Memoire de MASTER: « *DSL pour le model-checking d'un réseau de transport public par voie ferrée* »

Marius NGUENA KENGNI

Université de Genève

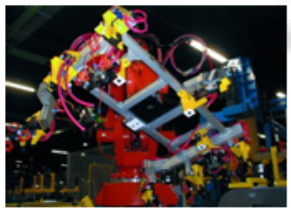
30 septembre 2009

Plan

Des bugs, aux conséquences désastreuses...



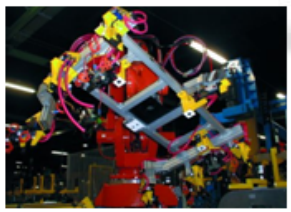






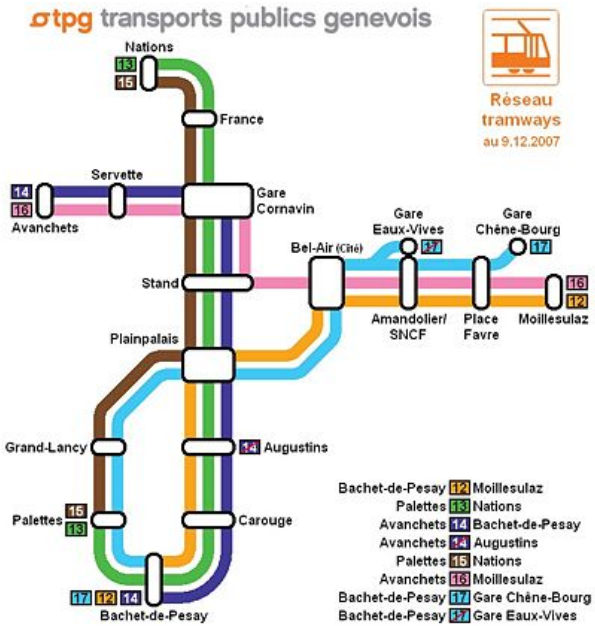




NOKIA
6101
NOKIA

YA
UN BUG!

Nécessité de « *vérifier* » des systèmes de réseau de transport public par trams...



Comment effectuer de telles vérifications ?

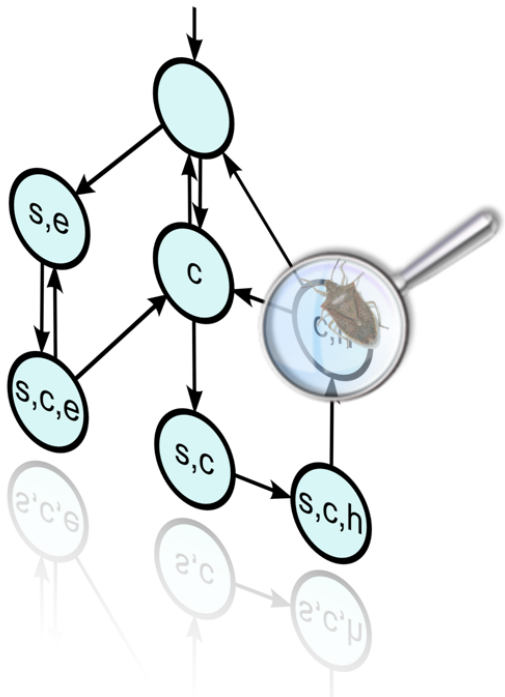
- Test

Comment effectuer de telles vérifications ?

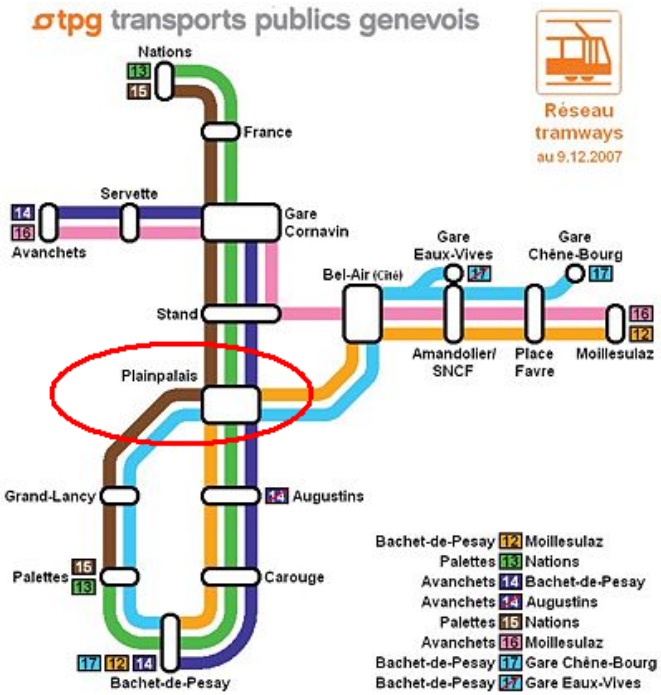
- Test
- Preuve par théorèmes

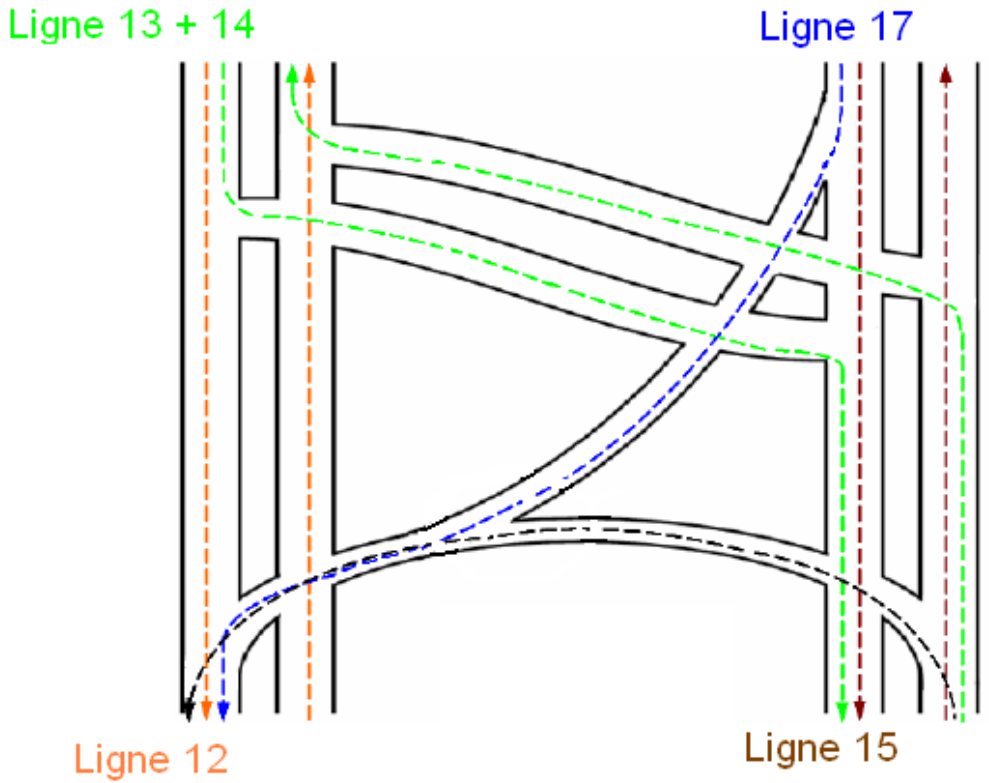
Comment effectuer de telles vérifications ?

- Test
- Preuve par théorèmes
- Model-checking
  - Analyse de l'espace d'états (fini !)



TRAMWAY à Genève

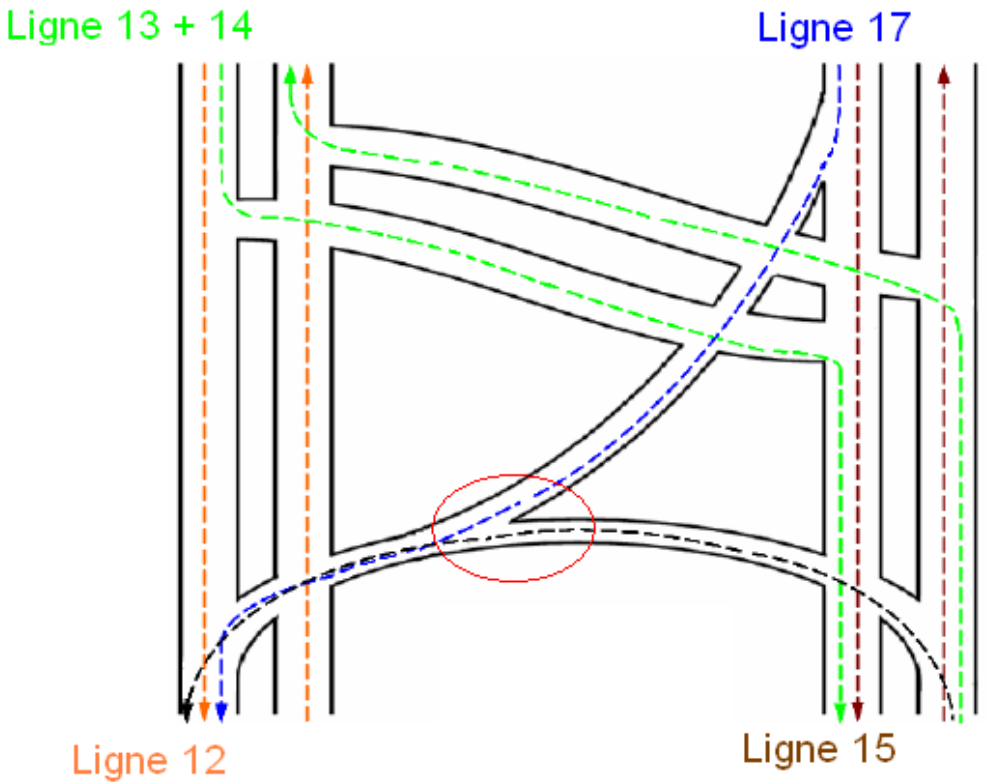




Carrefour de Plainpalais

TRAMWAY : caractéristiques

- Parallélisme
- Asynchrone
- Non déterminisme



Carrefour de Plainpalais

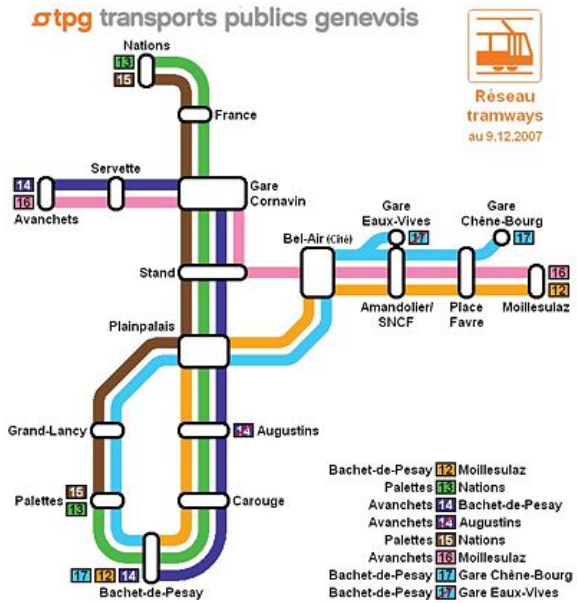
1. Model-checking sur un réseau de Tramway.

1. Model-checking sur un réseau de Tramway.
2. Développement d'un DSL pour les réseaux de Tramway.
   - Syntaxe (abstraite et concrète)
   - Sémantique

1. Model-checking sur un réseau de Tramway.
2. Développement d'un DSL pour les réseaux de Tramway.
   - Syntaxe (abstraite et concrète)
   - Sémantique
3. Sémantique et Transformations : Modularité ==> Réutilisabilité.

Le système *satisfait-il* *certaines propriétés?*

Le système *satisfait-il* *certaines propriétés?*



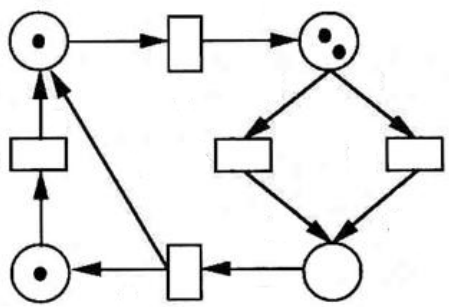
?

Le système *satisfait-il* *certaines propriétés?*

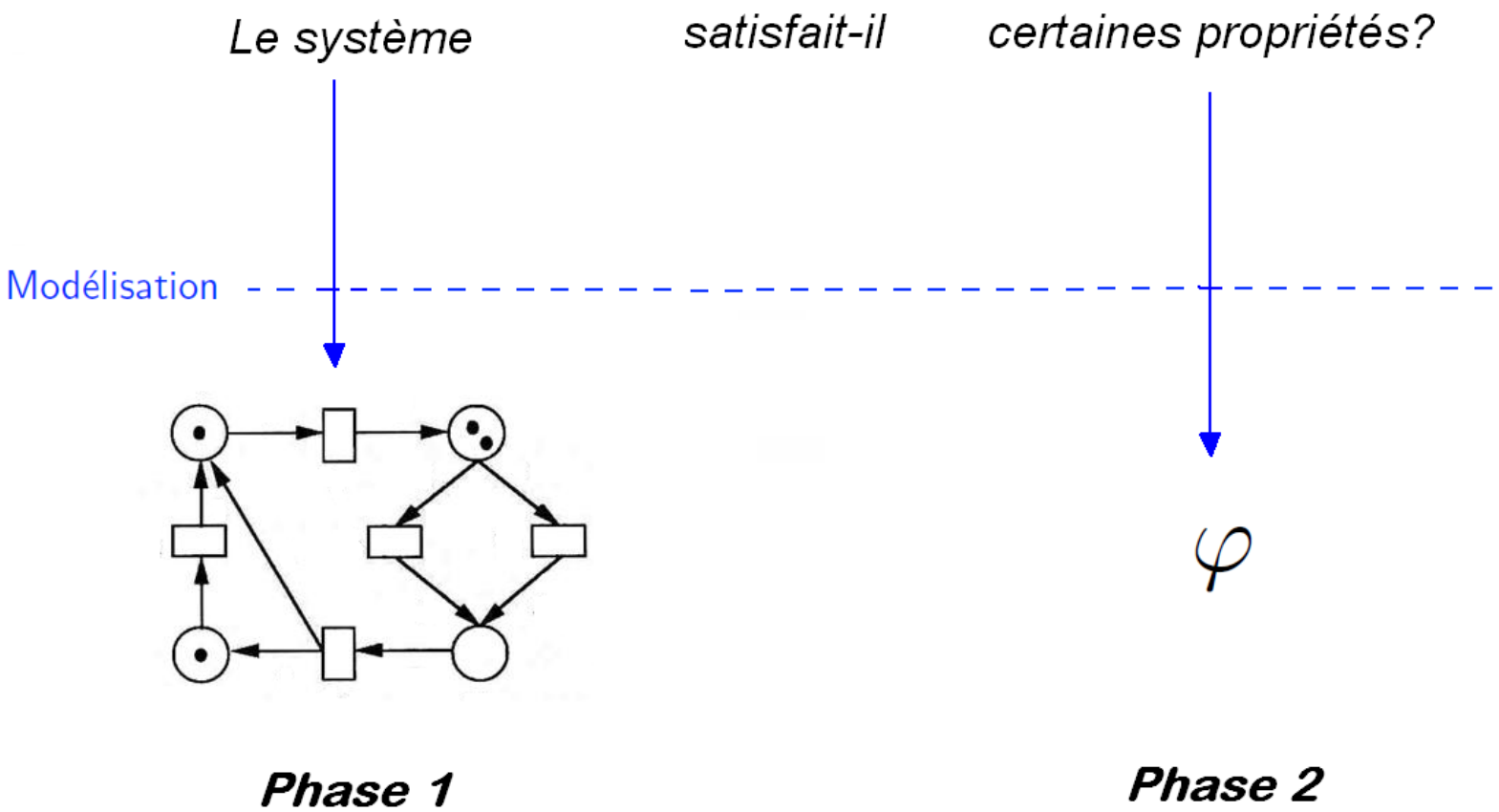
Modélisation

Le système *satisfait-il* *certaines propriétés?*

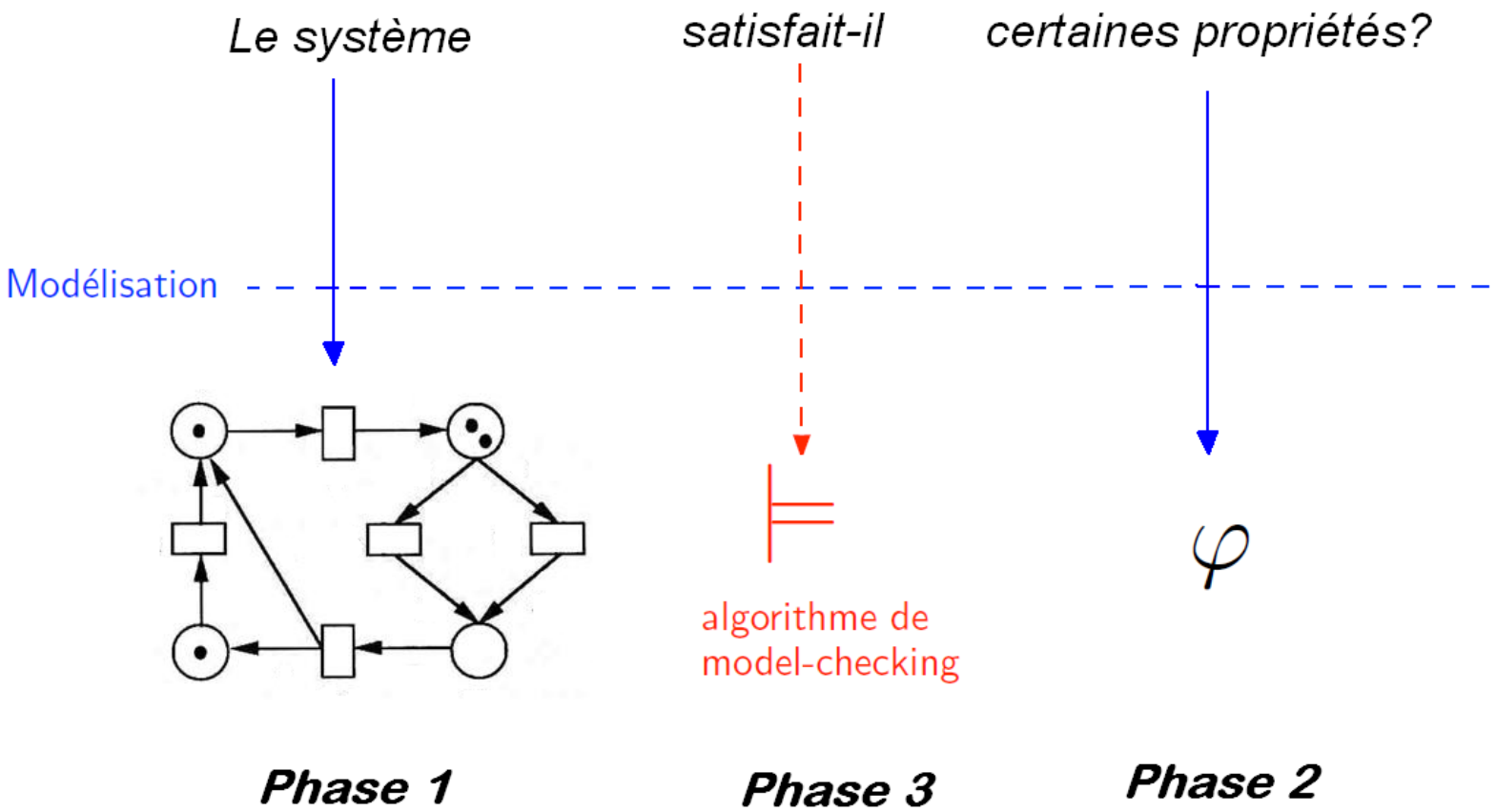
Modélisation



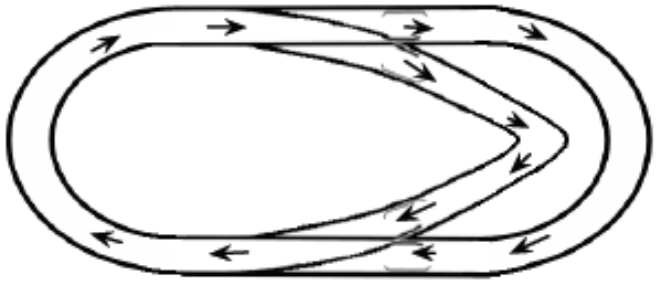
Phase 1


Le système
satisfait-il
certaines propriétés?
Modélisation
φ
Phase 1
Phase 2

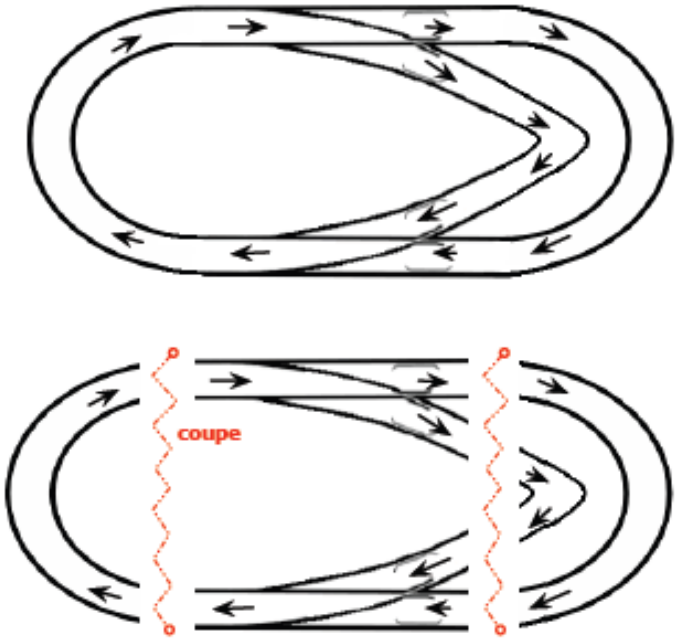
Le système est-il fidèle ? VALIDATION



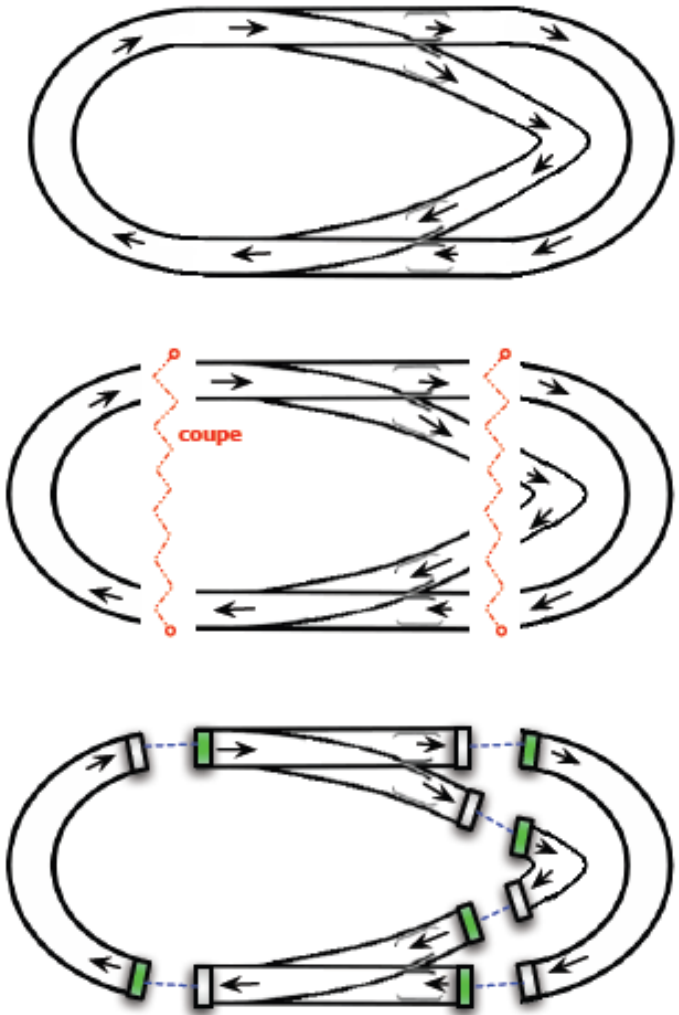
Phase 1 : Modélisation formelle



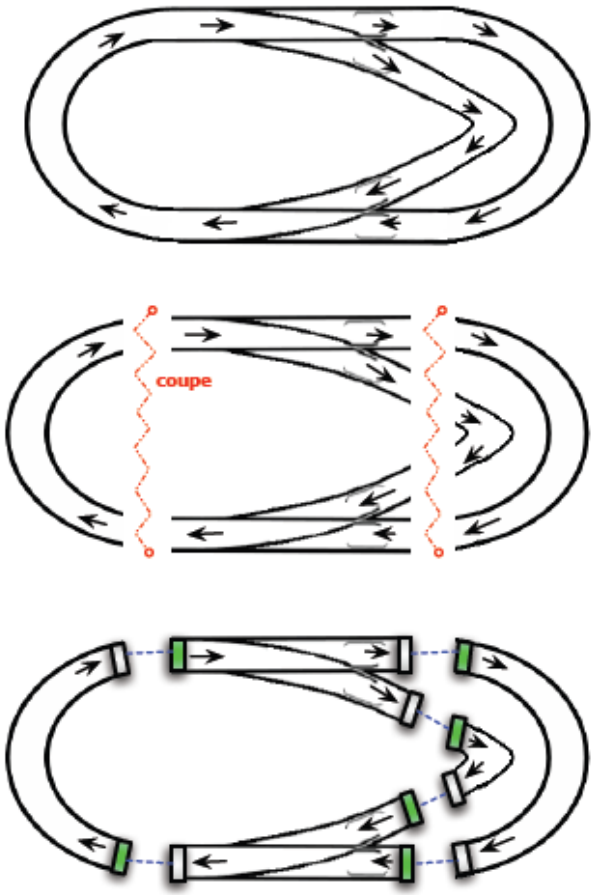
Phase 1 : Modélisation formelle



Phase 1 : Modélisation formelle

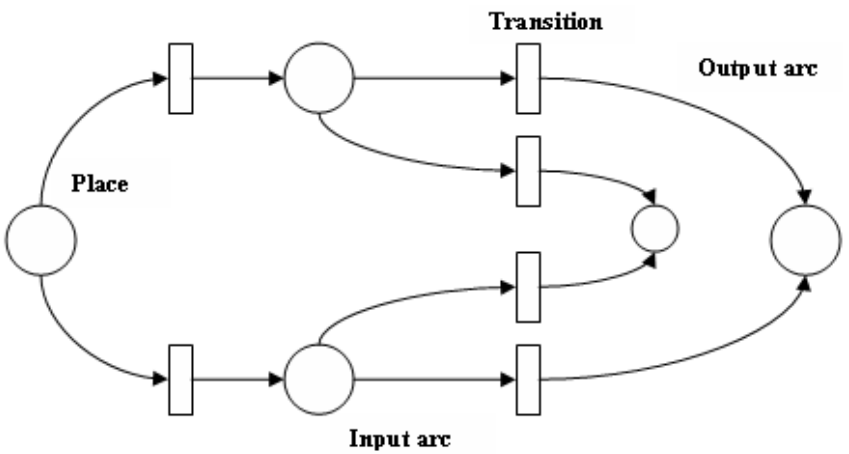


Phase 1 : Modélisation formelle



Un modèle formel :

Réseau de Petri



Phase 2 : Modélisation des propriétés

Propriété 1 : SECURITE

En langage naturel

" Les trams ne se croisent jamais "

En Logique mathématique

$$\forall s \in \boldsymbol{S}, \text{card(s)} \leq 1.$$

Phase 2 : Modélisation des propriétés

Propriété 2 : REGULARITE

En langage naturel

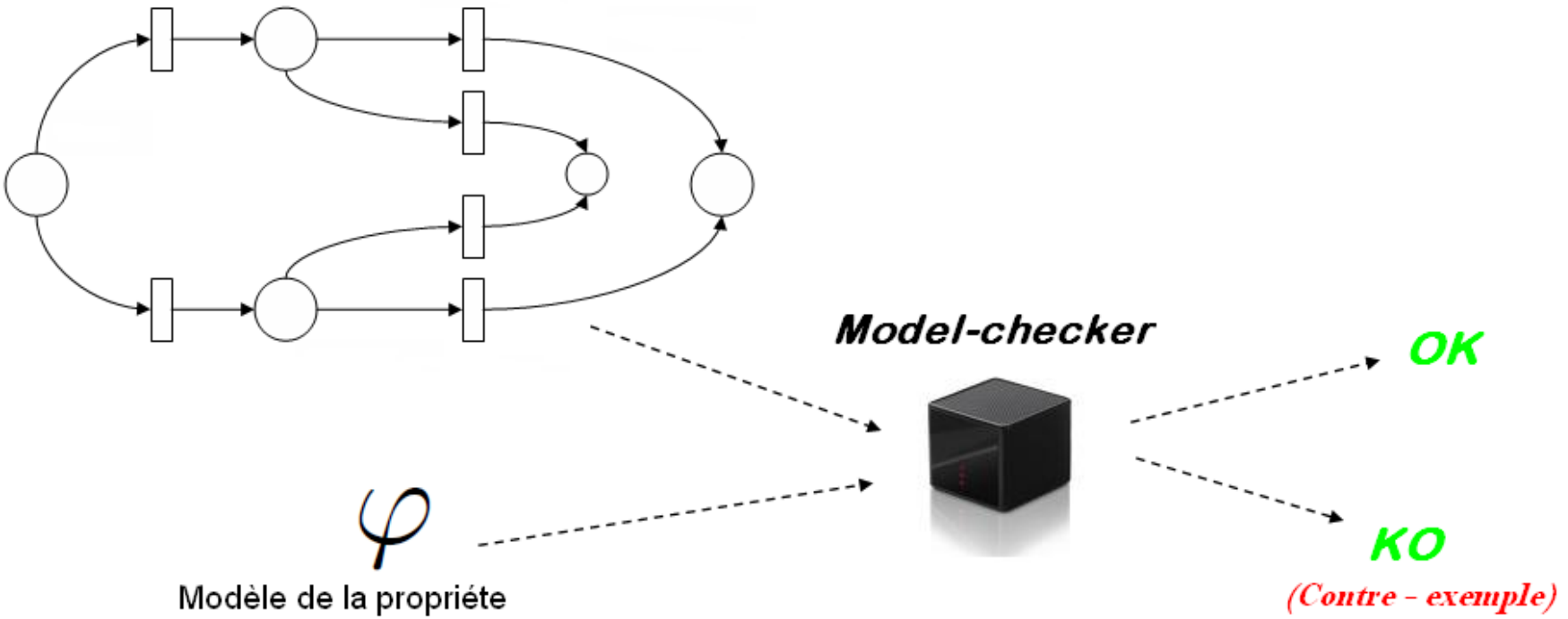
" Les heures d'arrivée des trams aux différents arrêts sont conformes aux tables horaires "

En Logique mathématique

$$\begin{cases} \forall t_{id} \in Trams, \\ \forall P_n, P_m \in Places | n, m \in \mathbb{N}^*, n < m, \\ \forall d_{P_i P_{i+1}} \in \boldsymbol{D}, \\ \forall \delta^-, \delta^+ \in \boldsymbol{D}_\delta, \\ Harr_{t_{id}}(P_m) \in [Harr_{t_{id}}(P_n) + \sum_{i=n}^{m} d_{P_i P_{i+1}} - \sum_{i=n}^{m} \delta^-_{P_i P_{i+1}}, \\ \qquad\qquad Harr_{t_{id}}(P_n) + \sum_{i=n}^{m} d_{P_i P_{i+1}} + \sum_{i=n}^{m} \delta^+_{P_i P_{i+1}}] \end{cases}$$

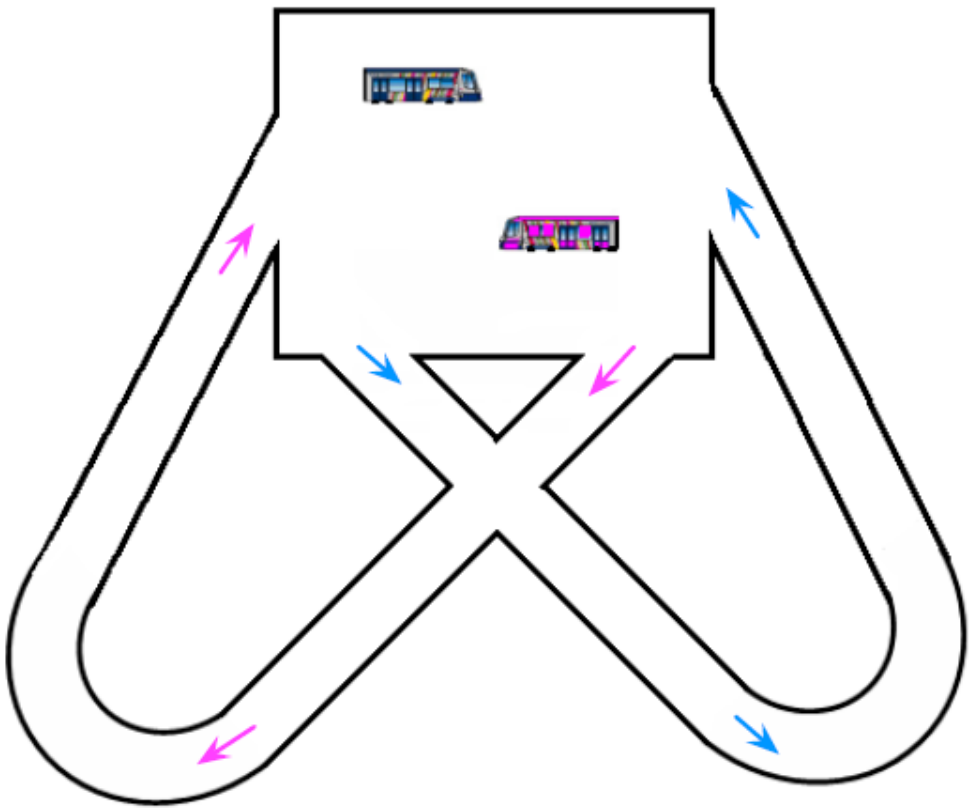
$$\boldsymbol{D}, \boldsymbol{D}_\delta \in \mathbb{N}^*$$

Phase 3 : Model-checking

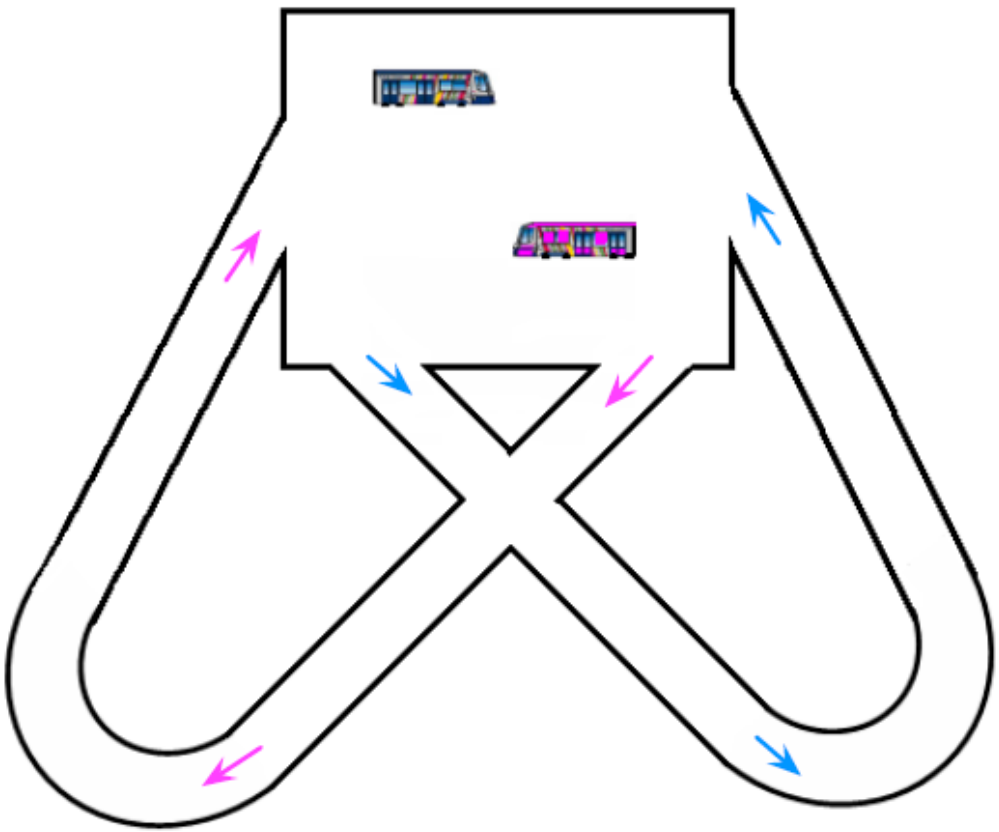


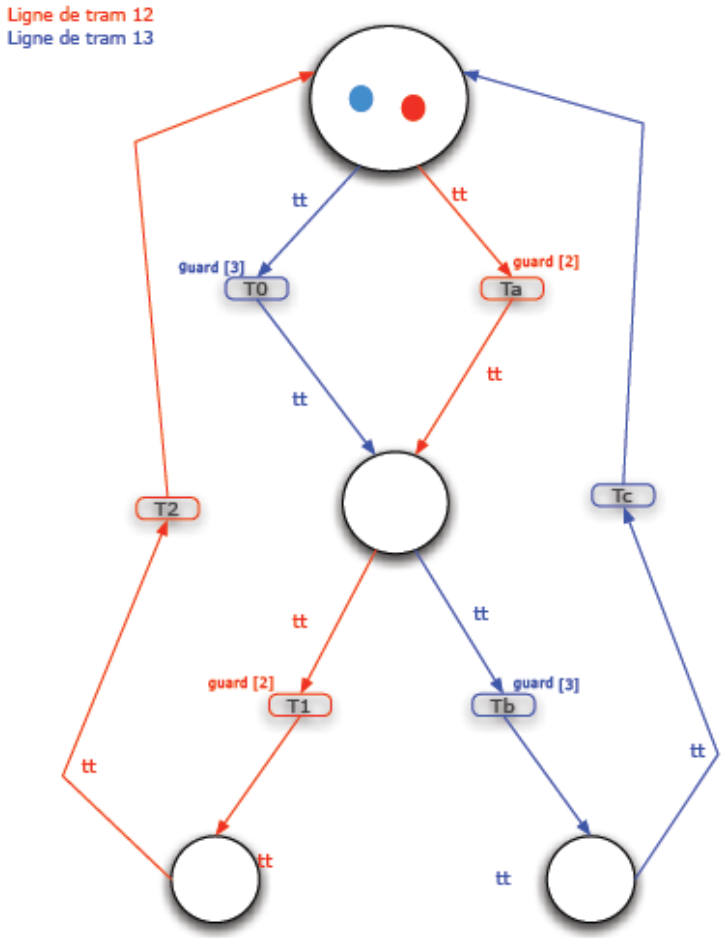
Maquette de test

Ligne de trams 13 et Ligne de trams 12.

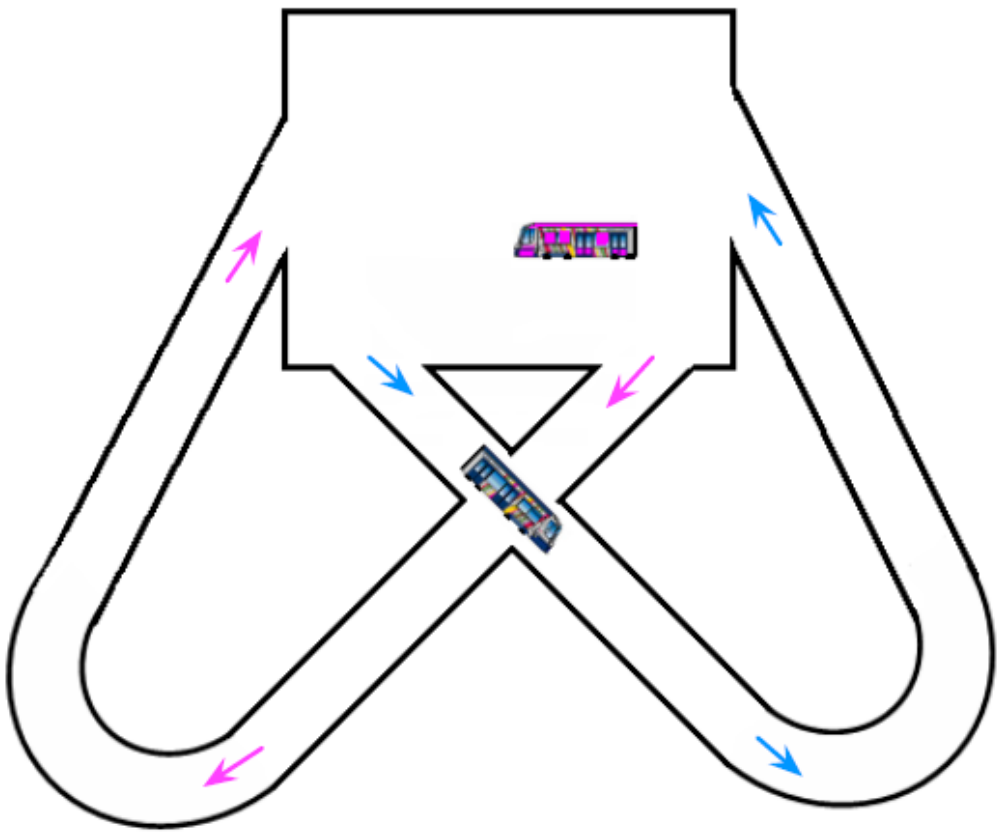


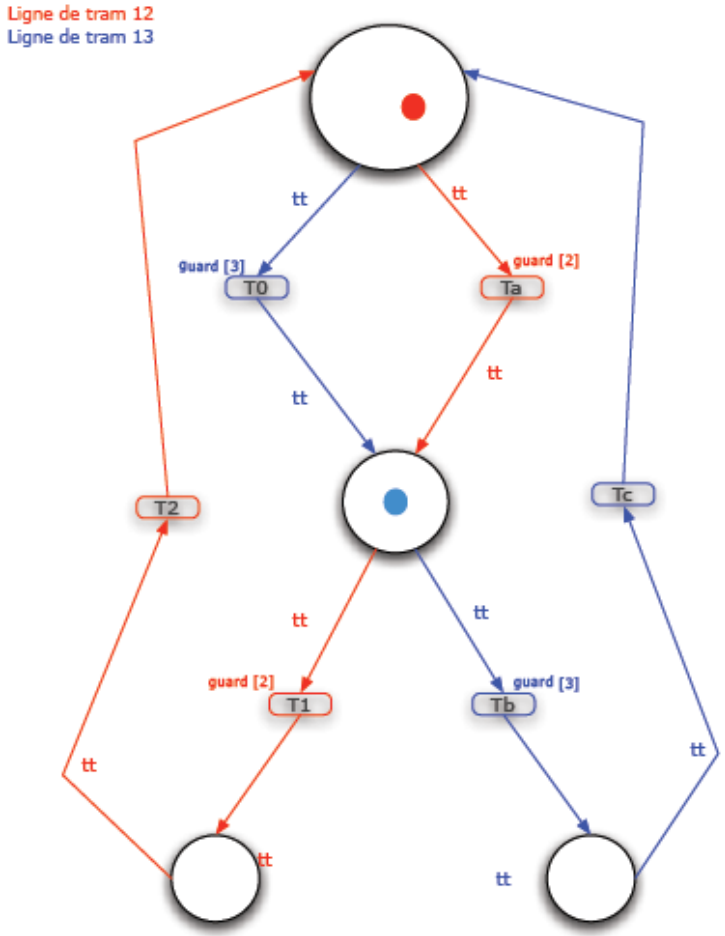
Maquette de test + RdP



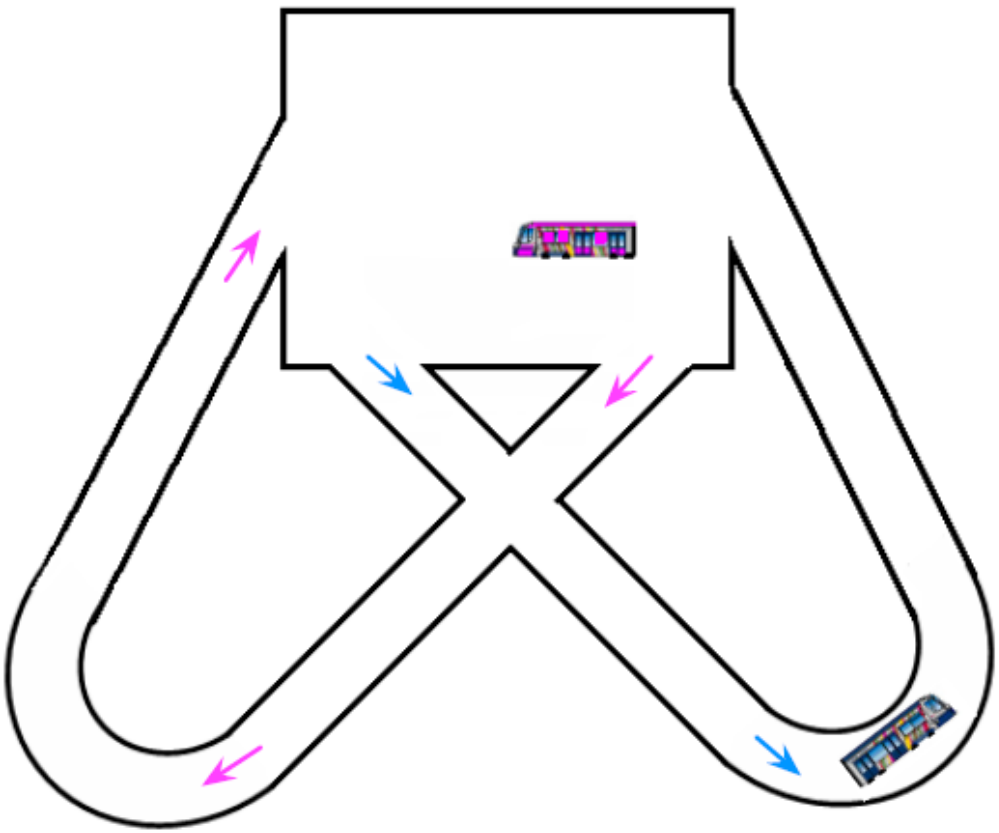


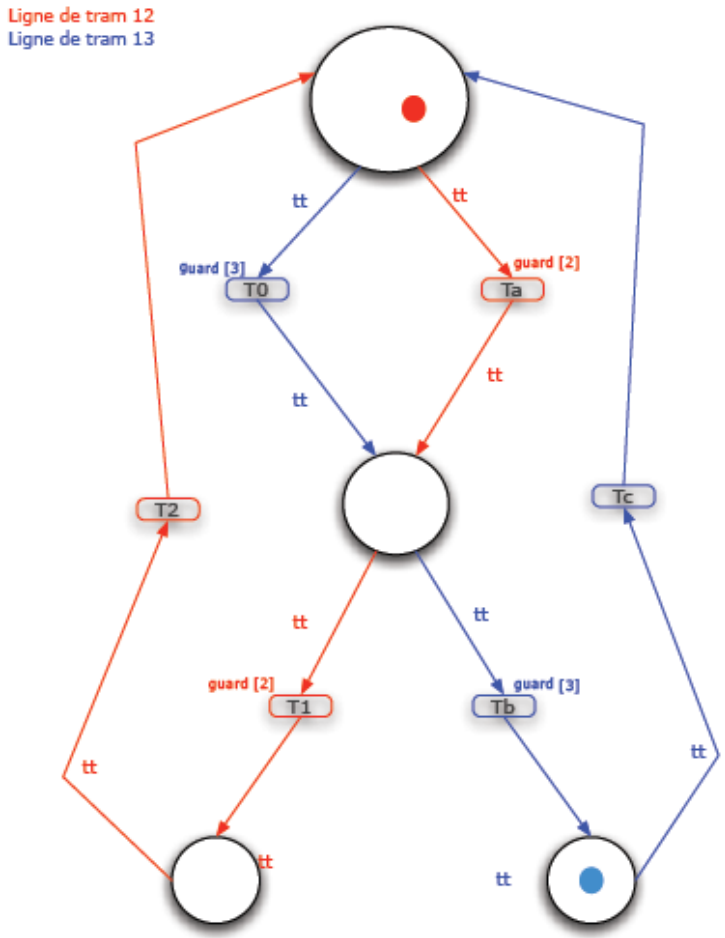
Maquette de test + RdP



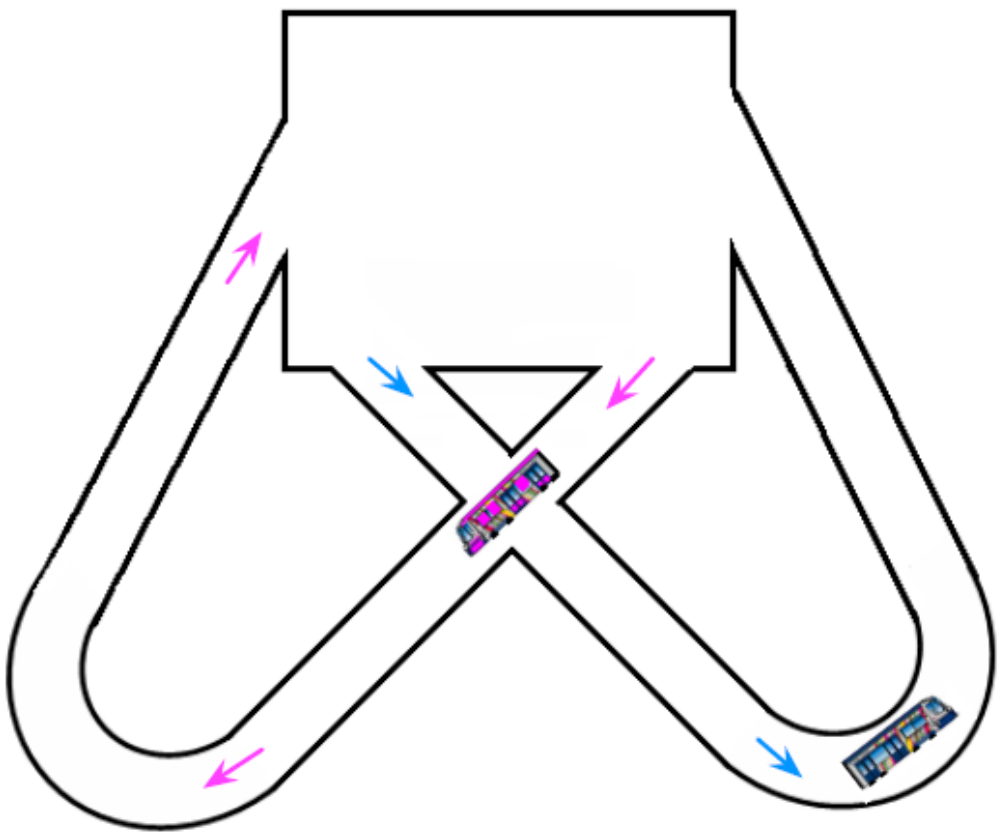


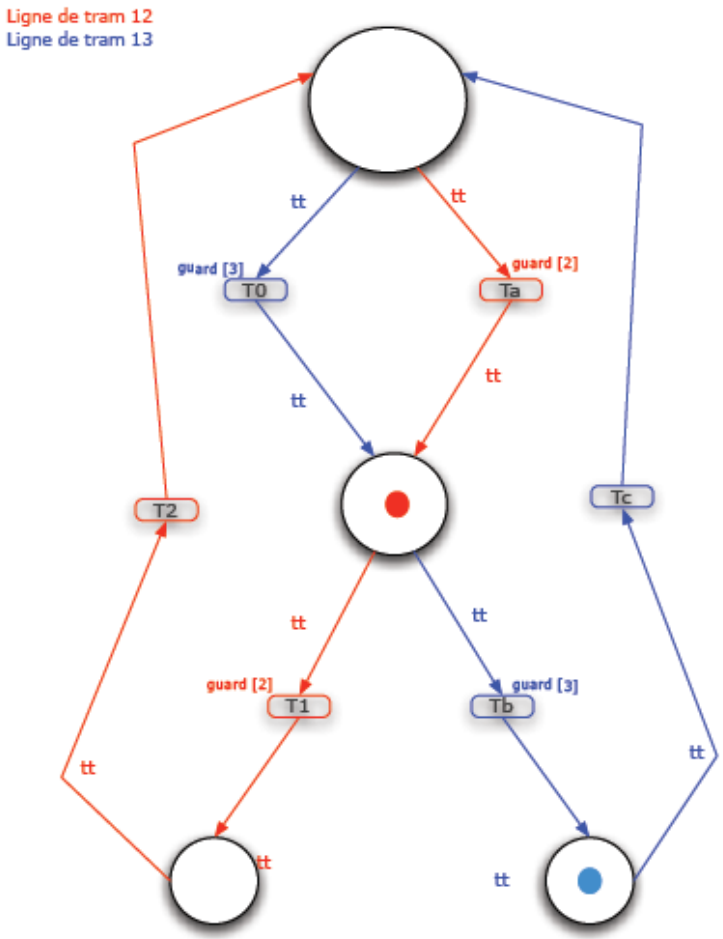
Maquette de test + RdP



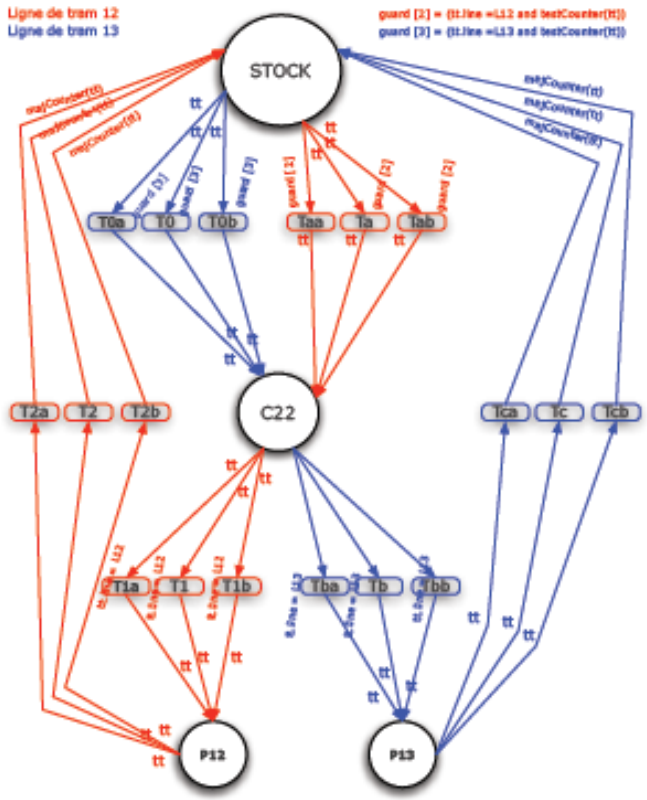


Maquette de test + RdP

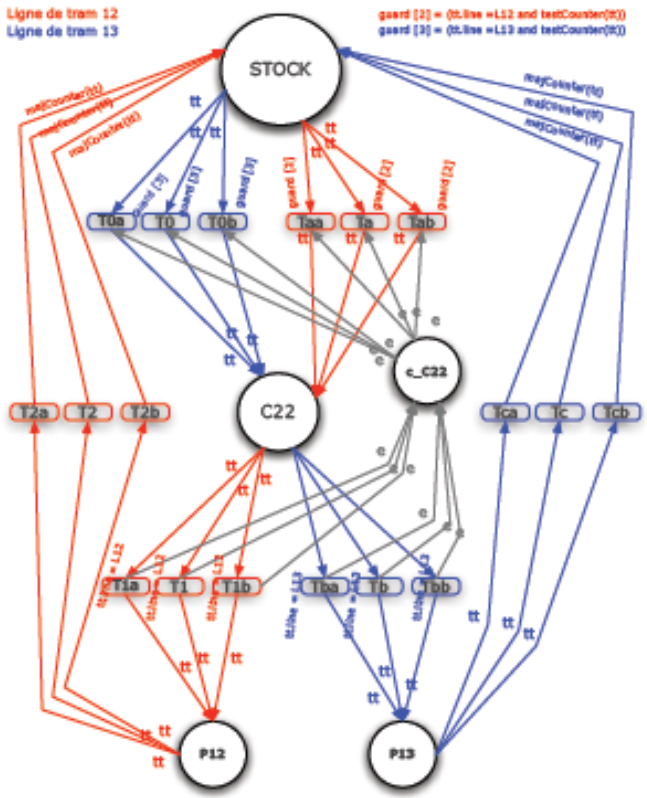




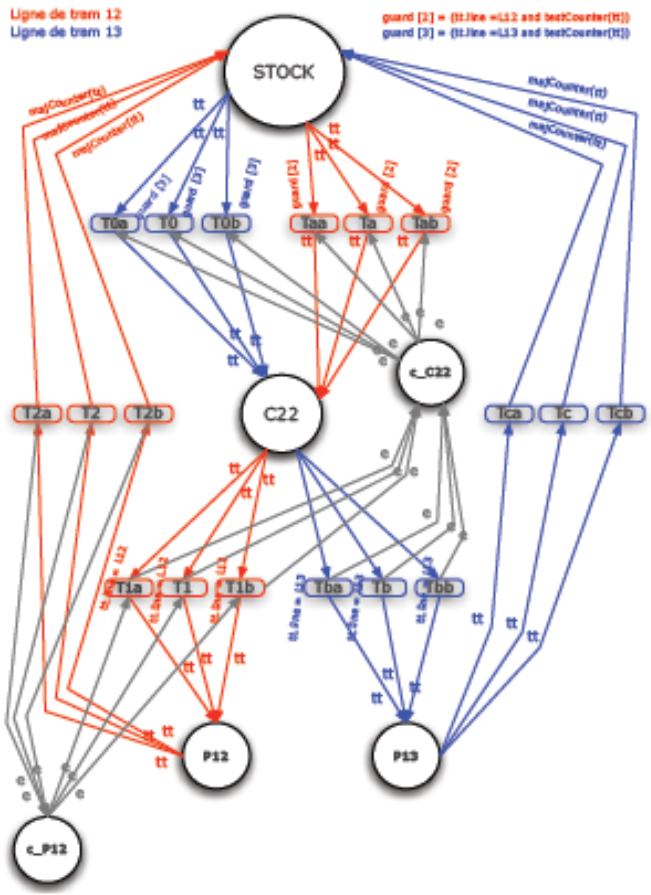
Délais


Ligne de tram 12
Ligne de tram 13
STOCK
T0a
T0
T0b
Taa
Ta
Tab
C22
T2a
T2
T2b
Tca
Tc
Tcb
T1a
T1
T1b
Tba
Tb
Tbb
P12
P13

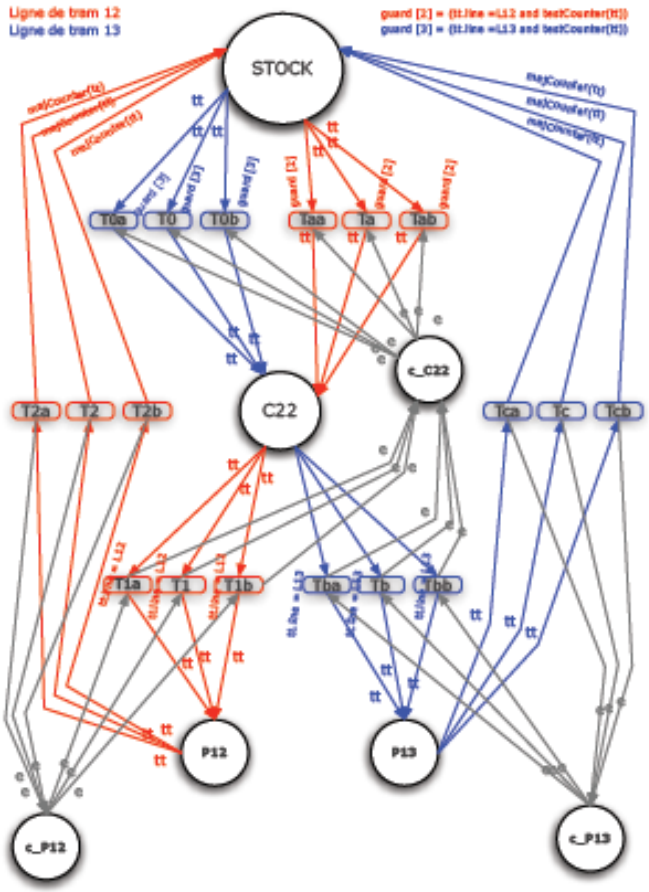
Délais + Capacité



Délais + Capacités



Délais + Capacités



Résultats obtenus

	CPNTools			HELENA		
Nombre de Cycles	1	2	3	1	2	3
Nombre de trams	5	5	5	5	5	5
Nombre d'états	472	5967	20630 (*PARTIAL*)	472	5967	**215136**

Tableau comparatif

CPNTools	HELENA
(+) Interface graphique	(-) Textuel

Tableau comparatif

CPNTools	HELENA
(+) Interface graphique	(-) Textuel
(+) RdP Hierarchiques	(-) Structure globale

Tableau comparatif

CPNTools	HELENA
(+) Interface graphique	(-) Textuel
(+) RdP Hierarchiques	(-) Structure globale
(+) RdP colorés temporisés	(-) RdP colorés

Tableau comparatif

CPNTools	HELENA
(+) Interface graphique	(-) Textuel
(+) RdP Hierarchiques	(-) Structure globale
(+) RdP colorés temporisés	(-) RdP colorés
(-) langage fonctionnel ML	(+/-) Developpé en ADA fonctions complexes(C), Semantique ambigüe places

Tableau comparatif

CPNTools	HELENA
(+) Interface graphique	(-) Textuel
(+) RdP Hierarchiques	(-) Structure globale
(+) RdP colorés temporisés	(-) RdP colorés
(-) langage fonctionnel ML	(+/-) Developpé en ADA fonctions complexes(C), Semantique ambigüe places
(-) Faible espace d'états	(+++) Memoire 10^8 états

Avantages

+ Précision de la réponse
+ Technique automatique
+ Contre-exemple ==> facilite le correction de l'erreur

Avantages et Inconvénients

+ Précision de la réponse
+ Technique automatique
+ Contre-exemple ==> facilite la correction de l'erreur
- Explosion combinatoire
- Uniquement pour des systèmes finis

DSL

GPL

DSL
Latex
Ant
HTML
BNF
BPMN
etc...

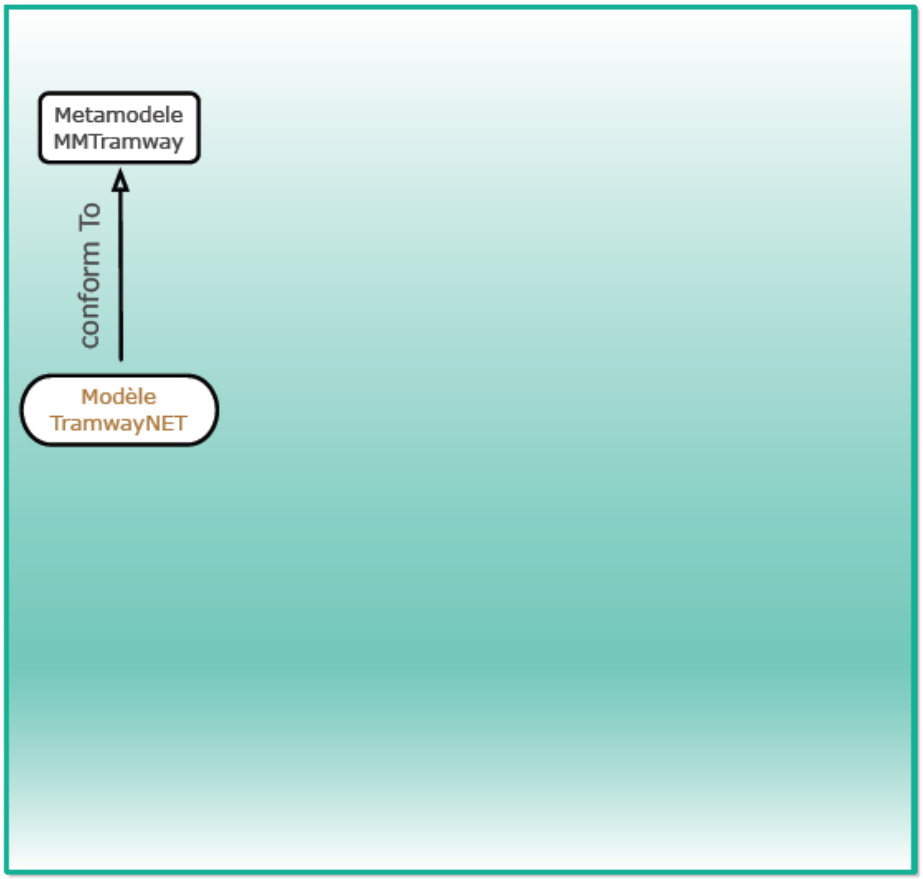
GPL
Ada
C++
Java
UML
Caml
etc...

Comme tout langage de programmation...

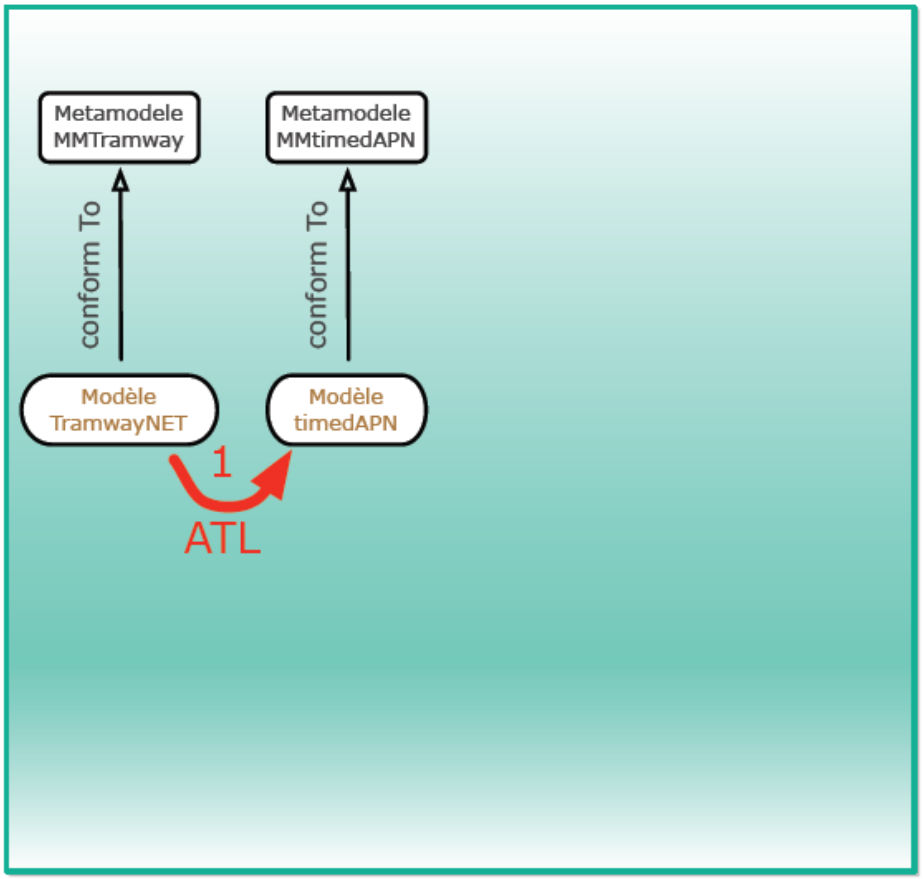
Syntaxe (abstraite et concrète)

Sémantique

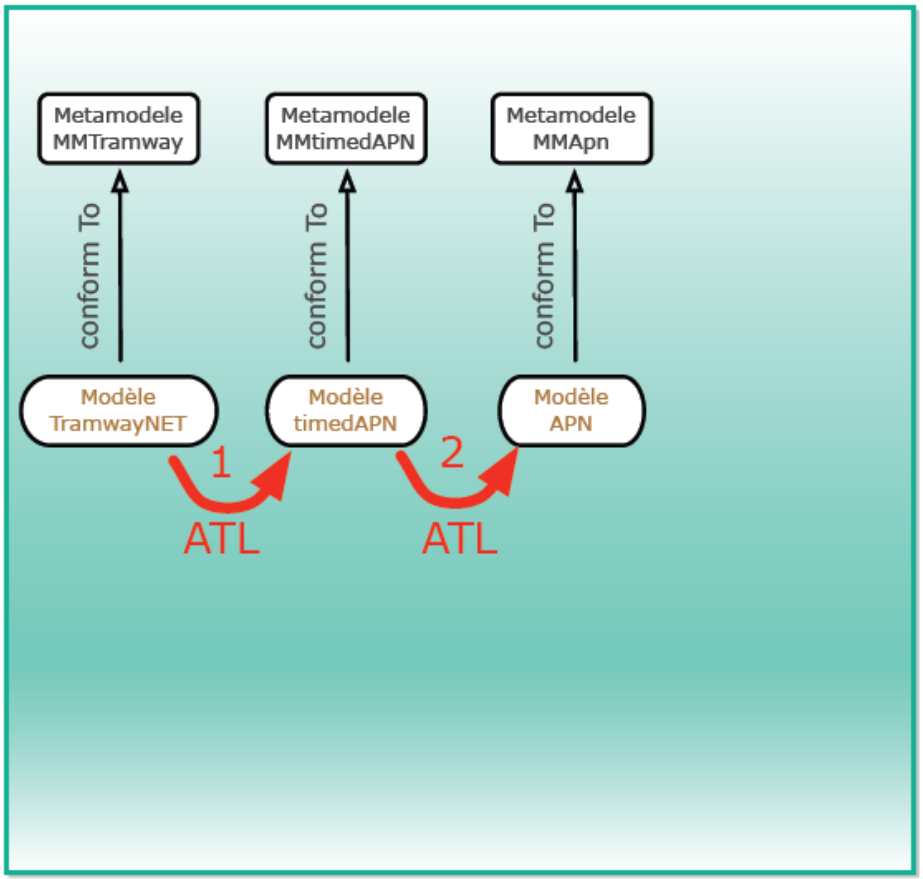
Overview conception du DSL



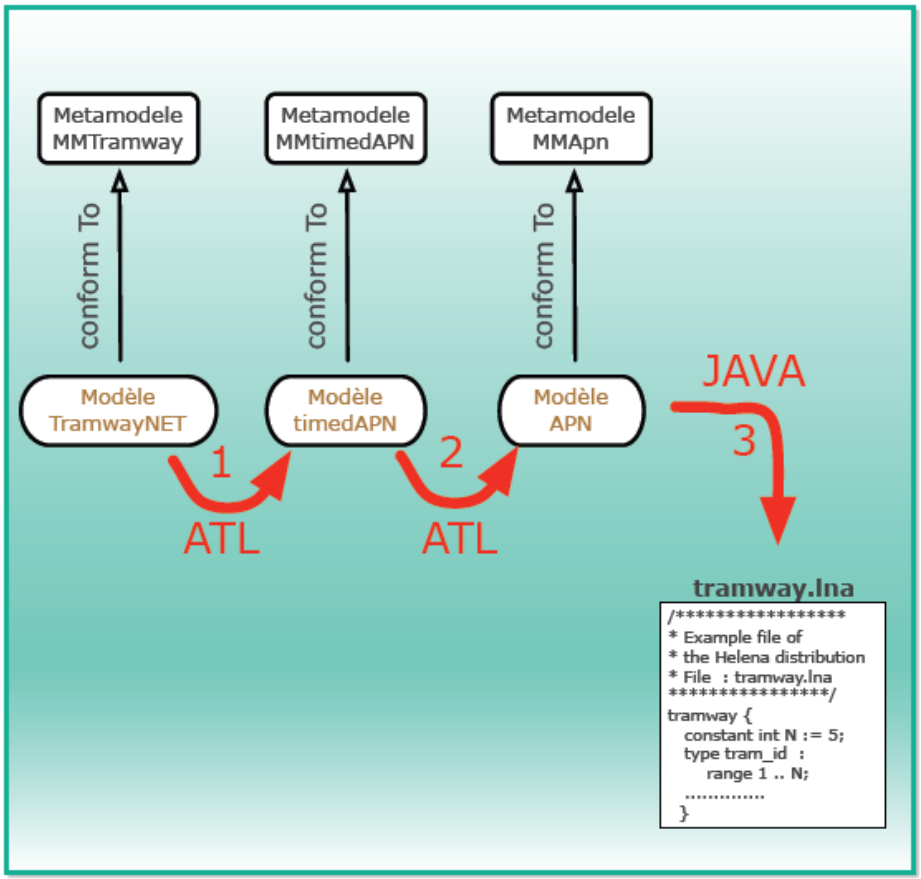
Overview conception du DSL



Overview conception du DSL



Overview conception du DSL



Programmation difficile

Comment on fait ça, st'plait ?

Programmation plus facile

Comment on fait ça, st'plait ?

- Abstractions et notations appropriées.

Réutilisation difficile

Je crée une librairie ou elle existe déjà ?

Réutilisation systématique

Je crée une librairie ou elle existe déjà ?

- Réutilisation guidée par les constructions du langage.
- Implementation proche de la conception.

Vérification difficile

Je voudrais savoir si mon programme termine et s'il calcule ce que je veux !

Vérification plus facile

Je voudrais savoir si mon programme termine et s'il calcule ce que je veux !

- Sémantique restreinte pour rendre certaines propriétés décidables.

Pourquoi utiliser un DSL ?

- Programmation plus facile dans un domaine bien spécifique
- Réutilisation systématique
- Vérification plus facile

Coût

- Documentation + Compilateur(analyse, vérification de types, génération de code, optimisation, etc...)

Inter-operabilité

- *Documentation + Compilateur(analyse, vérification de types, génération de code, optimisation, etc...)*
- Difficultés à faire collaborer les DSLs.

Debugger

- *Documentation + Compilateur(analyse, vérification de types, génération de code, optimisation, etc...)*
- *Difficultés à faire collaborer les DSLs.*
- Debugger inexistant ou généralement peu matures...

Difficultés

- Mise en oeuvre technique, apprentissage ATL
- Manipulation de gros métamodèles.

- Thème 1 : Le model-checking.
- Thème 2 : Conception et Implémentation d'un DSL textuel.
  - Syntaxe abstraite (metamodèle).
  - Syntaxe concrète.
  - Semantique.

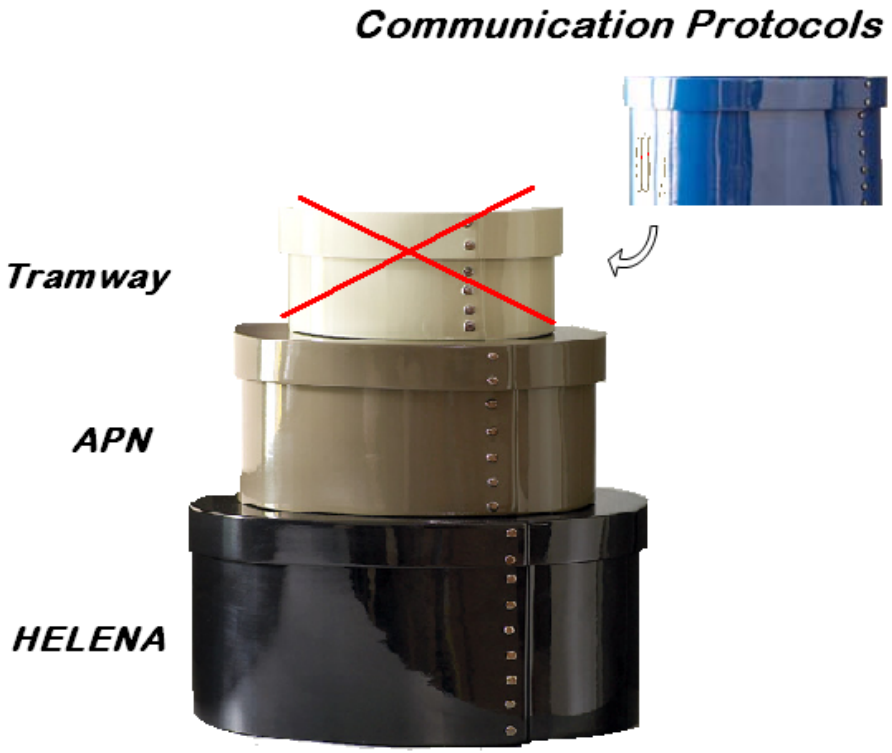
Modularite + Reutilisabilité

Ne mettez pas tous vos oeufs dans le même panier !

Modularite + Reutilisabilité

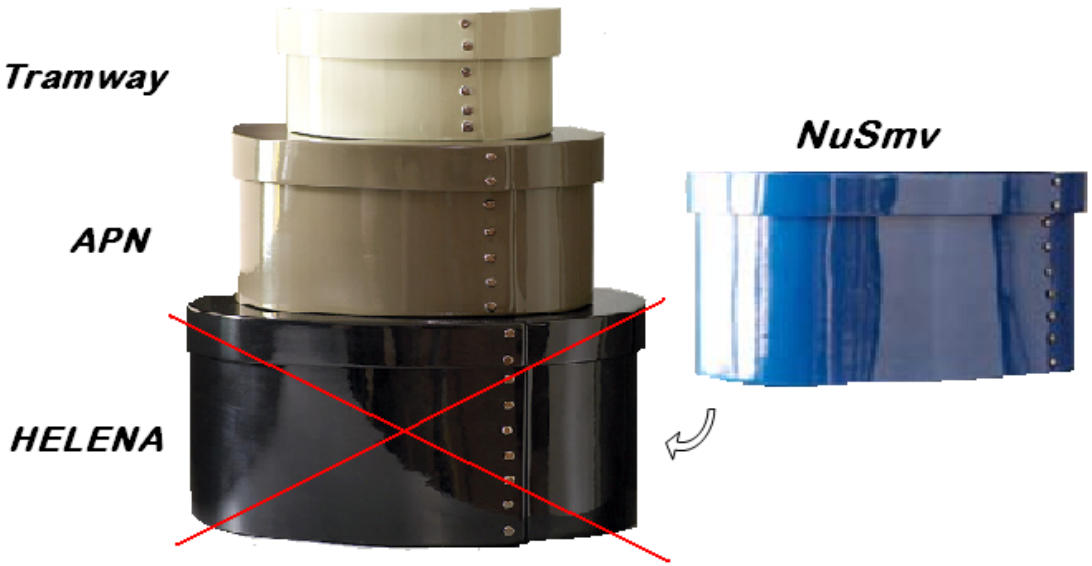
Ne mettez pas tous vos oeufs dans le même panier !

Modularite + Reutilisabilité



Ne mettez pas tous vos oeufs dans le même panier !

Modularite + Reutilisabilité



Ne mettez pas tous vos oeufs dans le même panier !

Ojectifs atteints ?

1. Model-checking sur un réseau de Tramway (95%).
2. Développement d'un DSL pour les réseaux de Tramway.
   - Syntaxe abstraite et concrète (100%)
   - Sémantique et Transformations : Modularité ==> Réutilisabilité. (90%)

- Développer un DSL ***graphique*** pour les réseaux de Tramway.

Merci pour votre attention, des questions ?