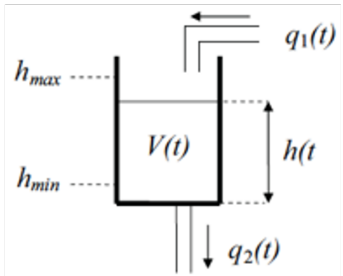
Le temps et les systèmes à événements discrets

Formation sur les Systèmes à Événements Discrets (SED)

2e édition
Mars 2025
Nantes

Introduction



Objet du monde réel

Systèmes Dynamiques

Abstraction

Point de vue

Systèmes à Avancement Temporel (SAT)

Systèmes Hybrides

Systèmes à Événements Discrets (SED)

SAT à temps continu

SAT à temps discret

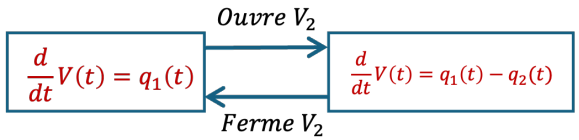
SED Logiques

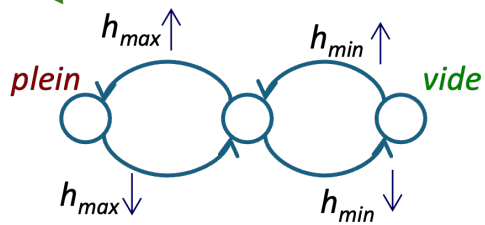
SED Temporisés

Modélisation

$\frac{d}{dt}V(t) = q_1(t) - q_2(t)$

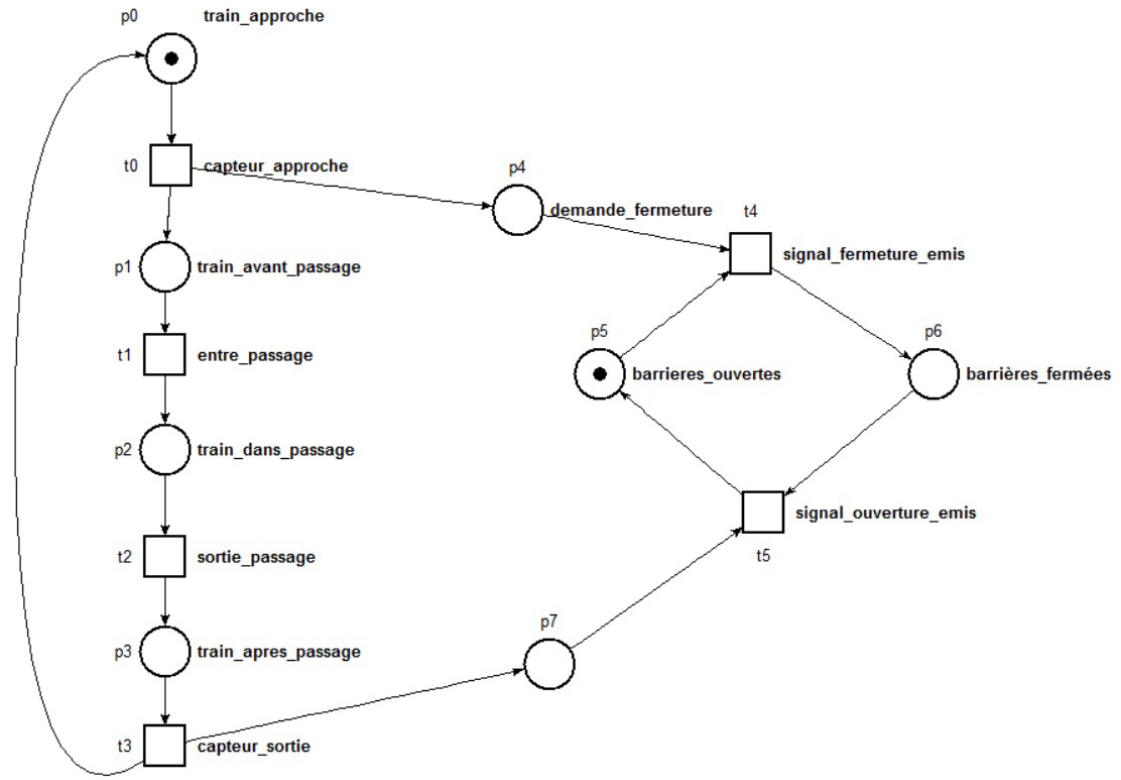
$V(k+1) - V(k) = T_e q_1(k) - T_e q_2(k)$





Introduction : Réseau ferroviaire

(voir formation sur les SED, 1ère édition, Nancy, Janvier 2024)

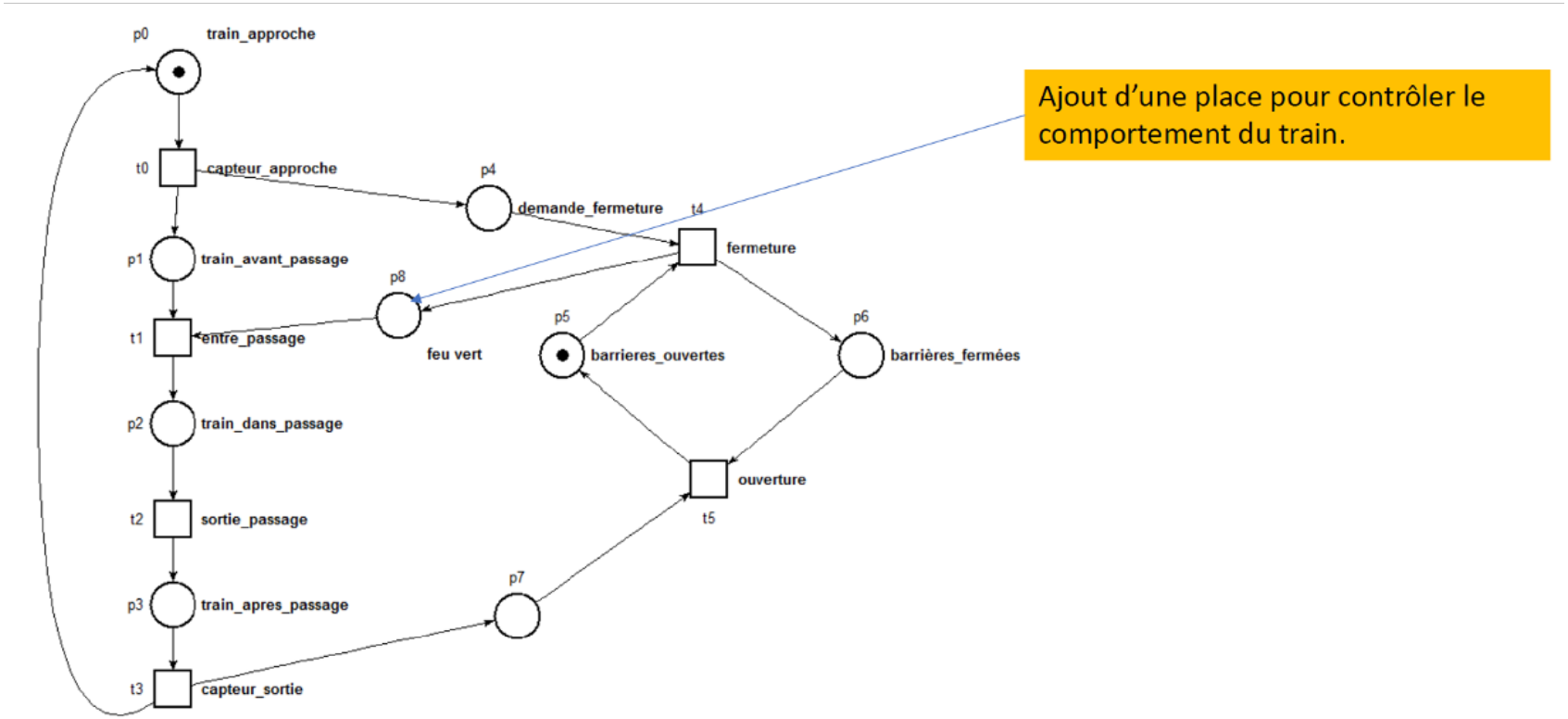


$\Longrightarrow$ Le réseau de Petri (autonome) est non borné.

En effet, on peut itérer les passages de trains (séquence $t_0 t_1 t_2 t_3$) sans commander la fermeture et l'ouverture de la barrière. Il va y avoir une accumulation de jetons dans les places p_4 et p_7 dont le marquage peut tendre à l'infini.

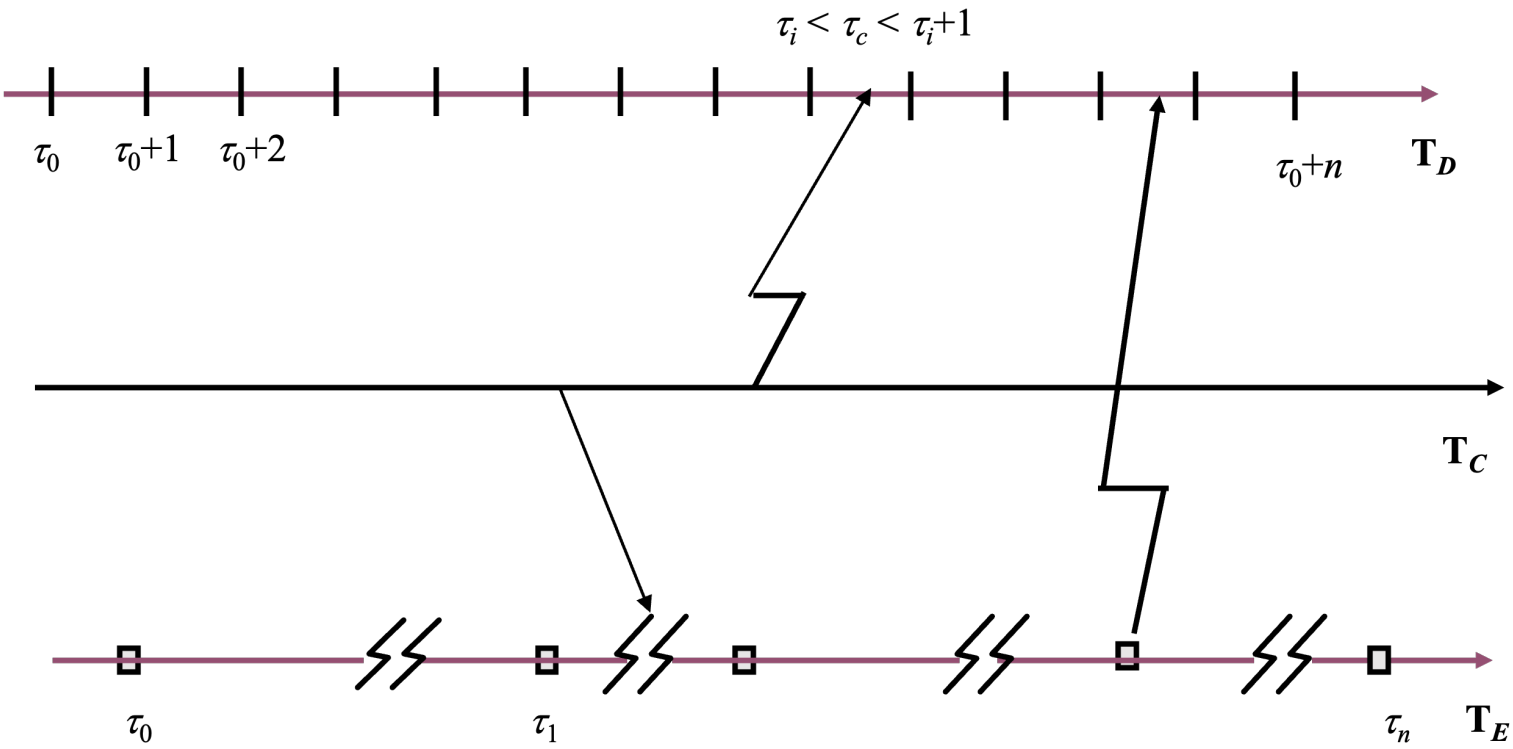
Introduction : Réseau ferroviaire

Une solution :



Une autre solution : jouer avec le temps !

Introduction

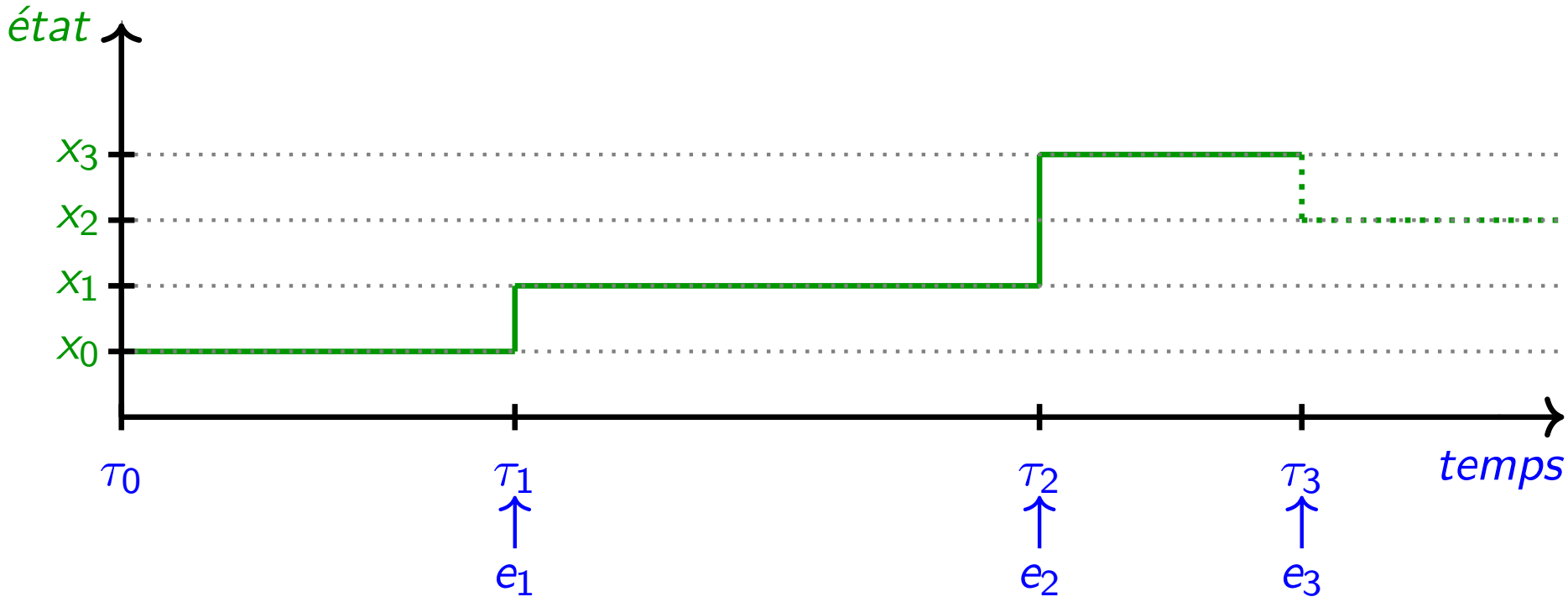


$\Longrightarrow$ Les modèles ont des perceptions différentes du temps.

$\Longrightarrow$ Les variables traitées appartiennent à des espaces mathématiques différents.

Introduction

Dans les systèmes à événements discrets, la trajectoire d'une variable d'état est constante par morceaux.



Plan

Objectif de ce cours

Attirer l'attention sur des subtilités de la prise en compte du temps et poser des bases pour les modèles SED temporisés.

Vers une description du comportement temporisé des SED

- Une **trajectoire d'évolution d'un SED** ne peut plus simplement se décrire par une séquence d'événements $(e_1, e_2, \ldots)$ ou une séquence d'états $(x_1, x_2, \ldots)$

- Soit $\tau_k, k = 1, 2, ...$, l'instant d'occurrence du k-ième événement (et donc de la k-ième transition d'état) avec τ_0 l'instant initial donné. Une trajectoire temporisée peut être décrite par

$((\mathbf{e_1}, \tau_1), (\mathbf{e_2}, \tau_2), ...)$ **séquence temporisée d'événements**

$((\mathbf{x_1}, \tau_1), (\mathbf{x_2}, \tau_2), ...)$ **séquence temporisée d'états**

Il y a lieu de **décrire les contraintes temporelles associées aux occurrences successives d'événements**
$\Longrightarrow$ **structure d'horloges** complémentant le modèle non temporisé (RdP, automate,...)

Struture d'horloges

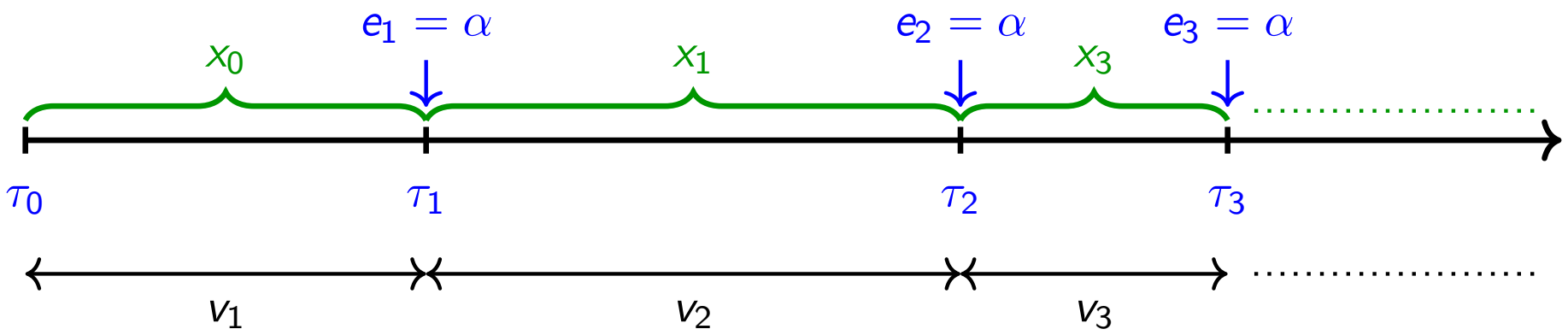
Principe

- A l'instant τ_{k-1}, supposons qu'un événement e_k est *activé* ou *validé*.
- Une ***horloge*** (se comportant comme un chronomètre à rebours) associée à cet événement est réglée à la valeur v_k reflétant la *durée d'activation avant que l'événement puisse survenir*.
- Le compte à rebours de l'horloge est immédiatement démarré et l'événement reste *activé* pendant le décompte.
- L'horloge atteint la valeur 0 à $\tau_k = \tau_{k-1} + v_k$, et à cet instant, l'événement peut/doit survenir.

Exemple élémentaire avec un seul événement

Un seul événement α activé depuis tout état atteint et

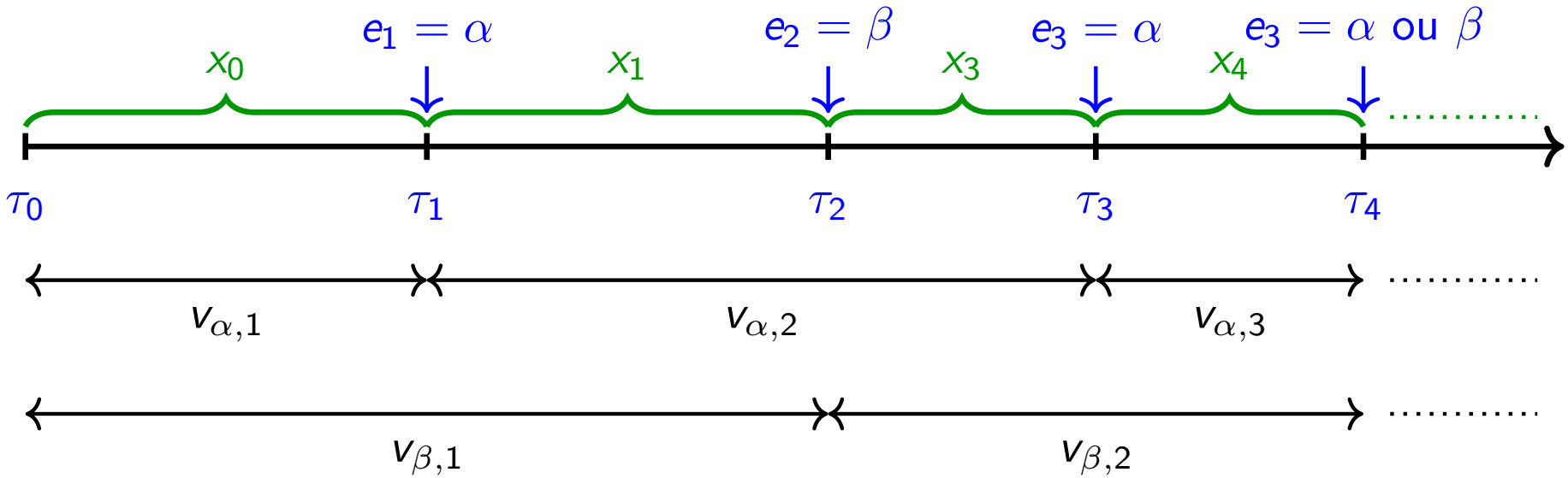
$$\mathbf{V} = \{v_1, v_2, v_3, \ldots\}.$$



Exemple élémentaire avec deux événements "permanents"

Deux événements α et β activés depuis tout état atteint et

$$\mathbf{V}_\alpha = \{v_{\alpha,1}, v_{\alpha,2}, v_{\alpha,3}, \ldots\} \quad , \quad \mathbf{V}_\beta = \{v_{\beta,1}, v_{\beta,2}, v_{\beta,3}, \ldots\}.$$

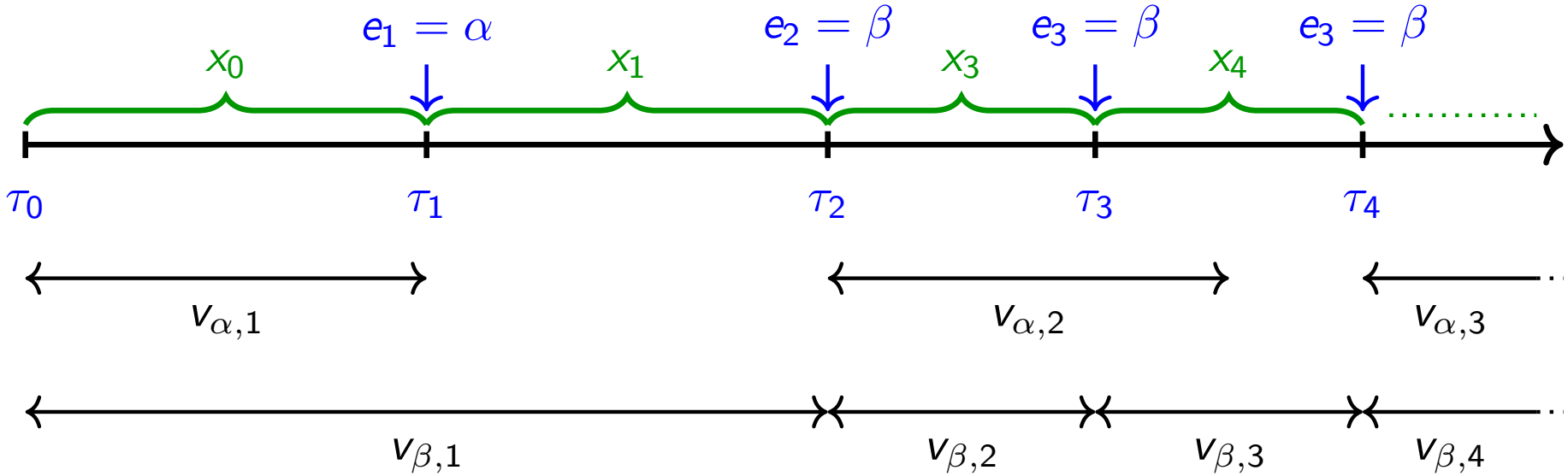


- On considère ici que l'événement qui survient est celui dont l'horloge atteint 0 en premier. En cas de simultanéité, il faut définir une règle pour lever l'indéterminisme.
- Quand un événement survient, son horloge est mise à jour avec la valeur suivante dans la liste.

Exemple avec deux événements mais "non permanents"

Deux événements α et β, $\mathbf{V}_\alpha = \{v_{\alpha,1}, v_{\alpha,2}, \ldots\}$, $\mathbf{V}_\beta = \{v_{\beta,1}, v_{\beta,2}, \ldots\}$:

état	x_0	x_1	x_2	x_3	x_4	$\ldots$
évenement(s) activé(s)	α, β	β	α, β	β	α, β	$\ldots$



- Entre τ_1 et τ_2, α n'est pas activé car x_1 est atteint.
- A τ_3, α est désactivé et sa valeur d'horologe courante est mise au rebus (une nouvelle valeur d'horloge $v_{\alpha,3}$ est considérée à τ_4).

Quelques règles à retenir des exemples

(i) Pour déterminer quel événement va survenir, on compare les valeurs d'horloge de tous les événements activés

(ii) Si un événement survient et qu'un autre événement γ était activé et le reste dans le nouvel état, alors l'horloge associée à γ poursuit son décompte (sans mise à jour)

(iii) Un événement γ est nouvellement activé quand

- γ vient de survenir et il est de nouveau activé dans le nouvel état
- un autre événement vient de survenir, γ était désactivé et il devient activé dans le nouvel état atteint

et alors son horloge est remise à jour avec la valeur suivante dans $\mathbf{V}_\gamma$

(iv) Un événement est désactivé quand un événement survient et qu'il n'est plus activé dans le nouvel état atteint (sa valeur d'horloge courante est mise au rebus, le décompte entamé est oublié)

Struture d'horloge, en conclusion

- La structure d'horloge (ou struture de temporisation) est le complément apporté à un modèle non temporisé de SED pour décrire son comportement temporisé :

$$\mathbf{V} = \{\mathbf{V}_\alpha : \alpha \in \Sigma\}$$

avec $\mathbf{V}_\alpha = \{v_{\alpha,1}, v_{\alpha,2}, \ldots\}$ la séquence d'horloge pour l'événement α, et Σ l'ensemble des événements.

- $\mathbf{V}$ est complètement spécifié en "entrée" du modèle.

- Plutôt qu'une séquence comme ci-dessus, il va être illustré par la suite que la durée d'activation v_α d'un événement α peut être définie selon l'état atteint dans le modèle.

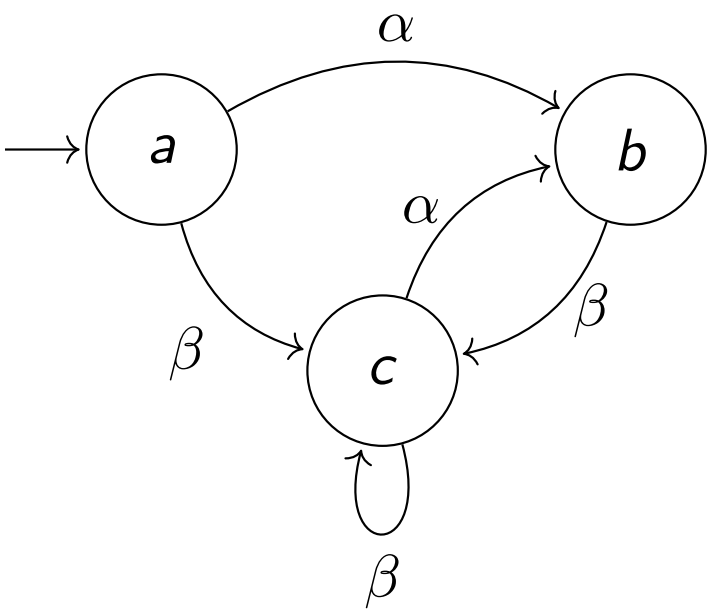
1. Introduction

2. Vers une description du comportement temporisé des SED

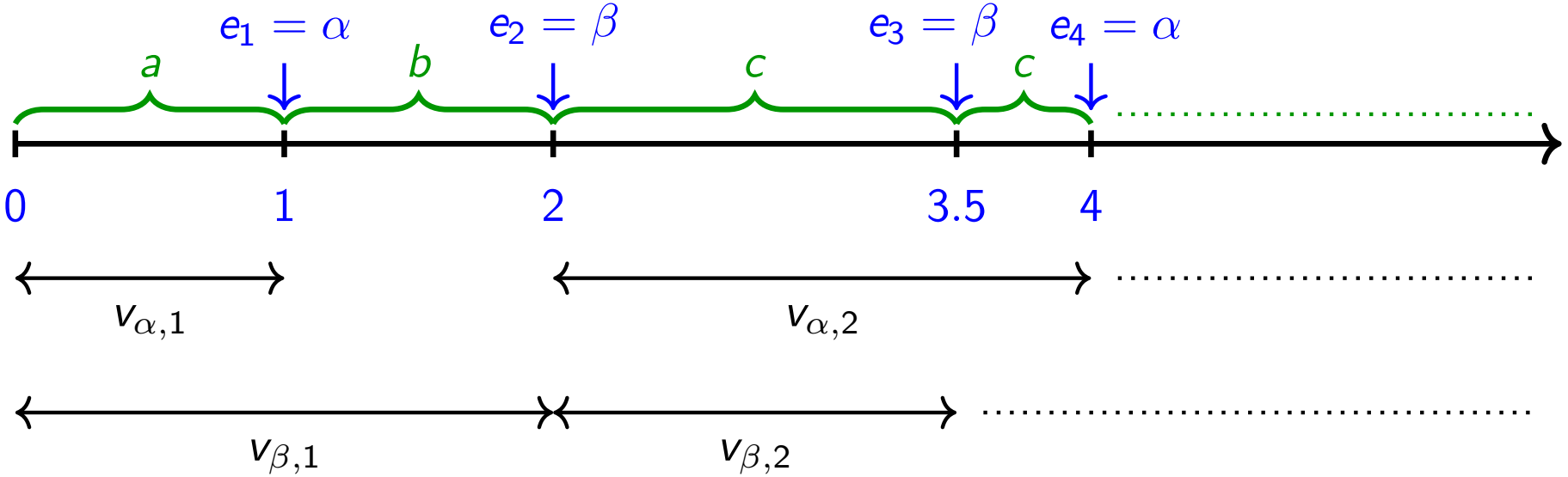
3. Illustrations au travers de plusieurs formalismes
   - Automates temporisés
   - Réseaux de Petri temporisés
   - Systèmes max-plus linéaires

4. Conclusion

Automate temporisé basique

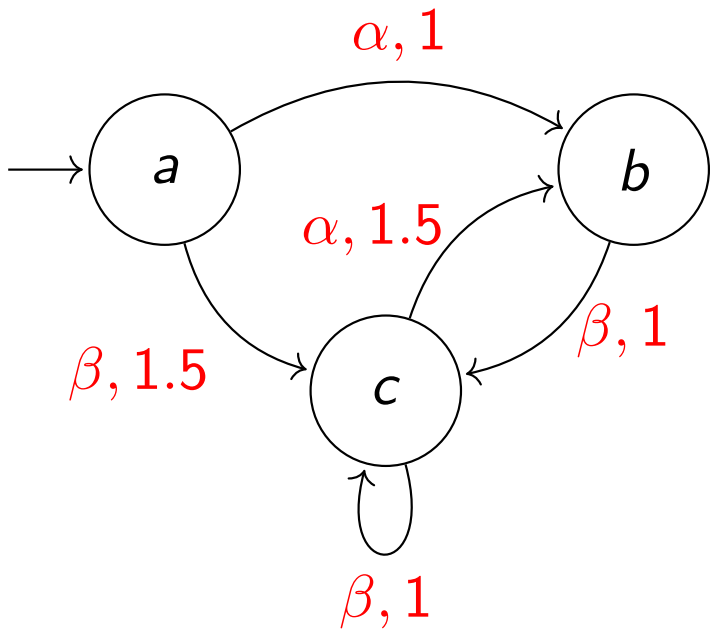


$$\mathbf{V}_\alpha = \{1, 2, 0.5, \ldots\} \ ,$$
$$\mathbf{V}_\beta = \{2, 1.5, 1, \ldots\}$$



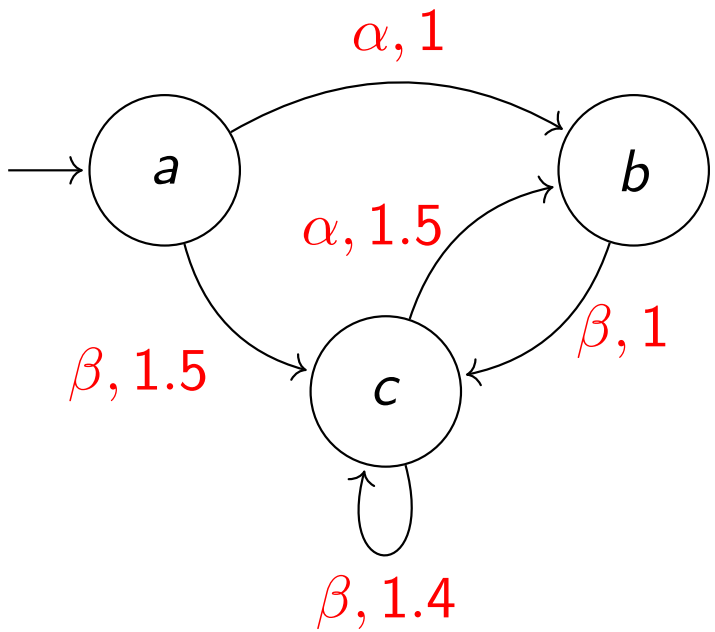
Automate temporisé basique (remanié)

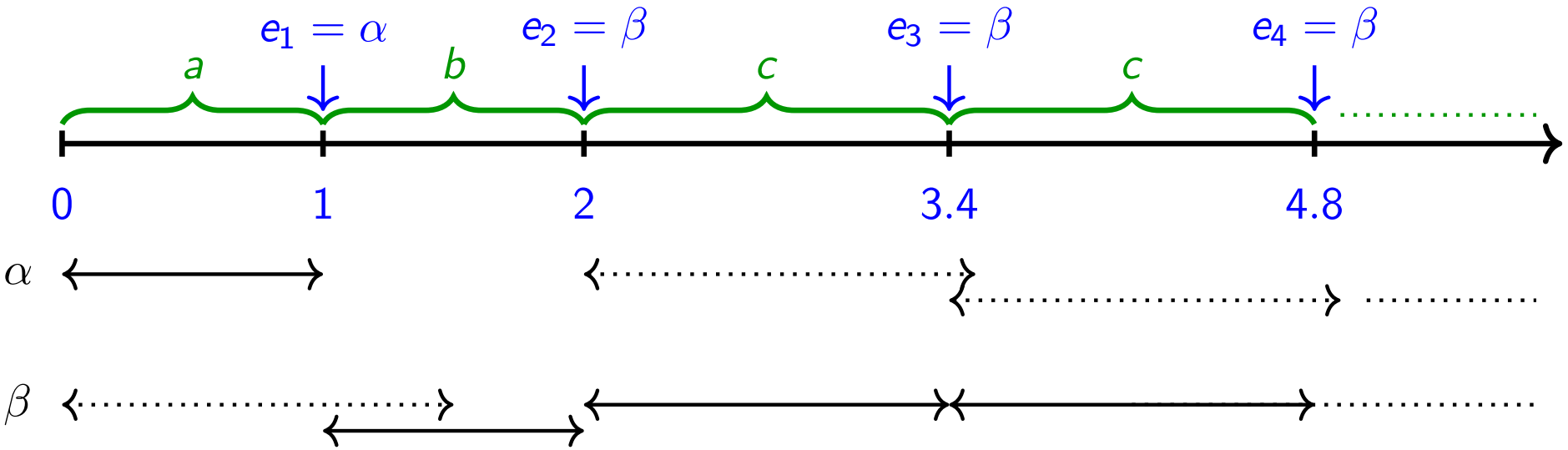
Une alternative est de définir les durées sur les transitions :



Quand un état est atteint, on peut considérer que les événements des transitions de sortie sont des nouvelles instances et qu'ils sont donc nouvellement activés : cela conduit alors à un comportement dynamique différent.

Automate temporisé basique (remanié)





Remarques

- Les automates temporisés présentés jusqu'ici ont été qualifiés de "basiques" car l'appellation "automates temporisés" est généralement utilisée pour désigner des automates avec des gardes (horloges globales qui peuvent être réinitialisées lors de transitions).
- Une alternative peut aussi être de considérer des intervalles comme durées d'activation des événements (modélisant typiquement une incertitude).
- La règle pour la remise à zéro des horloges peut varier selon les études.
- ...

Il existe un nombre important de types "d'automates temporisés" se distinguant principalement par : le nombre et la portée des horloges, leurs fonctionnements (domaine de définition et remise à zéro).

Réseau de Petri temporisé

Les réseaux de Petri temporisés décrivent la logique de changement d'états mais aussi la durée des états ou le "timing" des événements.

Quatre principales approches pour introduire les spécifications temporelles dans les modèles autonomes :

- **Temps associé aux jetons.** Les jetons portent une étiquette temporelle. Cette étiquette indique l'instant pour lesquels les jetons sont disponibles afin de franchir la transition ; cette étiquette peut être incrémentée à chaque franchissement d'une transition.
- **Temps associé aux arcs.** Un délai de transport est associé à chaque arc. Les jetons sont disponibles pour franchir la transition, une fois qu'ils ont atteint cette transition.
- **Temps associé aux places : RdP P-temporisés (TPPN).** Une durée (ou temporisation) d_i est associée à chaque place p_i du réseau.
- **Temps associé aux transitions : RdP T-temporisés (TTPN).** Une durée (ou temporisation) d_j est associée à chaque transition t_j du réseau.

RdP P-temporisé (TPPN)

RdP-P temporisé $= < P, T, Pre, Post, M_0, tempo >$ où *tempo* représente l'application qui associe à chaque place p_i une durée d_i.

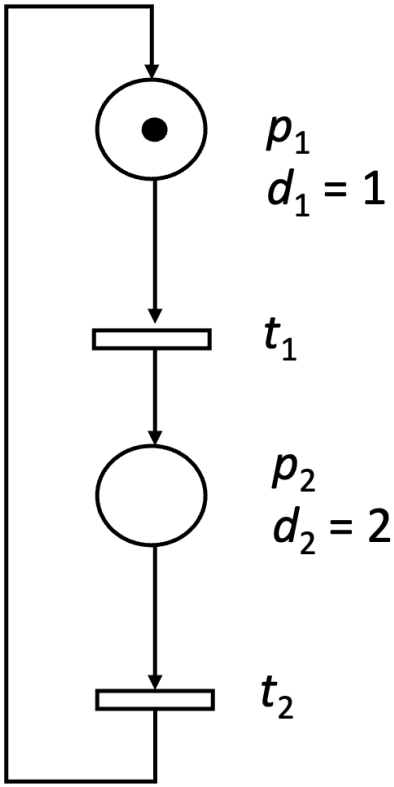
Principe

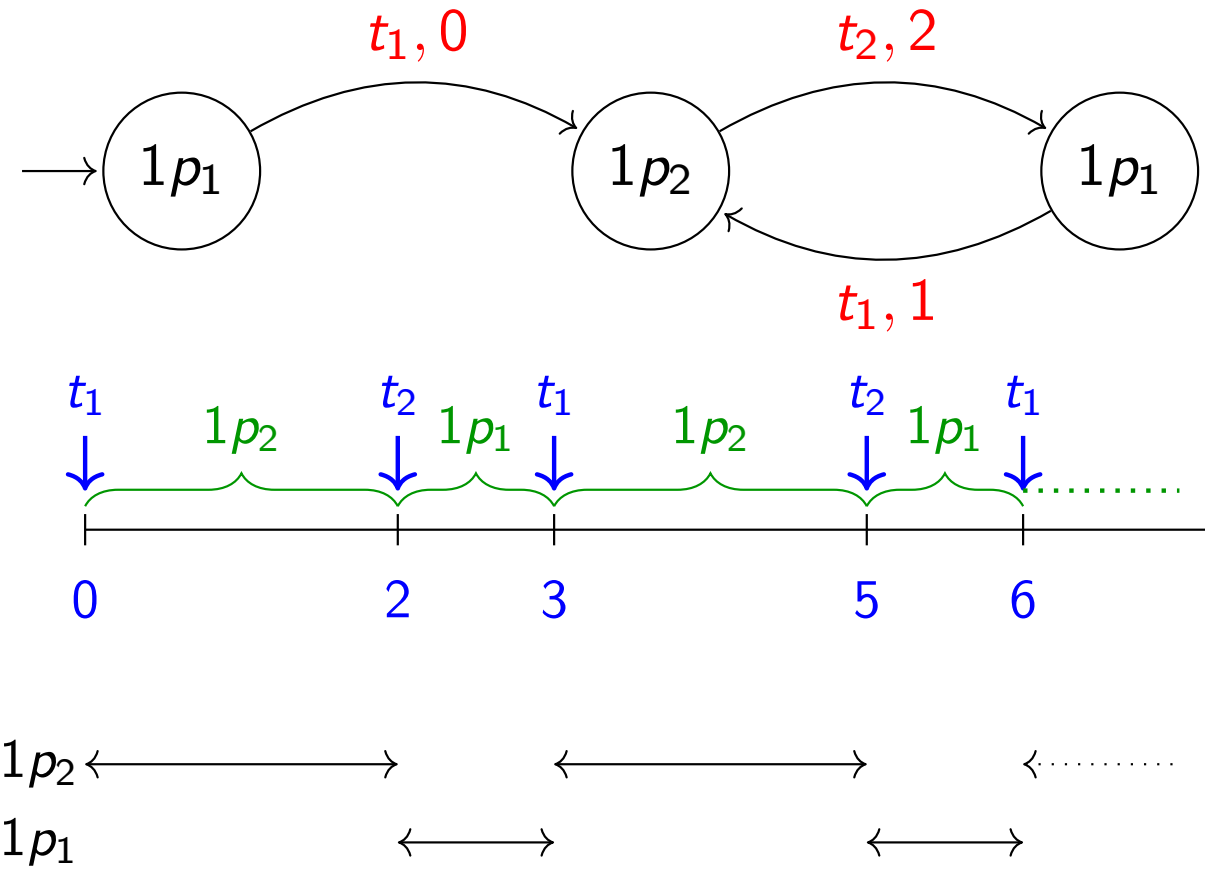
- Les jetons générés dans une place p_i deviennent indisponibles pendant la durée d_i.
- Puis ils deviennent disponibles après la durée d_i pour valider, voire franchir, la transition aval.
- Le marquage est décomposé en marques disponibles et marques indisponibles : $M = M^d + M^i$

Sémantique basée sur :

1. conditions temporelles, puis
2. conditions logiques.

RdP P-temporisé (TPPN) : exemple



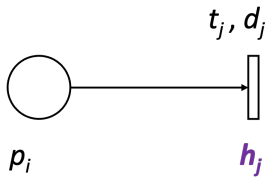


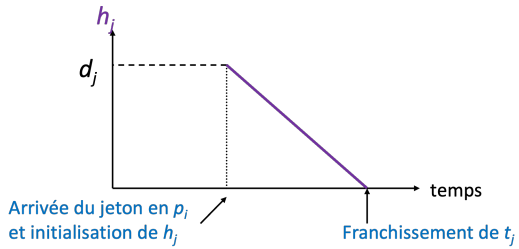
RdP T-temporisé (TTPN)

RdP-T temporisé $= < P, T, Pre, Post, M_0, tempo >$ où $tempo$ représente l'application qui associe à chaque transition t_j une durée d_j.

Principe

- Une transition t_j est validée si $M \geq Pre(\cdot, t_j)$.
- Puis elle est franchie (suivant la politique de franchissement) après la durée d_j : $M + C(\cdot, t_j) = M'$, où $C = Post - Pre$.

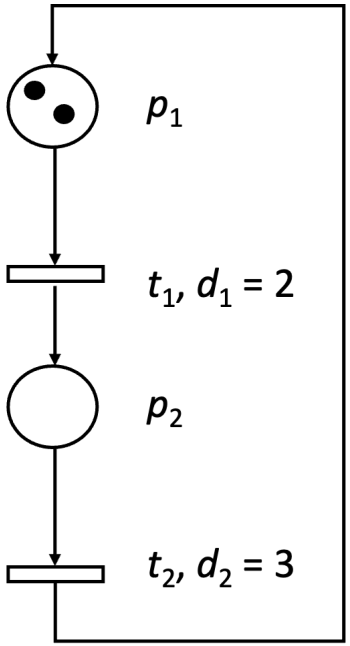




Sémantique basée sur :

1. conditions logiques, puis
2. conditions temporelles.

RdP T-temporisé (TTPN) : exemple



Durées fixes durant l'évolution (i.e., $d_1 = 2$ et $d_2 = 3$)

* Combien de fois la transition est-elle validée ?

RdP T-temporisé (TTPN) : Franchissements simultanés

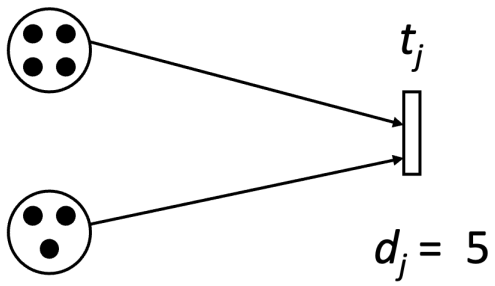
Degré de franchissement

Pour un marquage donné M, le **degré de franchissement** (enabling degree) d'une transition t_j est le nombre de fois que la transition peut être franchie avant de devenir infranchissable, i.e., $e_j(M) = \lfloor \min_{\forall p_i \in {}^\bullet t_j} \frac{M(p_i)}{Pre(p_i,t_j)} \rfloor$.

L'évolution du RdP dépend du nombre de franchissements simultanés autorisés.

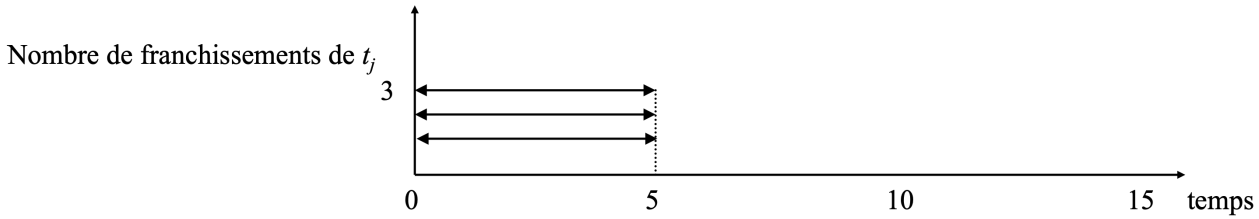
- Un **seul serveur (mono-serveur)** : la transition ne peut être franchie plusieurs fois simultanément. Modélise des processus en série.
- Une **infinité de serveurs** : la transition peut être franchie simultanément un nombre infini de fois (dépendra de son degré de franchissement). Modélise des processus en parallèle.
- Un **nombre limité de serveurs (k-serveurs)** : une transition peut être franchie plusieurs fois (k fois) simultanément. Modélise un degré maximum de parallélisme.

RdP T-temporisé (TTPN) : Franchissements simultanés

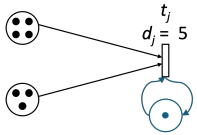


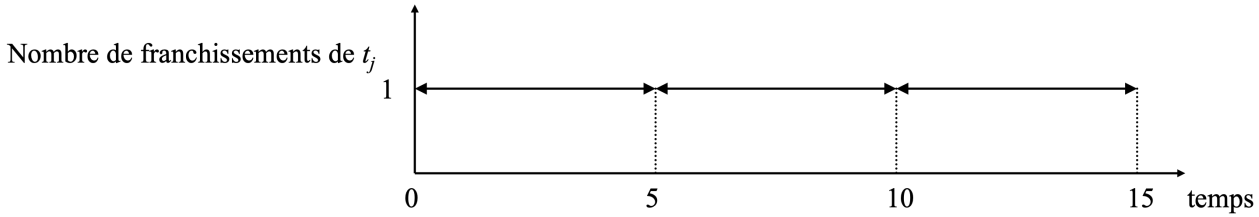
$e_j(M) = 3$

- cas : infinité de serveur

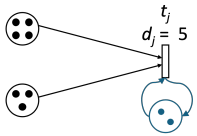


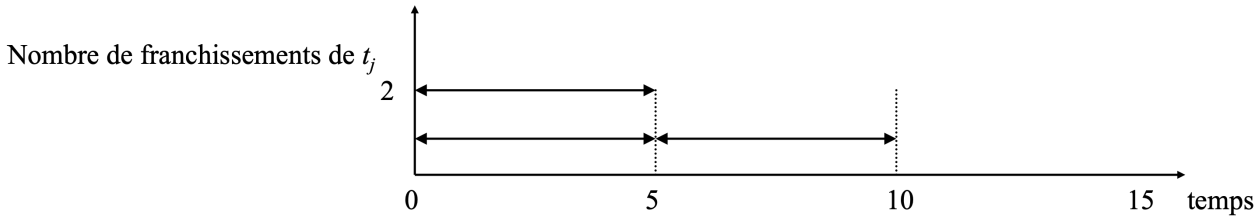
- cas : mono-serveur





- cas : 2-serveurs





RdP T-temporisé (TTPN) : Politiques de franchissement

Comment la transition est franchie ou comment s'affranchir d'un conflit ?

Deux possibilités :

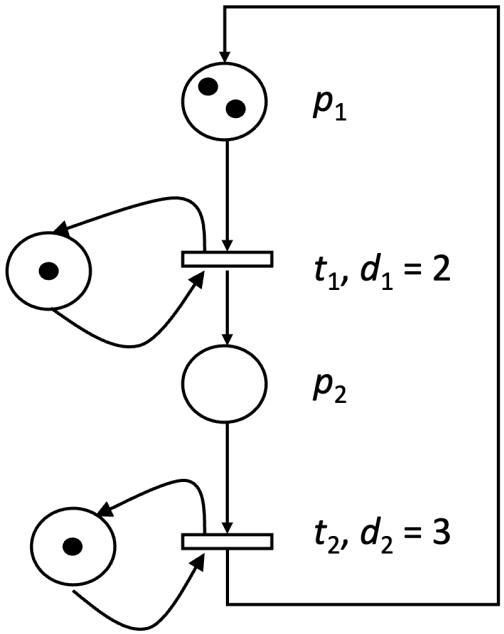
Politiques de franchissement

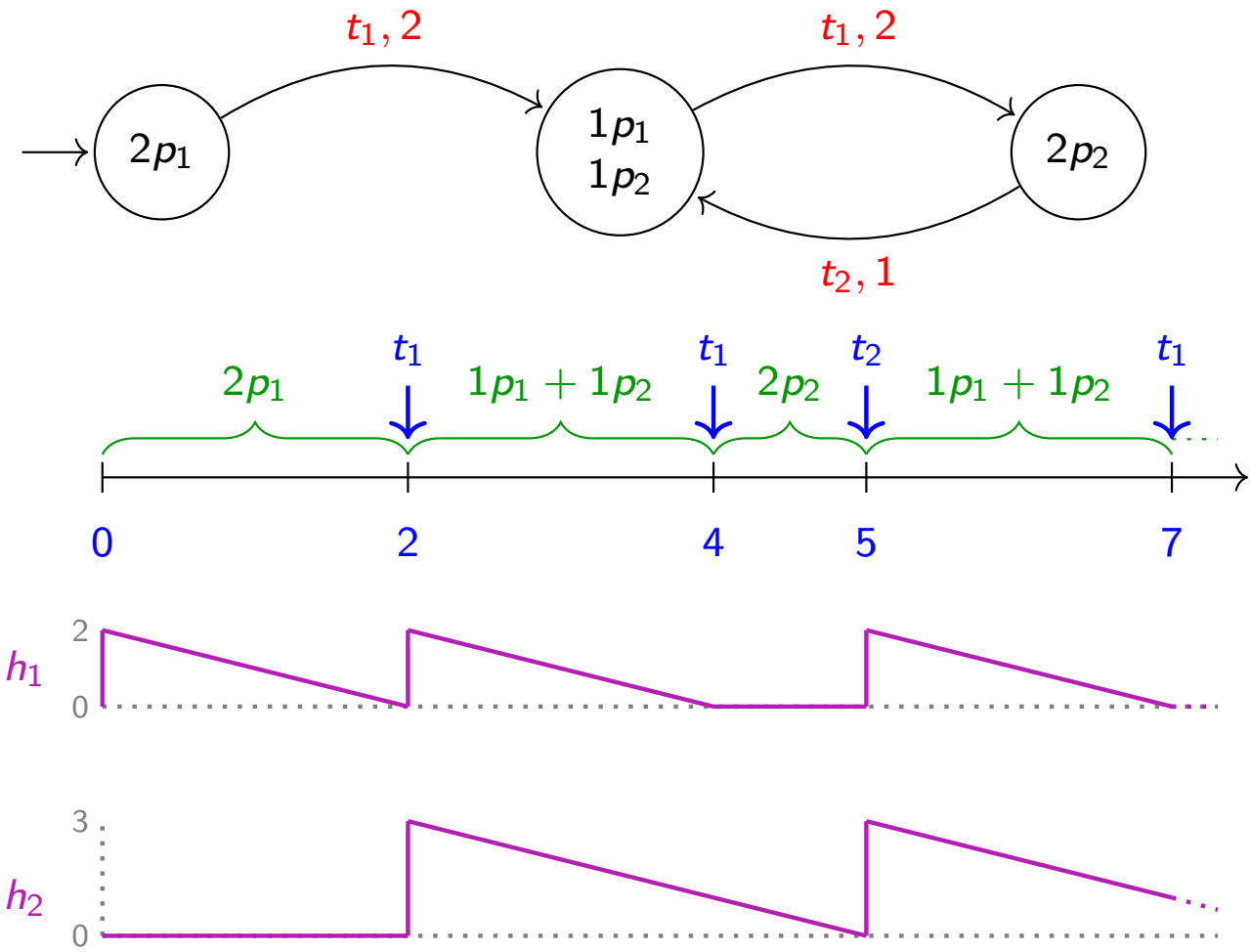
- **Modèle de préselection (Preselection policy)** :
  Lorsqu'une transition t_j est validée, les jetons nécessaires à son franchissement sont réservés et donc indisponibles pour les autres transitions.
  En cas de conflit : des règles prédéterminées permettent de choisir la transition validée : priorité, probabilité, index, etc.

- **Modèle concurrentiel ou modèle de compétition (Race policy)** :
  Lorsqu'une transition t_j est validée, les jetons nécessaires à son franchissement ne sont pas réservés et sont donc aussi disponibles pour la validation d'autres transitions.

RdP T-temporisé (TTPN) : Preselection policy

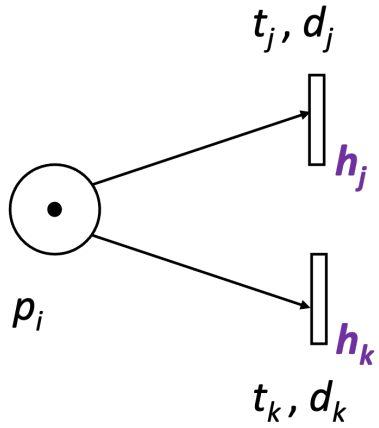
Fonctionnement au plus tôt ou à vitesse maximale : dès qu'une transition est validée, les marques nécessaires à son franchissement sont réservées.





RdP T-temporisé (TTPN) : Race policy

Dans le cas de transitions non persistantes
(i.e., franchissable mais non franchie) :
* Comment gérer la mémoire temporelle (l'échéancier) des transitions ? ou
* Comment considérer les horloges des transitions non franchies après le franchissement d'une transition ? Doit on garder en mémoire l'évolution passée ?



Deux mécanismes de base :

- Les valeurs actuelles des horloges des transitions sont perdues et de nouvelles valeurs sont générées si nécessaire (**restart process**)
- Les horloges des transitions non franchies gardent leurs valeurs et continueront à décroître dès qu'elles seront à nouveau franchissables (**continue process**)

Analyse des RdP temporisés

Comme pour les RdP autonomes, on peut étudier les propriétés qualitatives d'un RdP temporisé.

Il existe deux types de propriétés :

- Les **propriétés qui doivent être vraies pour tous les états** de l'espace d'états atteignables. Ces propriétés se conservent avec l'ajout du temps. Exemple : absence de blocage, bornitude, ...
- Les **propriétés qui doivent être vraies pour certains états**. Ces propriétés ne se conservent pas nécessairement avec l'ajout du temps. Exemple : vivacité, ...

Les invariants de places et de transitions sont identiques car la structure du graphe n'est pas modifiée.

- L'introduction du temps (et/ou des règles de priorité) peut **réduire l'espace d'atteignabilité**.
- Un RdP temporisé est nécessairement borné si le RdP autonome sous-jacent est borné. Par contre, un RdP temporisé peut être borné alors que le RdP autonome correspondant ne l'est pas.

Approche inspirée par la thèorie des signaux et systèmes

Exemple de signal considéré : **dateur**

Le dateur $d(k)$ représente la date de la $k^{ième}$ occurrence de l'événement labellisé par d.

Représentation d'état, fonction de transfert

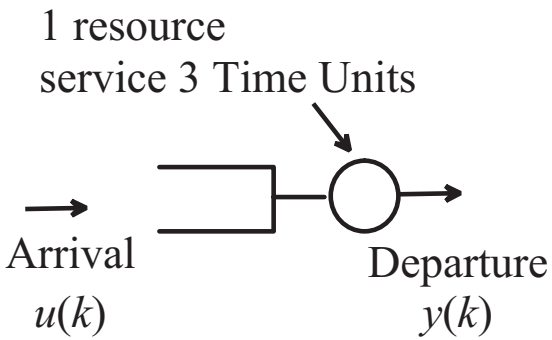
A partir de vecteurs de dateurs associés aux événements (notés x pour les événements internes, u pour les entrées, y pour les sortie), on peut décrire l'évolution dynamique de certaines classes de SED par une représentation d'état

$$\begin{cases} x(k) &= Ax(k-1) \oplus Bu(k) \\ y(k) &= Cx(k) \end{cases},$$

ou (via une tranformée sur les signaux idoine), par une fonction de transfert H en jeu dans la représentation entrée-sortie

$$Y = HU.$$

Exemple : file d'attente

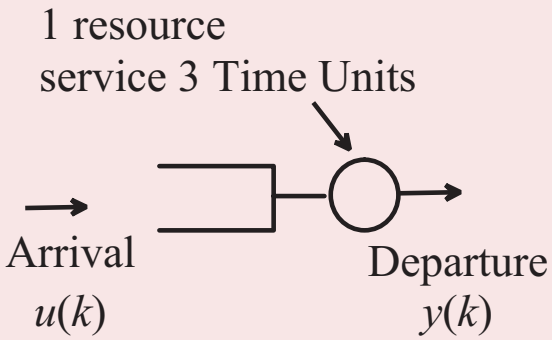


File d'attente avec un seul guichet

- $u(k)$ est la date de la $k^{\text{ième}}$ arrivée
- $y(k)$ est la date du $k^{\text{ième}}$ départ
- $x_1(k)$ est la date de début du $k^{\text{ième}}$ service
- $x_2(k)$ est la date de fin du $k^{\text{ième}}$ service

L'évolution dynamique peut être décrite par

$$\begin{aligned} x_1(k) &= \max(x_2(k-1), u(k)) \\ x_2(k) &= x_1(k) + 3 \\ y(k) &= x_2(k) \end{aligned}$$



Evolution dynamique décrite par

$$\begin{aligned} x_1(k) &= \max(x_2(k-1), u(k)) \\ x_2(k) &= x_1(k) + 3 \\ y(k) &= x_2(k) \end{aligned}$$

On peut ré-écrire la première égalité comme suit

$$x_1(k) = \max(\ -\infty + x_1(k-1)\ ,\ 0 + x_2(k-1)\ ,\ -\infty + x_1(k)\ , -\infty + x_2(k), 0 + u(k))$$

et en notant $\oplus$ l'opération max et $\otimes$ l'additon usuelle,

$$x_1(k) = \ -\infty \otimes x_1(k-1) \oplus 0 \otimes x_2(k-1) \oplus -\infty \otimes x_1(k) \oplus -\infty \otimes x_2(k) \oplus 0 \otimes u(k)$$

L'évolution dynamique peut ainsi être décrite par :

$$\begin{cases} \begin{pmatrix} x_1(k) \\ x_2(k) \end{pmatrix} &= \begin{pmatrix} -\infty & 0 \\ -\infty & -\infty \end{pmatrix} \otimes \begin{pmatrix} x_1(k-1) \\ x_2(k-1) \end{pmatrix} \oplus \begin{pmatrix} -\infty & -\infty \\ 3 & -\infty \end{pmatrix} \otimes \begin{pmatrix} x_1(k) \\ x_2(k) \end{pmatrix} \oplus \begin{pmatrix} 0 \\ -\infty \end{pmatrix} \otimes u(k) \\ y(k) &= \begin{pmatrix} -\infty & 0 \end{pmatrix} \otimes \begin{pmatrix} x_1(k) \\ x_2(k) \end{pmatrix} \end{cases}$$

et on peut en déduire ensuite une **représentation d'état standard mais sur une structure algébrique particulière.**

Théorie des systèmes max-plus linéaires

Apports

Modélisation, analyse, optimisation et commande de SED dont **le comportement temporisés** est principalement régi par l'apparition de **synchronisations**.

En bref

- Nouvelle théorie des systèmes basée sur l'algèbre max-plus (on parle aussi de "dioïdes")
- Apparue au début début des années 80
- Théorie "classique" des systèmes linéaires comme ligne directrice principale
- L'accent est mis sur les questions de performance (quantitative, temporelle) plutôt que sur les questions logiques
- Les résultats fondamentaux sont liés aux graphes d'événements temporisés (une sous-classe des RdP temporisés)

Avantages et inconvénients de la prise en compte du temps

Dans les systèmes à événements temporisés, il existe un compromis fondamental entre décidabilité, expressivité et complexité.

- **Decidabilité** : Un problème est décidable si un algorithme peut toujours donner une réponse correcte en un temps fini (ex. “le système atteint-il un état bloquant ?”).
- **Expressivité** : C’est la capacité d’un modèle à représenter des comportements complexes (ex. parallélisme, synchronisation, contraintes temporelles).
- **Complexité** : Mesure de la difficulté algorithmique pour analyser certaines propriétés du système (ex. temps de calcul pour vérifier qu’un état est atteignable).

Avantages et inconvénients de la prise en compte du temps

Les trois notions sont interdépendantes et influencent directement la modélisation et l'analyse des systèmes.
* Plus un modèle est puissant (expressif), plus il risque de devenir indécidable.
* Un modèle très expressif peut mieux décrire la réalité, mais il risque d'être plus difficile à analyser et rendre l'analyse plus coûteuse en temps et en espace.

Trouver un bon équilibre

Dans les systèmes à événements temporisés (automates, réseaux de Petri, ...) :

- **Décidabilité** : On la garde si on limite l'expressivité (ex. contraintes linéaires, temps fixes, bornes).
- **Expressivité** : On l'augmente en ajoutant des horloges complexes, des temps flottants ou stochastiques, mais on perd souvent la décidabilité.
- **Complexité** : Elle est souvent élevée, donc on la réduit en simplifiant le modèle (nombre d'horloges limité, abstraction du temps).

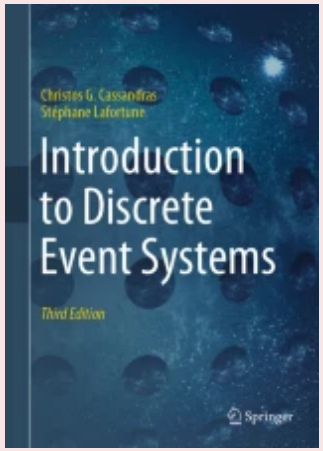
Finalement

Le temps s'écoule mais il ne passe que de temps en temps.

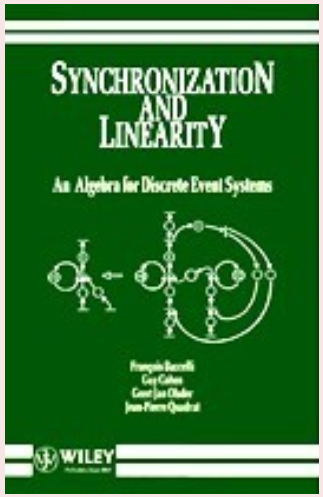
(I.D.)

Références

Largement et librement inspiré de



Introduction to Discrete Event Systems, Christos G. Cassandras , Stéphane Lafortune, 2021 (last edition), **Chapter 5 : Timed and Hybrid Systems**



Synchronization and linearity : An algebra for Discrete Event systems, Baccelli, Cohen, Olsder, Quadrat, 1992 (out of print but downloadable)

Équipe pédagogique

Auteur.rice.s : Isabel demongodin, Sébastien Lahaye.

Intervenant.e.s : Isabel demongodin, Sébastien Lahaye.

Crédits