Exécution répartie et temps-réel de réseaux de Petri

Supervision distribuée temps-réel

Olivier Baldellon, Matthieu Roy, Jean-Charles Fabre

Laboratoire d'Analyse et d'Architecture des Systèmes — CNRS

Algotel 2012

LAAS-CNRS

Plan

1. La supervision : qu'est-ce ? pourquoi ? comment ?
2. Une nouvelle sémantique
3. Le protocole distribué
4. Conclusion

Plan

1. La supervision : qu'est-ce ? pourquoi ? comment ?
2. Une nouvelle sémantique
3. Le protocole distribué
4. Conclusion

Quel est le problème ?

De plus de plus de systèmes deviennent extrêmement complexes :

- répartis ;
- temps-réel ;
- interagissent avec l'environnement ;
- amenés à être modifié.

Quel est le problème ?

De plus de plus de systèmes deviennent extrêmement complexes :

- répartis ;
- temps-réel ;
- interagissent avec l'environnement ;
- amenés à être modifié.

- Des erreurs apparaîtront à l'exécution
- Nécessité de détecter ces erreurs

Notre approche

La supervision

Notre approche peut se résumer en deux étapes :

Notre approche

La supervision

Notre approche peut se résumer en deux étapes :

Premièrement, on modélise les propriétés du système que l'on désire surveiller (Réseau de Petri).

Notre approche

La supervision

Notre approche peut se résumer en deux étapes :

Premièrement, on modélise les propriétés du système que l'on désire surveiller (Réseau de Petri).

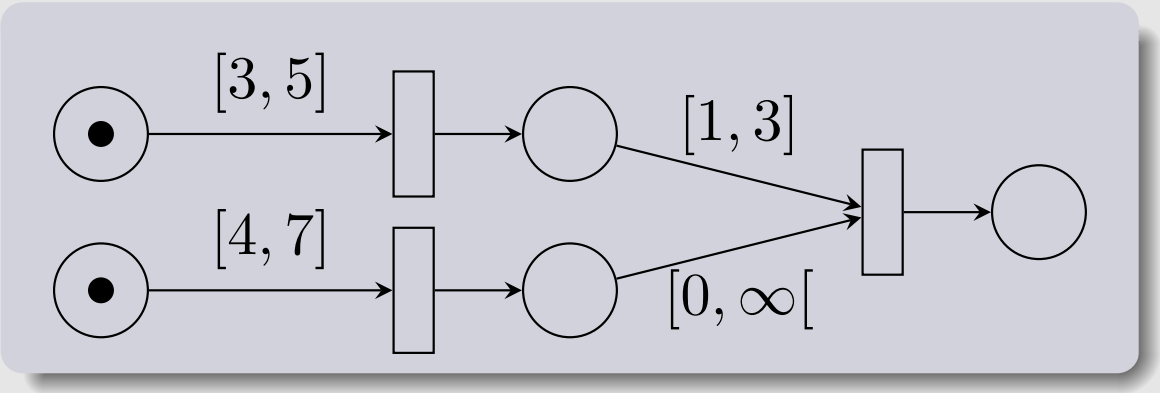
Dans un second temps nous avons besoin d'un moniteur qui

- observe l'exécution
- exécute le modèle à chaque fois qu'un évènement est détecté
- erreur lors de l'exécution du modèle = erreur dans le système

Réseaux de Petri A-temporisé

Le système doit être décrit en utilisant un réseau de Petri A-temporisé

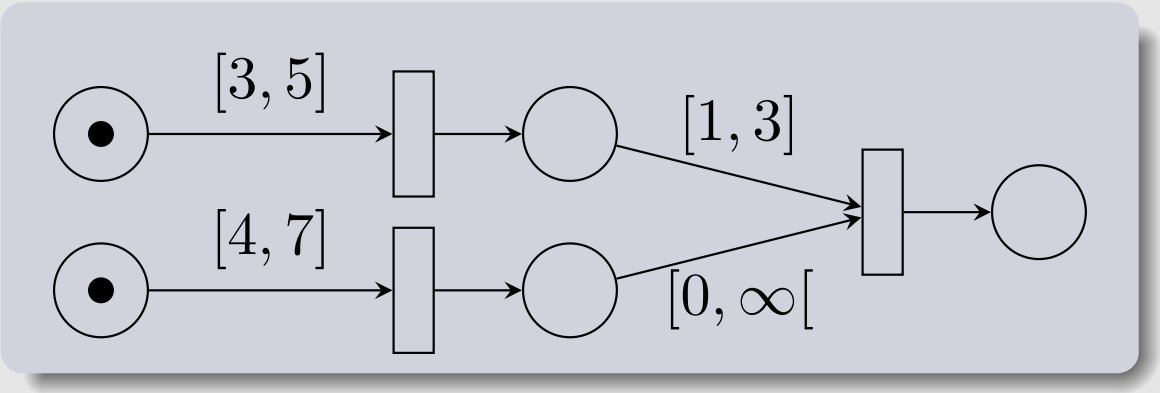
- $t = 0$



Réseaux de Petri A-temporisé

Le système doit être décrit en utilisant un réseau de Petri A-temporisé

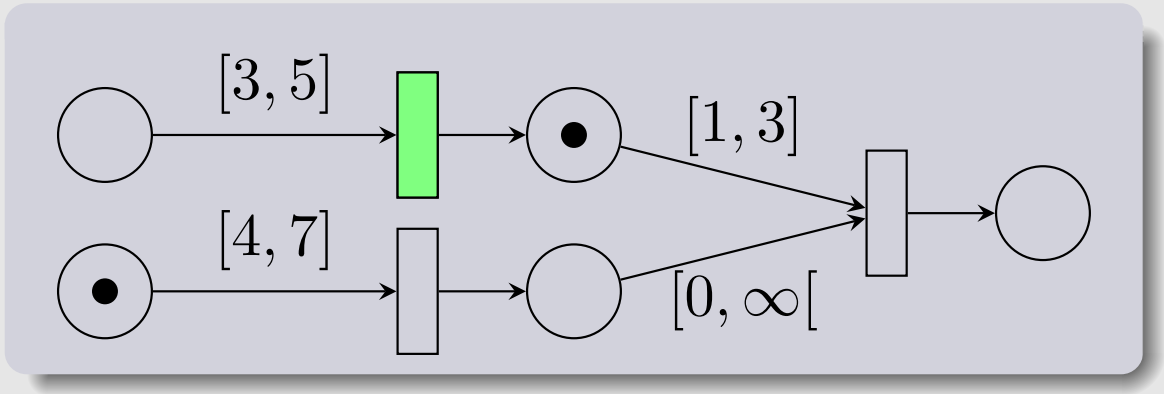
- $t = 0$
- $t = 1$



Réseaux de Petri A-temporisé

Le système doit être décrit en utilisant un réseau de Petri A-temporisé

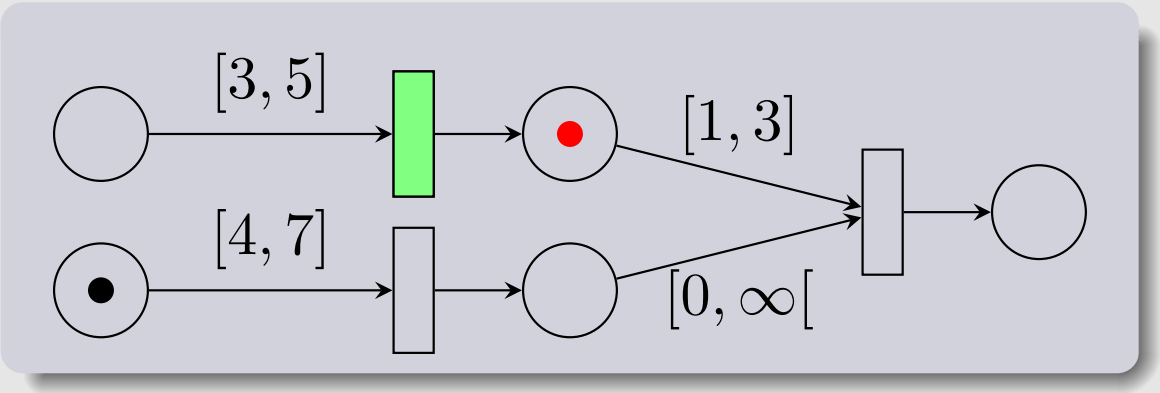
- $t = 0$
- $t = 1$
- $t = 3$



Réseaux de Petri A-temporisé

Le système doit être décrit en utilisant un réseau de Petri A-temporisé

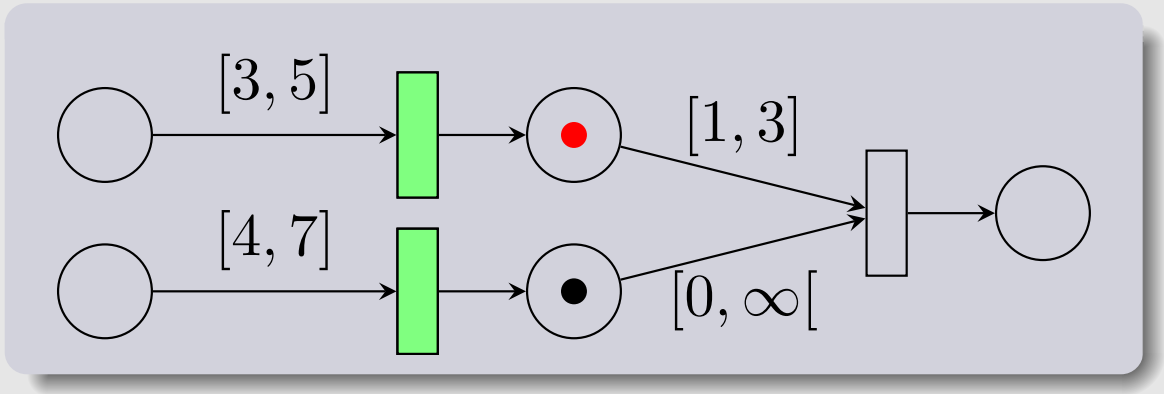
- $t = 0$
- $t = 1$
- $t = 3$
- $t = 6^+$



Réseaux de Petri A-temporisé

Le système doit être décrit en utilisant un réseau de Petri A-temporisé

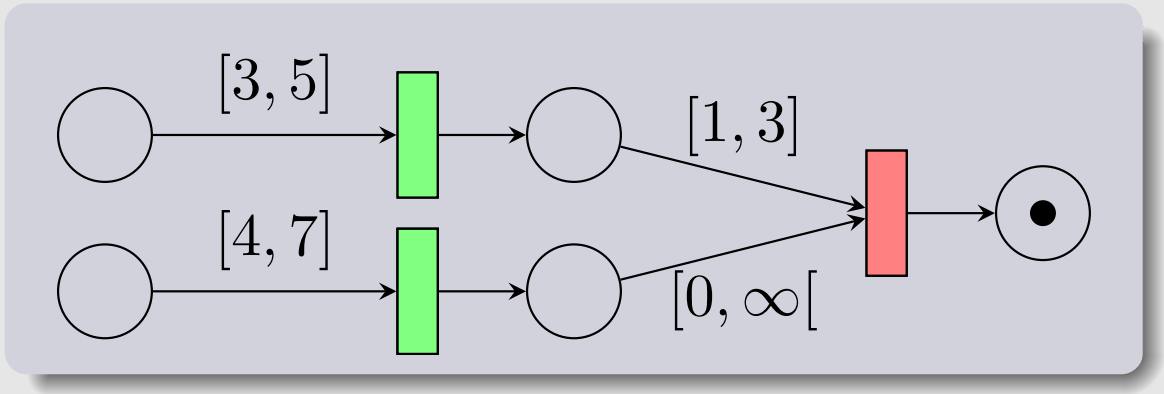
- $t = 0$
- $t = 1$
- $t = 3$
- $t = 6^+$
- $t = 7$



Réseaux de Petri A-temporisé

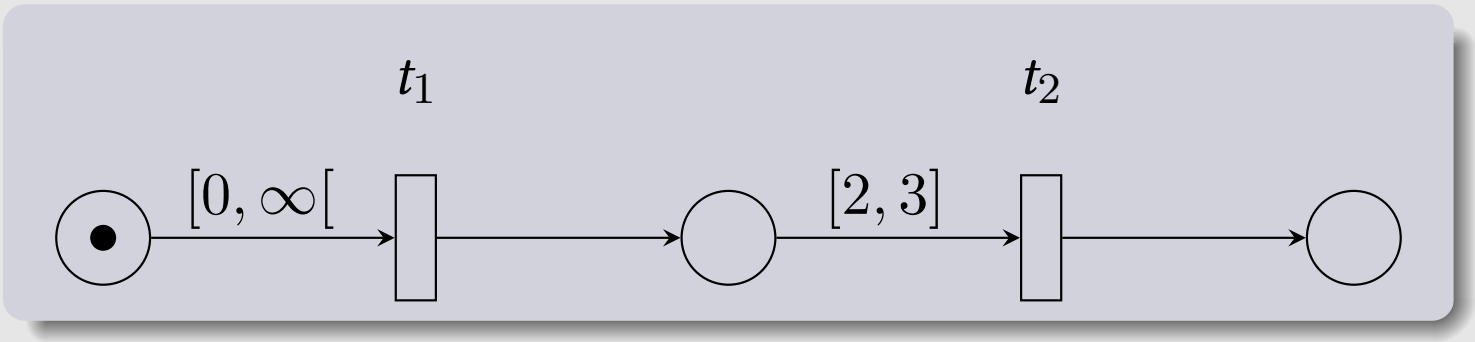
Le système doit être décrit en utilisant un réseau de Petri A-temporisé

- $t = 0$
- $t = 1$
- $t = 3$
- $t = 6^+$
- $t = 7$



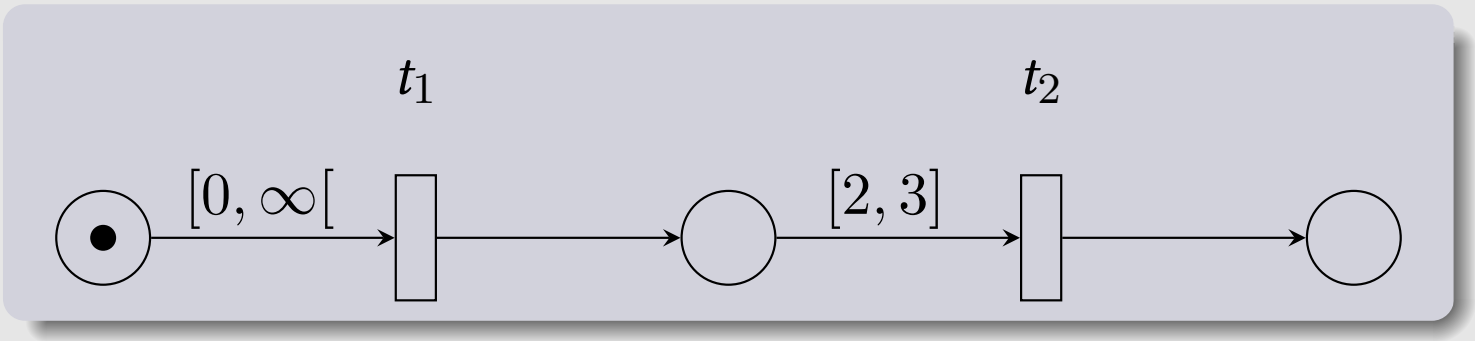
En pratique

- Le superviseur détecte des évènements du système
- Un évènement est un couple $(\mathit{transition}, \mathit{date})$
- Le superviseur exécute une observation (suite d'evènements)



En pratique

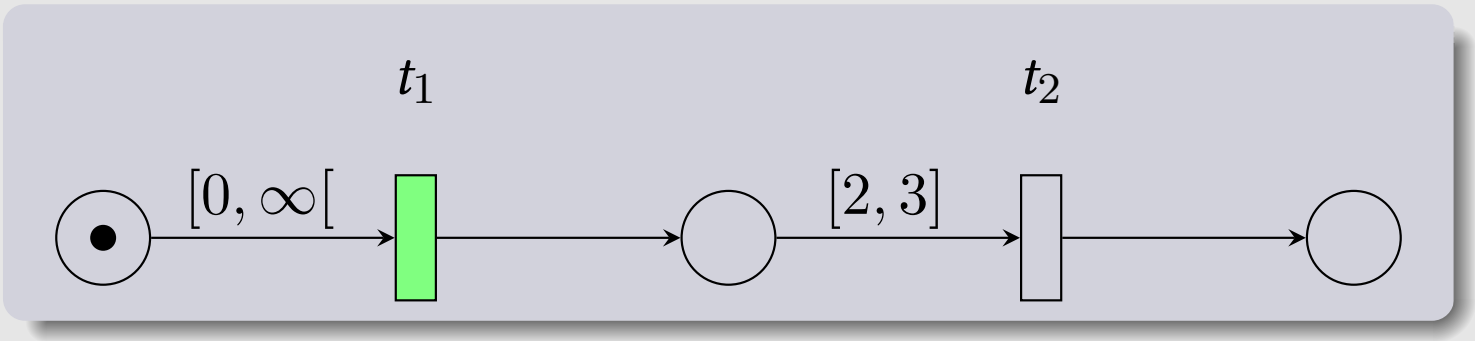
- Le superviseur détecte des évènements du système
- Un évènement est un couple $(transition, date)$
- Le superviseur exécute une observation (suite d'evènements)



- $(t_1, 2)$

En pratique

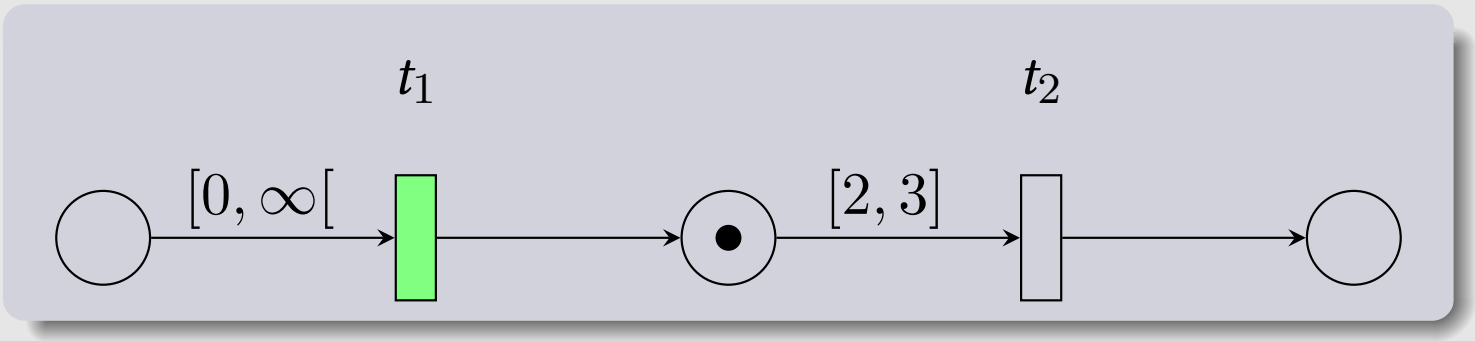
- Le superviseur détecte des évènements du système
- Un évènement est un couple $(transition, date)$
- Le superviseur exécute une observation (suite d'evènements)



- $(t_1, 2)$

En pratique

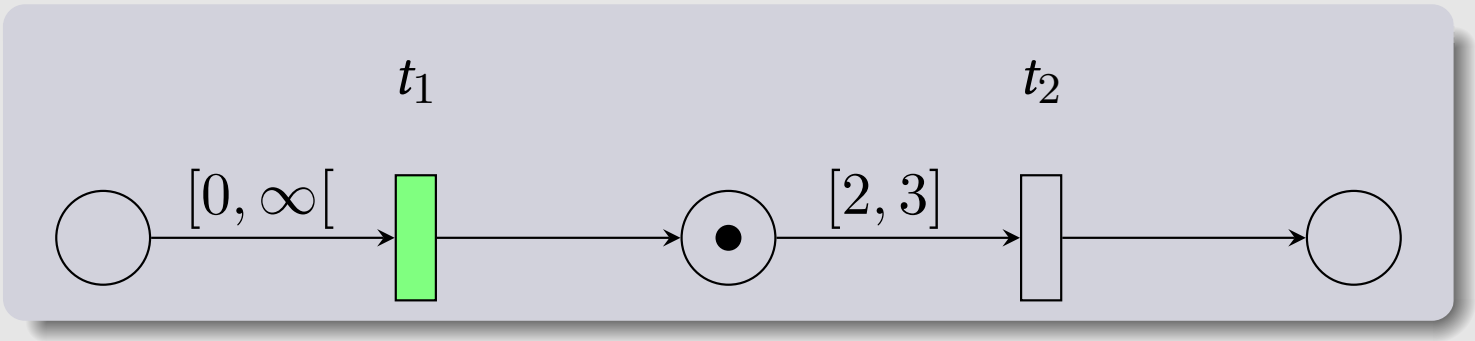
- Le superviseur détecte des évènements du système
- Un évènement est un couple $(transition, date)$
- Le superviseur exécute une observation (suite d'evènements)



- $(t_1, 2)$

En pratique

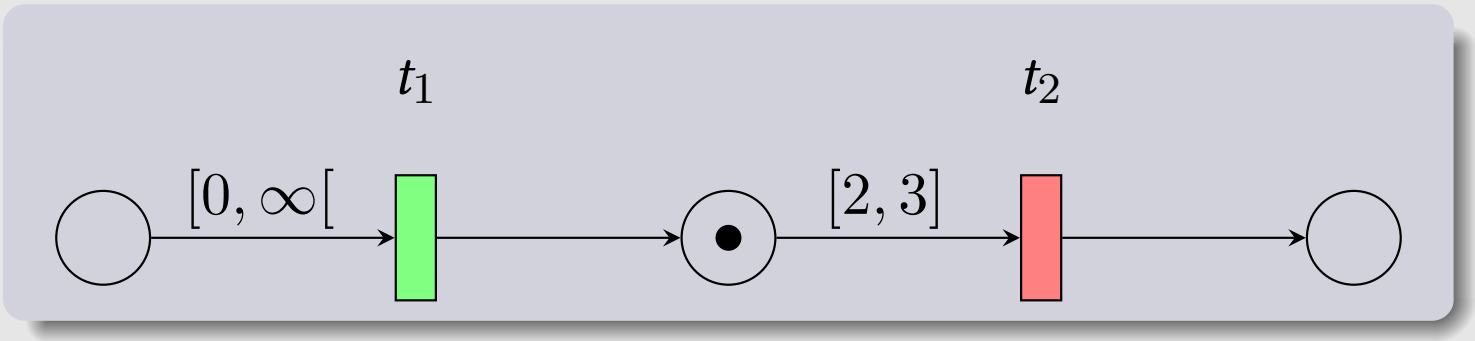
- Le superviseur détecte des évènements du système
- Un évènement est un couple $(transition, date)$
- Le superviseur exécute une observation (suite d'evènements)



- $(t_1, 2)$
- $(t_2, 7)$

En pratique

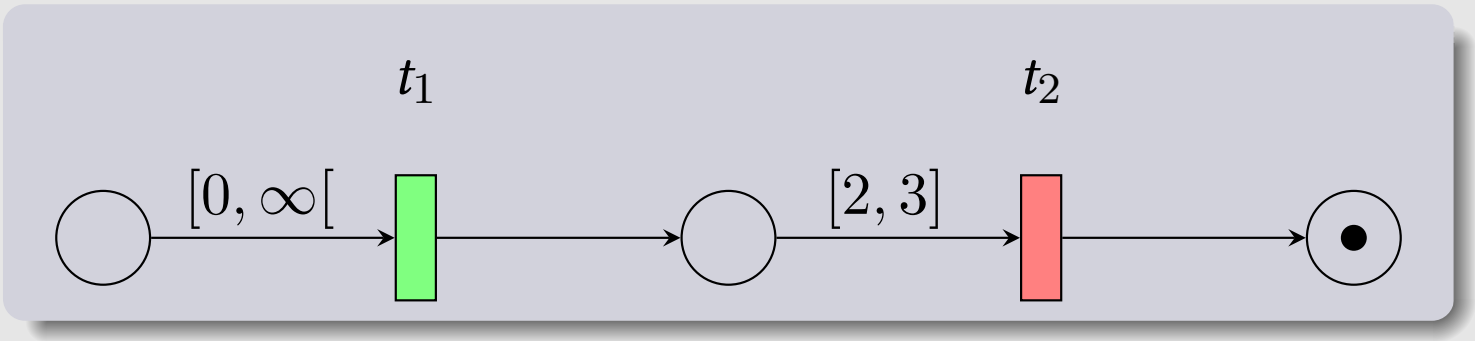
- Le superviseur détecte des évènements du système
- Un évènement est un couple $(transition, date)$
- Le superviseur exécute une observation (suite d'evènements)



- $(t_1, 2)$
- $(t_2, 7)$

En pratique

- Le superviseur détecte des évènements du système
- Un évènement est un couple $(transition, date)$
- Le superviseur exécute une observation (suite d'evènements)



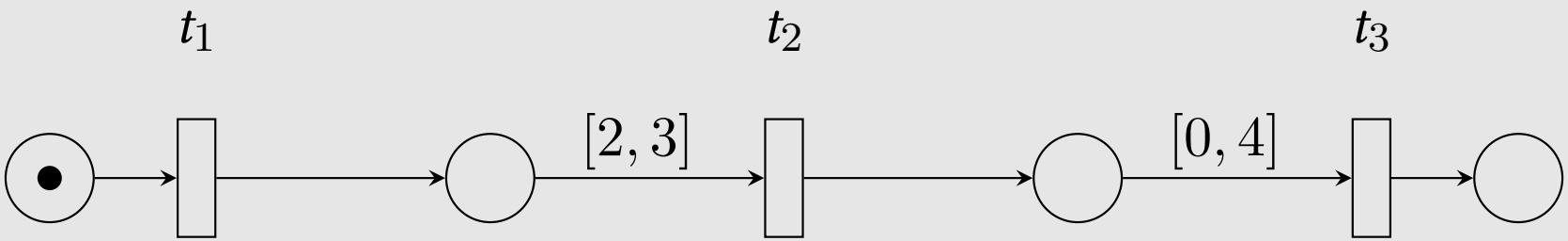
- $(t_1, 2)$
- $(t_2, 7)$

Plan

1. La supervision : qu'est-ce ? pourquoi ? comment ?
2. Une nouvelle sémantique
3. Le protocole distribué
4. Conclusion

Comment exécuter le réseau de Petri

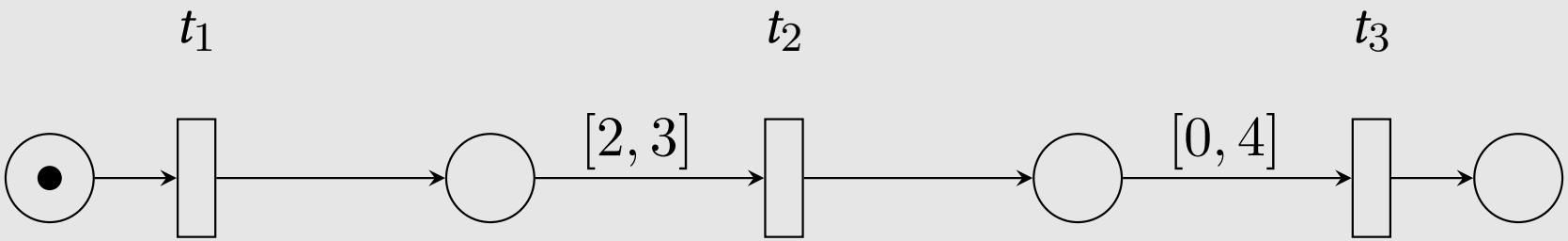
Que faire si un évènement est manquant ?



- $(t_2, 10)$

Comment exécuter le réseau de Petri

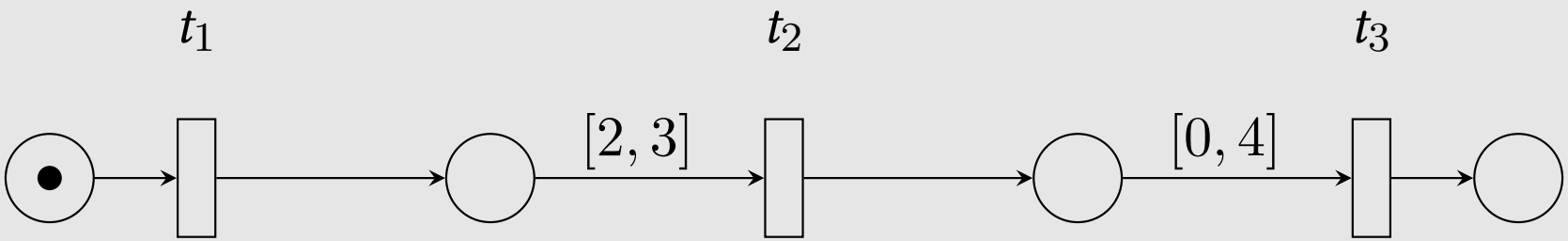
Que faire si un évènement est manquant ?



- $(t_2, 10)$
- $(t_3, 15)$

Comment exécuter le réseau de Petri

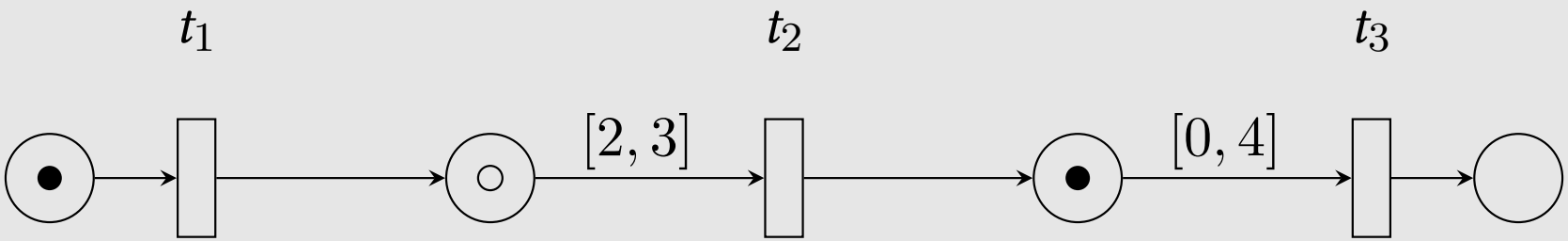
Que faire si un évènement est manquant ?



- $(t_2, 10)$
- $(t_3, 15)$
- ?

Comment exécuter le réseau de Petri

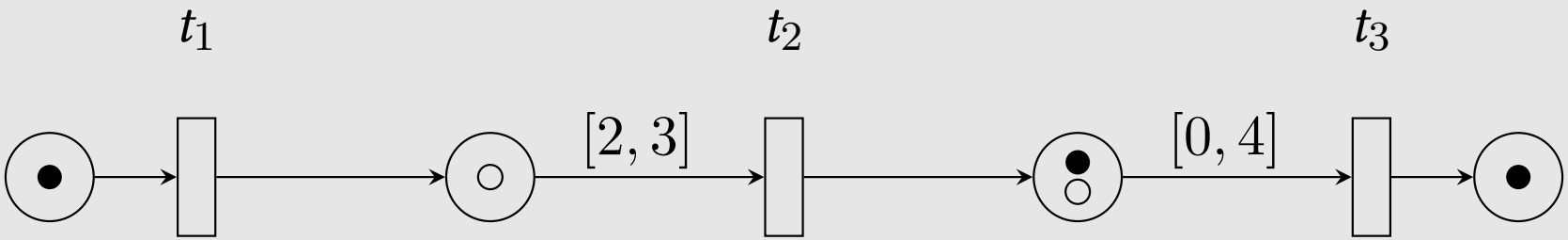
Que faire si un évènement est manquant ?



- $(t_2, 10)$

Comment exécuter le réseau de Petri

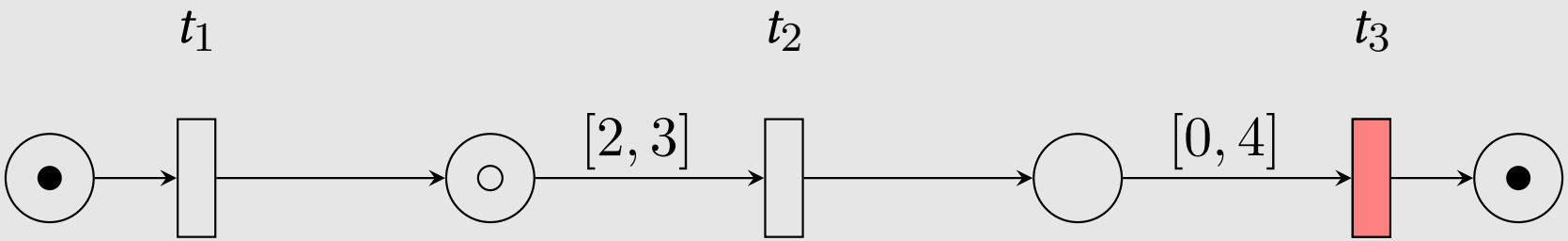
Que faire si un évènement est manquant ?



- $(t_2, 10)$
- $(t_3, 15)$

Comment exécuter le réseau de Petri

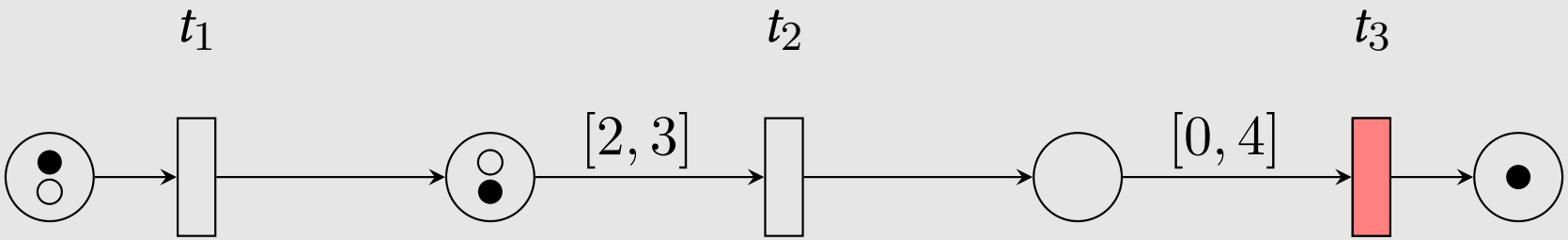
Que faire si un évènement est manquant ?



- $(t_2, 10)$
- $(t_3, 15)$

Comment exécuter le réseau de Petri

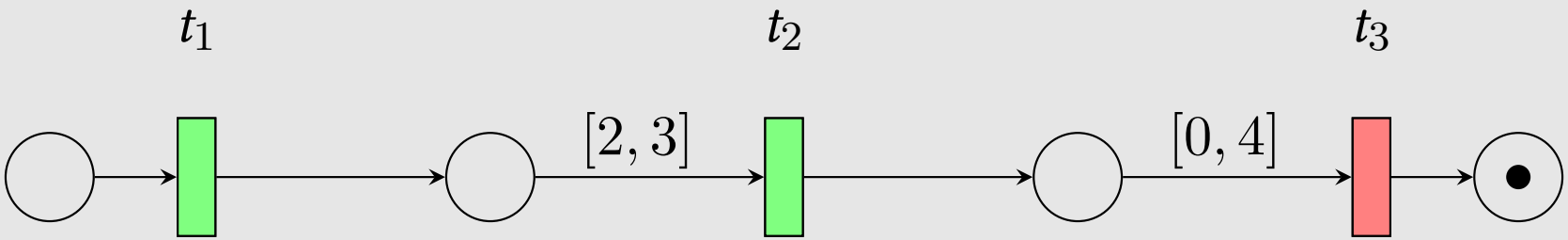
Que faire si un évènement est manquant ?



- $(t_2, 10)$
- $(t_3, 15)$
- $(t_1, 8)$

Comment exécuter le réseau de Petri

Que faire si un évènement est manquant ?



- $(t_2, 10)$
- $(t_3, 15)$
- $(t_1, 8)$

Les jetons négatifs

Pour évaluer la correction d'une exécution il faut :

Les jetons négatifs

Pour évaluer la correction d'une exécution il faut :

- Ajouter les jetons dans les places (positifs ou négatifs) : franchissement des transitions

Les jetons négatifs

Pour évaluer la correction d'une exécution il faut :

- Ajouter les jetons dans les places (positifs ou négatifs) : franchissement des transitions
- Comparer les dates d'apparition entre jetons négatifs et positifs

Les jetons négatifs

Pour évaluer la correction d'une exécution il faut :

- Ajouter les jetons dans les places (positifs ou négatifs) : franchissement des transitions
- Comparer les dates d'apparition entre jetons négatifs et positifs

Deux types d'erreurs

- Erreurs temporelles (transitions rouges)
- Évènements manquants (jeton « mort »)

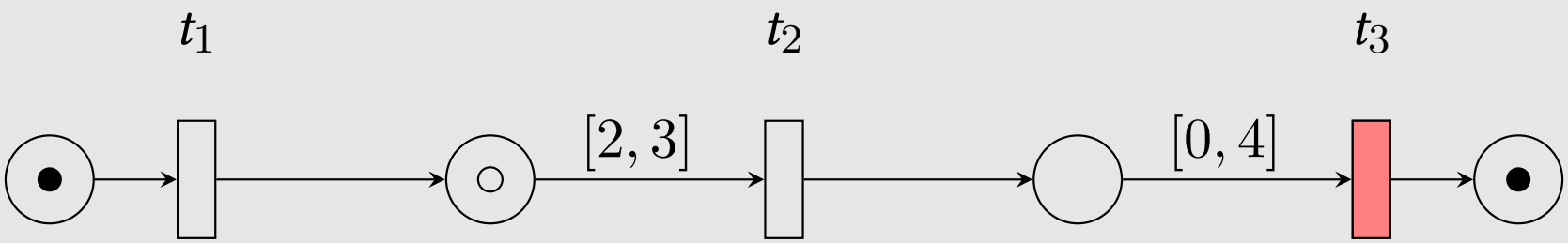
Les jetons négatifs

Pour évaluer la correction d'une exécution il faut :

- Ajouter les jetons dans les places (positifs ou négatifs) : franchissement des transitions
- Comparer les dates d'apparition entre jetons négatifs et positifs

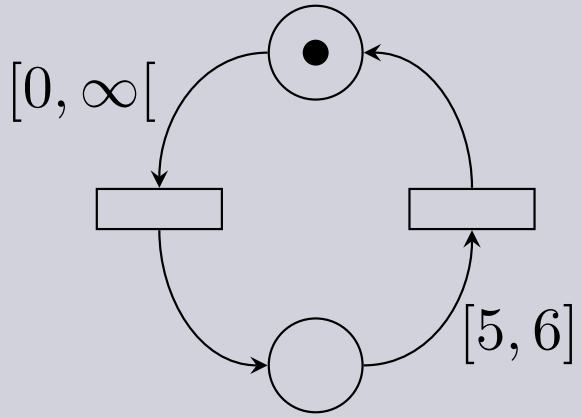
Deux types d'erreurs

- Erreurs temporelles (transitions rouges)
- Évènements manquants (jeton « mort »)



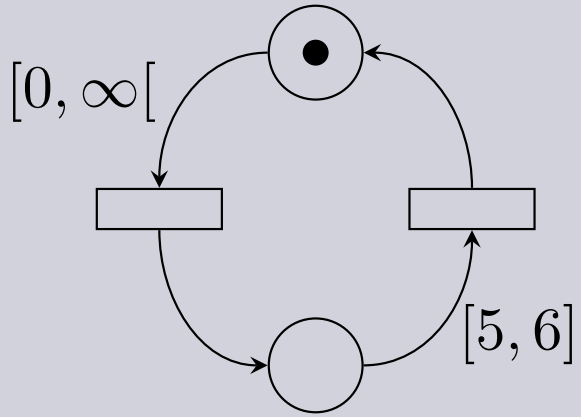
Jetons et identifiants

La problème



Jetons et identifiants

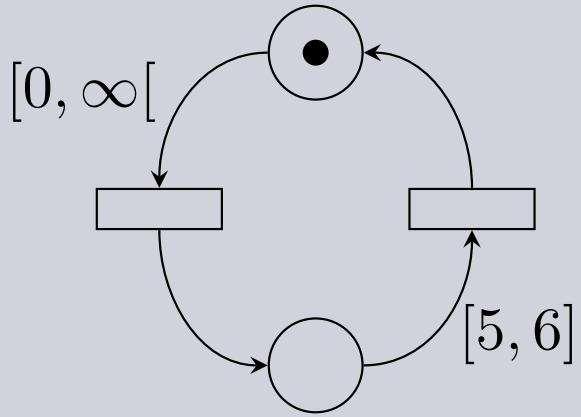
La problème



- $(t_1, 10)$ OK

Jetons et identifiants

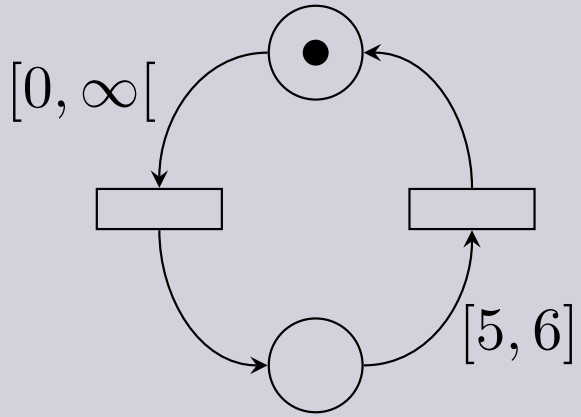
La problème



- $(t_1, 10)$ OK
- $(t_2, 12)$ ERREUR

Jetons et identifiants

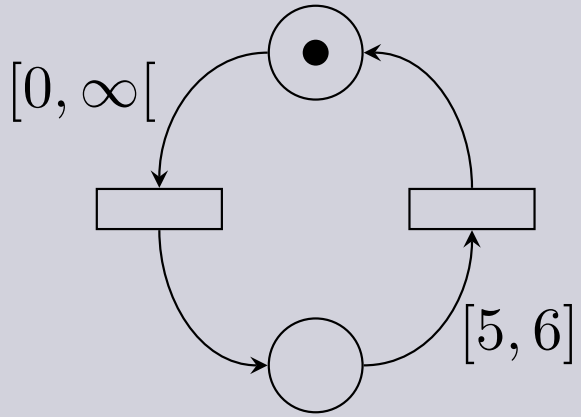
La problème



- $(t_1, 10)$ OK
- $(t_2, 12)$ ERREUR
- $(t_1, 14)$ OK

Jetons et identifiants

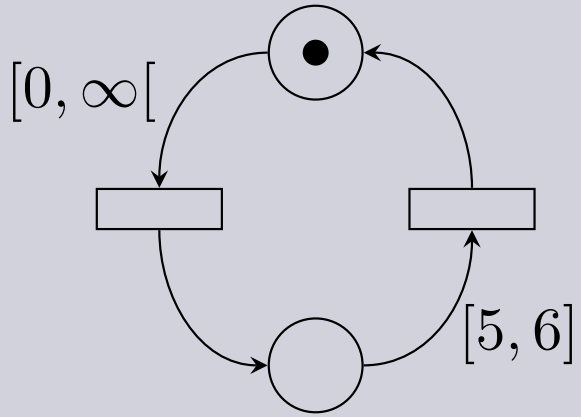
La problème



- $(t_1, 10)$ OK
- $(t_2, 12)$ ERREUR
- $(t_1, 14)$ OK
- $(t_2, 16)$ ERREUR

Jetons et identifiants

La problème

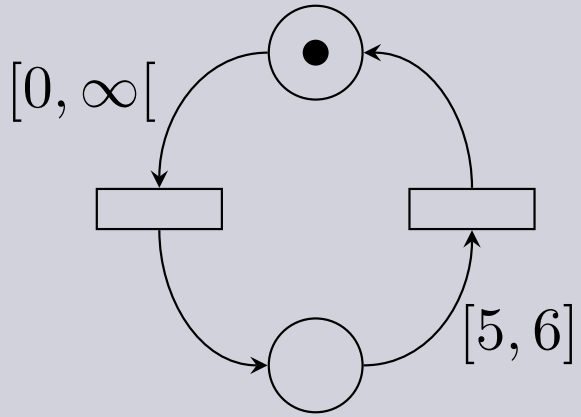


- $(t_1, 10)$ OK
- $(t_2, 12)$ ERREUR
- $(t_1, 14)$ OK
- $(t_2, 16)$ ERREUR

- $(t_1, 10)$ OK

Jetons et identifiants

La problème

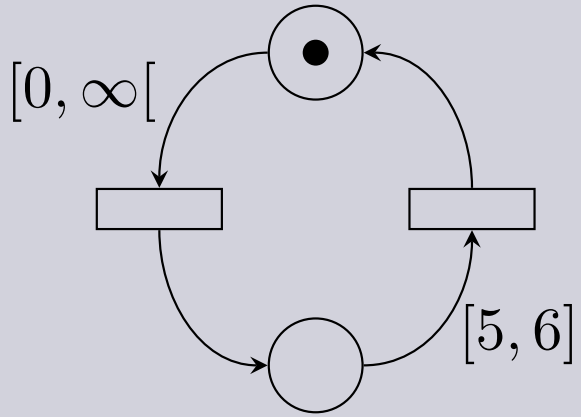


- $(t_1, 10)$ OK
- $(t_2, 12)$ ERREUR
- $(t_1, 14)$ OK
- $(t_2, 16)$ ERREUR

- $(t_1, 10)$ OK
-

Jetons et identifiants

La problème

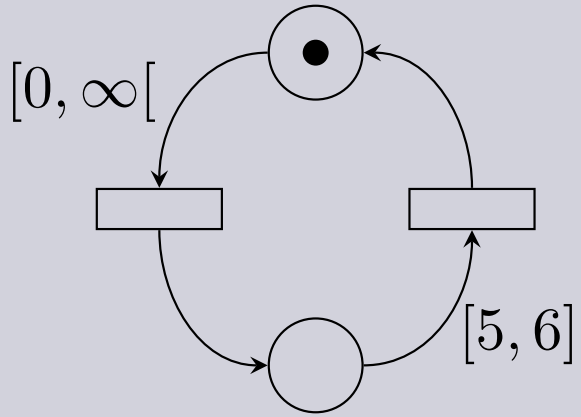


- $(t_1, 10)$ OK
- $(t_2, 12)$ ERREUR
- $(t_1, 14)$ OK
- $(t_2, 16)$ ERREUR

- $(t_1, 10)$ OK
-
-

Jetons et identifiants

La problème

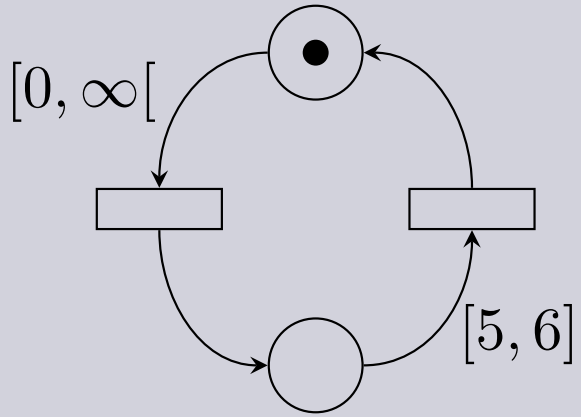


- $(t_1, 10)$ OK
- $(t_2, 12)$ ERREUR
- $(t_1, 14)$ OK
- $(t_2, 16)$ ERREUR

- $(t_1, 10)$ OK
-
-
- $(t_2, 16)$ OK

Jetons et identifiants

La problème



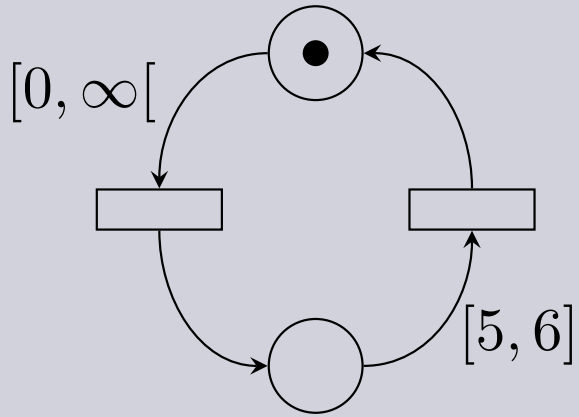
- $(t_1, 10)$ OK
- $(t_2, 12)$ ERREUR
- $(t_1, 14)$ OK
- $(t_2, 16)$ ERREUR

- $(t_1, 10)$ OK
-
-
- $(t_2, 16)$ OK

Notre moniteur ne détecte plus l'erreur !

Jetons et identifiants

La solution

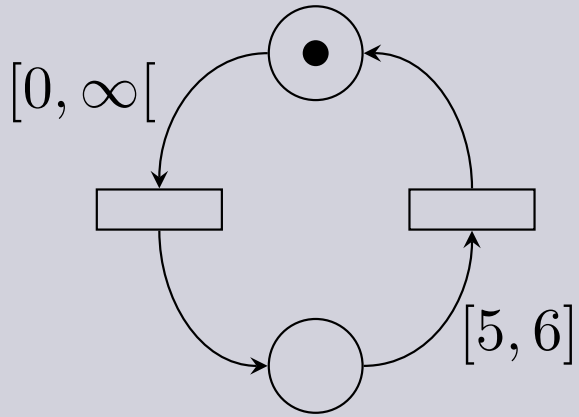


- $(t_1, 10, \{(p_1, 0), (p_2, 0)\})$ OK
- $(t_2, 12, \{(p_2, 0), (p_1, 1)\})$ ERREUR
- $(t_1, 14, \{(p_1, 1), (p_2, 1)\})$ OK
- $(t_2, 16, \{(p_2, 1), (p_1, 2)\})$ ERREUR

On ajoute à chaque jeton un identifiant qui permet de résoudre les cas d'ambiguïtés.

Jetons et identifiants

La solution



- $(t_1, 10, \{(p_1, 0), (p_2, 0)\})$ OK
- $(t_2, 12, \{(p_2, 0), (p_1, 1)\})$ ERREUR
- $(t_1, 14, \{(p_1, 1), (p_2, 1)\})$ OK
- $(t_2, 16, \{(p_2, 1), (p_1, 2)\})$ ERREUR

On ajoute à chaque jeton un identifiant qui permet de résoudre les cas d'ambiguïtés.

Notre moniteur détecte à nouveau l'erreur !

Plan

1. La supervision : qu'est-ce ? pourquoi ? comment ?
2. Une nouvelle sémantique
3. Le protocole distribué
4. Conclusion

Le protocole distribué

Description

Beaucoup de *threads* :

- un *thread* par place
- un *thread* par transition

Le protocole distribué

Description

Beaucoup de *threads* :

- un *thread* par place
- un *thread* par transition

Concrètement :

Le protocole distribué

Description

Beaucoup de *threads* :

- un *thread* par place
- un *thread* par transition

Concrètement :

- les évènements sont reçus par les *threads* des transitions ;

Le protocole distribué

Description

Beaucoup de *threads* :

- un *thread* par place
- un *thread* par transition

Concrètement :

- les évènements sont reçus par les *threads* des transitions ;
- les threads des transitions envoient les jetons (positifs ou negatifs) à ceux des places ;

Le protocole distribué

Description

Beaucoup de *threads* :

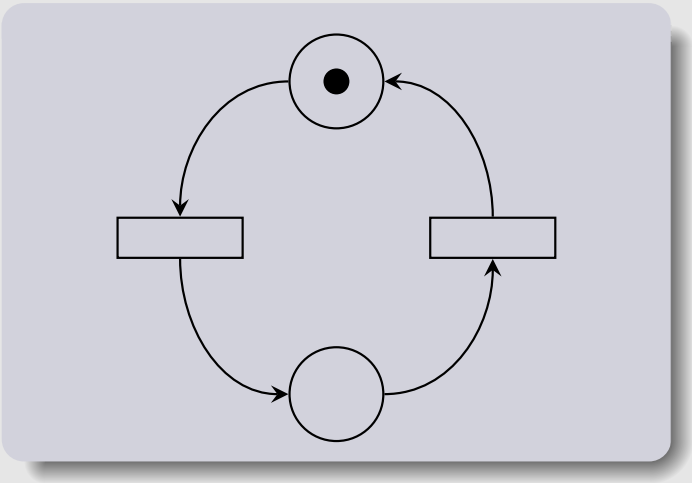
- un *thread* par place
- un *thread* par transition

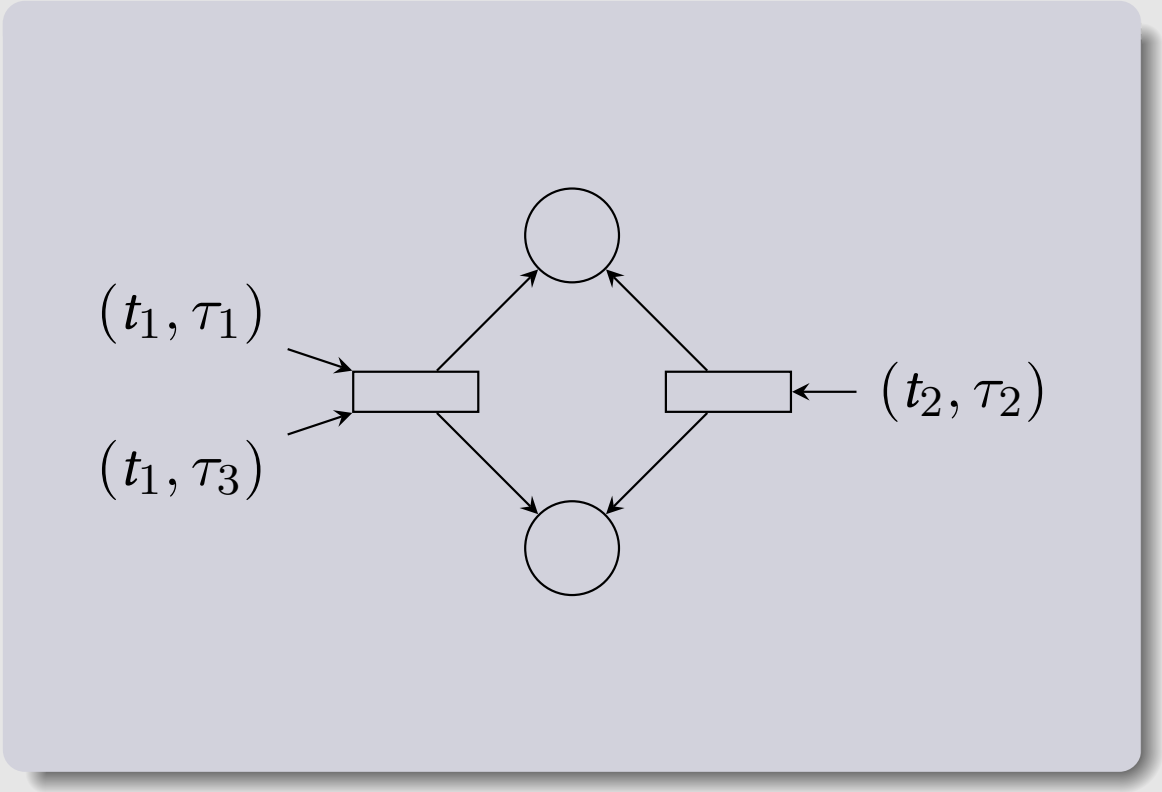
Concrètement :

- les évènements sont reçus par les *threads* des transitions ;
- les threads des transitions envoient les jetons (positifs ou negatifs) à ceux des places ;
- les places comparent le dates des jetons et détectent les erreur (erreurs temporelles ou évènements manquants).

Le protocole distribué

En image

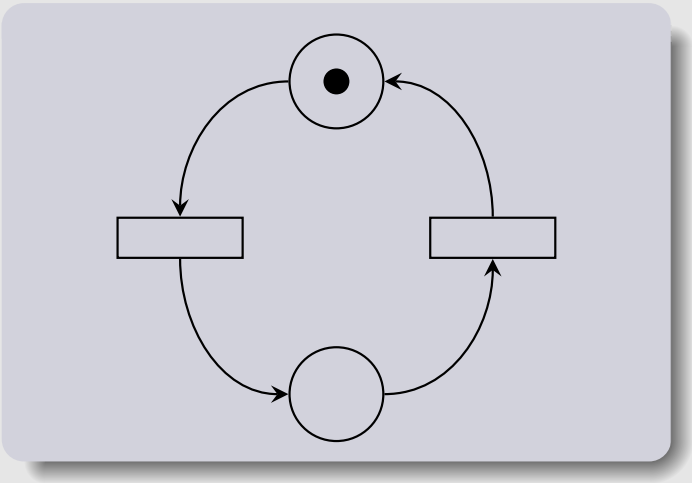


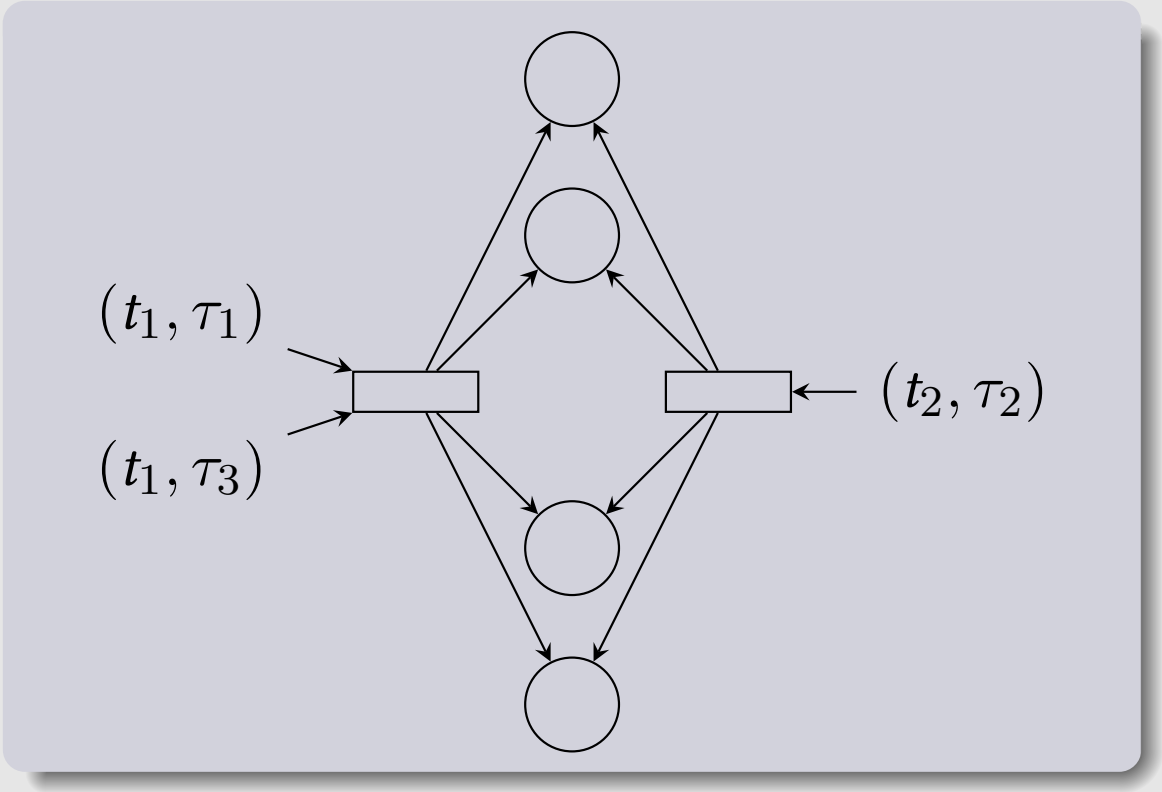


- Les threads peuvent être placés n'importe où.
- Ils sont facilement réplicables.

Le protocole distribué

En image





- Les threads peuvent être placés n'importe où.
- Ils sont facilement réplicables.

Implémentation et performance

- On a écrit un moniteur appelé « Minotor » en Erlang

Implémentation et performance

- On a écrit un moniteur appelé « Minotor » en Erlang
- il fonctionne !

Implémentation et performance

- On a écrit un moniteur appelé « Minotor » en Erlang
- il fonctionne !
- il passe très bien à l'échelle (plus de 2 000 000 threads sur une seule machine)

Implémentation et performance

- On a écrit un moniteur appelé « Minotor » en Erlang
- il fonctionne !
- il passe très bien à l'échelle (plus de 2 000 000 threads sur une seule machine)
- il permet de créer les places et les transitions de manière dynamique.

Plan

1. La supervision : qu'est-ce ? pourquoi ? comment ?
2. Une nouvelle sémantique
3. Le protocole distribué
4. Conclusion

Conclusions

et travaux en cours

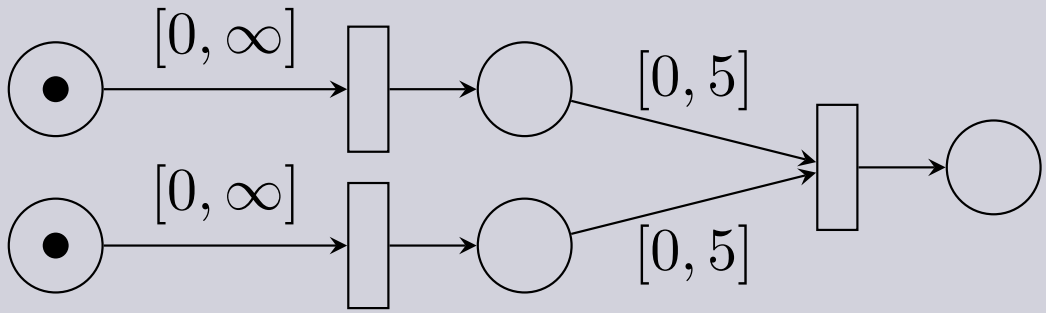
- Exécution distribuée de réseau de Petri qui ne nécessite pas une observation parfaite

Conclusions

et travaux en cours

- Exécution distribuée de réseau de Petri qui ne nécessite pas une observation parfaite

- **Transitions logiques** : pour augmenter l'expressivité, on peut ajouter des transitions qui ne correspondent à aucun évènement

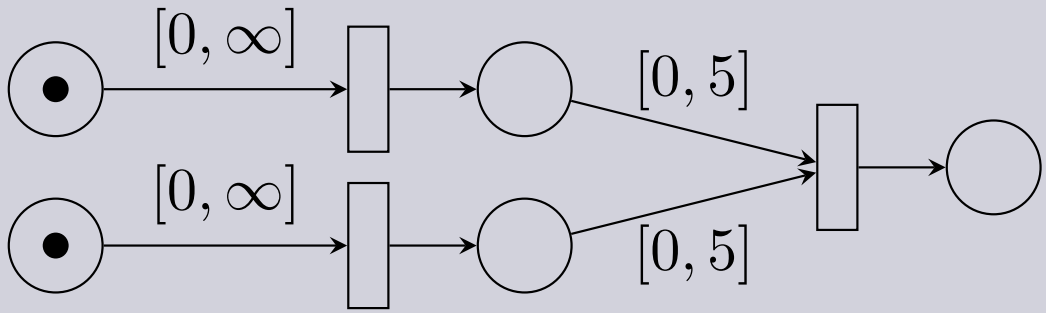


Conclusions

et travaux en cours

- Exécution distribuée de réseau de Petri qui ne nécessite pas une observation parfaite

- **Transitions logiques** : pour augmenter l'expressivité, on peut ajouter des transitions qui ne correspondent à aucun évènement



- **Transitions contextuelles** : à chaque évènement peut correspondre plusieurs transitions. (Nord, Sud, Est, ...).
  Introduction des probabilités pour gérer les évènements manquants.

Merci pour votre attention

Et maintenant, si vous avez des...