A Proposal For Relative Time Petri Nets - Foundations, Structure, Properties, Prototypes and Further Works

Marcel Jakobs
Christopher Israel
Heiko Winnebeck

18. Januar 2008

Vorschau

- Analyse und Spezifikation von nebenläufigen Systemen

Vorschau

- Analyse und Spezifikation von nebenläufigen Systemen
- Erreichbarkeitsanalyse (state space exploration)

Vorschau

- Analyse und Spezifikation von nebenläufigen Systemen
- Erreichbarkeitsanalyse (state space exploration)
- zeitgesteuerte Systeme (timed systems)

Vorschau

- Analyse und Spezifikation von nebenläufigen Systemen
- Erreichbarkeitsanalyse (state space exploration)
- zeitgesteuerte Systeme (timed systems)
- Vorstellung eines Petri Netzes basierend auf relativer Zeit

Präsentationsteil 1

Foundations und Structure

Heiko Winnebeck

Inhaltsverzeichnis

Einführung

Petri Netze ...

- sind zur Beschreibung von Systemen mit Nebenläufigkeit

Petri Netze ...

- sind zur Beschreibung von Systemen mit Nebenläufigkeit
- besitzen eine formale mathematische Definition

Petri Netze ...

- sind zur Beschreibung von Systemen mit Nebenläufigkeit
- besitzen eine formale mathematische Definition
- besitzen eine intuitive graphische Darstellung

Petri Netze ...

- sind zur Beschreibung von Systemen mit Nebenläufigkeit
- besitzen eine formale mathematische Definition
- besitzen eine intuitive graphische Darstellung
- ermöglichen durch Analysetechniken das dynamische Verhalten des Modells zu erforschen

Petri Netze ...

- sind zur Beschreibung von Systemen mit Nebenläufigkeit
- besitzen eine formale mathematische Definition
- besitzen eine intuitive graphische Darstellung
- ermöglichen durch Analysetechniken das dynamische Verhalten des Modells zu erforschen
- Entwurf/Modellierung und Steuerung von Industrieanlagen

Problem: state space explosion

state space explosion ...

- durch eine Fülle an Zuständen in nebenläufigen Systemen

Problem: state space explosion

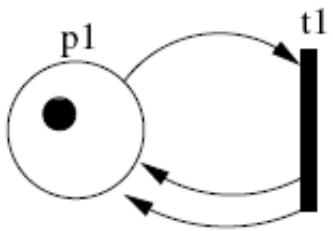
state space explosion ...

- durch eine Fülle an Zuständen in nebenläufigen Systemen
- durch Ergänzung des Netzes mit einer globalen Zeit (timed systems)

Problem: state space explosion

state space explosion ...

- durch eine Fülle an Zuständen in nebenläufigen Systemen
- durch Ergänzung des Netzes mit einer globalen Zeit (timed systems)



Zeit in Petri Netzen

- quantitative Messung der Zeit durch globale Zeitsteuerung

Zeit in Petri Netzen

- quantitative Messung der Zeit durch globale Zeitsteuerung
- Zeitstempel an jedes Token anhängen
  dabei beruht jeder Zeitstempel auf einer absolute globale Zeit,
  welche gleichbleibend ansteigt

Zeit in Petri Netzen

- quantitative Messung der Zeit durch globale Zeitsteuerung
- Zeitstempel an jedes Token anhängen
  dabei beruht jeder Zeitstempel auf einer absolute globale Zeit, welche gleichbleibend ansteigt
- Intervall Semantiken
  Angabe eines Bereiches von Zeiten, der für eine bestimmte Spanne gültig ist

Grundlagen

unbestimmtes Intervall (Intervals of Uncertainty)

- Werte für die das Netz schaltet oder blockt.

unbestimmtes Intervall (Intervals of Uncertainty)

- Werte für die das Netz schaltet oder blockt.

Definition:
Sei $\phi(\lambda, t) \in [0, 1]$ die Wahrscheinlichkeit, dass λ bei t gilt.
Ein unbestimmtes Intervall für λ ist ein Paar von Zeitgrenzen (timing bounds) $[lo_\lambda, hi_\lambda]$.

- $lo_\lambda \leq hi_\lambda$
- $\forall k < lo_\lambda : \phi(\lambda, k) = 0$
- $\forall k \geq hi_\lambda : \phi(\lambda, k) = 1$
- $\forall k' \geq k : \phi(\lambda, k') \geq \phi(\lambda, k)$

multi-sets

multi-set ...

- wird auch bag genannt

multi-sets

multi-set ...

- wird auch bag genannt
- ist eine Funktion einer Menge S über $\mathbb{N}$

multi-sets

multi-set ...

- wird auch bag genannt
- ist eine Funktion einer Menge S über $\mathbb{N}$
- wird repräsentiert durch : $A = \Sigma_{s \in S} A(s)$

multi-sets

multi-set ...

- wird auch bag genannt
- ist eine Funktion einer Menge S über $\mathbb{N}$
- wird repräsentiert durch : $A = \Sigma_{s \in S}\ A(s)$
- S_{MS} ist die Menge aller multi-sets von S

multi-sets

Definition:
Seien $A, B, C \in S_{MS}$

- $s \in A$, wenn $A(s) > 0$

multi-sets

Definition:
Seien $A, B, C \in S_{MS}$

- $s \in A$, wenn $A(s) > 0$
- die Größe von A (geschrieben #A) ist äquivalent zu $A = \Sigma_{s \in S}\ A(s)'s$

multi-sets

Definition:
Seien $A, B, C \in S_{MS}$

- $s \in A$, wenn $A(s) > 0$
- die Größe von A (geschrieben #A) ist äquivalent zu $A = \Sigma_{s \in S}\ A(s)'s$
- A ist eine Untermenge von B ($A \subseteq B$), wenn $\forall s \in S : A(s) \leq B(s)$

multi-sets

Definition:
Seien $A, B, C \in S_{MS}$

- $s \in A$, wenn $A(s) > 0$
- die Größe von A (geschrieben #A) ist äquivalent zu $A = \Sigma_{s \in S}\ A(s)'s$
- A ist eine Untermenge von B ($A \subseteq B$), wenn $\forall s \in S : A(s) \leq B(s)$
- C ist die Summe zweier multi-sets A und B ($C = A + B$), wenn $C(s) = A(s) + B(s)$

multi-sets

Definition:
Seien $A, B, C \in S_{MS}$

- $s \in A$, wenn $A(s) > 0$
- die Größe von A (geschrieben #A) ist äquivalent zu $A = \Sigma_{s \in S}\ A(s)'s$
- A ist eine Untermenge von B ($A \subseteq B$), wenn $\forall s \in S : A(s) \leq B(s)$
- C ist die Summe zweier multi-sets A und B ($C = A + B$), wenn $C(s) = A(s) + B(s)$
- C ist die Differenz zweier multi-sets A und B ($C = A - B$), wenn $B \subseteq A$ *and* $C(s) = A(s) - B(s)$

multi-sets

Definition:
Sei $A \in S_{MS}$, so schreiben wir A als $\{a_i\}_{i=1}^{n}$

- $n = \#A$

multi-sets

Definition:
Sei $A \in S_{MS}$, so schreiben wir A als $\{a_i\}_{i=1}^{n}$

- $n = \#A$
- $\forall 1 \leq i \leq n : a_i \in A$

multi-sets

Definition:
Sei $A \in S_{MS}$, so schreiben wir A als $\{a_i\}_{i=1}^{n}$

- $n = \#A$
- $\forall 1 \leq i \leq n : a_i \in A$
- $\forall s \in S$: *wenn* $sub(s) = \{m \mid a_m = s\}$
  die Indices der Sequenz die s abbilden sind, dann ist
  $\#sub(s) = A(s)$

Petri Netze

Petri Netze sind bipartite gerichtete Graphen.

Petri Netze

Petri Netze sind bipartite gerichtete Graphen.

- die Knoten werden durch *places* und *transitions* dargestellt

Petri Netze

Petri Netze sind bipartite gerichtete Graphen.

- die Knoten werden durch *places* und *transitions* dargestellt
- ein Zustand wird durch *tokens* angezeigt, welche an den *places* angesiedelt sind

Petri Netze

Petri Netze sind bipartite gerichtete Graphen.

- die Knoten werden durch *places* und *transitions* dargestellt
- ein Zustand wird durch *tokens* angezeigt, welche an den *places* angesiedelt sind
- die *transitions* repräsentieren Effekte der *events* in Netzen

Petri Netze

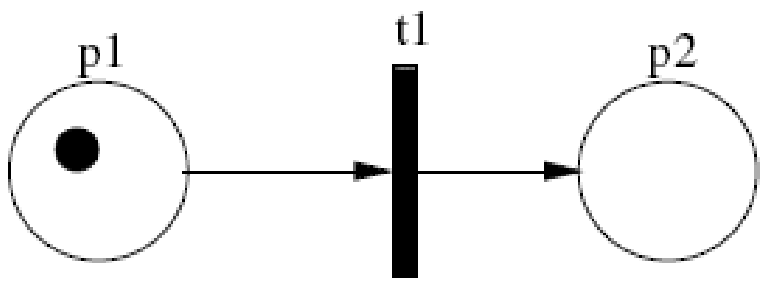
Petri Netze sind bipartite gerichtete Graphen.

- die Knoten werden durch *places* und *transitions* dargestellt
- ein Zustand wird durch *tokens* angezeigt, welche an den *places* angesiedelt sind
- die *transitions* repräsentieren Effekte der *events* in Netzen
- *markings* sind Funktionen, die angeben wie viele *tokens* in den *places* der Netzes beinhaltet sind

Petri Netze

Petri Netze sind bipartite gerichtete Graphen.

- die Knoten werden durch *places* und *transitions* dargestellt
- ein Zustand wird durch *tokens* angezeigt, welche an den *places* angesiedelt sind
- die *transitions* repräsentieren Effekte der *events* in Netzen
- *markings* sind Funktionen, die angeben wie viele *tokens* in den *places* der Netzes beinhaltet sind



Petri Netze

- *arcs*:
  sind die gerichteten Kanten des Netzes

Petri Netze

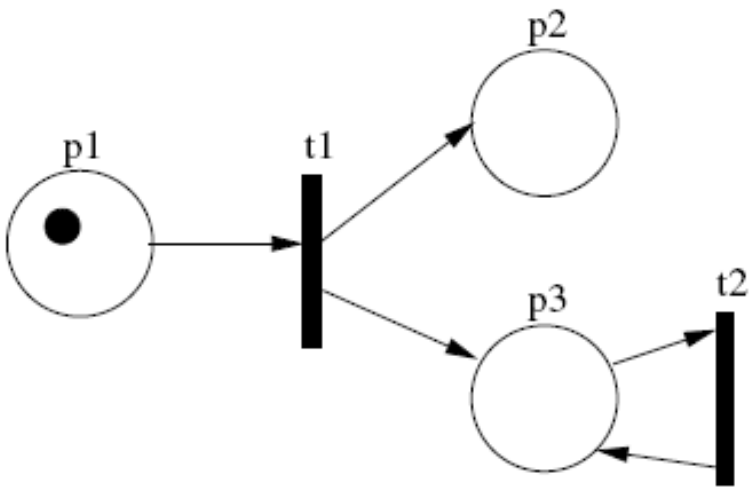
- *arcs*:
  sind die gerichteten Kanten des Netzes
- *input arcs*:
  verbinden *places* und *transitions*

Petri Netze

- *arcs*:
  sind die gerichteten Kanten des Netzes
- *input arcs*:
  verbinden *places* und *transitions*
- *output arcs*:
  verbinden *transitions* und *places*

Petri Netze

- *arcs*:
  sind die gerichteten Kanten des Netzes
- *input arcs*:
  verbinden *places* und *transitions*
- *output arcs*:
  verbinden *transitions* und *places*



Petri Netze

Definition:
Sei (P, T, A, M_0) ein Petri Netz.

- Die Transition $t \in T$ ist für ein *marking* $M \in \mathbb{M}$ aktiv

  geschrieben:
  $M[t\rangle, wenn\ \forall p \in P : M(p) \geq A(p, t)]$

Petri Netze

Definition:
Sei (P, T, A, M_0) ein Petri Netz.

- Die Transition $t \in T$ ist für ein *marking* $M \in \mathbb{M}$ aktiv

  geschrieben:
  $M[t\rangle, wenn\ \forall p \in P : M(p) \geq A(p,t)]$

- Wenn $t \in T$ für ein *marking* $M \in \mathbb{M}$ aktiv ist, feuert t und produziert ein *marking* $M' \in \mathbb{M}$

  geschrieben:
  $M[t\rangle\ M', gdw.\ \forall p \in P : M'(p) = M(p) - A(p,t) + A(t,p)]$

Zustandsräume (state spaces)

Ein *state space* ist eine Datenstruktur, die die Menge an dynamischen Zuständen, die ein System aufweisen kann, repräsentiert.

Definition:
Ein *state space* eines Modells ist ein Tupel (S, E, Δ, S_I)

- S ist die Menge der Zustände
- E ist die Menge der Events, welche auftreten können
- Δ ist die Menge der Transitionen
- $S_I \subseteq S$ ist Anfangszustand des Modells

Zustandsräume (state spaces)

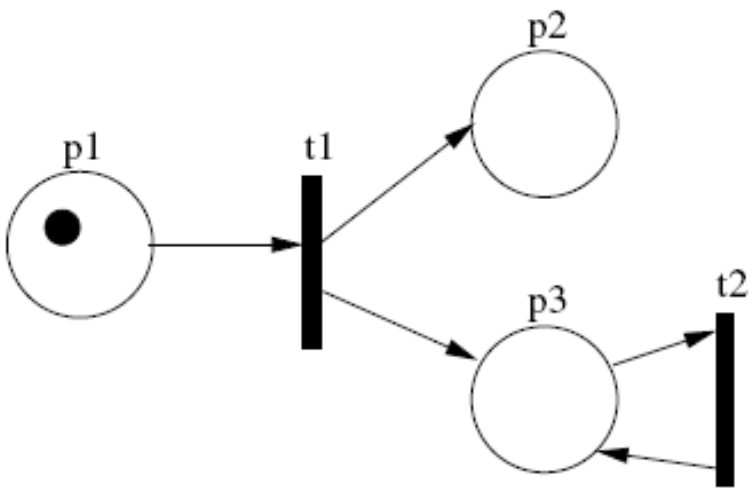
Der *state space* eines Petri Netzes (P, T, A, M_0) ist das Tupel (S, E, Δ, S_i), mit

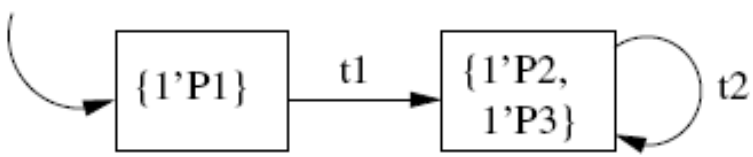
- $S = \mathbb{M}$
- $E = T$
- $(M_1, t, M_2) \in \Delta$ wenn $M_1[t\rangle M_2$
- $S_I = \{M_0\}$

Zustandsräume (state spaces)

Der *state space* eines Petri Netzes (P, T, A, M_0) ist das Tupel (S, E, Δ, S_i), mit

- $S = \mathbb{M}$
- $E = T$
- $(M_1, t, M_2) \in \Delta$ wenn $M_1[t\rangle M_2$
- $S_I = \{M_0\}$





Zustandsräume (state spaces)

Seien $s_1, s_2 \in S$. s_2 ist direkt von s_1 erreichbar, wenn es ein Event $e \in E$ gibt, so das $(s_1, e, s_2) \in \Delta$.

kurz geschrieben als $s_1 \rightarrow_e s_2$ oder $s_1 \rightarrow s_2$

Zustandsräume (state spaces)

Seien $s_1, s_2 \in S$. s_2 ist direkt von s_1 erreichbar, wenn es ein Event $e \in E$ gibt, so das $(s_1, e, s_2) \in \Delta$.

kurz geschrieben als $s_1 \rightarrow_e s_2$ oder $s_1 \rightarrow s_2$

Seien $s_{dest}, s_{src} \in S$. s_{dest} ist erreichbar von s_{src}, wenn es eine Folge von *events* $\{e_i\}_{i=1}^{n-1}$ und eine Folge von *states* $\{s_i\}_{i=1}^{n}$ gibt, so das:

- $s_{src} = s_1$ und $s_{dest} = s_n$
- $\forall 1 \leq i \leq n-1 : (s_i, e_i, s_{i+1}) \in \Delta$

kurz geschrieben als $s_{src} \rightarrow^* s_{dest}$

Relative Time Petri Nets

Struktur von Relative Time Petri Nets

Die Herangehensweise Zeit mit Petri Netzen zu verbinden heißt *relative time petri nets* (RTPNs).

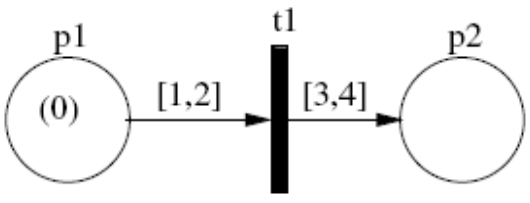
Struktur von Relative Time Petri Nets

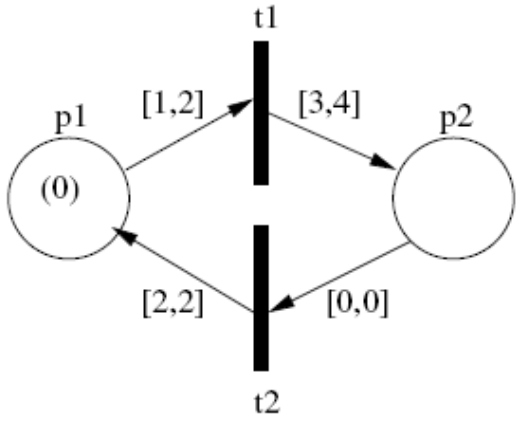
Die Herangehensweise Zeit mit Petri Netzen zu verbinden heißt *relative time petri nets* (RTPNs).

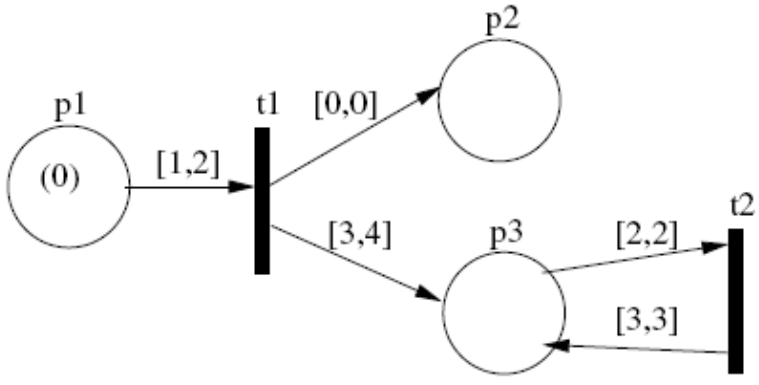
Definition:
Ein *relative time petri net* ist ein Tupel (P, T, A, M_0), mit

- P ist die Menge der *places*
  T ist die Menge der *transitions*
  $P \bigcap T = \emptyset$
- $A \in (((P \ x \ T) \bigcup (T \ x \ P)) \ x \ (\mathbb{N} \ x \ \mathbb{N}))_{MS}$
  ist das multi-set der *arcs*
- M_0 ist der Anfangszustand

Struktur von Relative Time Petri Nets







Struktur von Relative Time Petri Nets

Seien $M_1, M_2 \in \mathbb{M}^*$. M_1 ist ein *timed subset* von M_2,

geschrieben:

$M_1 \ll M_2$,

wenn $\forall p \in P : M_1(p) = \{x_i\}_{i=1}^n$, und $\exists \{y_i\}_{i=1}^n \subseteq M_2(p)$,

so das $x_i \leq y_i \forall i$

Schluss

Vielen Dank für die Aufmerksamkeit!

Structure

Inhaltsverzeichnis

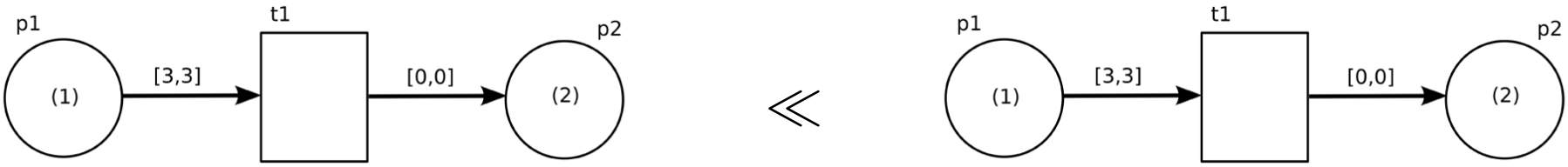
Timed Subset

Definition: *Timed Subset*

Seien $A, B \in \mathbb{M}^*$, dann ist A ein „*Timed Subset*“ von B, geschrieben $A \ll B$ genau dann wenn:
$\forall p \in P : A(p) = \{x_i\}_{i=1}^{n},\ \exists \{y_i\}_{i=1}^{n} \subseteq B(p),\ x_i \leq y_i\ \forall i$

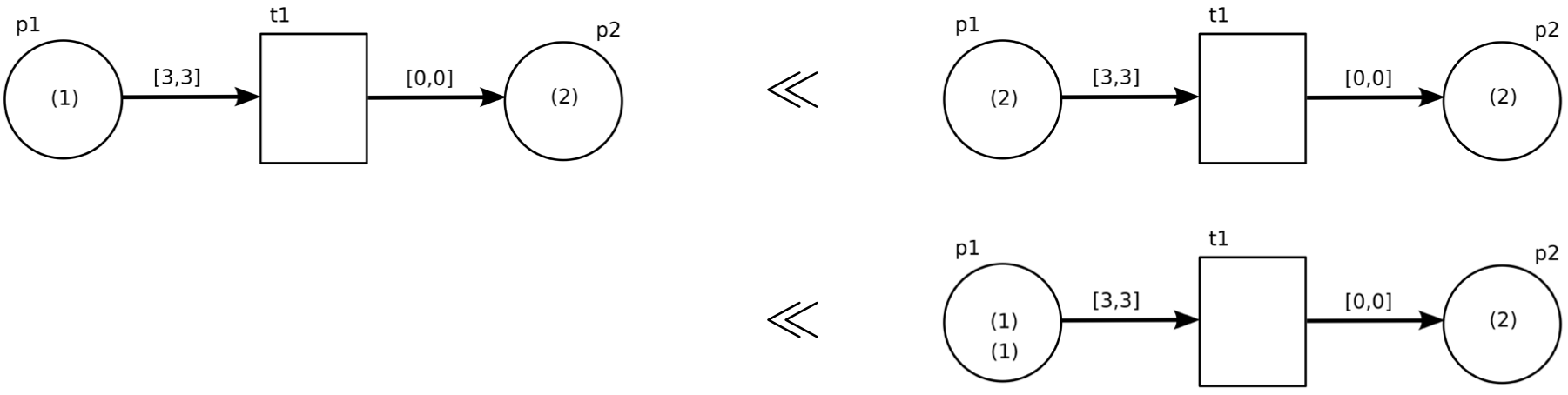
Eine *Marking* ist also ein „*Timed Subset*“ eines anderen *Markings*, wenn das andere *Marking* mindestens genauso viele (oder mehr) *Tokens* hat, wie das erste *Marking*, die mindestens genauso alt sind.

Timed Subset - Beispiel

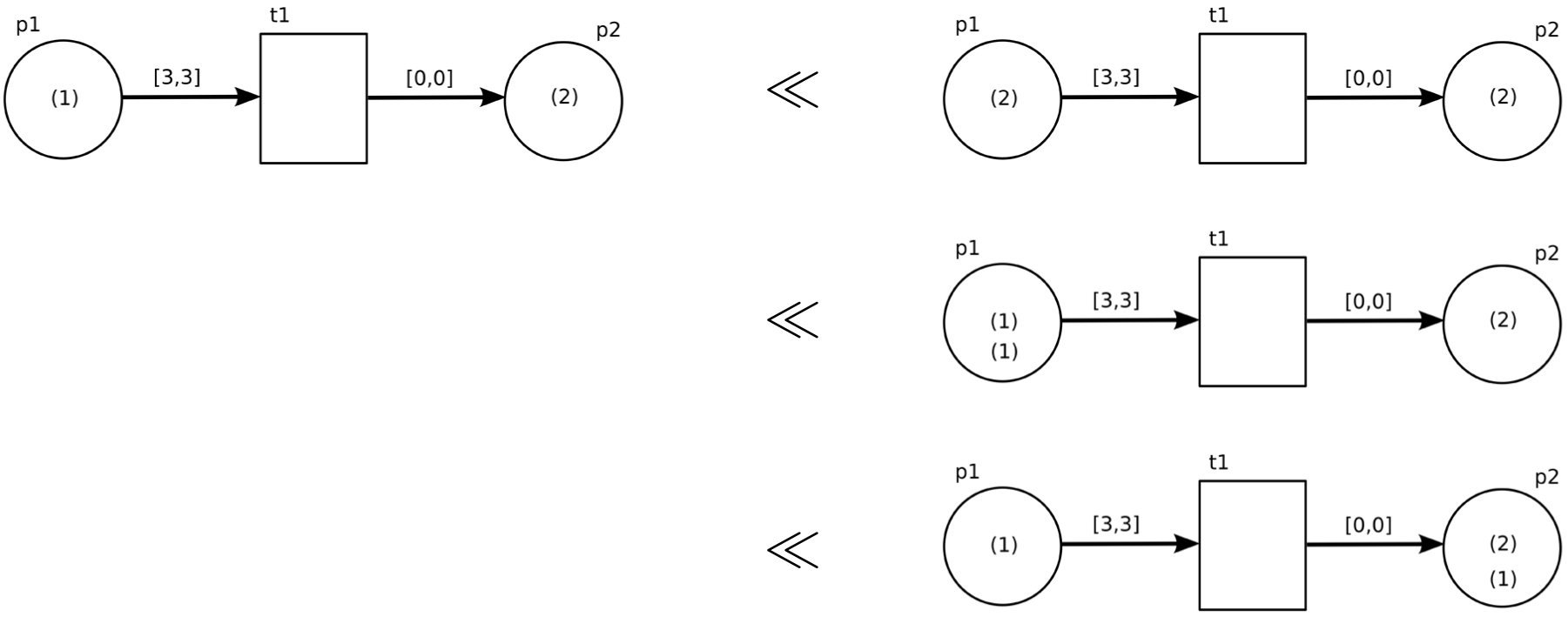


Timed Subset - Beispiel

Timed Subset - Beispiel

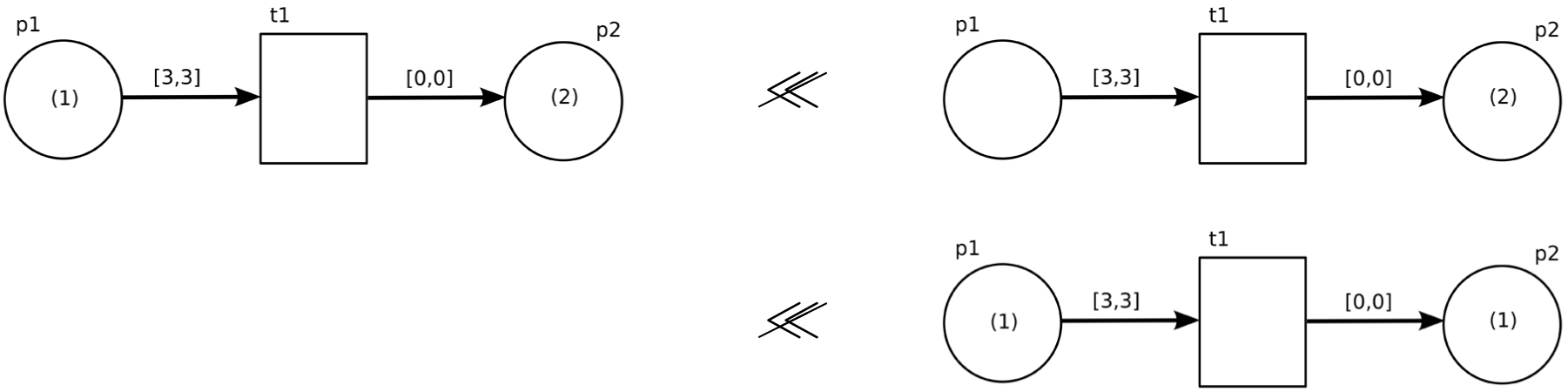


Timed Subset - Beispiel

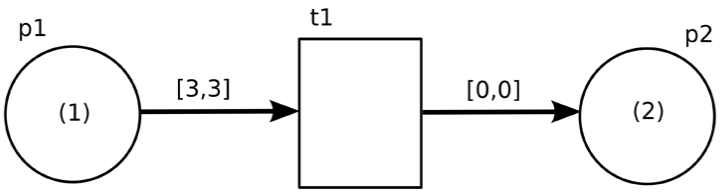


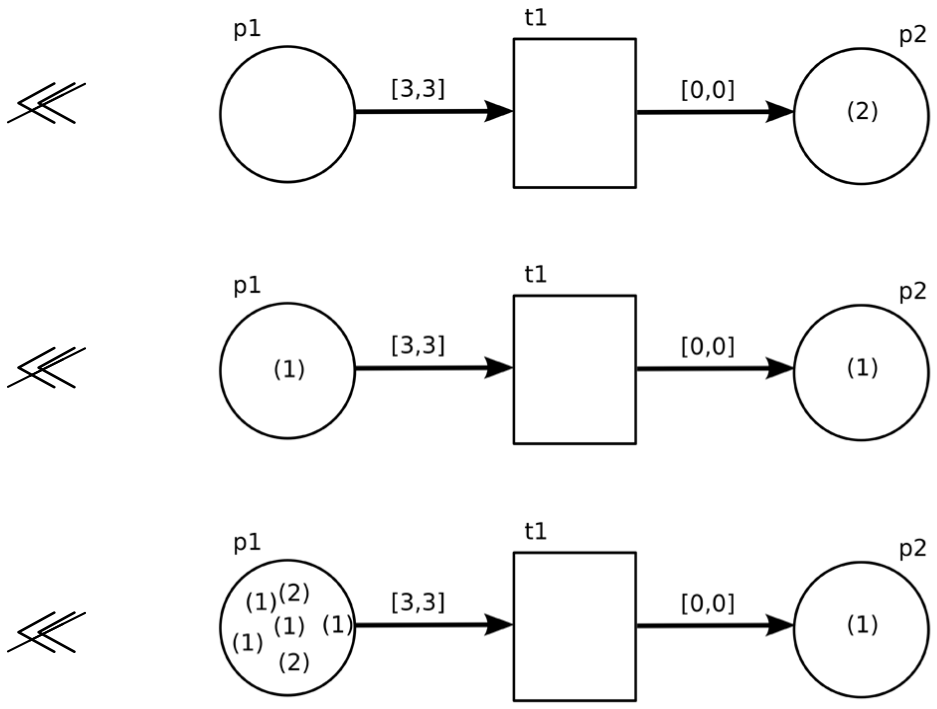
Timed Subset - Beispiel

Timed Subset - Beispiel



Timed Subset - Beispiel





Timed Difference

Definition: *Timed Difference*

Seien $A, B, C \in \mathbb{M}^*$, und $A \ll B$. Dann ist C eine „*T*imed Difference“ zwischen A und B, geschrieben $C = B \backslash A$ genau dann wenn:

$C \subseteq B,\ \sharp C = \sharp B - \sharp A,\ A \ll (B - C)$

C ist also eine Untermenge von B, die man von B abziehen kann, wobei A ein *Timed Subset* der entfernten Elemente ist. Die Anzahl der Elemente der resultierenden Menge entspricht der Anzahl der Elemente von B weniger der von A.
Es ist zu beachten, dass es mehrere verschiedene *Timed Differences* für zwei *Markings* geben kann.

Timed Difference - Beispiel

- $A = \{0, 1\}$ $B = \{0, 2, 2, 4\}$

Timed Difference - Beispiel

- $A = \{0, 1\}$ $B = \{0, 2, 2, 4\}$ wie man sieht: $A \ll B$

Timed Difference - Beispiel

- $A = \{0, 1\}$ $B = \{0, 2, 2, 4\}$ wie man sieht: $A \ll B$
- $\{0, 1\} \ll \{0, 2\} \rightarrow \{2, 4\} = B \backslash A$
- $\{0, 1\} \ll \{0, 4\} \rightarrow \{2, 2\} = B \backslash A$
- $\{0, 1\} \ll \{2, 2\} \rightarrow \{0, 4\} = B \backslash A$
- $\{0, 1\} \ll \{2, 4\} \rightarrow \{0, 2\} = B \backslash A$

Minimal Timed Difference

Definition: *Minimal Timed Difference*

Seien $A, B, C \in \mathbb{M}^*$, und $A \ll B$.
Dann ist C eine „*Minimal Timed Difference*“ zwischen A und B, geschrieben $C = B \backslash_{min} A$ genau dann wenn:
$C = B \backslash A$ und $\forall D \in \mathbb{M}^*$ mit $D = B \backslash A$ gilt, dass $D \ll C$.

Die *Minimal Timed Difference* ist also die *Timed Difference*, von der alle anderen *Timed Differences Timed Subsets* sind.
Die „*Minimal Timed Difference*“ zwischen zwei *Markings* wird dazu verwendet, um zu entscheiden, welche Tokens beim Feuern einer *Transition* zu entfernen sind.

Minimal Timed Difference - Beispiel

- $A = \{0, 1\}$ $B = \{0, 2, 2, 4\}$

Minimal Timed Difference - Beispiel

- $A = \{0, 1\}$ $B = \{0, 2, 2, 4\}$
- *Timed Differences*: $\{2, 4\}$ $\{2, 2\}$ $\{0, 4\}$ $\{0, 2\}$

Minimal Timed Difference - Beispiel

- $A = \{0, 1\}$ $B = \{0, 2, 2, 4\}$
- *Timed Differences*: $\{2, 4\}$ $\{2, 2\}$ $\{0, 4\}$ $\{0, 2\}$
- $\{0, 2\} \ll \{0, 4\} \ll \{2, 2\} \ll \{2, 4\}$

Minimal Timed Difference - Beispiel

- $A = \{0, 1\}$ $B = \{0, 2, 2, 4\}$
- *Timed Differences*: $\{2, 4\}$ $\{2, 2\}$ $\{0, 4\}$ $\{0, 2\}$
- $\{0, 2\} \ll \{0, 4\} \ll \{2, 2\} \ll \{2, 4\}$
  $\Rightarrow \{2, 4\} = B \backslash_{min} A$

Minimal Timed Difference - Berechnung

Die *Minimal Timed Difference* zwischen zwei *Markings* $A \ll B$ wird wie folgt berechnet:

- Man fängt an, indem man $A_0 = A$ und $B_0 = B$ setzt.
- Dann setzt man $A_i = A_{i-1} - \{max(A_{i-1})\}$, d.h. man entfernt das größte Element von A
- man setzt $B_i = B_{i-1} - \{min\{b \in B_i | b \geq max(A_{i-1})\}\}$, d.h. man entfernt das kleinste Element, das noch mindestens so groß ist, wie das, was gerade aus A_i entfernt wurde
- Die letzen beiden Schritte werden wiederholt, bis A_i die leere Menge ist
- B_i enthält dann die *Minimal Timed Difference*

Aging

- „*Aging*“ ist der Mechanismus, der die Modellierung des Zeitverlaufs ermöglicht
- Beim Altern des Netzes werden die *Timestamps* aller Tokens im Netz um einen Wert δ erhöht.
- Das *Marking* M' ist dann gleich eines um δ gealterten *Markings*

Definition: *Aging*

$M' = age(M, \delta)$ gdw. $\forall p \in P, d \in \mathbb{Z} : M'(p, d) = M(p, d + \delta)$

Zusammenfassung

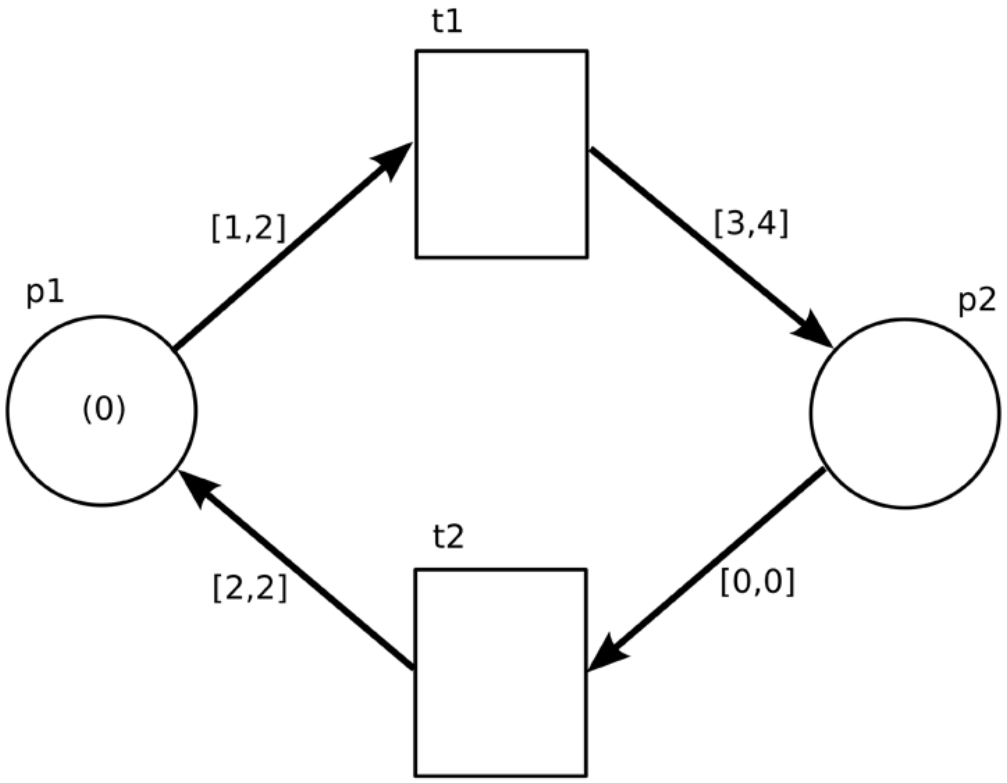
- Jeder Token ist mit einem *Timestamp* markiert
- Der *Timestamp* eines Tokens gibt an, wie lange der Token nach dem Feuern der *Transition* verfügbar war.
- Durch das *Aging* werden alle *Timestamps* des Netzes um einen Wert δ erhöht
- *Transitions* feuern sofort, d. h. wenn eine *Transition* feuert verschwinden die Tokens auf der Eingabeseite und es erscheinen Tokens auf der Ausgabeseite, ohne dass dabei Zeit verstreicht

Zusammenfassung

- Wenn die obere Intervallgrenze aller *Input Arcs* einer *Transition* voll erfüllt ist, so feuert die *Transition* sofort. Innerhalb des Intervalls kann die *Transition* feuern
- Ein *Timestamp* kann einen negativen Wert haben - dies bedeutet, dass der Token noch nicht verfügbar ist. Wenn das Intervall der *Output Arc* einer *Transition* positive Werte enthält, so erhalten die Tokens auf der Ausgabeseite auch negative *Timestamps*
- Die Nummern an den Eingangskanten einer *Transition* geben an, wann eine *Transition* feuern darf

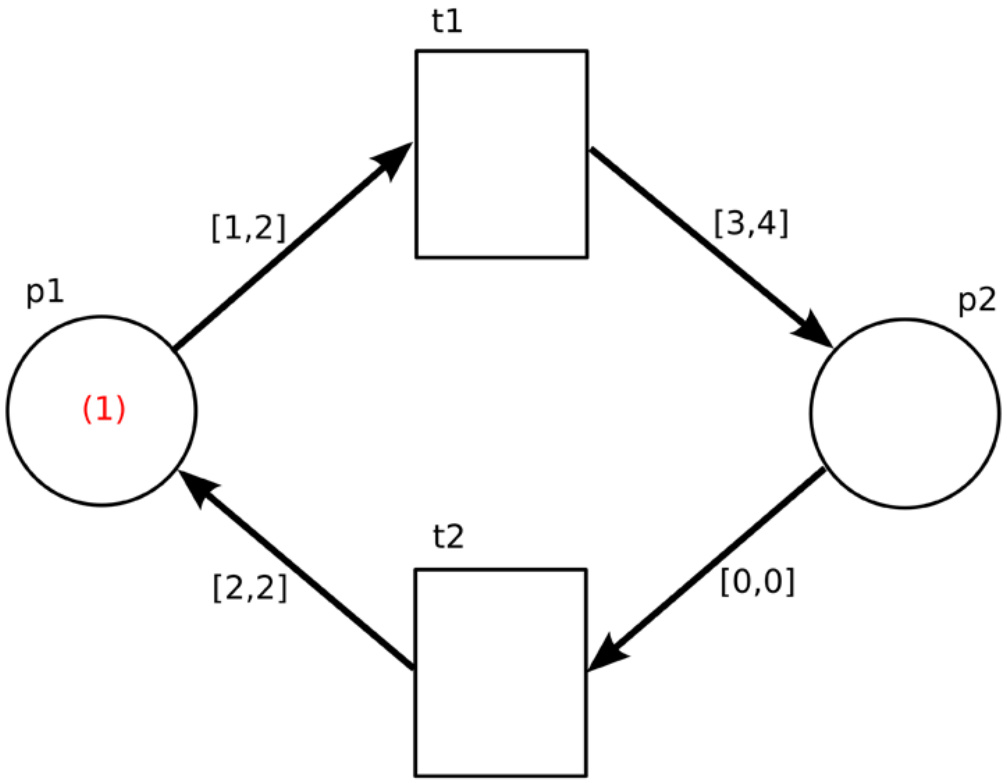
Netzbeispiel

Anfangszustand



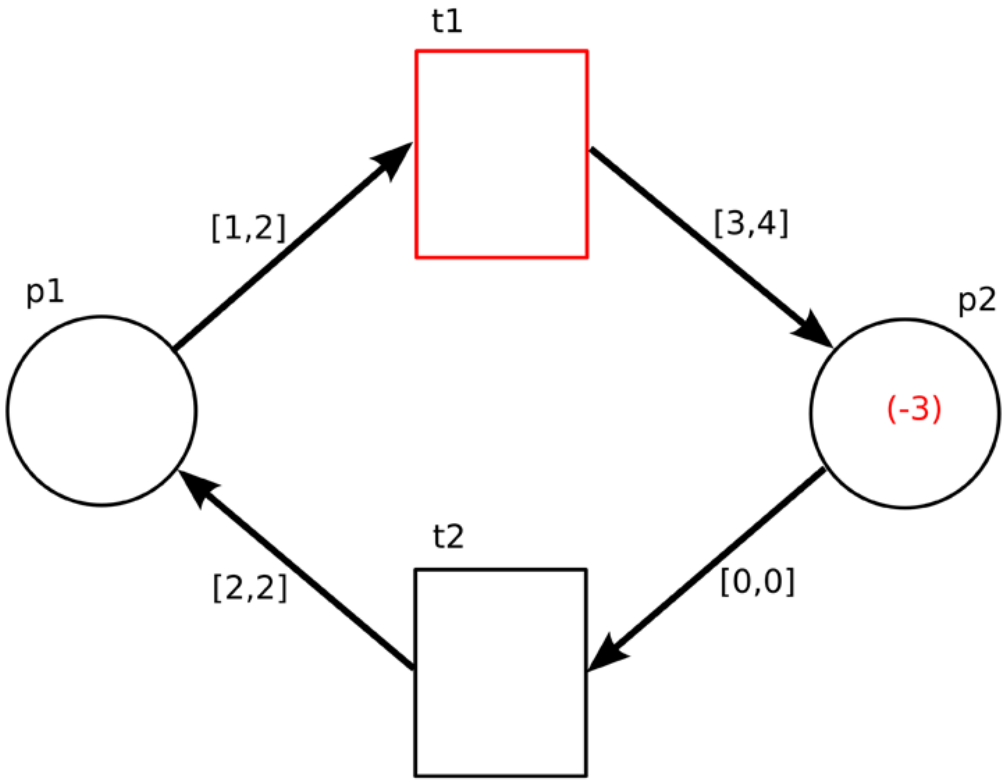
Netzbeispiel

Aging um 1



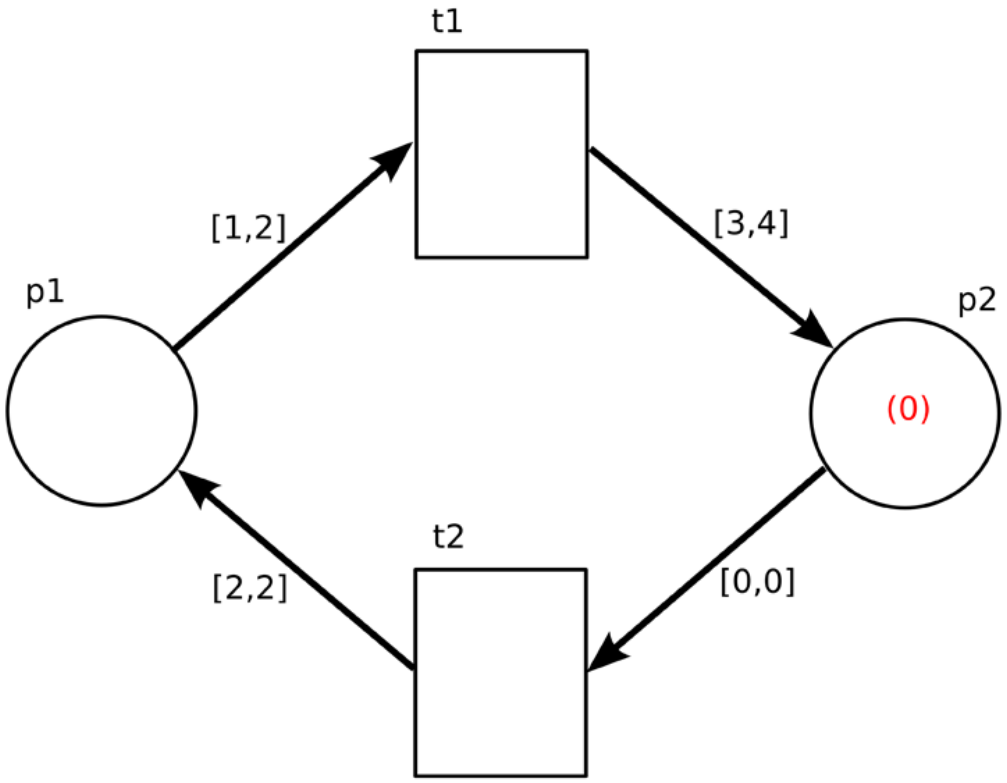
Netzbeispiel

t1 feuert



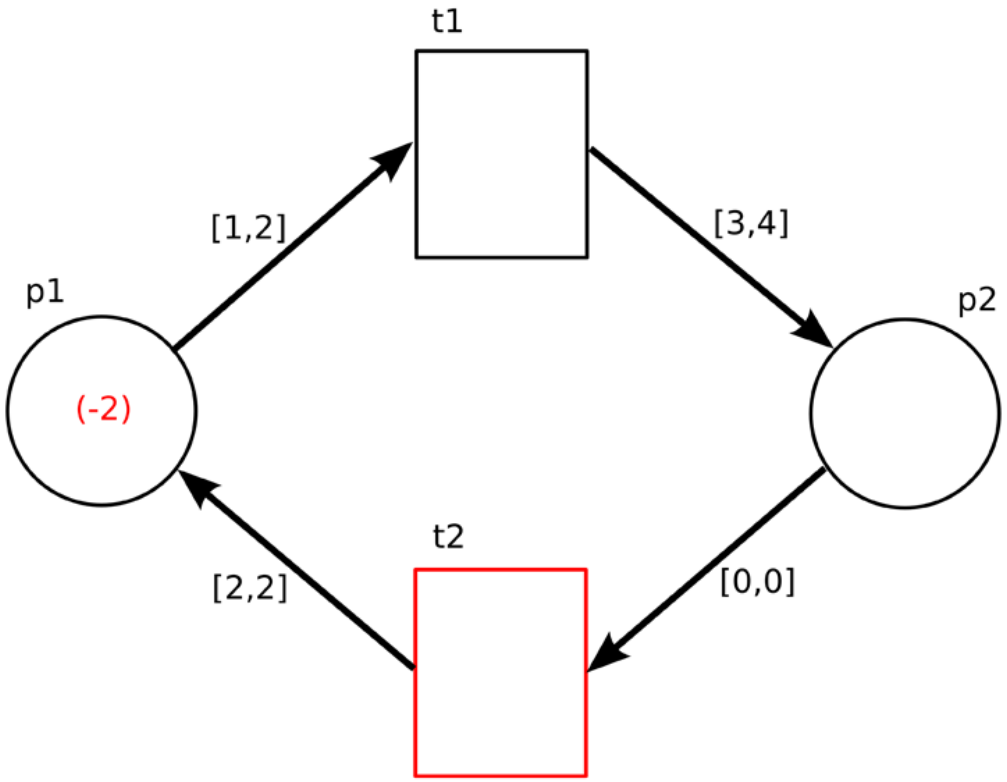
Netzbeispiel

Aging um 3



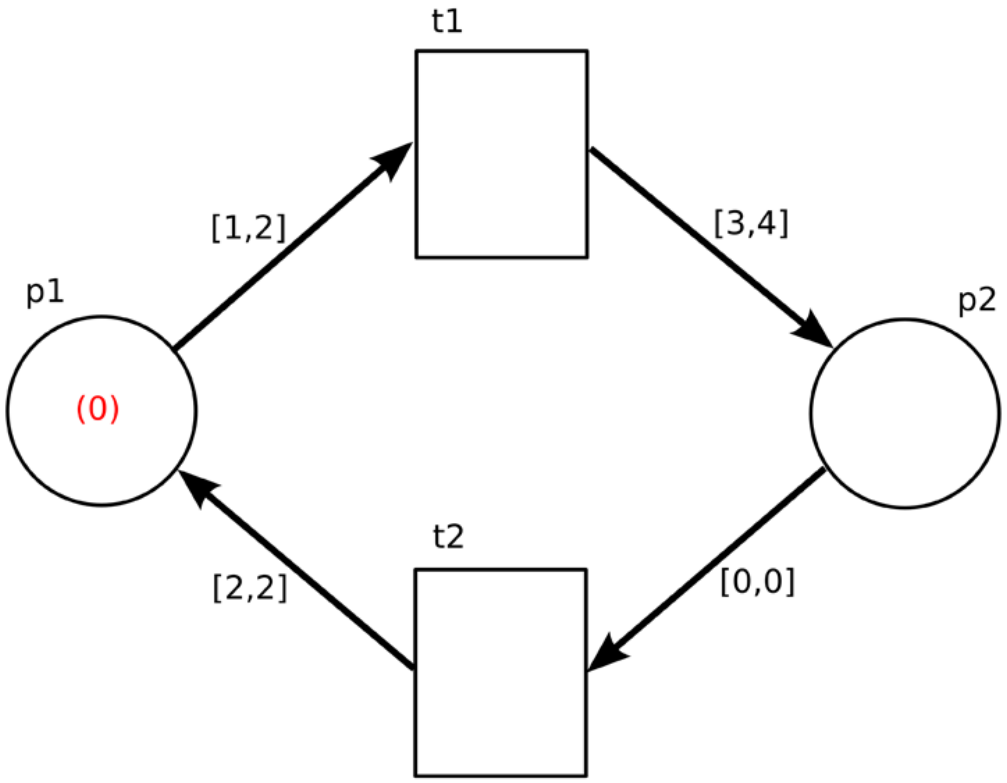
Netzbeispiel

t2 feuert



Netzbeispiel

Aging um 2 & zurück zum Anfangszustand



Events

Definition: *Event*

- Ein „*Event*“ ist ein Tupel (t, M_{cons}, M_{prod})
  - $t \in T$ ist die feuernde *Transition*
  - M_{cons}, $M_{prod} \in \mathbb{M}^*$ sind Relative Time *Markings*
  - M_{cons} ist das *Marking*, das beim Feuern der *Transition* auf der Input-Seite entfernt wird

    $\forall p \in P$: sei $X = \{(p, t, [a_i, b_i])\}_{i=1}^{n} \subseteq A$ das vollständige Multi-Set der *Input Arcs* für t, dann ist $M_{cons}(p) = \{x_i\}_{i=1}^{n}$ mit $a_i \leq x_i \leq b_i \forall i$
  - M_{prod} ist das *Marking*, das beim Feuern der *Transition* auf der Output-Seite hinzugefügt wird

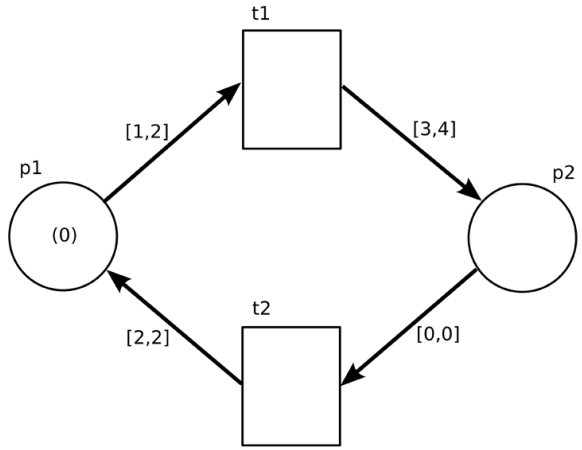
    $\forall p \in P$: sei $X = \{(t, p, [a_i, b_i])\}_{i=1}^{n} \subseteq A$ das vollständige Multi-Set der *Output Arcs* für t, dann ist $M_{prod}(p) = \{x_i\}_{i=1}^{n}$ mit $-a_i \geq x_i \geq -b_i \forall i$
  - Der Index in M_{cons} entspricht also dem der zugehörigen *Input Arc* und der Index in M_{prod} dem der zugehörigen *Output Arc*

Da die Guards der Input- und Output-Arcs ein Intervall definieren, können zu einer *Transition* mehrere *Events* existieren:

Events - Beispiel

Die *Input Arc* der *Transition* t_1 hat das Intervall $[1, 2]$, die *Output Arc* das Intervall $[3, 4]$
Dadurch ergeben sich vier mögliche *Events*:

- $(t_1, \{1\}, \{-3\})$
- $(t_1, \{1\}, \{-4\})$
- $(t_1, \{2\}, \{-3\})$
- $(t_1, \{2\}, \{-4\})$



Bei *Transition* t_2 ergibt sich durch die Intervalle $[0, 0]$ und $[2, 2]$ nur ein *Event*:

- $(t_2, \{0\}, \{-2\})$

Events: Regeln

Definition: enabled

Ein *Event* $E = (t, M_{cons}, M_{prod})$ ist „enabled“ für ein *Marking* $M \in \mathbb{M}^*$ nach einem Delay δ, geschrieben $M[\delta, E\rangle$ gdw. $M_{cons} \ll age(M, \delta)$

Das *Marking* muss also um δ ge-aged werden, um in das Intervall der *Input Arcs* der *Transition* zu fallen.

Definition: firing

Wenn das *Event* $E = (t, M_{cons}, M_{prod})$ für ein *Marking* $M \in \mathbb{M}^*$ nach einem Delay δ „enabled“ ist, dann ist das *Marking* $M \in \mathbb{M}^*$, das durch das firing der *Transition* produziert wird, geschrieben $M[\delta, E\rangle M'$:
$M' = age(M, \delta) \setminus_{min} M_{cons} + M_{prod}$

Man entfernt also aus M diejenigen Tokens, deren *Timestamp* eine *Minimal Timed Difference* zu M_{cons} hat.

Statespaces

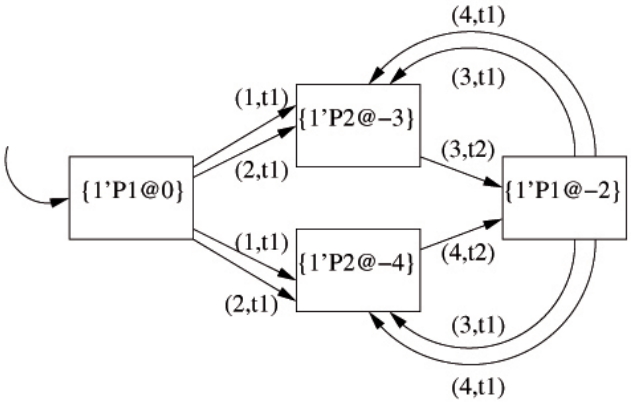
Definition: *Statespace*

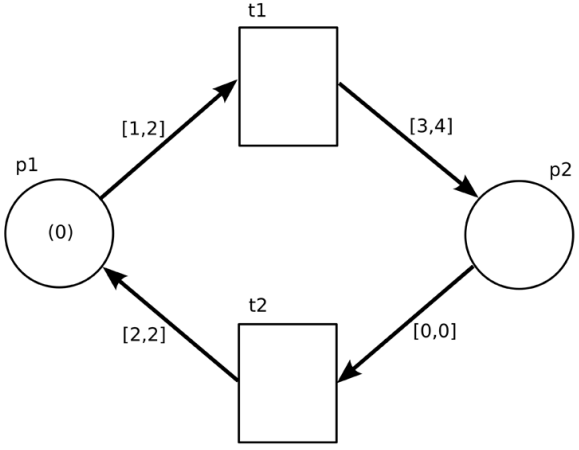
Der *Statespace* für ein Relative Time Petri Net ist das Tupel (S, E, Δ, S_I) mit:

- den möglichen *Markings* $S = \mathbb{M}^*$
- den *Events* $E = \mathbb{N} \times T$
- den Zustandsübergängen Δ, für die gilt:
  $(M_1, (\delta, t), M_2) \in \Delta$ gdw.
  $\exists M_{cons}, M_{prod} \in \mathbb{M}^* : M_1[\delta, (t, M_{cons}, M_{prod})\rangle M_2$
- den Initialzustand $S_I = \{M_0\}$

Die *Events* benötigen anders als im normalen Petrinetzmodell neben den *Transitions* auch noch einen Delaywert $\delta \in T$, um den das Netz ge-aged werden muss um t zu enablen.

Statespaces - Beispiel





- an den Kanten sind die *Events* vermerkt
- es ist nicht nötig, M_{cons}, oder M_{prod} anzugeben, da das Delay und die feuernde *Transition* ausreichen, um das *Event* zu bestimmen
- die Boxen für die States sind mit den *Markings* beschriftet: $\{1'P1@0\}$ bedeutet, dass das erste Token der *Place* P1 den *Timestamp* 0 besitzt
- ein Problem besteht noch, wenn man zulässt, dass das Delay über die Intervallgrenzen hinaus wächst, da dadurch der Zustandsraum ins unendliche wachsen kann.

Vielen Dank für die Aufmerksamkeit

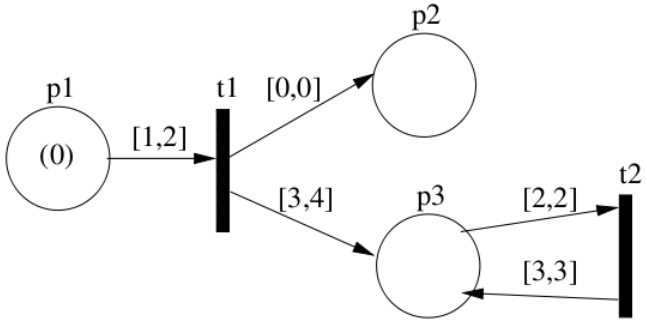
Properties, Prototypes and Related Work

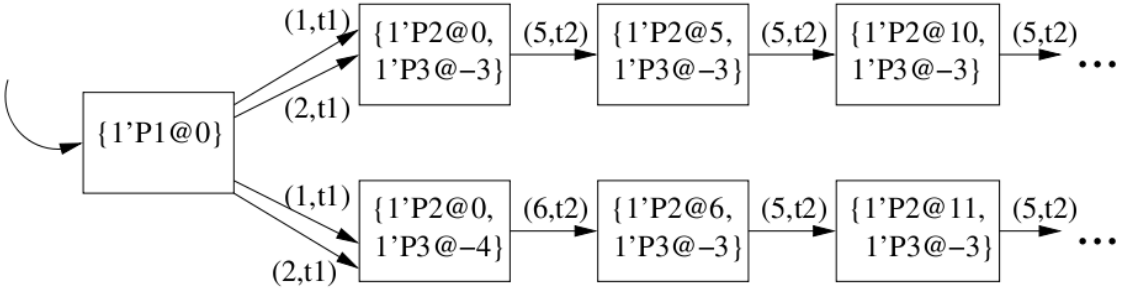
Inhaltsverzeichnis

Problembeschreibung

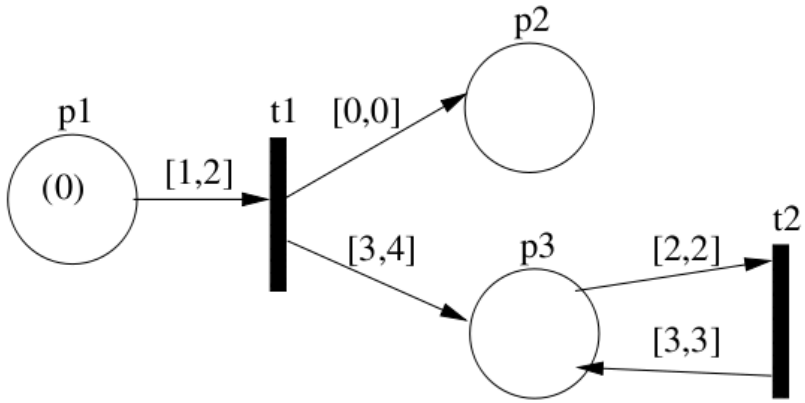
Problem: unendliche *Statespaces*

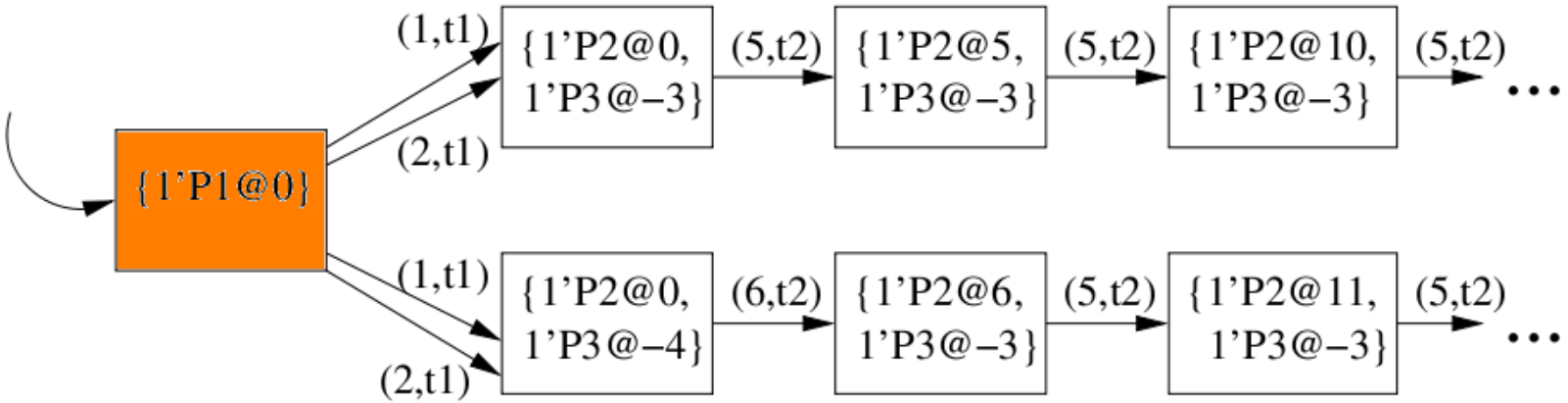
- *Outgoing Arc* einer *Place* = *Input Arc* einer *Transition*.
- Falls eine *Place* keine *Outgoing Arcs* besitzt, bleibt ein *Token* dort unendlich lang, solange das restliche *Petri Net* noch nicht *dead* ist (also *deadlock free* oder *living*)
- Dieses *Token* wird mit jedem *Step* älter, wodurch der *Statespace* unendlich weiter wächst.



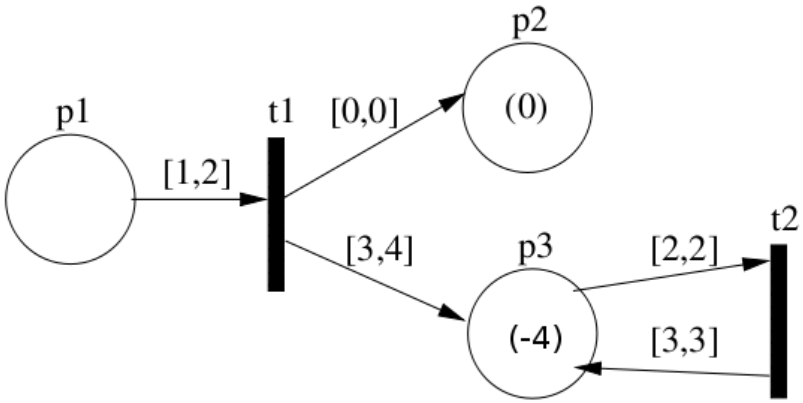


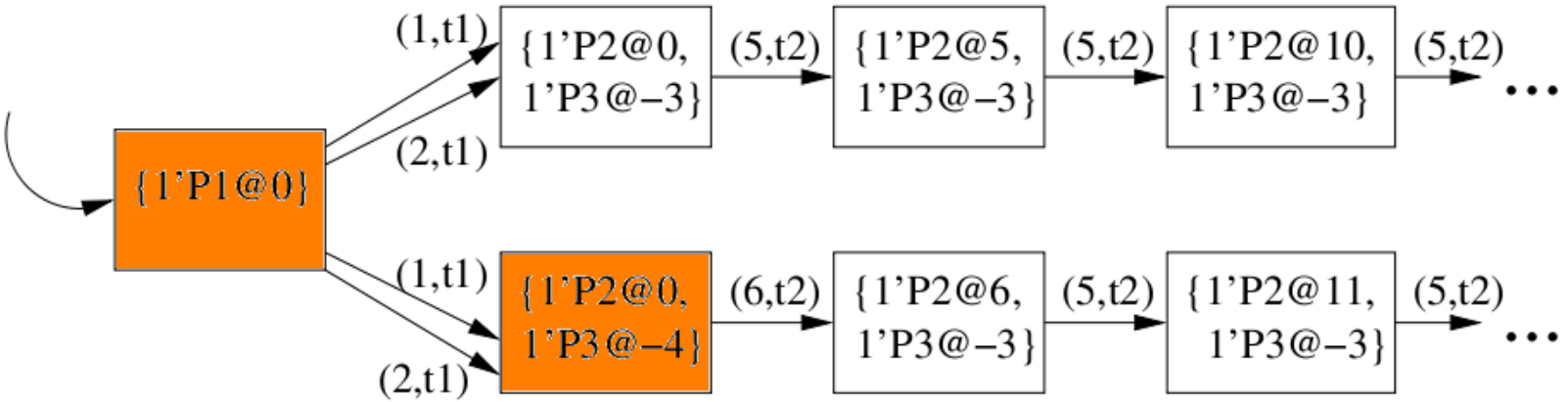
Beispiel


p2
p1
t1
[0,0]
(0)
[1,2]
[3,4]
p3
t2
[2,2]
[3,3]

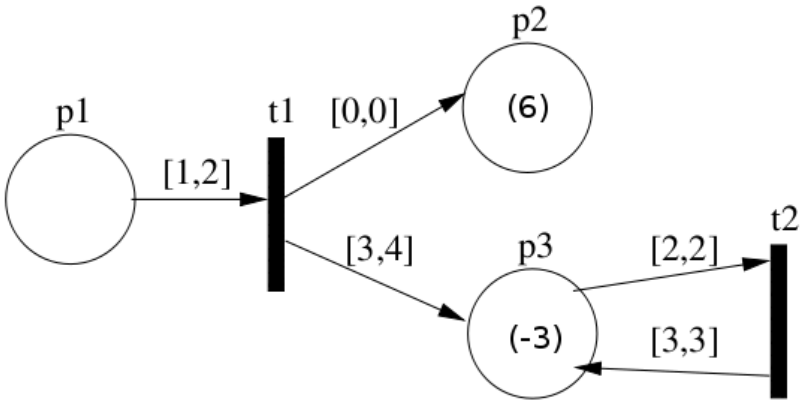

(1,t1)
{1'P2@0,
1'P3@−3}
(5,t2)
{1'P2@5,
1'P3@−3}
(5,t2)
{1'P2@10,
1'P3@−3}
(5,t2)
{1'P1@0}
(2,t1)
(1,t1)
{1'P2@0,
1'P3@−4}
(6,t2)
{1'P2@6,
1'P3@−3}
(5,t2)
{1'P2@11,
1'P3@−3}
(5,t2)
(2,t1)

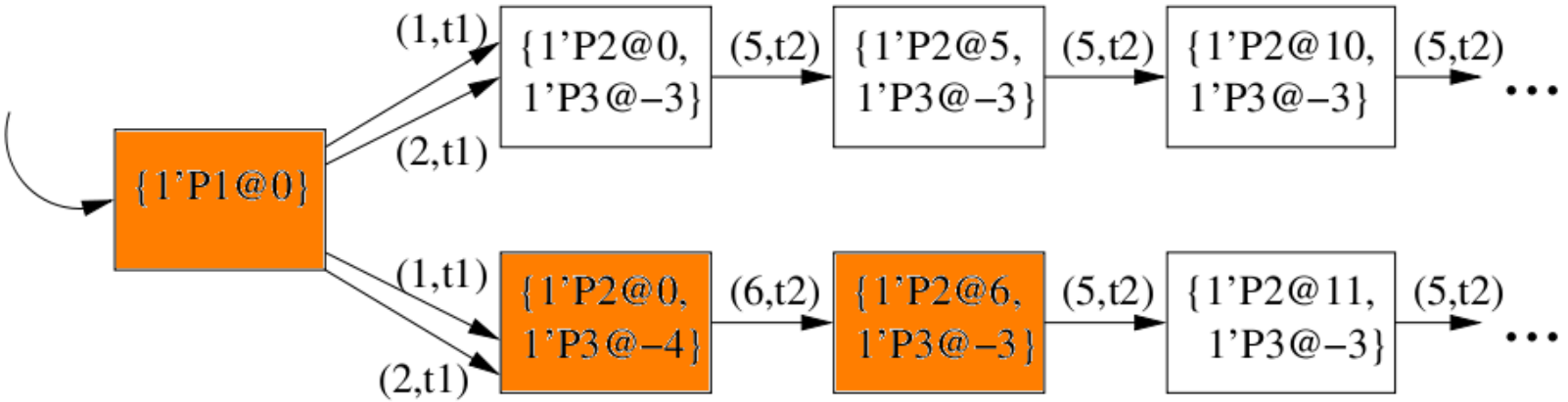
Beispiel


p1
t1
[0,0]
p2
(0)
[1,2]
[3,4]
p3
[2,2]
t2
(-4)
[3,3]

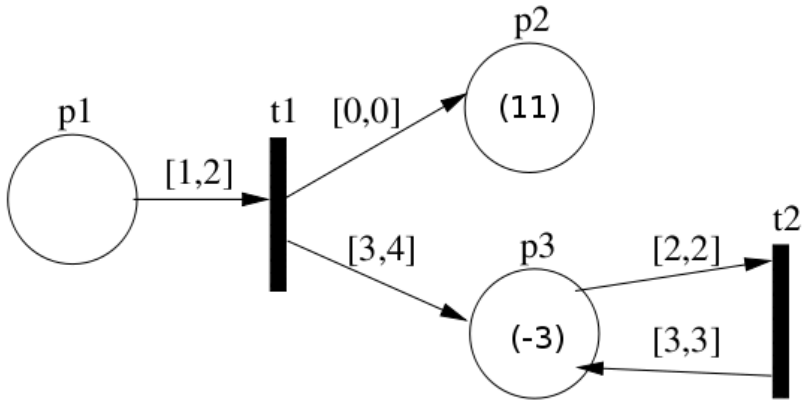

{1'P1@0}
(1,t1)
(2,t1)
{1'P2@0, 1'P3@−3}
(5,t2)
{1'P2@5, 1'P3@−3}
(5,t2)
{1'P2@10, 1'P3@−3}
(5,t2)
(1,t1)
(2,t1)
{1'P2@0, 1'P3@−4}
(6,t2)
{1'P2@6, 1'P3@−3}
(5,t2)
{1'P2@11, 1'P3@−3}
(5,t2)

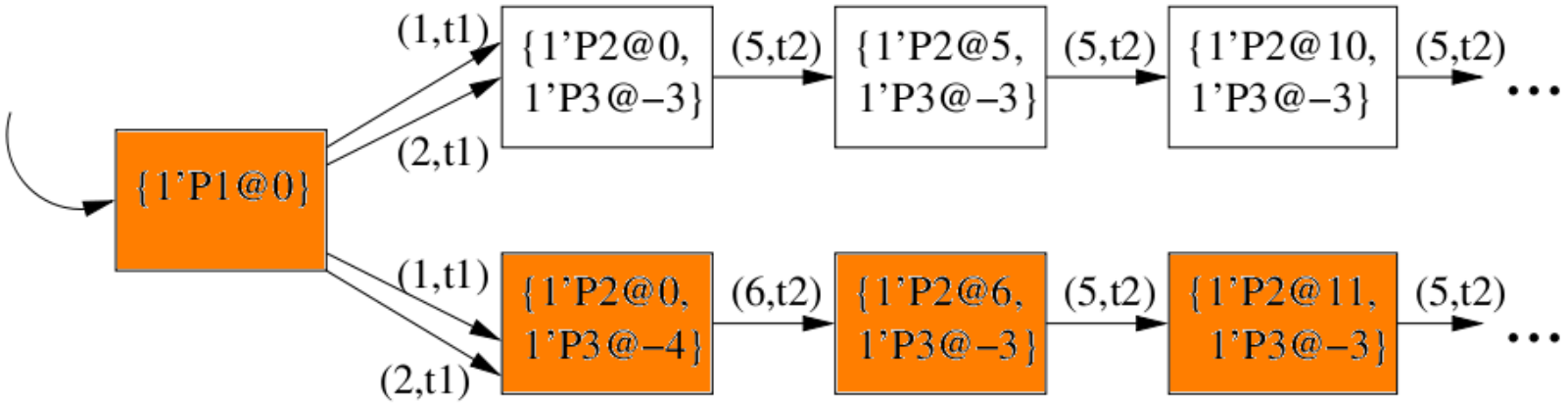
Beispiel


p1
t1
[0,0]
p2
(6)
[1,2]
[3,4]
p3
[2,2]
t2
(-3)
[3,3]


{1'P1@0}
(1,t1)
(2,t1)
{1'P2@0, 1'P3@–3}
(5,t2)
{1'P2@5, 1'P3@–3}
(5,t2)
{1'P2@10, 1'P3@–3}
(5,t2)
(1,t1)
(2,t1)
{1'P2@0, 1'P3@–4}
(6,t2)
{1'P2@6, 1'P3@–3}
(5,t2)
{1'P2@11, 1'P3@–3}
(5,t2)

Beispiel


p1
t1
[0,0]
p2
(11)
[1,2]
[3,4]
p3
[2,2]
t2
(-3)
[3,3]


{1'P1@0}
(1,t1)
(2,t1)
{1'P2@0, 1'P3@−3}
(5,t2)
{1'P2@5, 1'P3@−3}
(5,t2)
{1'P2@10, 1'P3@−3}
(5,t2)
(1,t1)
(2,t1)
{1'P2@0, 1'P3@−4}
(6,t2)
{1'P2@6, 1'P3@−3}
(5,t2)
{1'P2@11, 1'P3@−3}
(5,t2)

Lösung mit *Truncation*

Time Cap

Wir schauen uns für jede *Place* die *Outgoing Arcs* an und begrenzen nach jedem *Step* die *Tokens* auf die oberste Grenze der Intervalls der *Outgoing Arcs*
Dieser Wert wird wie folgt berechnet:

Time Cap

Wir schauen uns für jede *Place* die *Outgoing Arcs* an und begrenzen nach jedem *Step* die *Tokens* auf die oberste Grenze der Intervalls der *Outgoing Arcs*
Dieser Wert wird wie folgt berechnet:

Definition: *Time Cap* $\omega(p)$

Ein *Time Cap* $(\omega(p))$ für ein *Relative Time Petri Net* (P,T,A,M_0) ist eine Zuordnung $\omega : P \rightarrow \mathbb{N}$, so dass
$\forall p \in P : \omega(p) = max\{b | \exists a \in \mathbb{N}, t \in T : [a, b] \in A(p, t)\}$.

Dies entspricht der höchsten Zahl, die an einer der *Outgoing Arcs* der *Place* notiert ist.

Truncation

In jedem *Step* wird jedes *Token* auf das errechnete *Time Cap* für die *Place* gesetzt (abgeschnitten).

Definition: *Truncation*

$M|_\omega \in \mathbb{M}$ ist die *Truncation* von M, genau wenn $\forall p \in P, d \in \mathbb{Z}$ gilt:

$$M|_\omega(p,d) = \begin{cases} M(p,d) & d < \omega(p) \\ \sum_{i=\omega(p)}^{\infty} & d = \omega(p) \\ 0 & d > \omega(p) \end{cases}$$

Alle *Tokens* $> \omega(p)$ (*Time Cap*) sind nun auf $\omega(p)$ gekürzt worden. Alle *Tokens* $\leq \omega(p)$ bleiben wie sie waren.

Argumentativer Beweis

Jedes aktivierte *Event* eines *Markings* ist auch nach einer *Truncation* aktiviert:

"$\Rightarrow$"

$\omega(p)$ ist das minimale *Marking* einer *Place*, so dass alle *Events* aktiviert sind. Wenn nun alle *Tokens* auf $\omega(p)$ gekürzt werden, so bleiben auch alle *Events*, die aktiviert waren weiterhin aktiviert.

Argumentativer Beweis

Jedes aktivierte *Event* eines *Markings* ist auch nach einer *Truncation* aktiviert:

"$\Rightarrow$"

$\omega(p)$ ist das minimale *Marking* einer *Place*, so dass alle *Events* aktiviert sind. Wenn nun alle *Tokens* auf $\omega(p)$ gekürzt werden, so bleiben auch alle *Events*, die aktiviert waren weiterhin aktiviert.

"$\Leftarrow$"

Da nur verlängert wird, bleiben alle aktivierten *Events* weiterhin aktiviert. Alle anderen bleiben unberührt, da $Tokens < \omega(p)$ unberührt bleiben.

Argumentativer Beweis

Jedes aktivierte *Event* eines *Markings* ist auch nach einer *Truncation* aktiviert:

"$\Rightarrow$"

$\omega(p)$ ist das minimale *Marking* einer *Place*, so dass alle *Events* aktiviert sind. Wenn nun alle *Tokens* auf $\omega(p)$ gekürzt werden, so bleiben auch alle *Events*, die aktiviert waren weiterhin aktiviert.

"$\Leftarrow$"

Da nur verlängert wird, bleiben alle aktivierten *Events* weiterhin aktiviert. Alle anderen bleiben unberührt, da *Tokens* $< \omega(p)$ unberührt bleiben.

Somit sind im gekürzten und ungekürzten *State* die gleichen *Events* aktiviert (und die gleichen deaktiviert)

Formaler Beweis

Sei $M \in \mathbb{M}$ ein *Marking* für das Netz (P,T,A,M_0). Dann ist jedes *Event*, das in M aktiviert ist auch in $M|_\omega$ aktiviert und umgekehrt.

Formaler Beweis

Sei $M \in \mathbb{M}$ ein *Marking* für das Netz (P,T,A,M_0). Dann ist jedes *Event*, das in M aktiviert ist auch in $M|_\omega$ aktiviert und umgekehrt.

"$\Rightarrow$"

Sei (t, M_{cons}, M_{prod}) aktiviert für $M|_\omega$ nach *Delay* δ. Dann ist $M_{cons} \ll age(M, \delta) \Rightarrow M_{cons}|_\omega \ll age(M|_\omega, \delta)$. Aber $M_{cons} = M_{cons}|_\omega$, da die *Tokens* in M_{cons} innerhalb der Intervalle für t liegen müssen. Daher ist $M_{cons} \ll age(M|_\omega, \delta)$, also ist (t, M_{cons}, M_{prod}) in $M|_\omega$ aktiviert.

Formaler Beweis

Sei $M \in \mathbb{M}$ ein *Marking* für das Netz (P,T,A,M_0). Dann ist jedes *Event*, das in M aktiviert ist auch in $M|_\omega$ aktiviert und umgekehrt.

"$\Rightarrow$"

Sei (t, M_{cons}, M_{prod}) aktiviert für $M|_\omega$ nach *Delay* δ. Dann ist $M_{cons} \ll age(M, \delta) \Rightarrow M_{cons}|_\omega \ll age(M|_\omega, \delta)$. Aber $M_{cons} = M_{cons}|_\omega$, da die *Tokens* in M_{cons} innerhalb der Intervalle für t liegen müssen. Daher ist $M_{cons} \ll age(M|_\omega, \delta)$, also ist (t, M_{cons}, M_{prod}) in $M|_\omega$ aktiviert.

"$\Leftarrow$"

Sei (t, M_{cons}, M_{prod}) aktiviert für $M|_\omega$ nach *Delay* δ. Da $M|_\omega \ll M$ und $M_{cons} \ll age(M|_\omega, \delta) \ll age(M, \delta)$ ist $M_{cons} \ll age(M, \delta)$ weil die *Timestamps* in M größer sind als die in $M|_\omega$. Also ist (t, M_{cons}, M_{prod}) in M nach *Delay* δ aktiviert.

Truncated Marking

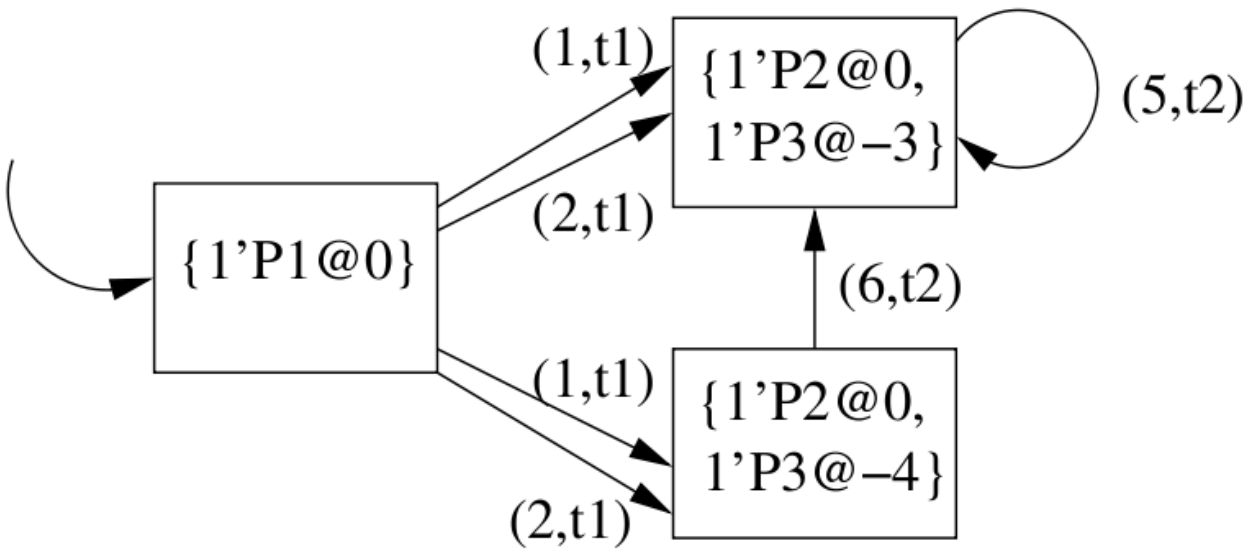
Wir definieren das *Marking*, welches durch das *Firing* eines *Events* entsteht nach der *Truncation* als *Truncated Marking*

Definition: *Truncated Marking*

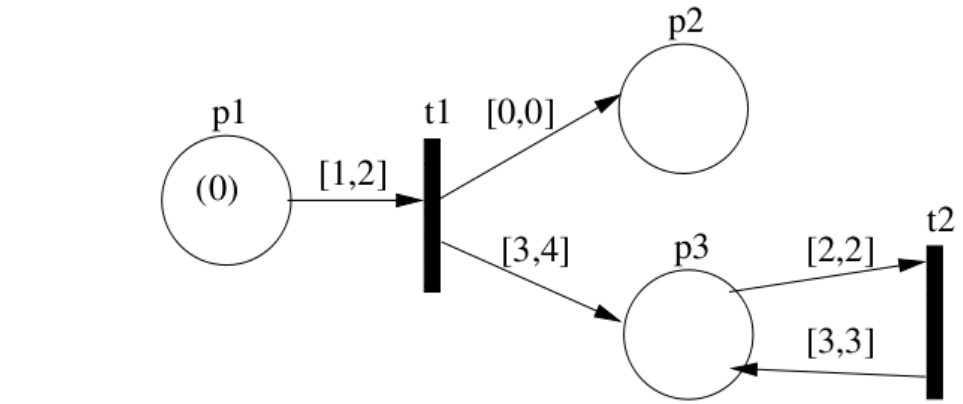
Sei der *Event* (t, M_{cons}, M_{prod}) für das *Marking* $M \in \mathbb{M}^*$ nach *Delay* δ aktiviert, dann ist das *Truncated Marking* $M' \in \mathbb{M}^*$ was durch das *Firing* eines *Events* entsteht:
$M' = age(M, \delta)|_\omega \setminus_{min} M_{cons} + M_{prod}$.

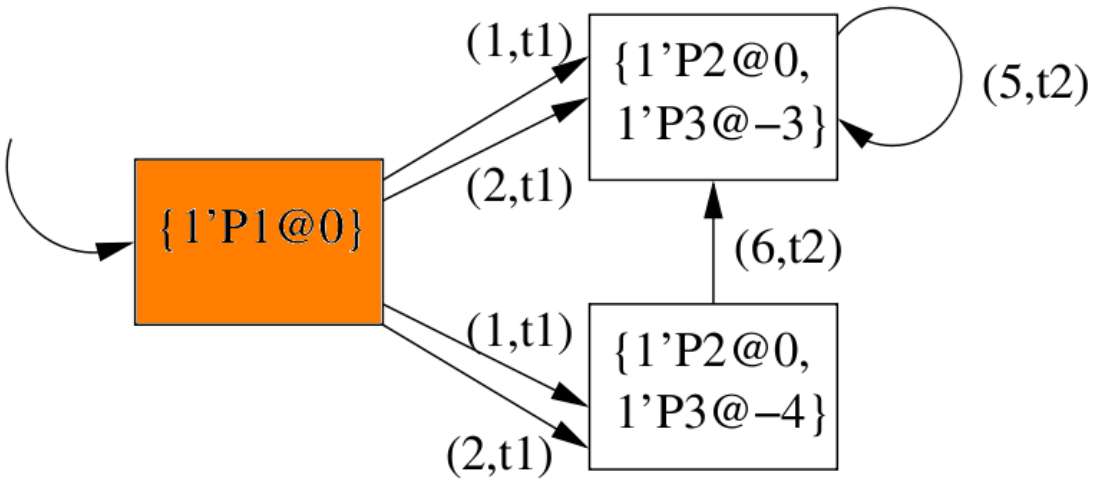
Truncated Statespace

Der *Truncated Statespace*, welcher durch die *Truncation* entsteht ist nicht mehr unendlich:

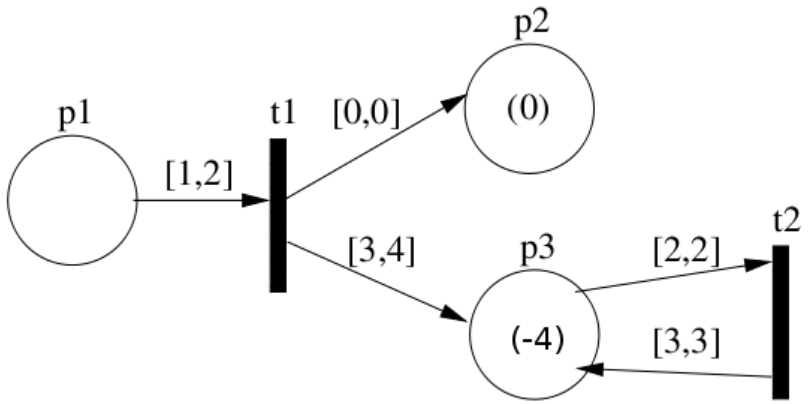


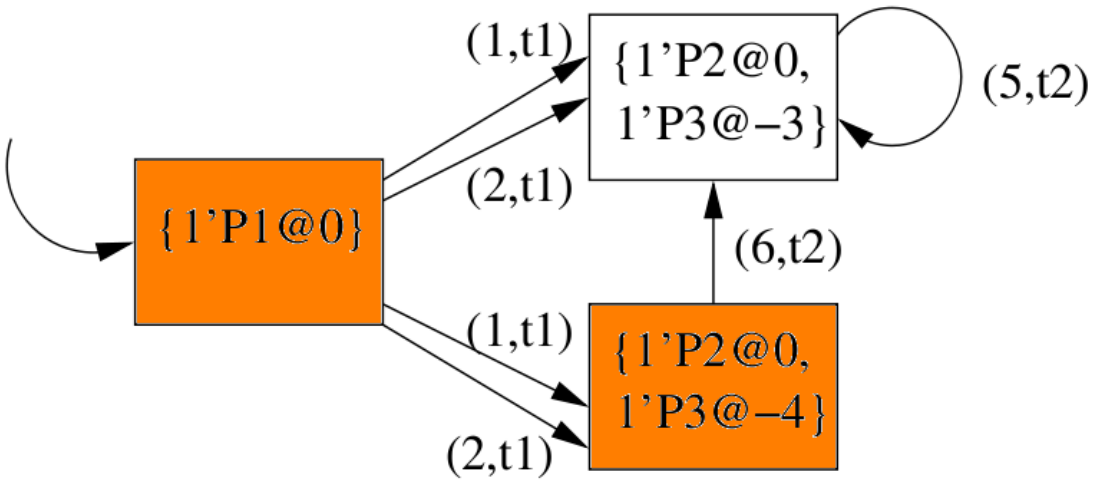
Beispiel


p1
(0)
t1
[1,2]
[0,0]
p2
[3,4]
p3
[2,2]
t2
[3,3]

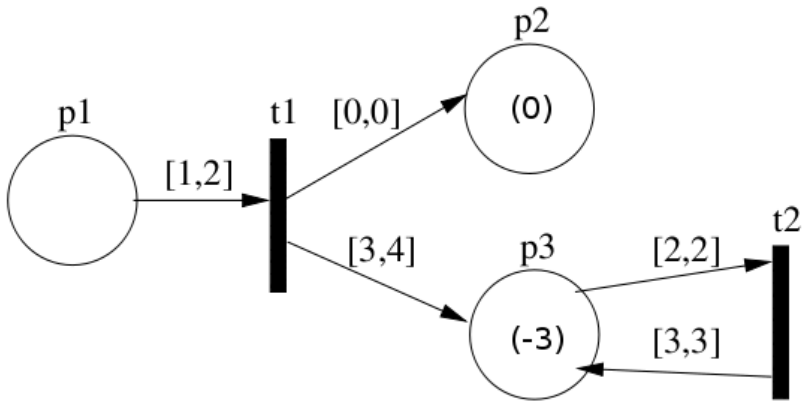

{1'P1@0}
(1,t1)
(2,t1)
{1'P2@0,
1'P3@−3}
(5,t2)
(6,t2)
(1,t1)
(2,t1)
{1'P2@0,
1'P3@−4}

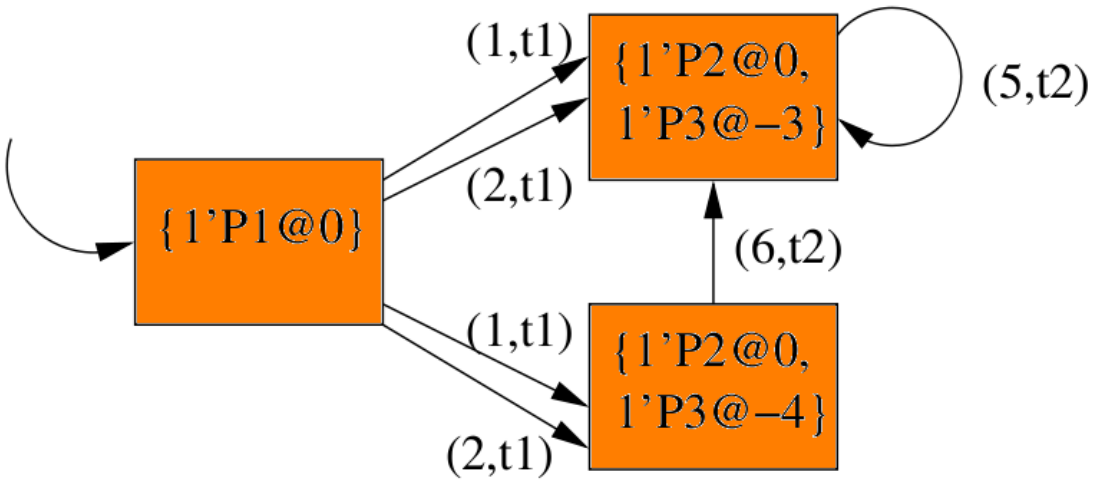
Beispiel


p2
p1
t1
[0,0]
(0)
[1,2]
t2
[3,4]
p3
[2,2]
(-4)
[3,3]


(1,t1)
{1'P2@0,
1'P3@−3}
(5,t2)
{1'P1@0}
(2,t1)
(6,t2)
(1,t1)
{1'P2@0,
1'P3@−4}
(2,t1)

Beispiel


p1
t1
[1,2]
[0,0]
p2
(0)
[3,4]
p3
(-3)
[2,2]
t2
[3,3]


{1'P1@0}
(1,t1)
(2,t1)
{1'P2@0,
1'P3@−3}
(5,t2)
(6,t2)
(1,t1)
(2,t1)
{1'P2@0,
1'P3@−4}

Certain Events

Events, die immer den *Upper Bound* jedes Intervals konsumieren feuern auf jeden Fall sobald das *Event* aktiviert ist.

Definition: *Certain Event*

Ein *Event* (t, M_{cons}, M_{prod}) ist *Certain* (sicher), genau dann wenn $\forall p \in P : Sei(p, t, [a_i, b_i])_{i=1}^{n} \subseteq A$ die komplette Multimenge der *Input Arcs* von t. Dann gilt $M_{cons}(p) = b_i{}_{i=1}^{n}$.

Earliest Certain Events

Das *Delay*, was ab einem *Marking* M den ersten *Certain Event* aktiviert kann man als obere Grenze für das *Delay* eines *States* ansehen.

Definition: *Earliest Certain Events*

Das oder die *Certain Event(s)*, die von einem *Marking* $M \in \mathbb{M}^*$ nach dem kürzesten *Delay* δ_0 aktiviert sind, bezeichnet man als *Earliest Certain Events*.
Das impliziert, dass jeder andere *Certain Event* erst nach δ_1 aktiveriert wird mit $\delta_1 \geq \delta_0$.

Earliest Certain Events

Das *Delay*, was ab einem *Marking* M den ersten *Certain Event* aktiviert kann man als obere Grenze für das *Delay* eines *States* ansehen.

Definition: *Earliest Certain Events*

Das oder die *Certain Event(s)*, die von einem *Marking* $M \in \mathbb{M}^*$ nach dem kürzesten *Delay* δ_0 aktiviert sind, bezeichnet man als *Earliest Certain Events*.
Das impliziert, dass jeder andere *Certain Event* erst nach δ_1 aktiveriert wird mit $\delta_1 \geq \delta_0$.

Definition: *Earliest Certain Event Time*

Die *Earliest Certain Event Time* eines *Markings* M ist das *Delay* δ, nach dem die *Earliest Certain Events* von M aktiviert sind.

Absolute Zeit

Trace

Um von der relativen zu einer absoluten Zeit zu kommen vollzieht man die durchlaufenen *States* und die dazu führenden *Events* nach.

Definition: *Trace*

Sei (S, E, Δ, S_1) ein *Statespace*. Ein *Trace* ist eine Sequenz strikt alternierender *States* und *Events*, geschrieben $[s_0, e_1, s_1, e_2, s_2, ...]$.
Dabei gilt $\forall i \in [0, \infty] : (s_i, e_{i+1}, s_{i+1}) \in \Delta$.
Die Menge aller *Traces* bezeichnen wir als $\mathbb{T}_R$.

Aggregate Delay Function

Die *Aggregate Delay Function* addiert alle *Delays* eines *Traces* bis zu einer bestimmten Stelle n auf.

Definition: *Aggregate Delay Function*

Die *Aggregate Delay Function* ist eine Zuordnung:
clock : $\mathbb{T}_R x \mathbb{N} \rightarrow \mathbb{N}$, für die gilt:

$$\forall \xi = [s_0, (\delta_1, t_1), s1, ...], n \leq \frac{|\xi|}{2} : \textbf{clock}(\xi, n) = \sum_{i=1}^{n} \delta_i$$

Definition Absolute Zeit

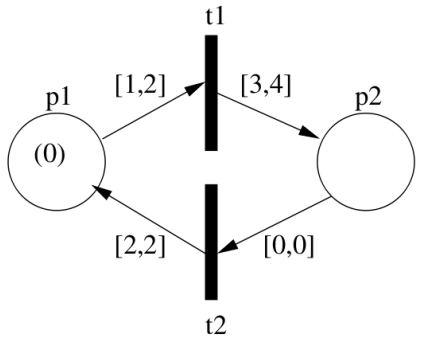
Der *Absolute Time Trace* kann wie folgt aus dem relativen mittels der *Aggregate Delay Function* erstellt werden.

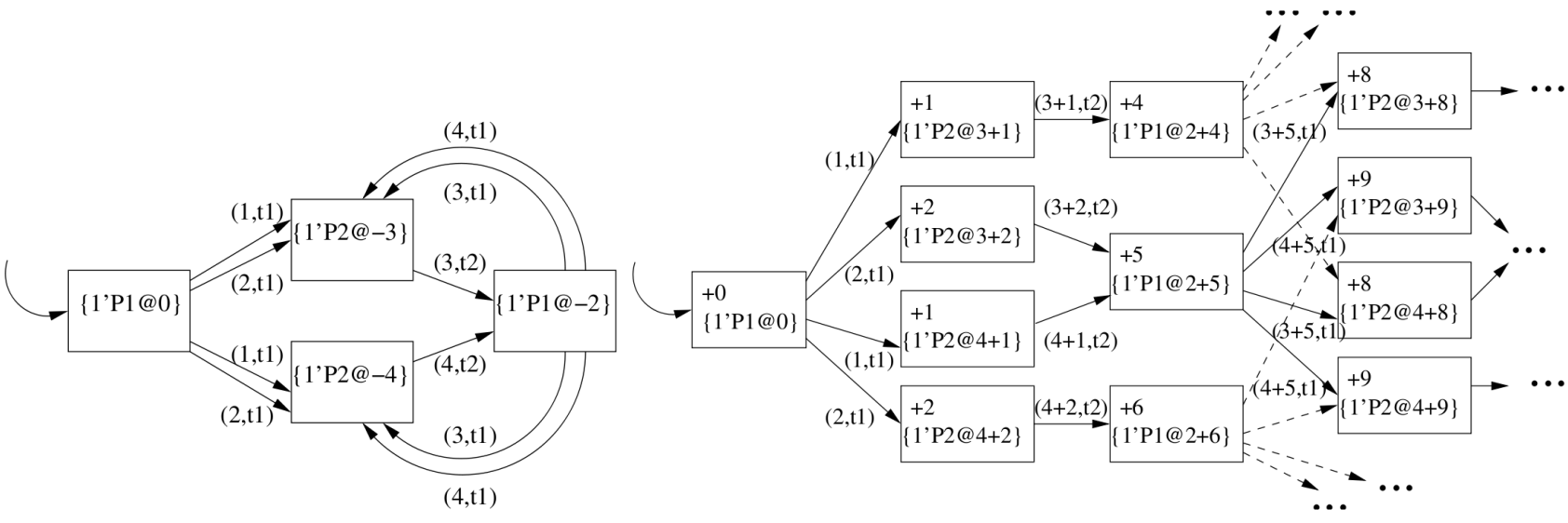
Definition: *Absoulte Time Trace*

Sei $\xi^* = [s_0, (\delta_1, t_1), s_1, ...]$ ein *Relative Time Trace*. Dann ist der zugehörige *Absolute Time Trace* definiert als:
$\textbf{absolute}(\xi^*) = [s_0, (\textbf{clock}(\xi^*, 1), t_1),$
$\textbf{age}(s_1, \textbf{clock}(\xi^*, 2), t_2),$
$\textbf{age}(s_2, \textbf{clock}(\xi^*, 2)), ...]$

Hiermit kann man zu jedem *Event* die absolut vergangene Zeit ausrechnen.

Beispiel


t1
p1
[1,2]
[3,4]
p2
(0)
[2,2]
[0,0]
t2


{1'P1@0}
{1'P2@−3}
{1'P2@−4}
{1'P1@−2}
(1,t1)
(2,t1)
(3,t2)
(4,t2)
(3,t1)
(4,t1)
+0
{1'P1@0}
+1
{1'P2@3+1}
+2
{1'P2@3+2}
+1
{1'P2@4+1}
+2
{1'P2@4+2}
+4
{1'P1@2+4}
+5
{1'P1@2+5}
+6
{1'P1@2+6}
+8
{1'P2@3+8}
+9
{1'P2@3+9}
+8
{1'P2@4+8}
+9
{1'P2@4+9}
(3+1,t2)
(3+2,t2)
(4+1,t2)
(4+2,t2)
(3+5,t1)
(4+5,t1)

Prototypes and Related Work

Prototypes

Die im Paper erwähnte MOSES Implementation lief auf keinem unserer Systeme, weswegen eine Aussage zu dieser Software leider nicht möglich ist.

Erwähnenswert ist, dass die *Statespaces* für relative Zeiten etwa linear wachsen, für absolute Zeiten jedoch exponentiell.

packets	1	2	3	4	5	6	7
absolute	70 311	2.240 16.083	24.231 230.749	84.796 855.451	200.453 2.086.960	378.978 4.018.015	620.371 6.648.616
relative	96 421	1.093 11.558	2.107 22.841	3.121 34.124	4.135 45.407	5.149 56.690	6.163 67.973

Related Work

Ähnliche Ansätze sind auch zu finden in:

- interval timed coloured Petri nets (van der Aalst) - Idee der *Timestamps*
- Recoverabilityof Communication Protocols (P. Merlin und D.J. Farber) - Relative statt absolute Zeit
- Lazy transition systems: application to timing optimization of asynchronous circuits (viele, siehe Paper) - Relative Zeit in Bezug auf Erreichbarkeitsanalyse

Vielen Dank für Ihre Aufmerksamkeit