l'introduction des MFs dans les activités de développement des systèmes**

- **Celle-ci repose sur :**

➔La définition de langages formels par le biais de la ***méta-modélisation***

➔L'utilisation de ***transformations de modèles*** pour générer des modèles décrits dans ces langages formels

3. Intégration des Méthodes Formelles avec l'IDM (Suite)

Combinaison de l'IDM avec les MFs

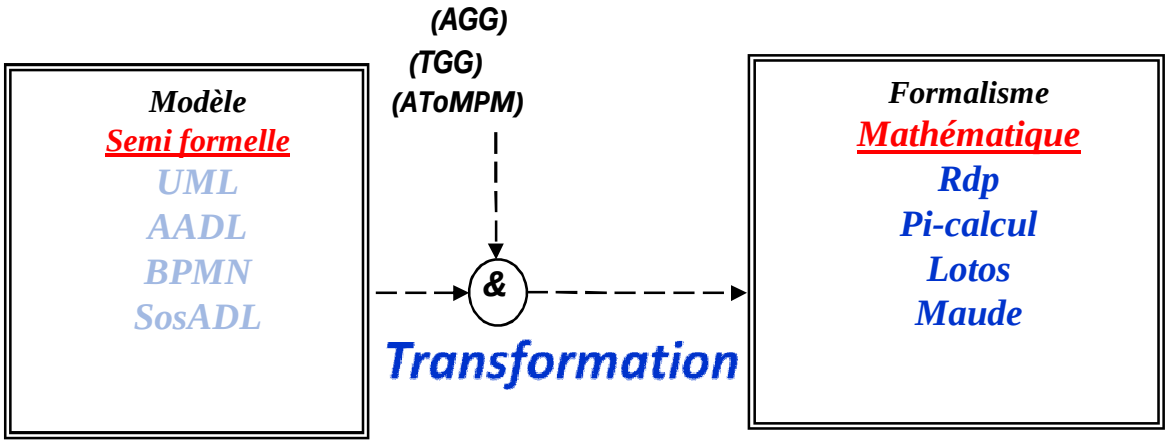
- Chacune des deux approches a des points forts et des points faibles
- Les deux approches peuvent être combinées dans le sens où les inconvénients de l'un peuvent être surmontés grâce aux apports de l'autre

	Avantages	*Inconvénients*
IDM	√ Notations conviviales et souvent graphiques √ Génération automatique d'outils de développement √ Transformations automatiques de modèles	✗ Manque de sémantiques formelles ✗ Analyse de modèles impossible
MFs	√ Fondements mathématiques rigoureux √ Analyse formelle de modèles	✗ Notations complexes ✗ Absence d'outils de développement ✗ Manque d'intégration

Sommaire

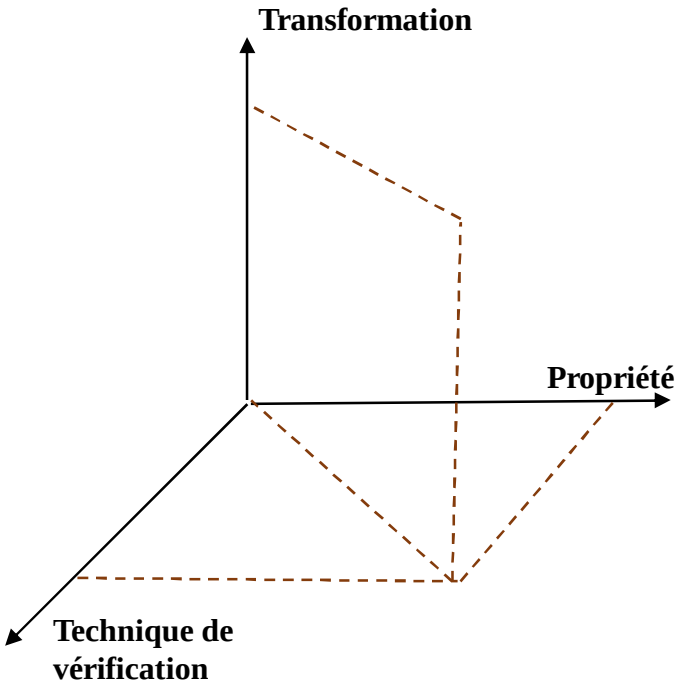
4. Vérification d'une Transformation: Problématique

- Ces transformations et leurs outils : manque de vérification



- **Propriétés à Vérifier:**
  - **Terminaison de la transformation**
  - **Préservation de la sémantique du modèle source**
  - **Confluence**
  - **Invariants**
  - **Complétude**
  - **Absence d'interblocage & boucle infinie ...**

Approche de trois dimensions [Amrani et al. 2015]



Transformation de Modèles : outils

- Le choix de l'outil de transformation est très important.

- Outil dédie seulement a la transformation:

  -Atom3

  -AToMPM

  -TGG

  -AGG

- Outil dédie à la transformation et à la validation:

  -Moment-2

  -Groove

  -Viatra

Propriété: en générale

- Dans le domaine de vérification formelle:
  - _Propriété sûreté (**safety**)
  - _Propriété de vivacité **(liveness**)

4. Vérification d'une Transformation:

Propriété: fonctionnelle ou comportementale

- Concerne le comportement de la transformation elle-même, il existe 2 types:

1)Terminaison:

- Assurer que la transformation termine après un nombre des étapes finie.
- Assurer l'existence d'un modèle cible.

2)Confluence:

- l'ordre d'application des règles de transformation n'est pas important.
- Assurer que la transformation toujours produire le même résultat.

Problème :Le système de réécriture de graph est **indécidable**.

4.Vérification d'une Transformation

Propriété : liée à la transformation

- Concerne les modelés (source et cible) et la préservation des propriétés par la transformation, il existe trois types:

1) Conformité:

- Un modèle est valide s'il est conforme à son méta modèle.
- Est Vérifié automatiquement dans des Framework de modélisation.

2) Relation syntaxique :

- Certains éléments(source)seront transformé en d'autres éléments (cible)

3) Relation sémantique :

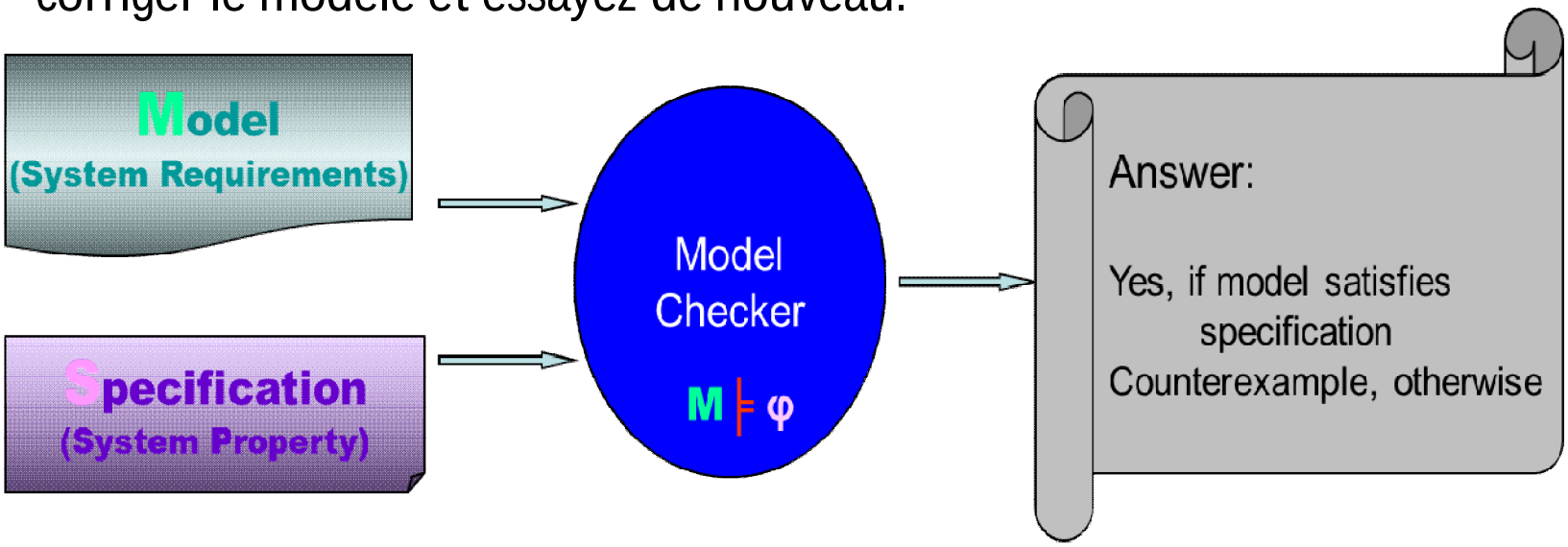
- Liée la signification des deux modèles (source et cible)
- Par exemple construire les LTS de la sémantique des deux modèles.

Techniques de Vérification Formelle

- **Model checking**

Pour augmenter notre confiance dans l'exactitude du modèle:

- Vérification: Le modèle satisfait des propriétés importantes du système.
- Débogage: étude des contre-exemples, localiser la source de l'erreur, corriger le modèle et essayez de nouveau.

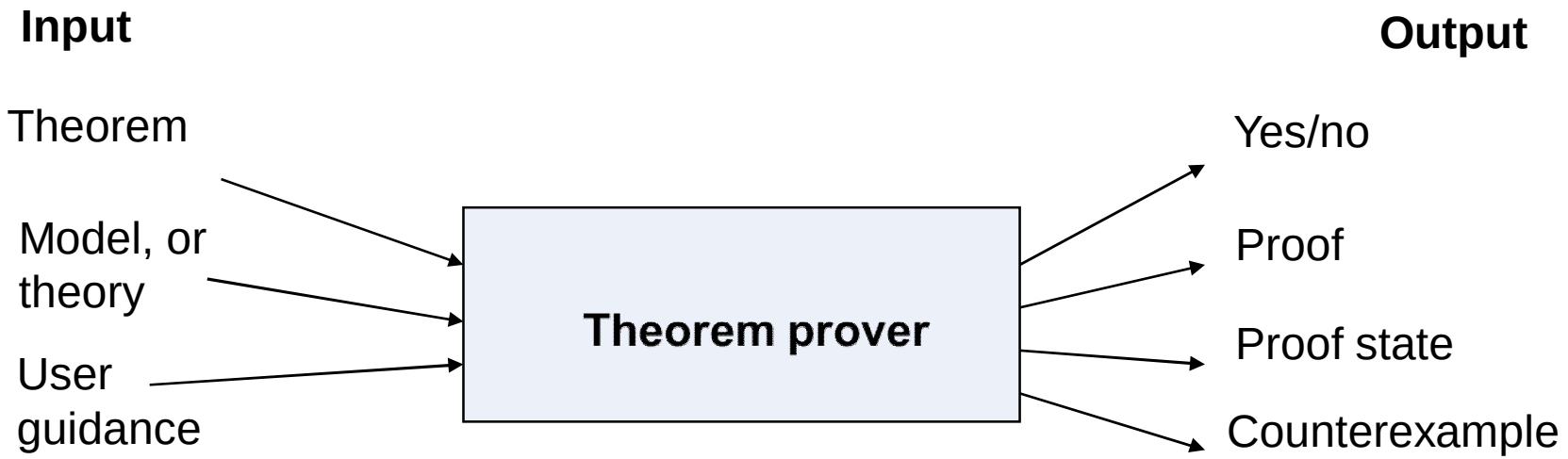


Inconvénient: Problème d'explosion combinatoire de graphe d 'état

4. Vérification d'une Transformation:

Techniques de Vérification Formelle

- **Theorem proving**

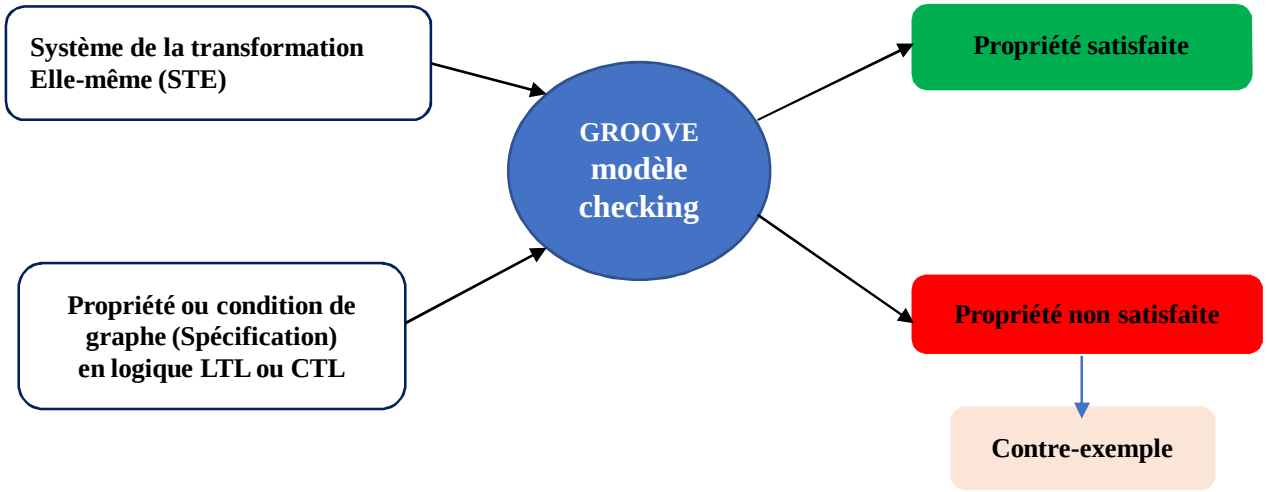


- Le Système et les propriétés sont des formules logique.
- Espace infini.
- Exige Haut degré d'expertise humaine.
- Nombre d'outils du domaine public:

  Coq, Isabelle, PVS, HOL

Exemple :Approche GROOVE modèle-checking

- Cette approche permet de vérifier les transformations de graphes en utilisant l'outil GROOVE et son modèle checking. GROOVE considère la transformation comme un Système de Transition étiqueté (**STE**) contenant un ensemble des états et des transitions.



- Chaque transition connecte deux états et étiqueté avec le nom de la règle appliqué.
- Chaque état contient les propriétés ou les conditions de graphes qui sont valides.
- Ces conditions sont exprimées avec la logique LTL ou CTL.
- Modèle checking répond avec oui si le système de la transformation (STE) satisfait la propriété (la spécification) sinon il répond avec non et génère *un contre-exemple.*

Exemple: Approche basée sur démonstrateur de théorèmes

- Cette approche considère la transformation comme une fonction mathématique qui compose de plusieurs fonction récursives. Elle consiste en deux parties:

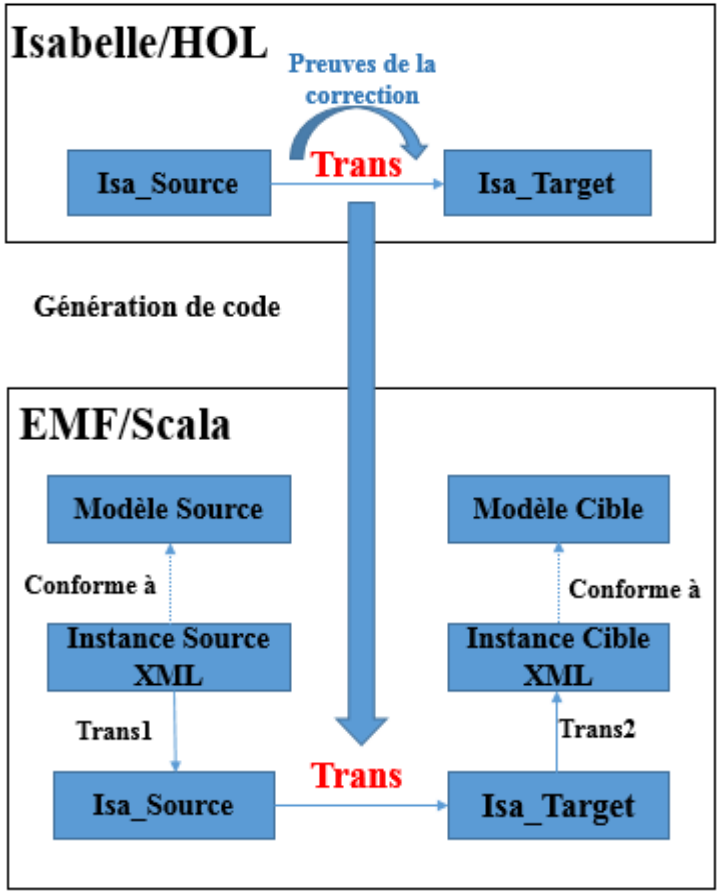
1. En Isabelle/HOL:

- Description de modèle source
- Description de modèle cible
- Description de la transformation
- Spécification des propriétés attendues
- Preuve de correction des propriétés attendues

2. En EMF/Scala:

- Définition des méta-modèles en utilisant EMF
- Génération de code Scala de la fonction Trans
- Définition des fonctions Trans1 et Trans2
- Composition des fonctions :

Trans2 (Trans (Trans1(modèle source)))-->modèle cible



4. Vérification d'une Transformation
Étude de cas : Transformation des modèles BPMN vers les réseaux de Petri en utilisant Isabelle/HOL

1. Spécification de la transformation

- ✓ Description des modèles BPMN avec ***Isa_BMPN***
- ✓ Description des modèles réseaux de Petri avec **Isa_CPN**
- ✓ Description de la transformation avec la fonction **Trans**

2. Vérification de la transformation

- ✓ La préservation de certain structure de modèle source par la transformation

3. Exécution de la transformation

- ✓ Définition de méta-modèle ***BPMN***
- ✓ Définition des fonctions : **Trans1** et **Trans2**
- ✓ Exécution de la transformation **: Trans2 (Trans (Trans1 (BPMN)) ------> réseaux de Petri**

Sommaire

1) *Introduction*

2) *Modèles Formels: Réseaux de Petri*

3) *Intégration des Méthodes Formelles avec l'IDM*

4) *Vérification d'une Transformation*

5) ***Conclusion***

5.Conclusion

- Les méthodes formelles sont utilisées pour le development de logiciels sur.
- les RDPs sont des *methodes formelles qui* offrent une description d'un système en faisant une distinction claire entre l'aspect statique et l'aspect dynamique
- les RDPs Autorisent différentes interprétation sémantique du comportement des systèmes
- Les RDPs permettent la modélisations de: la concurrence, la causalité, la séquence, le conflit et le choix .
- Intégration des Méthodes Formelles avec l'IDM offert la possibilité de vérifier les propriétés attendues des systèmes.
- La vérification & validation des transformations de modèles en utilisant les techniques de vérification formelles augmente la qualité des logiciels.

Références

1. Thèse de Doctorat en sciences de Mr Kerkouche Elhillali (Université Mentouri Constantine, 2010)

2. Amrani, Moussa, et al. "Formal verification techniques for model transformations: A tridimensional classification." The Journal of Object Technology 14.3 (2015): 1-1.

3. Cours de Alloua Chaoui, Université de Constantine 2

4. Jensen, Kurt, and Lars M. Kristensen. Coloured Petri nets: modelling and validation of concurrent systems. Springer Science & Business Media, 2009.

5. Thèse de Doctorat de Mr Said Meghzili (Université Abdelhamid Mehri- Constantine 2)