Systèmes à événements discrets à paramètres temporels

Olivier H. Roux

École Centrale de Nantes / IRCCyN

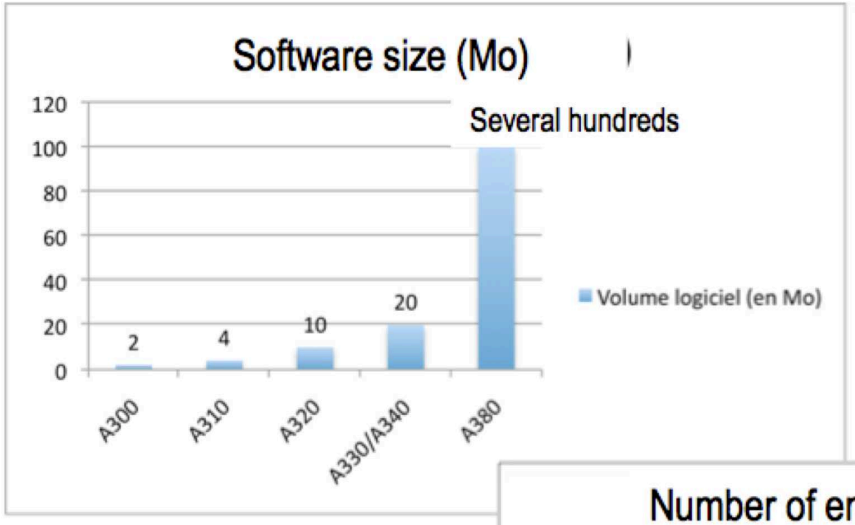
Modélisation des Systèmes Réactifs (MSR'13)

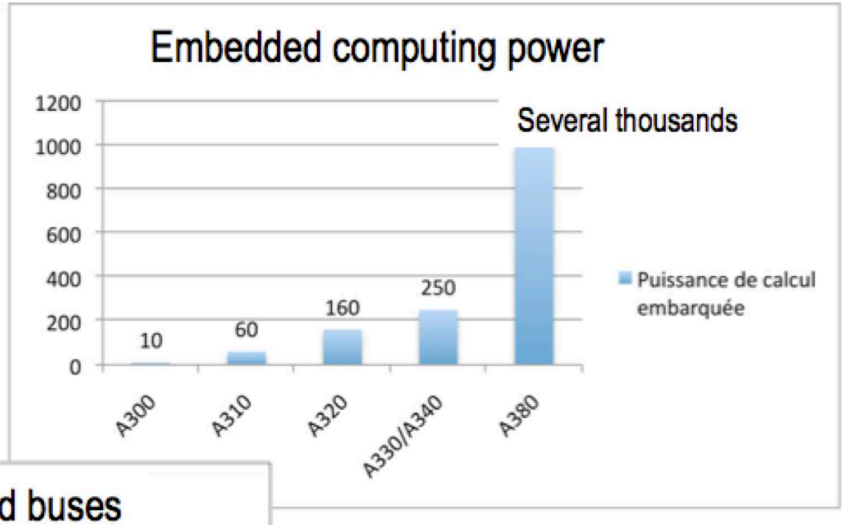
Rennes, 13 novembre 2013

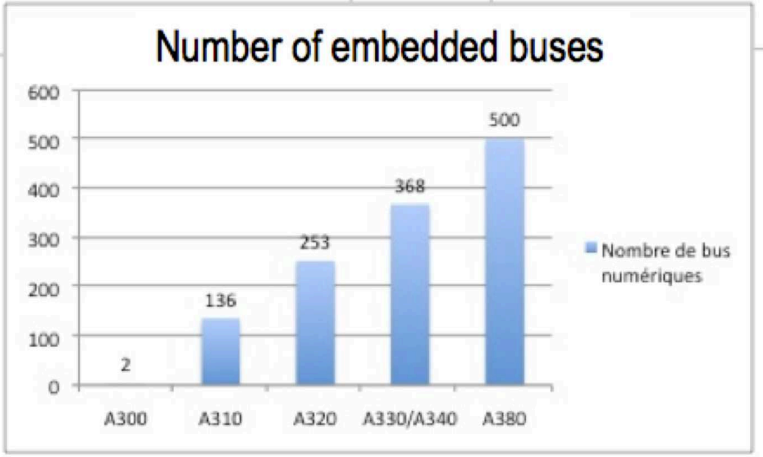
Avant propos

Résultats issus de travaux effectués avec :

- Aleksandra Jovanović
- Didier Lime
- Claude Martinez
- Louis-Marie Traonouez

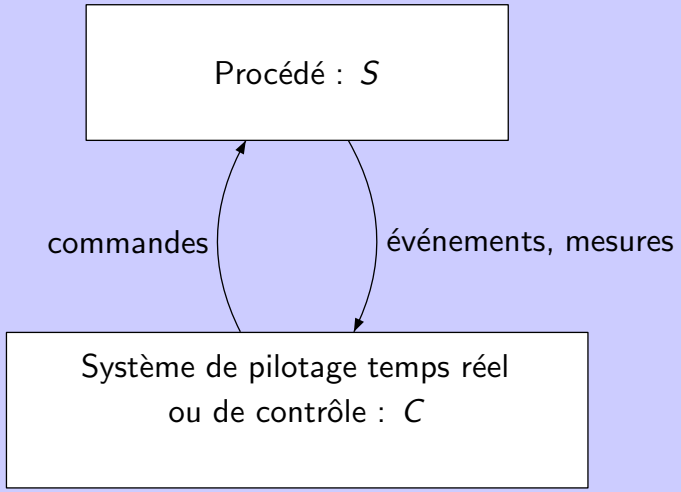




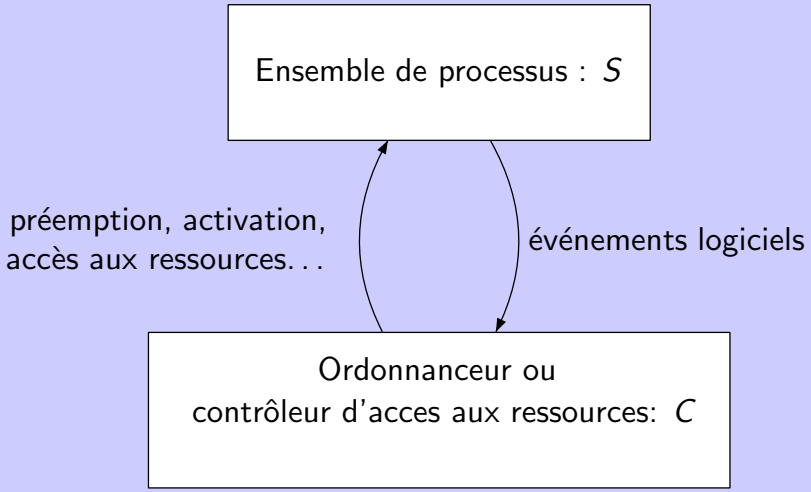


- Exponential increase of embedded systems in the aircraft
  - Computers, buses, middleware, software

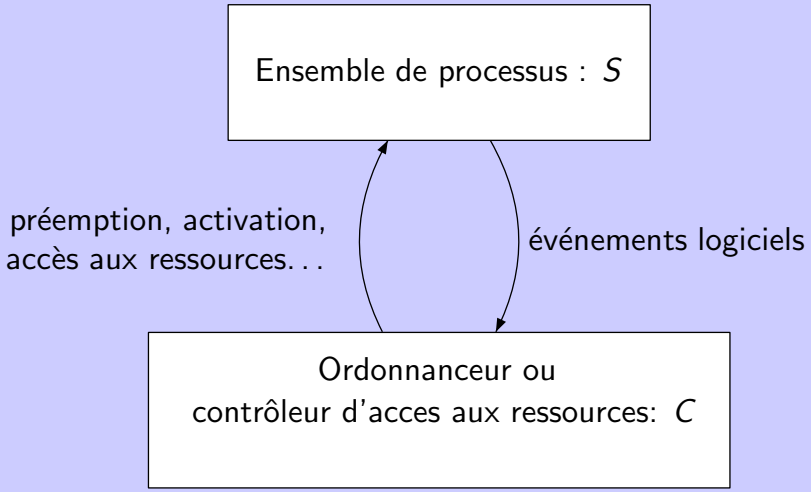
Systèmes Réactifs



Systèmes Réactifs



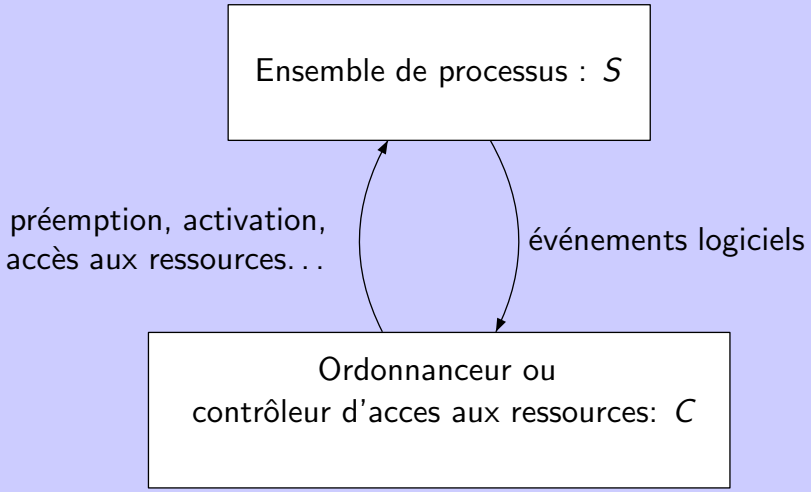
Systèmes Réactifs



Spécifications :

- Spécification fonctionnelle
- Spécification non fonctionnelle
  - **ressources** (Energie, processeur, mémoire ...)
  - **temporelle**

Systèmes Réactifs



Spécifications :

- Spécification fonctionnelle
- Spécification non fonctionnelle
  - **ressources** (Energie, processeur, mémoire ...)
  - **temporelle**

Objectif :

Les spécifications doivent être respectées

En entrée :

- Modélisation du système S
- Spécification des propriétés de contrôle : φ

En entrée :

- Modélisation du système S
- Spécification des propriétés de contrôle : φ

Problème de la vérification de modèle (model-checking) :

- Modélisation du système contrôlé $S \parallel C$.

$$\text{Est-ce que } S \parallel C \models \varphi \quad ? \tag{1}$$

En entrée :

- Modélisation du système S
- Spécification des propriétés de contrôle : φ

Problème de la vérification de modèle (model-checking) :

- Modélisation du système contrôlé $S \parallel C$.

$$\text{Est-ce que } S \parallel C \models \varphi \quad ? \tag{1}$$

Problème du contrôle :

$$\text{Existe-t-il } C \text{ tel que } S \parallel C \models \varphi \quad ? \tag{2}$$

- Si oui : synthétiser ce contrôleur.

En entrée :

- Modélisation du système S
- Spécification des propriétés de contrôle : φ

Problème de la vérification de modèle (model-checking) :

- Modélisation du système contrôlé $S \parallel C$.

$$\text{Est-ce que } S \parallel C \models \varphi \quad ? \tag{1}$$

Problème du contrôle :

$$\text{Existe-t-il } C \text{ tel que } S \parallel C \models \varphi \quad ? \tag{2}$$

- Si oui : synthétiser ce contrôleur.

Problème :

Si la réponse est négative, que fait-on ?

Problème de la vérification paramétrée :

- Modélisation paramétrée du système contrôlé $S(p) \parallel C(p)$.

Existe-t-il des valeurs $\nu(p)$ des paramétres p

$$\text{tels que } S(\nu(p)) \parallel C(\nu(p)) \models \varphi(\nu(p)) \quad ? \tag{3}$$

- Si oui : synthétiser l'ensemble des valeurs $\nu(p)$.

Problème de la vérification paramétrée :

- Modélisation paramétrée du système contrôlé $S(p) \parallel C(p)$.

Existe-t-il des valeurs $\nu(p)$ des paramétres p

$$\text{tels que } S(\nu(p)) \parallel C(\nu(p)) \models \varphi(\nu(p)) \quad ? \tag{3}$$

- Si oui : synthétiser l'ensemble des valeurs $\nu(p)$.

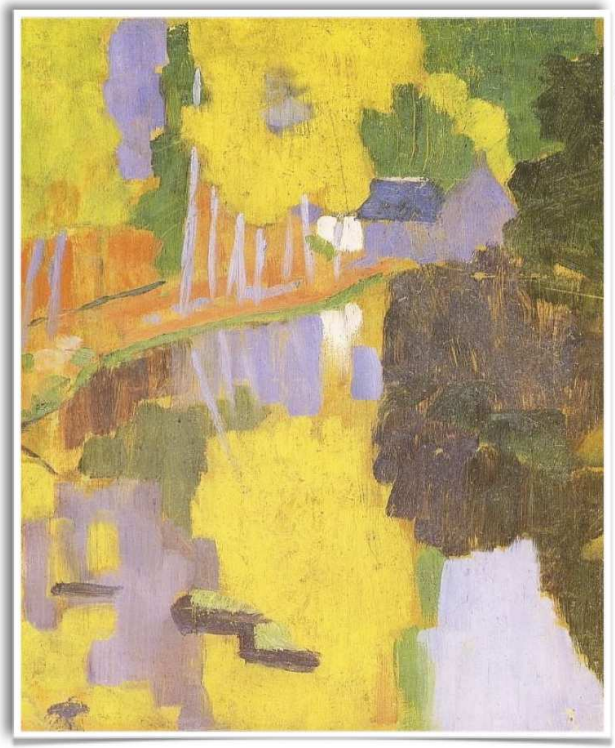
Problème du contrôle paramétré :

Existe-t-il des valeurs $\nu(p)$ des paramétres p et un contrôleur $C((\nu(p))$

$$\text{tels que} S(\nu(p)) \parallel C(\nu(p)) \models \varphi(\nu(p)) \quad ? \tag{4}$$

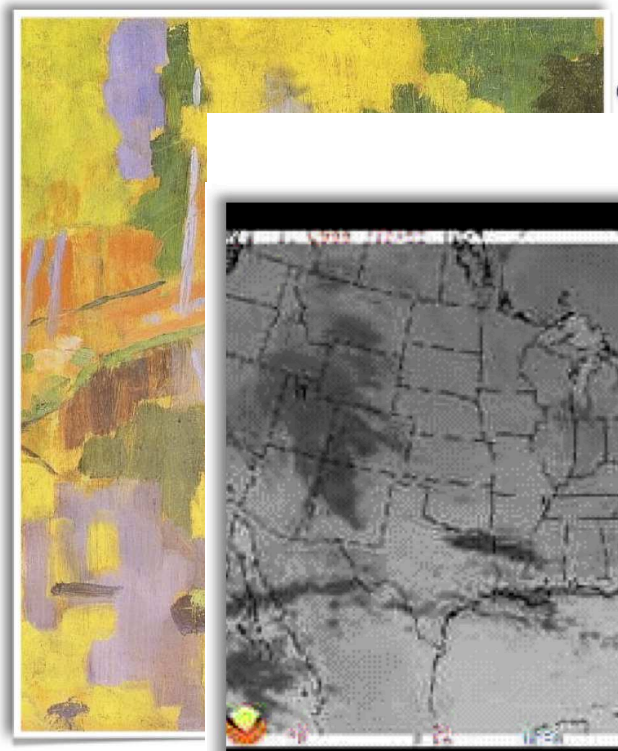
- Si oui : synthétiser l'ensemble des valeurs $\nu(p)$ et le contrôleur $C(\nu(p))$.

Modéliser c'est abstraire



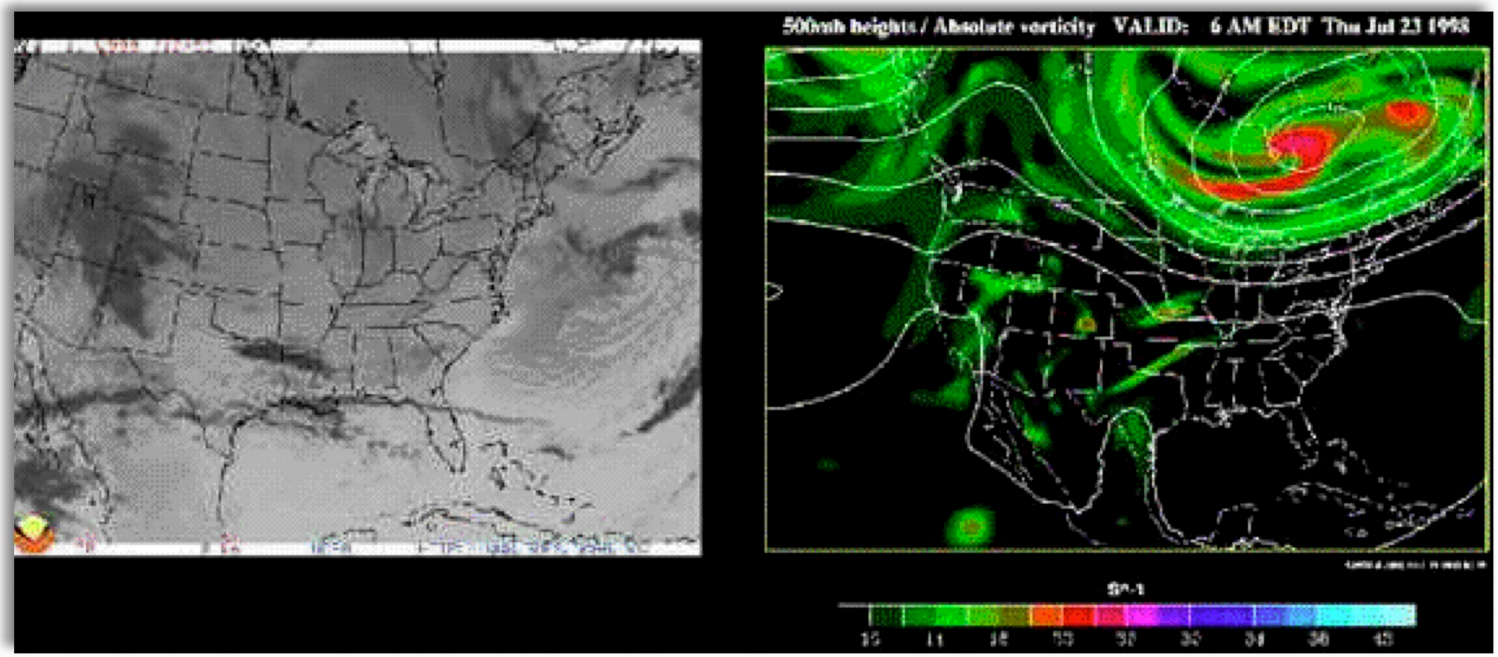
Le Talisman
de P. Sérusier
(1888)

Modéliser c'est abstraire

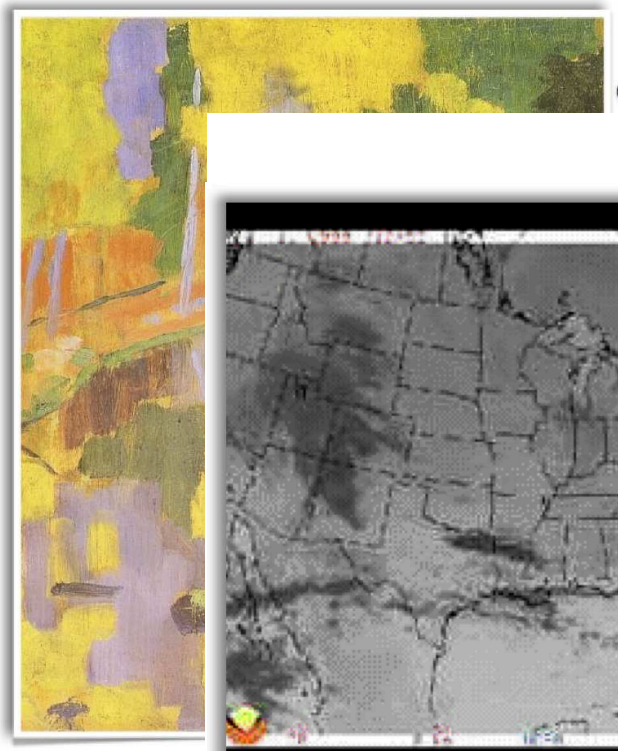


Le Talisman
de P. Sérusier

Carte météo

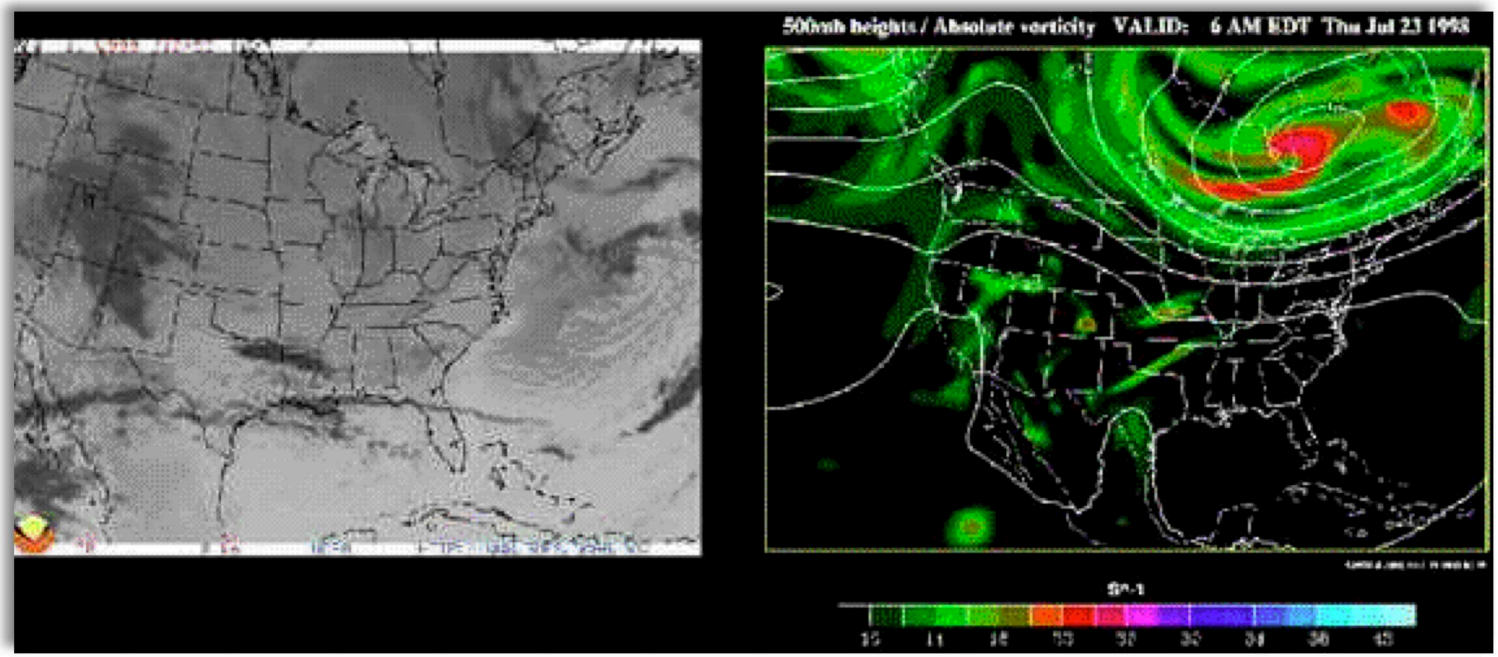


Modéliser c'est abstraire



Le Talisman
de P. Sérusier

Carte météo



Compromis **Expressivité du modèle - Ce que l'on peut (veut) faire avec**

Applications **temps réel** :

- Spécification **fonctionnelle**
- Spécification **temporelle**

$\Rightarrow$ temps logique généralement insuffisant.

Applications **temps réel** :

- Spécification **fonctionnelle**
- Spécification **temporelle**

$\Rightarrow$ temps logique généralement insuffisant.

Modèle formel

Extensions **temporisées** des

- Algèbre de processus
- Réseau de Petri
- Automate d'états fini

Applications **temps réel** :

- Spécification **fonctionnelle**
- Spécification **temporelle**

$\Rightarrow$ temps logique généralement insuffisant.

Modèle formel

Extensions **temporisées** des

- Algèbre de processus
- Réseau de Petri
- Automate d'états fini

Formalisation de la spécification

- observateurs
- logique temporelle (LTL, CTL)
- logique temporelle temporisée (TCTL)

Applications **temps réel** :

- Spécification **fonctionnelle**
- Spécification **temporelle**

$\Rightarrow$ temps logique généralement insuffisant.

Modèle formel

Extensions **temporisées** des

- Algèbre de processus
- Réseau de Petri
- Automate d'états fini

Formalisation de la spécification

- observateurs
- logique temporelle (LTL, CTL)
- logique temporelle temporisée (TCTL)

Contraintes temporelles paramétrées sur

le modèle et/ou la spécification

Outline

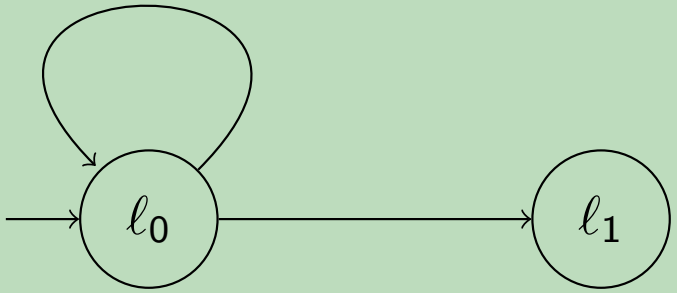
Outline

Timed Automata

A timed automaton is made of

- a transition system

Example



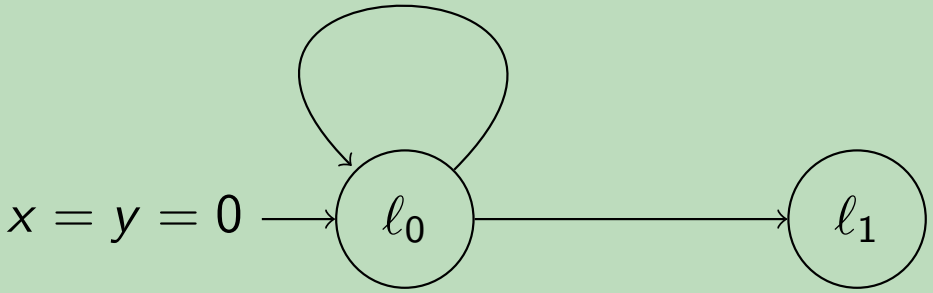
Timed Automata

A timed automaton is made of

- a transition system
- a set of clocks

Example

Clocks (x, y):



Timed Automata

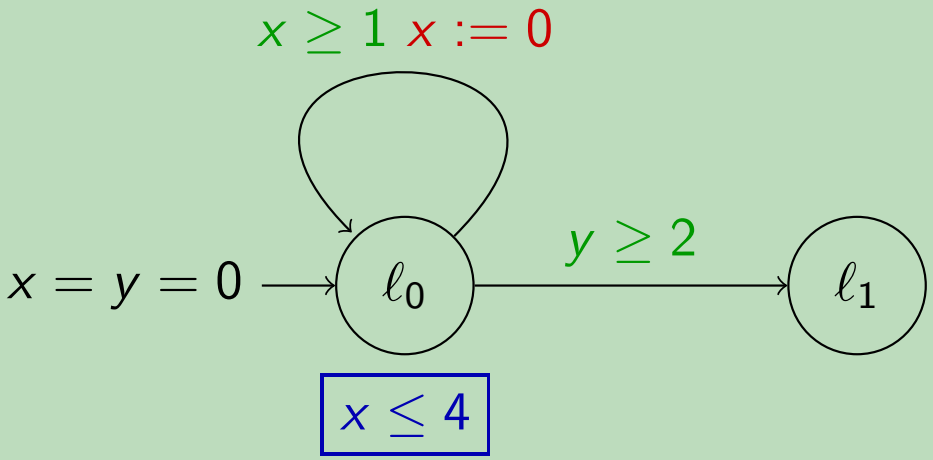
A timed automaton is made of

- a transition system
- a set of clocks
- timing constraints on locations and edges

Example

Clocks (x, y):

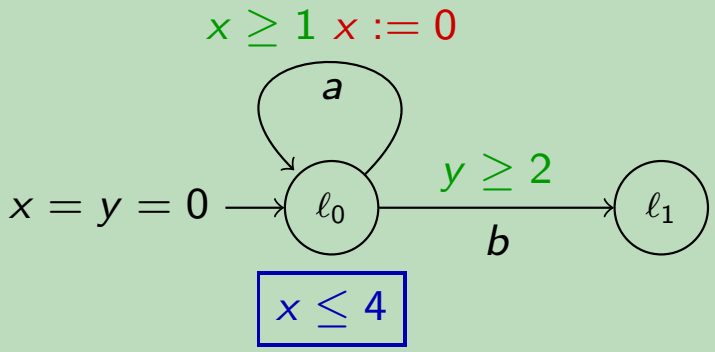
- guards
- invariants
- resets

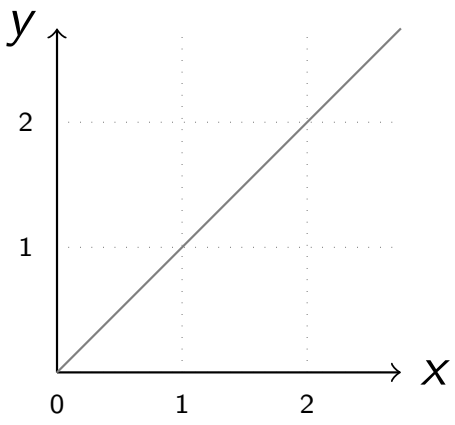


Timed Automata (TA) [AD94]

Exemple

- a transition system
- a set of clocks (here: x, y)
- timing constraints

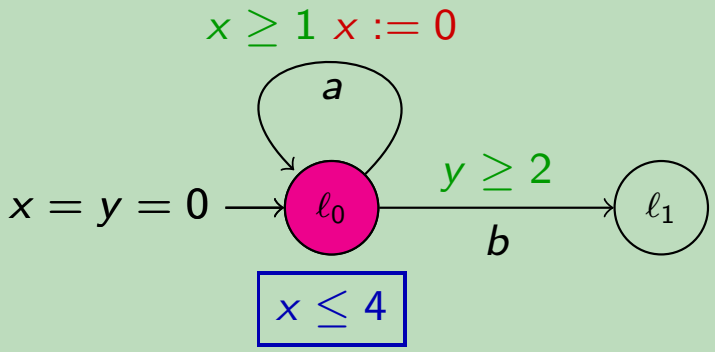


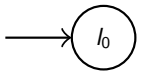


Timed Automata (TA) [AD94]

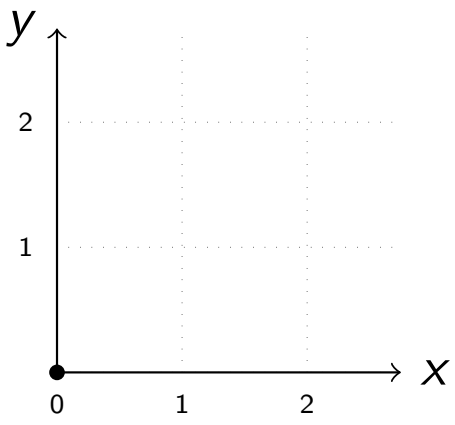
Exemple

- a transition system
- a set of clocks (here: x, y)
- timing constraints





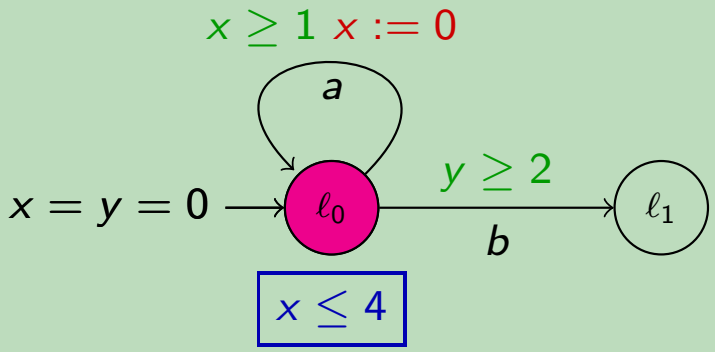
$(0, 0)$

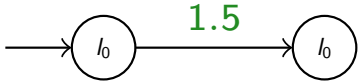


Timed Automata (TA) [AD94]

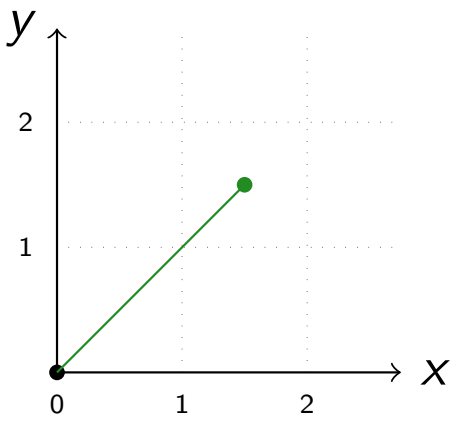
Exemple

- a transition system
- a set of clocks (here: x, y)
- timing constraints





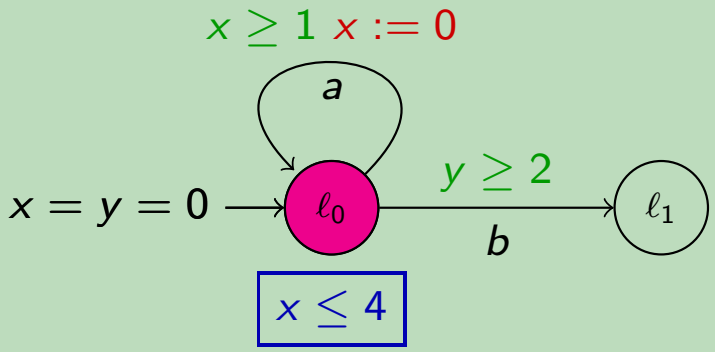
$(0, 0)$ $(1.5, 1.5)$

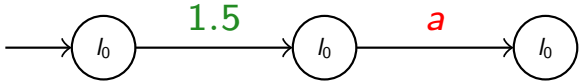


Timed Automata (TA) [AD94]

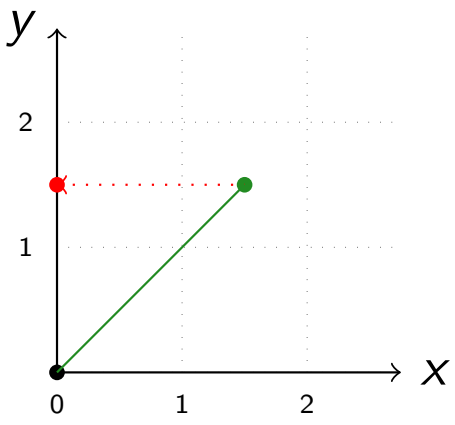
Exemple

- a transition system
- a set of clocks (here: x, y)
- timing constraints





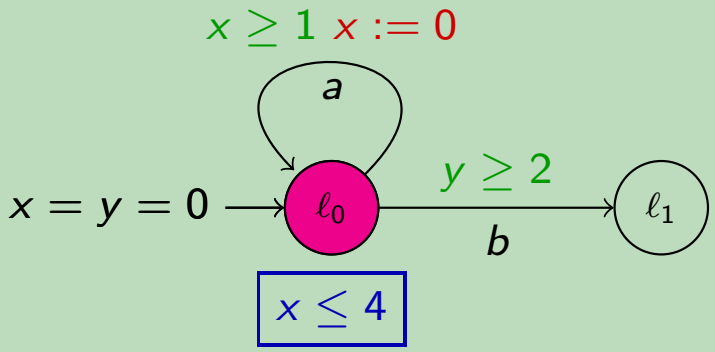
$(0, 0)$ $(1.5, 1.5)$ $(0, 1.5)$

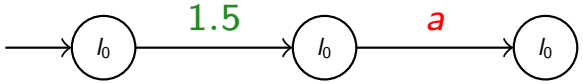


Timed Automata (TA) [AD94]

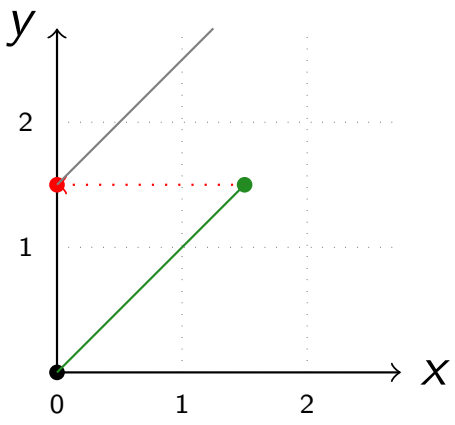
Exemple

- a transition system
- a set of clocks (here: x, y)
- timing constraints





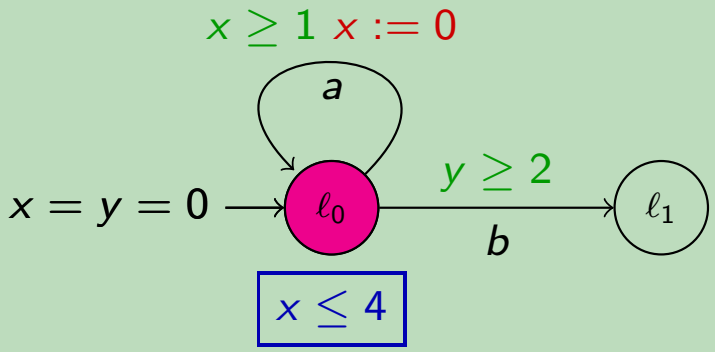
$(0, 0)$ $(1.5, 1.5)$ $(0, 1.5)$

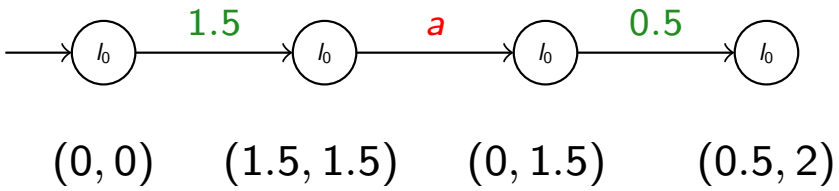


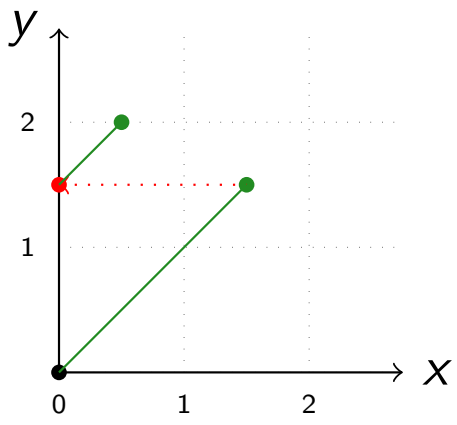
Timed Automata (TA) [AD94]

Exemple

- a transition system
- a set of clocks (here: x, y)
- timing constraints



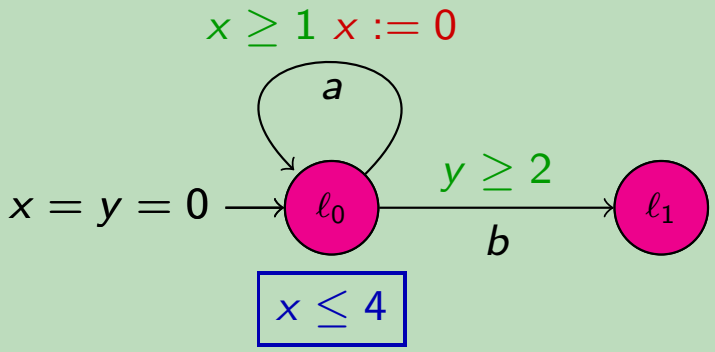


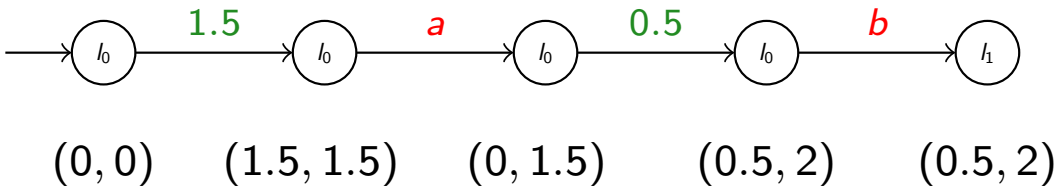


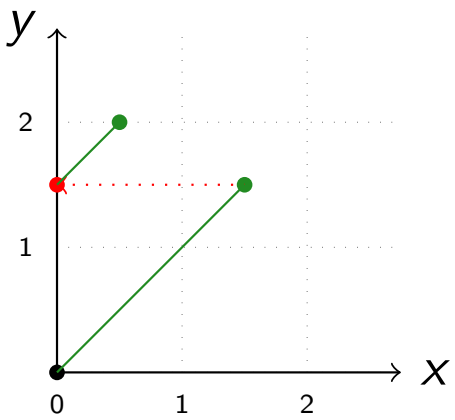
Timed Automata (TA) [AD94]

Exemple

- a transition system
- a set of clocks (here: x, y)
- timing constraints



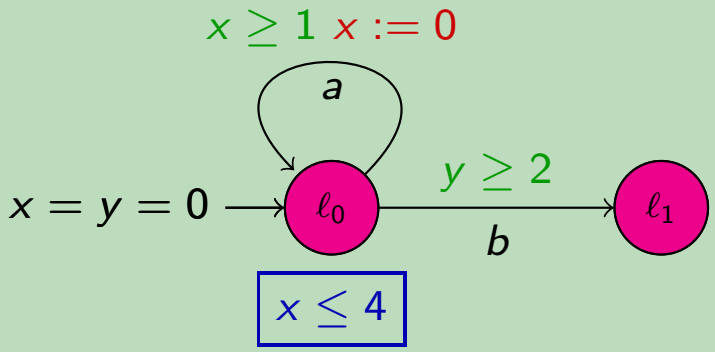


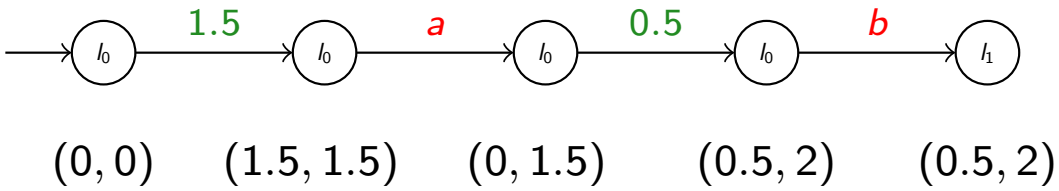


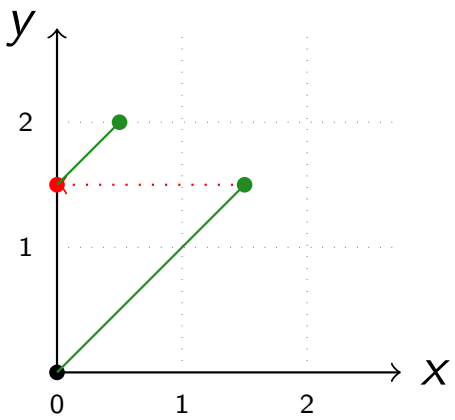
Timed Automata (TA) [AD94]

Exemple

- a transition system
- a set of clocks (here: x, y)
- timing constraints







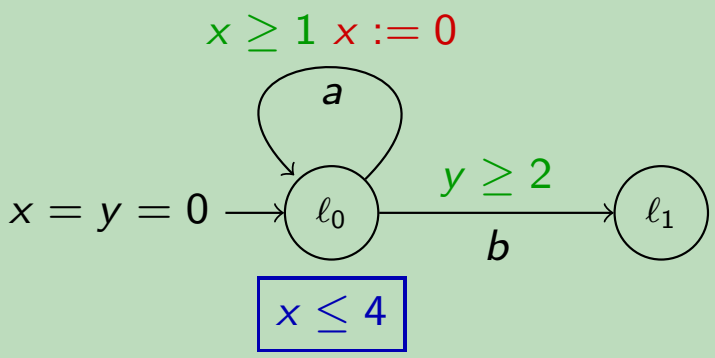
Infinité de branchements et d'états

Abstractions symboliques de l'espace d'états : graphe des **régions**, graphe des **zones**.

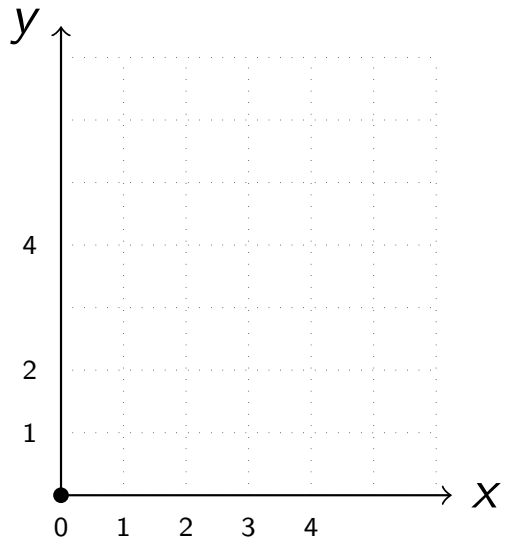
Timed Automata (TA) [AD94]

Exemple

- a transition system
- a set of clocks (here: x, y)
- timing constraints



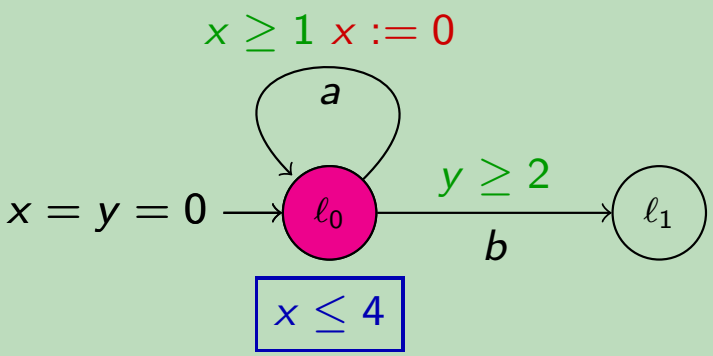
ℓ_0



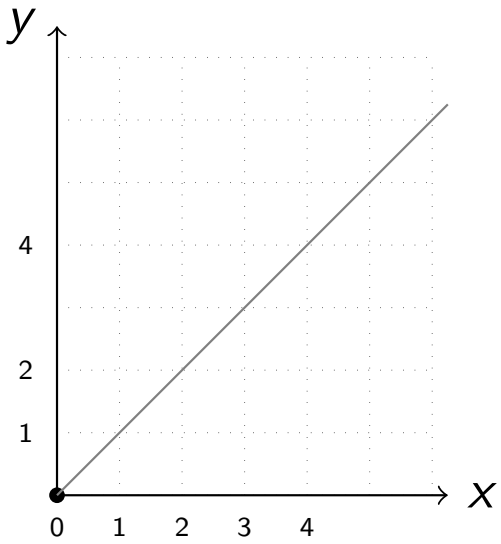
Timed Automata (TA) [AD94]

Exemple

- a transition system
- a set of clocks (here: x, y)
- timing constraints



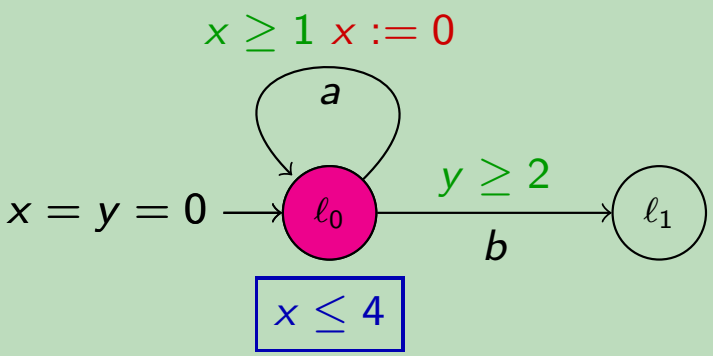
ℓ_0
$x = y$



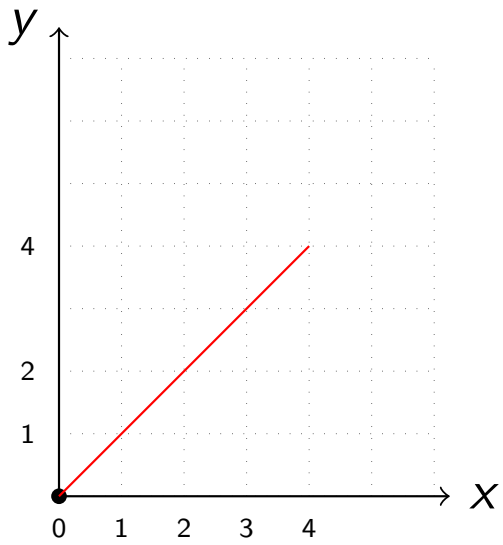
Timed Automata (TA) [AD94]

Exemple

- a transition system
- a set of clocks (here: x, y)
- timing constraints



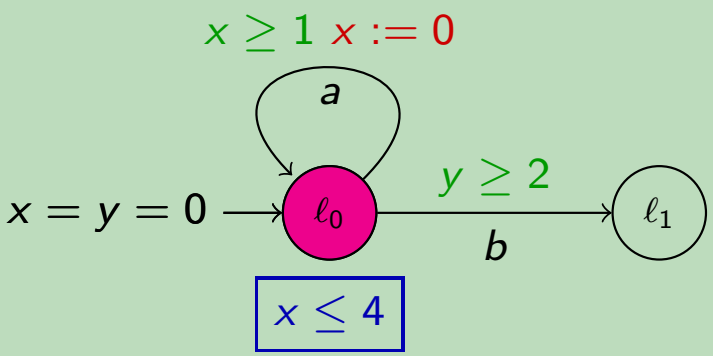
ℓ_0
$x = y$
$x \leq 4$



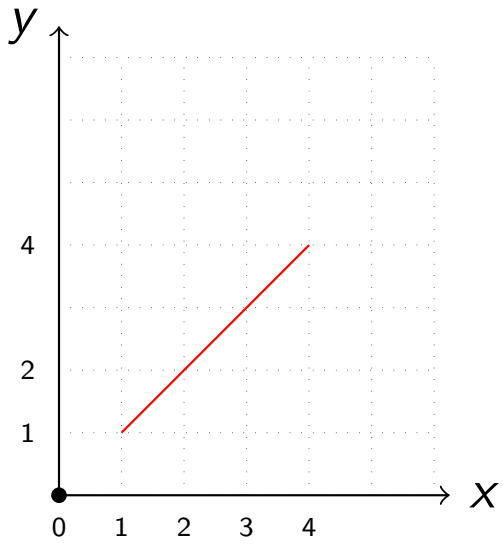
Timed Automata (TA) [AD94]

Exemple

- a transition system
- a set of clocks (here: x, y)
- timing constraints



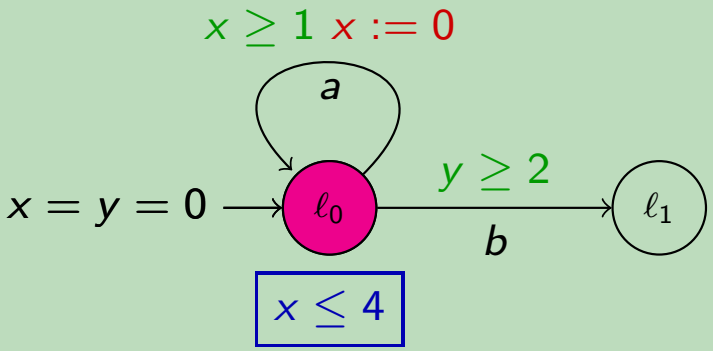
ℓ_0
$x = y \quad \xrightarrow{a}$
$x \leq 4$



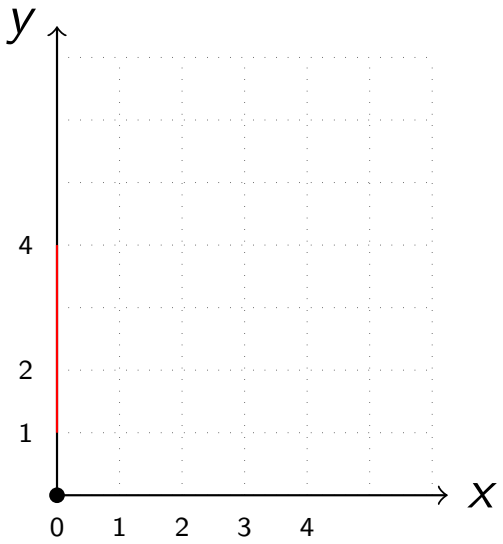
Timed Automata (TA) [AD94]

Exemple

- a transition system
- a set of clocks (here: x, y)
- timing constraints



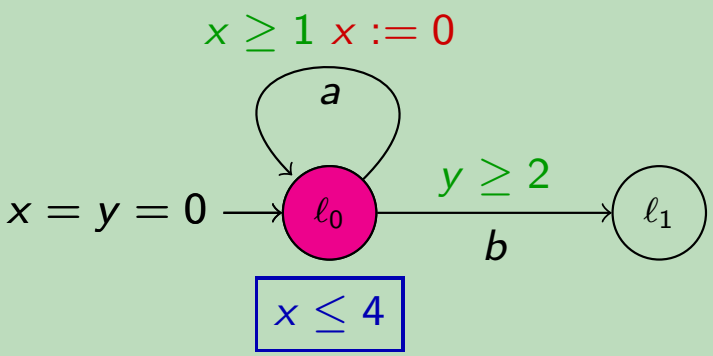
$$\begin{array}{lcl} \ell_0 & & \ell_0 \\ x = y & \xrightarrow{a} & 1 \leq y - x \leq 4 \\ x \leq 4 & & x = 0 \end{array}$$



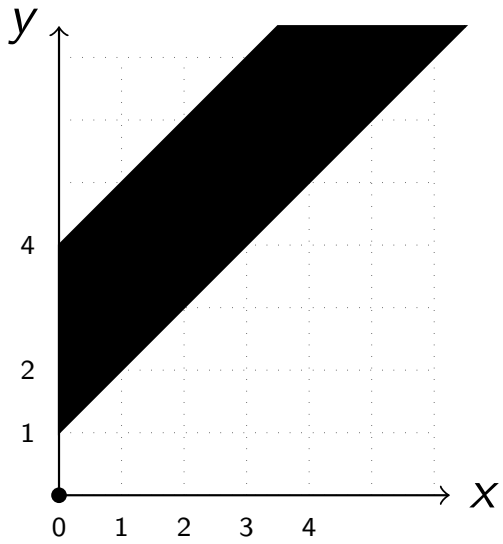
Timed Automata (TA) [AD94]

Exemple

- a transition system
- a set of clocks (here: x, y)
- timing constraints



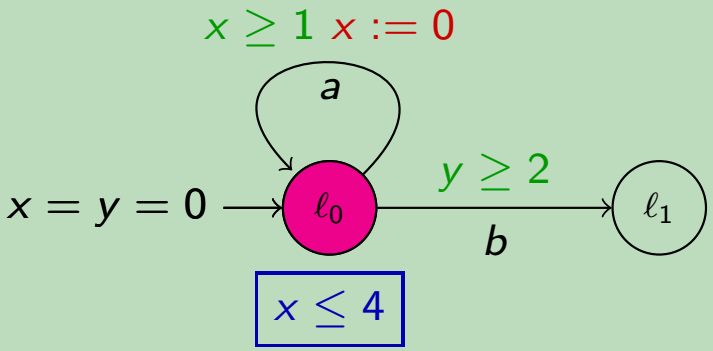
$$\begin{array}{lcl} \ell_0 & & \ell_0 \\ x = y & \xrightarrow{a} & 1 \leq y - x \leq 4 \\ x \leq 4 & & \end{array}$$



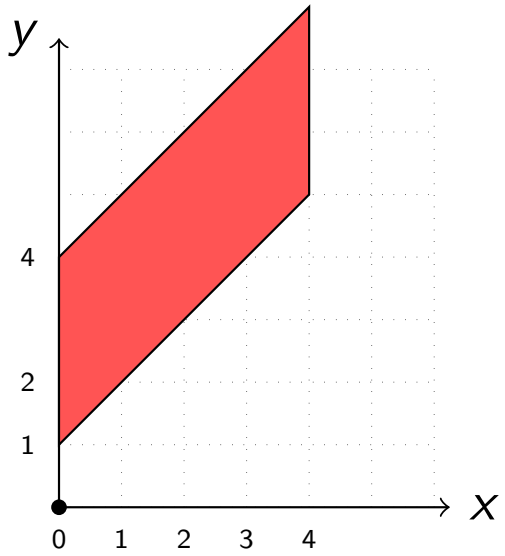
Timed Automata (TA) [AD94]

Exemple

- a transition system
- a set of clocks (here: x, y)
- timing constraints



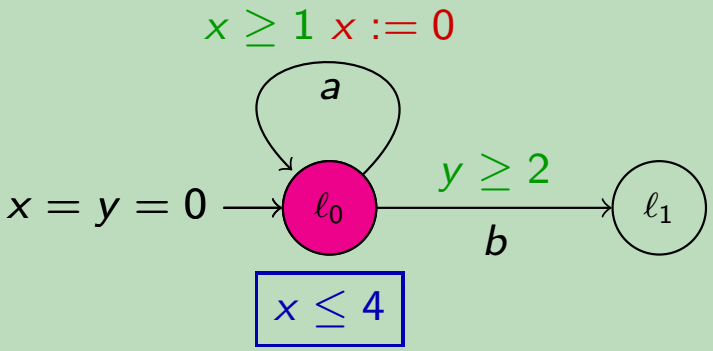
$$\begin{array}{lcl} \ell_0 & & \ell_0 \\ x = y & \xrightarrow{a} & 1 \leq y - x \leq 4 \\ x \leq 4 & & x \leq 4 \end{array}$$



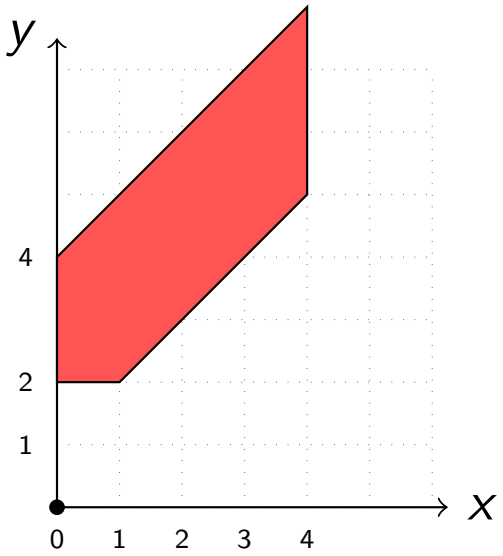
Timed Automata (TA) [AD94]

Exemple

- a transition system
- a set of clocks (here: x, y)
- timing constraints



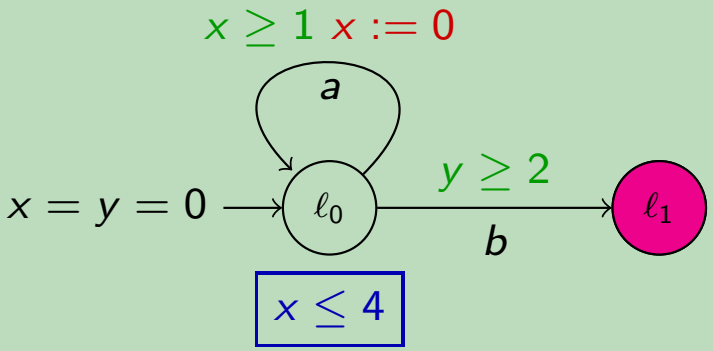
$$\begin{array}{l} \ell_0 \\ x = y \\ x \leq 4 \end{array} \xrightarrow{a} \begin{array}{l} \ell_0 \\ 1 \leq y - x \leq 4 \\ x \leq 4 \end{array} \xrightarrow{b}$$



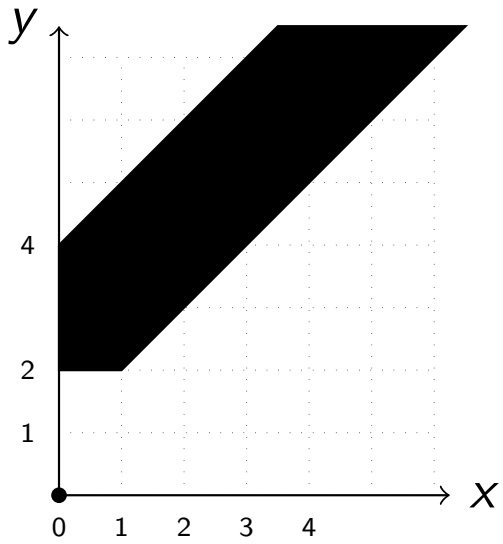
Timed Automata (TA) [AD94]

Exemple

- a transition system
- a set of clocks (here: x, y)
- timing constraints



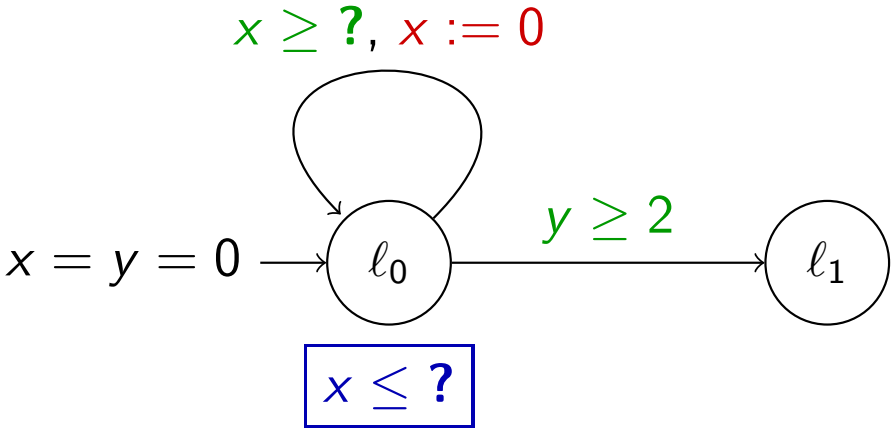
$$\begin{array}{l} \ell_0 \\ x = y \\ x \leq 4 \end{array} \xrightarrow{a} \begin{array}{l} \ell_0 \\ 1 \leq y - x \leq 4 \\ x \leq 4 \end{array} \xrightarrow{b} \begin{array}{l} \ell_1 \\ 1 \leq y - x \leq 4 \\ y \geq 2 \end{array}$$



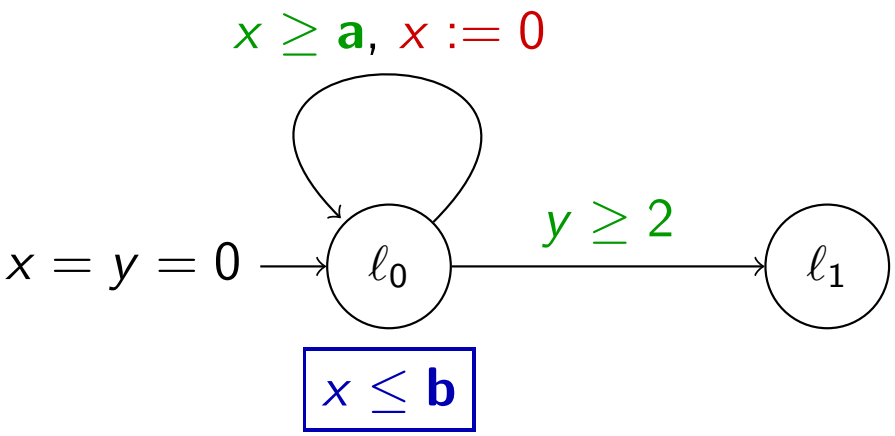
Theorem [AD94]

L'accessibilité est décidable pour les TA et PSPACE-complet.

Parametric Timed Automata (PTA) [AHV93]

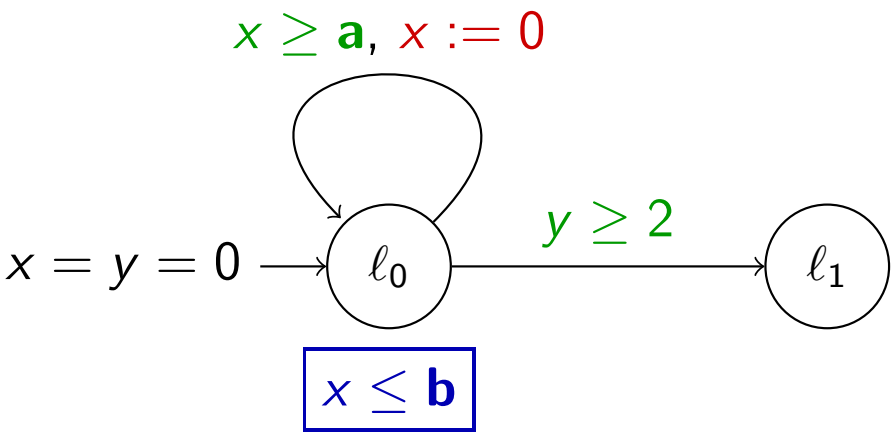


Parametric Timed Automata (PTA) [AHV93]



- Un PTA est un automate temporisé dont les bornes des contraintes d'horloges sont des **expressions linéaires** sur un ensemble fini de paramètres **P**

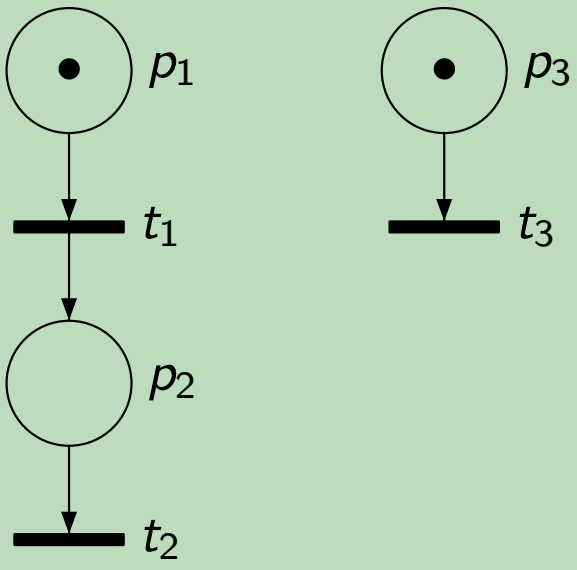
Parametric Timed Automata (PTA) [AHV93]



- Un PTA est un automate temporisé dont les bornes des contraintes d'horloges sont des **expressions linéaires** sur un ensemble fini de paramètres **P**
- Pour un PTA $\mathcal{A}$ et une valuation des parametres $v : P \to \mathbb{Q}$, "$v(\mathcal{A})$" est un a **automate temporisé**.

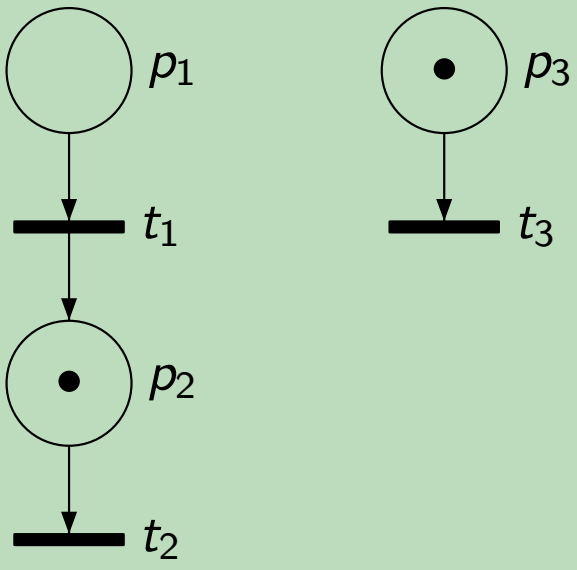
Réseau de Petri T-temporel (TPN)

Exemple



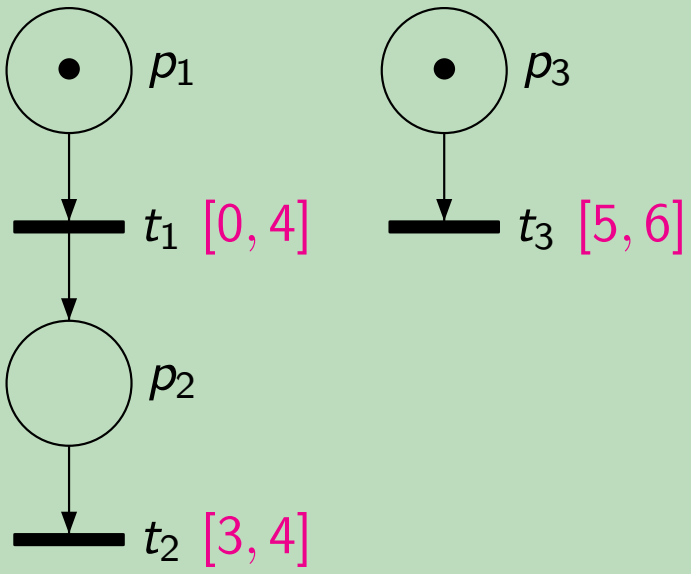
Réseau de Petri T-temporel (TPN)

Exemple



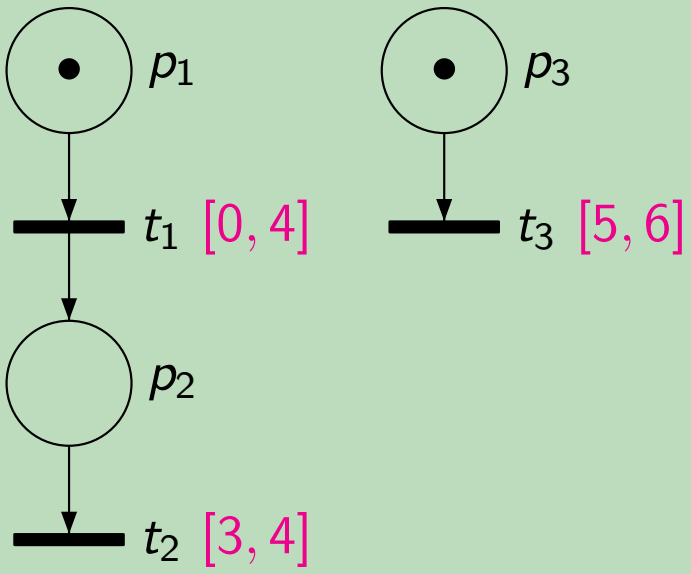
Réseau de Petri T-temporel (TPN)

Exemple



Réseau de Petri T-temporel (TPN)

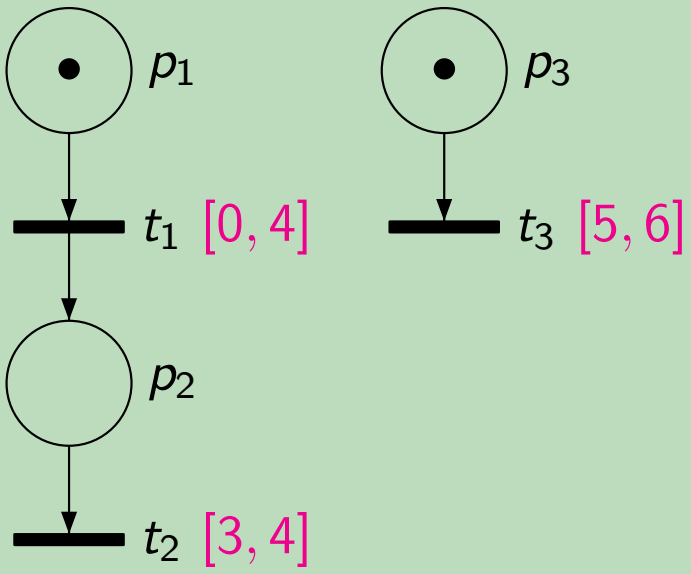
Exemple



$\{p_1, p_3\}$
$\nu(t_1) = 0$
$\nu(t_3) = 0$

Réseau de Petri T-temporel (TPN)

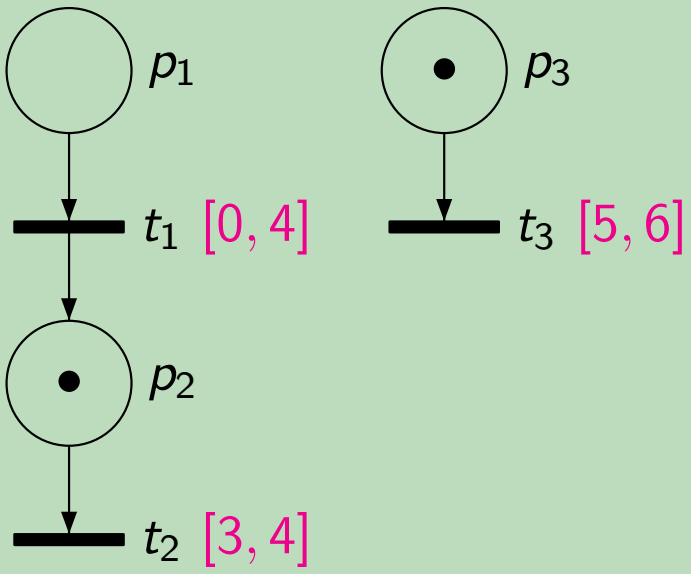
Exemple



$$\begin{array}{lcl} \{p_1, p_3\} & & \{p_1, p_3\} \\ \nu(t_1) = 0 & \xrightarrow{4} & \nu(t_1) = 4 \\ \nu(t_3) = 0 & & \nu(t_3) = 4 \end{array}$$

Réseau de Petri T-temporel (TPN)

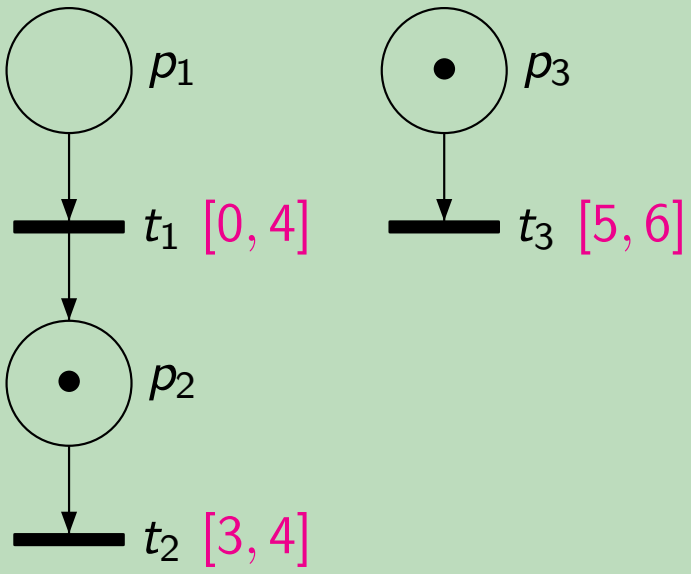
Exemple



$$\begin{array}{c} \{p_1, p_3\} \\ \nu(t_1) = 0 \\ \nu(t_3) = 0 \end{array} \xrightarrow{4} \begin{array}{c} \{p_1, p_3\} \\ \nu(t_1) = 4 \\ \nu(t_3) = 4 \end{array} \xrightarrow{t_1} \begin{array}{c} \{p_2, p_3\} \\ \nu(t_2) = 0 \\ \nu(t_3) = 4 \end{array}$$

Réseau de Petri T-temporel (TPN)

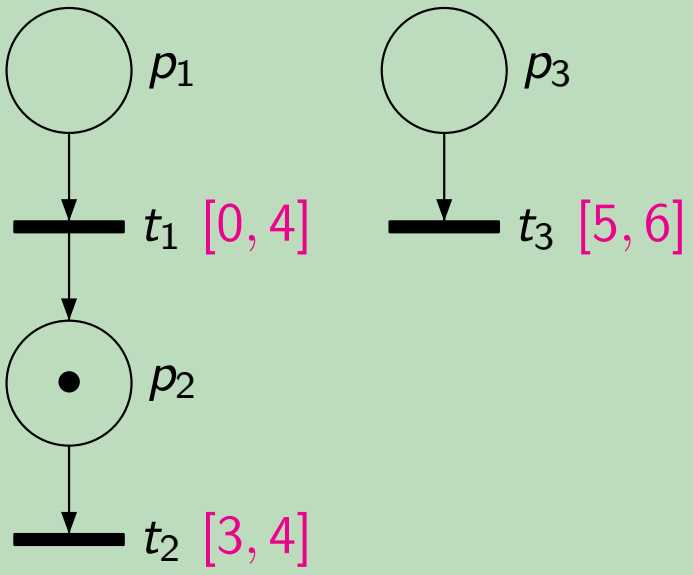
Exemple



$$\begin{array}{c}\{p_1,p_3\}\\ \nu(t_1)=0\\ \nu(t_3)=0\end{array} \xrightarrow{4} \begin{array}{c}\{p_1,p_3\}\\ \nu(t_1)=4\\ \nu(t_3)=4\end{array} \xrightarrow{t_1} \begin{array}{c}\{p_2,p_3\}\\ \nu(t_2)=0\\ \nu(t_3)=4\end{array} \xrightarrow{1.3} \begin{array}{c}\{p_2,p_3\}\\ \nu(t_2)=1.3\\ \nu(t_3)=5.3\end{array}$$

Réseau de Petri T-temporel (TPN)

Exemple



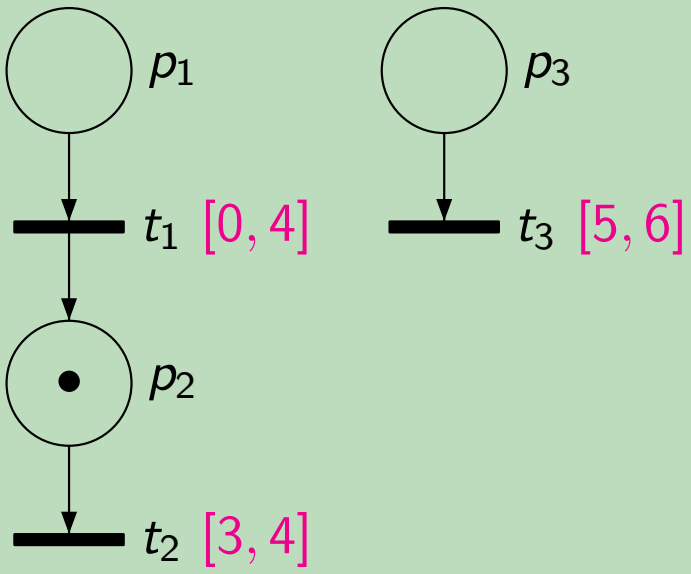
$$\begin{array}{c}\{p_1,p_3\}\\ \nu(t_1)=0\\ \nu(t_3)=0\end{array} \xrightarrow{4} \begin{array}{c}\{p_1,p_3\}\\ \nu(t_1)=4\\ \nu(t_3)=4\end{array} \xrightarrow{t_1} \begin{array}{c}\{p_2,p_3\}\\ \nu(t_2)=0\\ \nu(t_3)=4\end{array} \xrightarrow{1.3} \begin{array}{c}\{p_2,p_3\}\\ \nu(t_2)=1.3\\ \nu(t_3)=5.3\end{array} \xrightarrow{t_3} \cdots$$

Abstractions symboliques de l'espace d'états

Graphe des **classes**, graphe des **régions**, graphe des **zones**

Réseau de Petri T-temporel (TPN)

Exemple



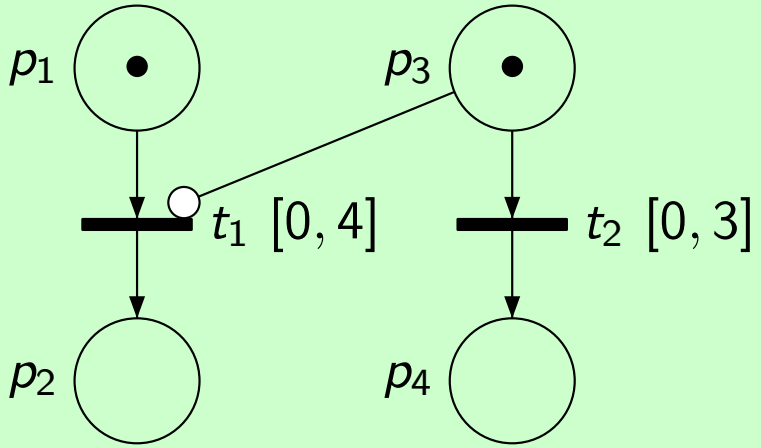
$$\begin{array}{l}\{p_1,p_3\}\\ \nu(t_1)=0\\ \nu(t_3)=0\end{array} \xrightarrow{4} \begin{array}{l}\{p_1,p_3\}\\ \nu(t_1)=4\\ \nu(t_3)=4\end{array} \xrightarrow{t_1} \begin{array}{l}\{p_2,p_3\}\\ \nu(t_2)=0\\ \nu(t_3)=4\end{array} \xrightarrow{1.3} \begin{array}{l}\{p_2,p_3\}\\ \nu(t_2)=1.3\\ \nu(t_3)=5.3\end{array} \xrightarrow{t_3} \cdots$$

Theorem [BD91]

L'accessibilité est décidable pour les TPN bornés et PSPACE-complet.

Des arcs particuliers

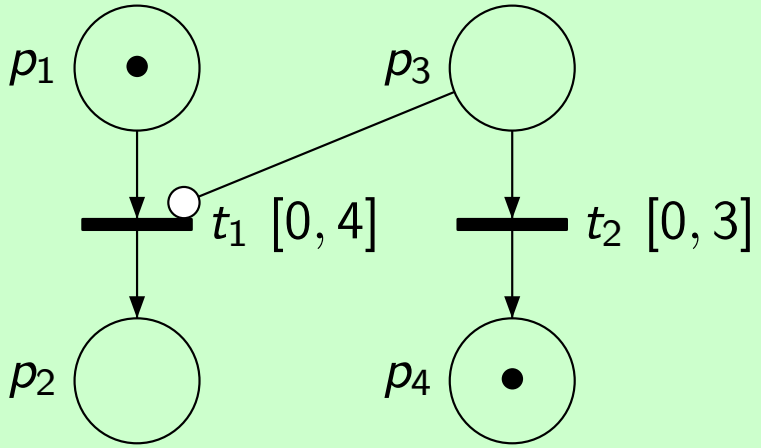
Example (Arc inhibiteur logique)



Peut-on tirer t_1 avant t_2 ?

Des arcs particuliers

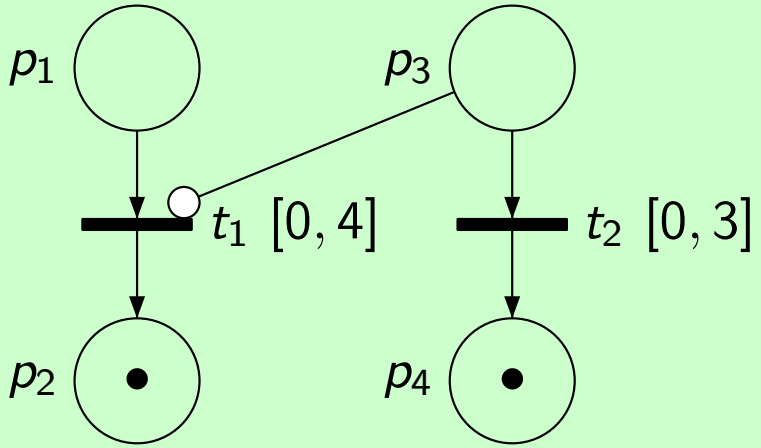
Example (Arc inhibiteur logique)



Peut-on tirer t_1 avant t_2 ?

Des arcs particuliers

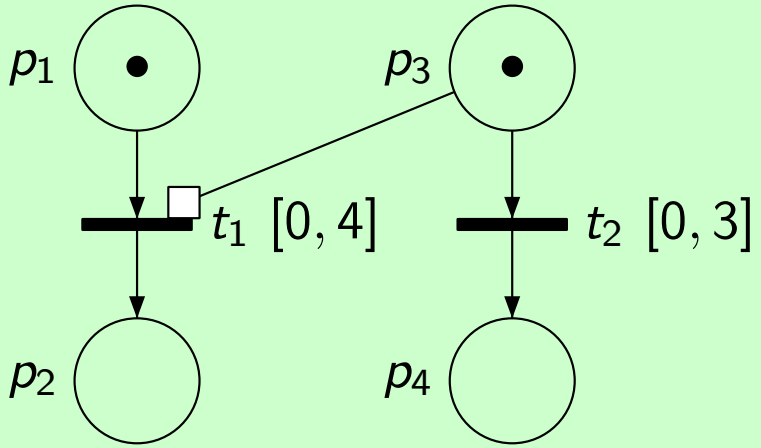
Example (Arc inhibiteur logique)



Peut-on tirer t_1 avant t_2 ?

Des arcs particuliers

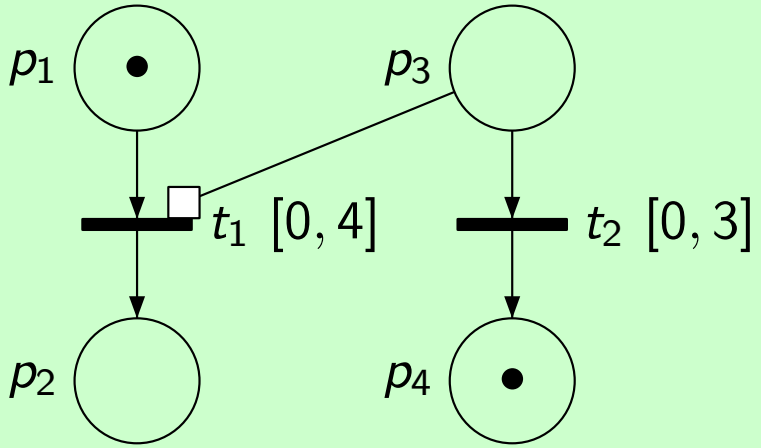
Example (Arc de lecture)



Peut-on tirer t_2 avant t_1 ?

Des arcs particuliers

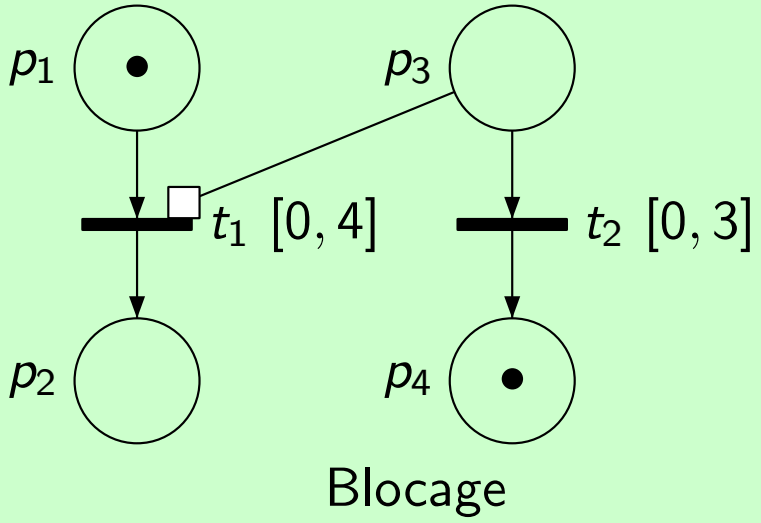
Example (Arc de lecture)



Peut-on tirer t_2 avant t_1 ?

Des arcs particuliers

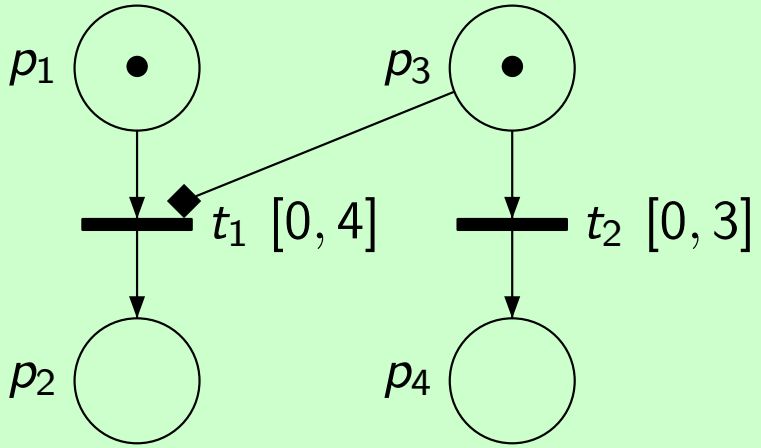
Example (Arc de lecture)



Peut-on tirer t_2 avant t_1 ?

Des arcs particuliers

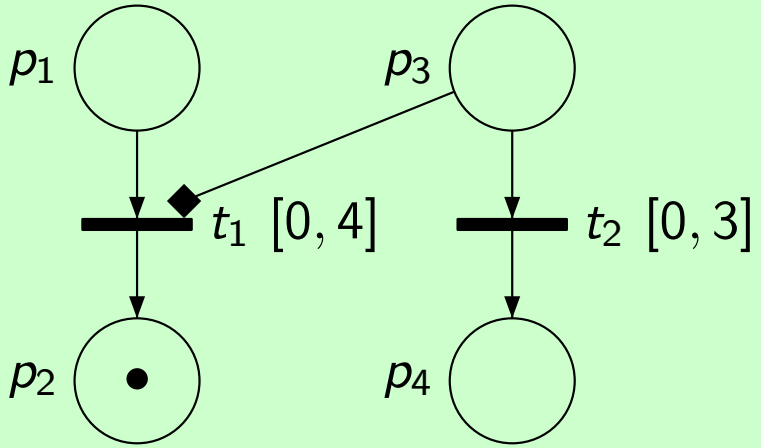
Example (Arc de vidange)



Peut-on tirer t_1 avant t_2 ?

Des arcs particuliers

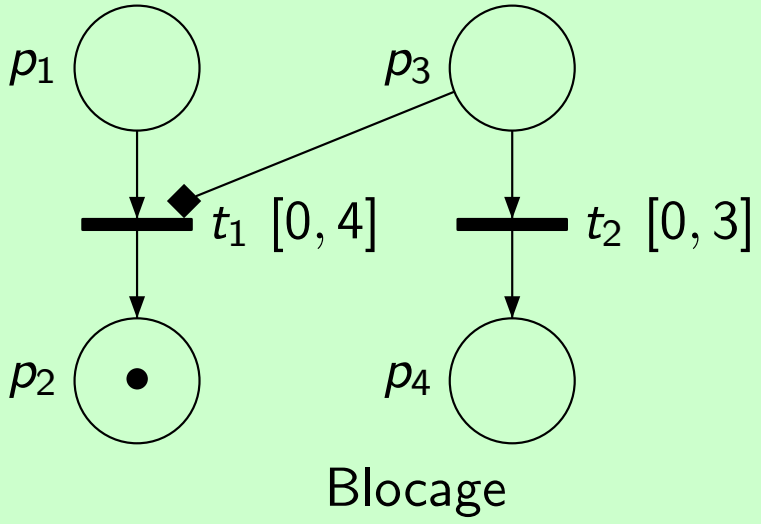
Example (Arc de vidange)



Peut-on tirer t_1 avant t_2 ?

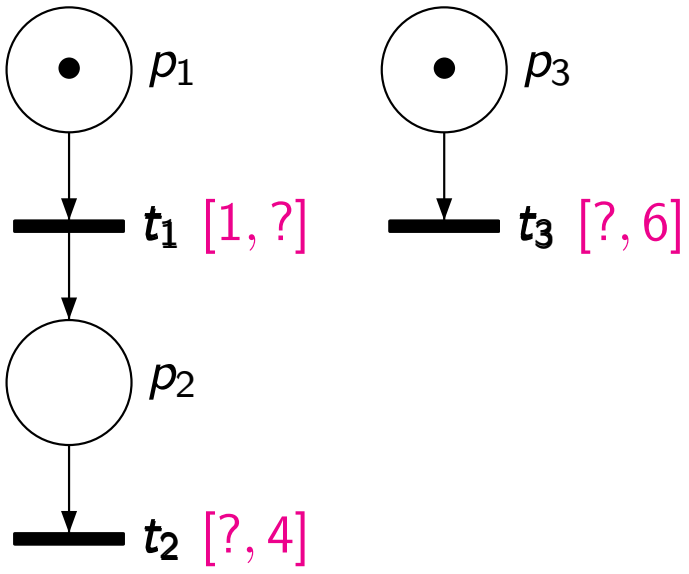
Des arcs particuliers

Example (Arc de vidange)

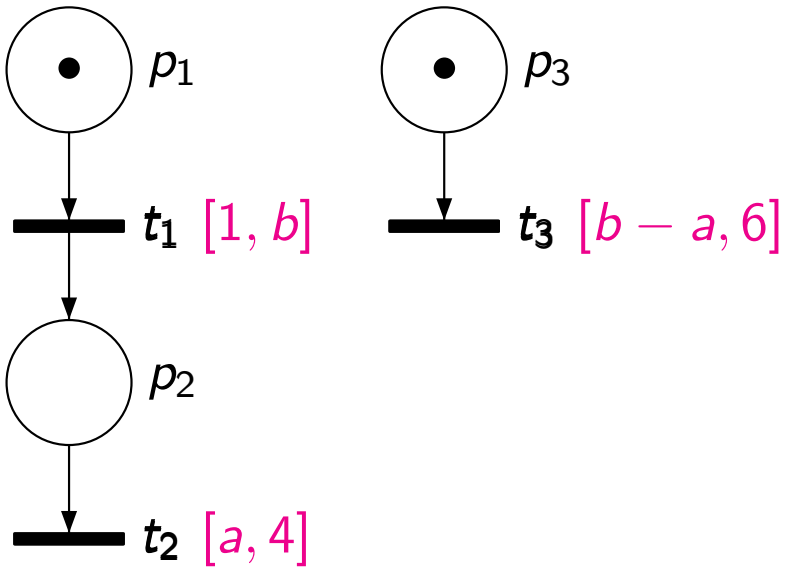


Peut-on tirer t_1 avant t_2 ?

Parametric Time Petri Nets (PTPN) [TLR09]

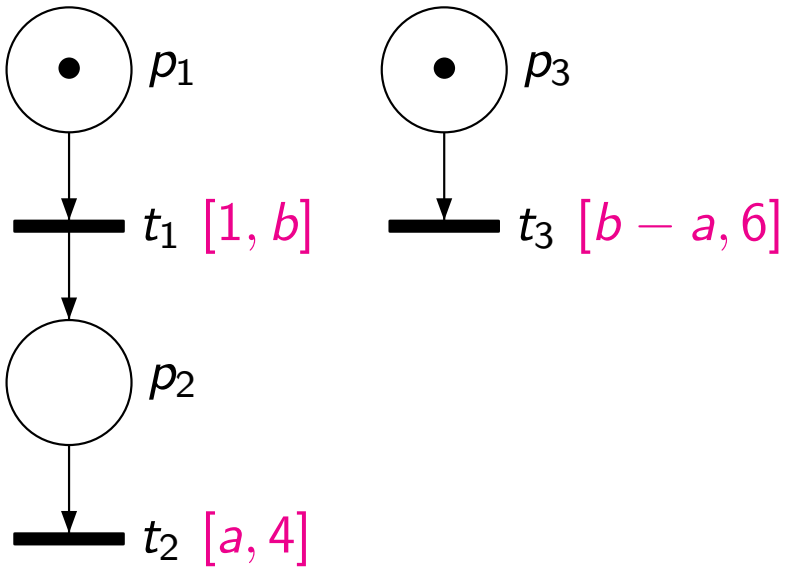


Parametric Time Petri Nets (PTPN) [TLR09]



- Un PTPN est un réseau de Petri temporel dont les bornes des intervalles sont des **expressions linéaires** sur un ensemble fini de paramètres **P**

Parametric Time Petri Nets (PTPN) [TLR09]



- Un PTPN est un réseau de Petri temporel dont les bornes des intervalles sont des **expressions linéaires** sur un ensemble fini de paramètres **P**
- Pour un PTPN $\mathcal{N}$ et une valuation des parametres $v : P \to \mathbb{Q}$, "$v(\mathcal{N})$" est un **réseau de Petri temporel**.

Outline

Les problèmes paramétrés

- Soit φ une propriété et $\mathcal{A}$ un PTA (ou un PTPN)
- Soit $V_\varphi(\mathcal{A})$ l'ensemble des valeurs v des paramètres t.q.. $v(\mathcal{A}) \models \varphi$.

φ-emptiness problem

L'ensemble $V_\varphi(\mathcal{A})$ est-il vide ?

φ-synthesis problem

Pouvons nous synthétiser l'ensemble des valeurs des paramètres $V_\varphi(\mathcal{A})$?

Les problèmes paramétrés

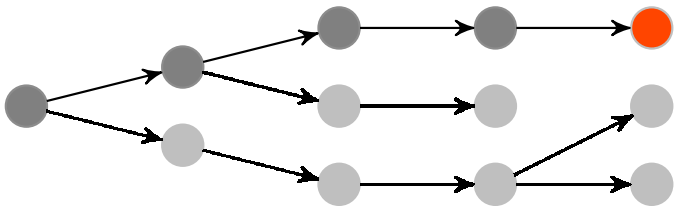
- Soit φ une propriété et $\mathcal{A}$ un PTA (ou un PTPN)
- Soit $V_\varphi(\mathcal{A})$ l'ensemble des valeurs v des paramètres t.q.. $v(\mathcal{A}) \models \varphi$.

φ-emptiness problem

L'ensemble $V_\varphi(\mathcal{A})$ est-il vide ?

φ-synthesis problem

Pouvons nous synthétiser l'ensemble des valeurs des paramètres $V_\varphi(\mathcal{A})$?



(a) Reachability (EF)

Les problèmes paramétrés

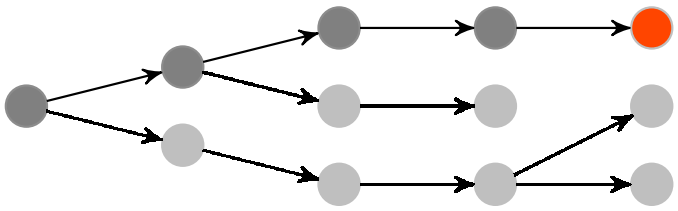
- Soit φ une propriété et $\mathcal{A}$ un PTA (ou un PTPN)
- Soit $V_\varphi(\mathcal{A})$ l'ensemble des valeurs v des paramètres t.q.. $v(\mathcal{A}) \models \varphi$.

φ-emptiness problem

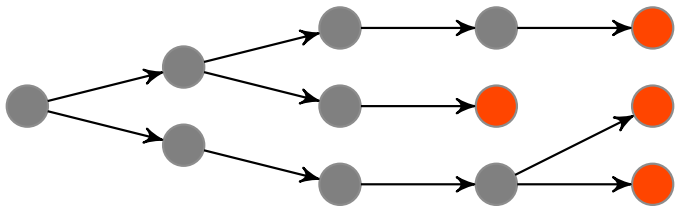
L'ensemble $V_\varphi(\mathcal{A})$ est-il vide ?

φ-synthesis problem

Pouvons nous synthétiser l'ensemble des valeurs des paramètres $V_\varphi(\mathcal{A})$?



(a) Reachability (EF)



(b) Unavoidability (AF)

Résultats de (In)décidabilité

Theorem [AHV93]

The EF-emptiness problem for PTA is **undecidable**.

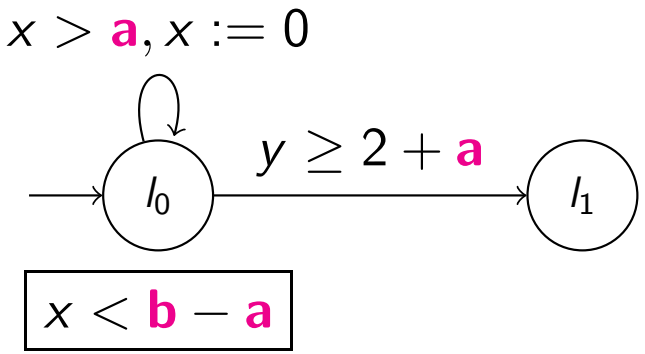
Proof:

- Reduction from the 2-counter machine halting problem
- PTA is constructed such that the machine halts iff $V \neq \emptyset$
- If the machine does not halt, the PTA cannot reach l_{halt}: $V = \emptyset$;

L/U-automata [HRSV02]

- In L/U-automata, parameters are either **upper bounds** (e.g. $x \leq b$ or $x \geq 1 - b$) or **lower bounds** on clocks.

L/U

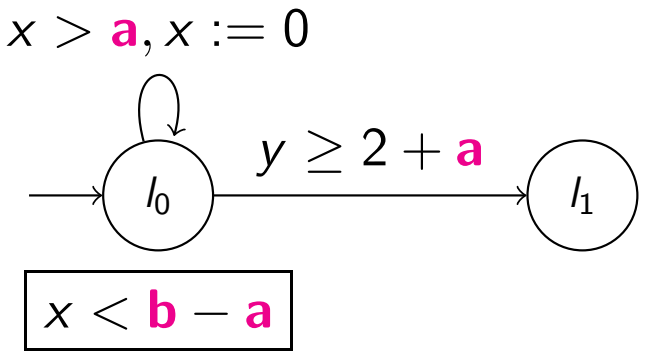


- **All** the possible behaviors can be obtained by setting lower bounds to 0 and upper bounds to $+\infty$.

L/U-automata [HRSV02]

- In L/U-automata, parameters are either **upper bounds** (e.g. $x \leq b$ or $x \geq 1 - b$) or **lower bounds** on clocks.

L/U



- **All** the possible behaviors can be obtained by setting lower bounds to 0 and upper bounds to $+\infty$.

Theorem [HRSV02]

The EF-emptiness problem for L/U-automata is decidable and PSPACE-complete.

L/U-automata: AF-Emptiness

Theorem [JLR13a]

The AF-emptiness problem for **U**-automata is undecidable.

- Proved by adapting the 2-counter machine reduction of [AHV93].
- Machine halts iff l_{halt} is unavoidable in the U-automaton

L/U-automata: Synthesis

Theorem [JLR13a]

For L/U automata, the solution to the EF-synthesis problem cannot be represented using any formalism for which emptiness of the intersection with equality constraints is decidable.

(in particular, not **finite unions of convex polyhedra**)

L/U-automata: Synthesis

Theorem [JLR13a]

For L/U automata, the solution to the EF-synthesis problem cannot be represented using any formalism for which emptiness of the intersection with equality constraints is decidable.

(in particular, not **finite unions of convex polyhedra**)

Proof (inspired from [BT09]):

- In a PTA, if p_i is both a lower bound and an upper bound:
  - Replace p_i by a fresh parameter p_i^u when p_i is an upper-bound;
  - Replace p_i by a fresh parameter p_i^l when p_i is a lower-bound.
- $V' = \bigcap_i p_i^u = p_i^l$

Bounded Integer Parametric Problems

- L/U automata:
  - Mixed decidability results regarding **emptiness**;
  - **Synthesis** of all parameter values is not feasible.

Bounded Integer Parametric Problems

- L/U automata:
  - Mixed decidability results regarding **emptiness**;
  - **Synthesis** of all parameter values is not feasible.
- Pragmatic restriction: if we look for parameter values as **bounded integers** "everything" is decidable and computable!

Bounded Integer Parametric Problems

- L/U automata:
  - Mixed decidability results regarding **emptiness**;
  - **Synthesis** of all parameter values is not feasible.
- Pragmatic restriction: if we look for parameter values as **bounded integers** "everything" is decidable and computable!

Theorem [JLR13a]

EF-emptiness for possibly **unbounded integer** values is undecidable.

Proof: As in [AHV93]

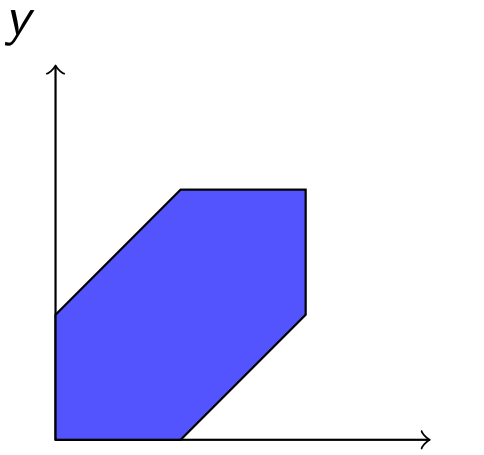
Outline

Symbolic States for PTA

- We do **not** want to **enumerate** all possible values of the parameters;

Symbolic States for PTA

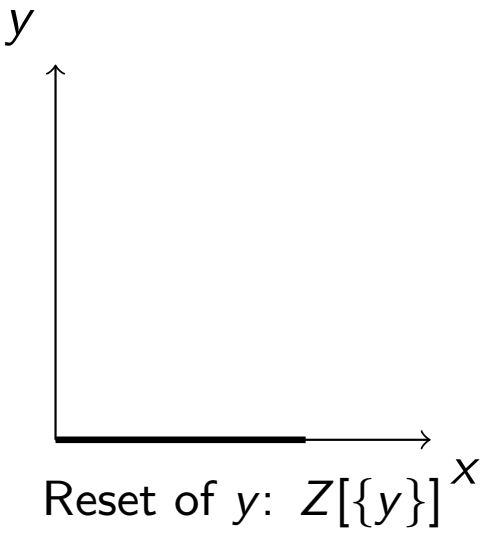
- We do **not** want to **enumerate** all possible values of the parameters;
- We extend the classical **symbolic** approach to model-checking in dense-time:
  - Symbolic state (l, Z): location + clock zone - set of clock valuations defined by a clock constraint
  - Extension: parametric symbolic state (l, Z): location + polyhedron constraining **both** clocks and parameters



Reset of y: $Z[\{y\}]$

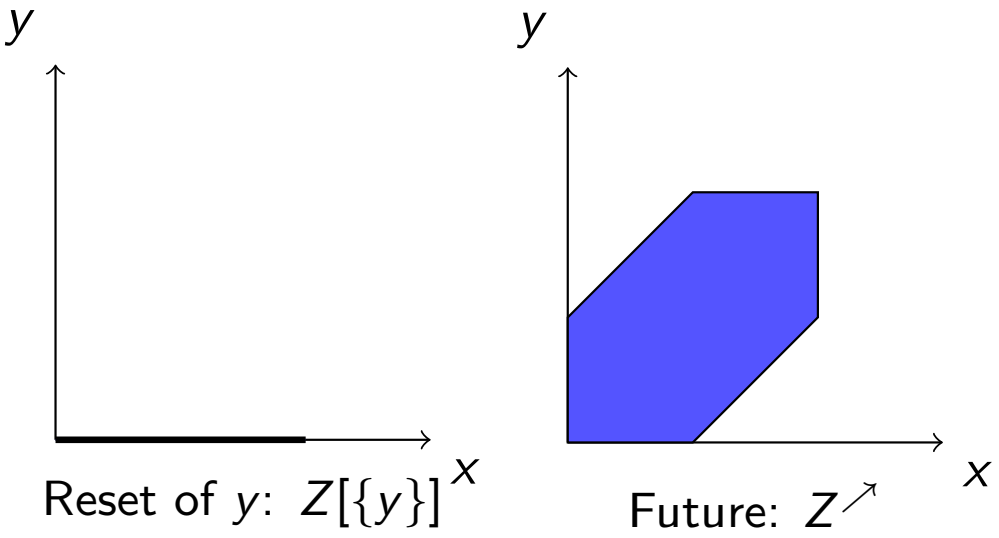
Symbolic States for PTA

- We do **not** want to **enumerate** all possible values of the parameters;
- We extend the classical **symbolic** approach to model-checking in dense-time:
  - Symbolic state (l, Z): location + clock zone - set of clock valuations defined by a clock constraint
  - Extension: parametric symbolic state (l, Z): location + polyhedron constraining **both** clocks and parameters



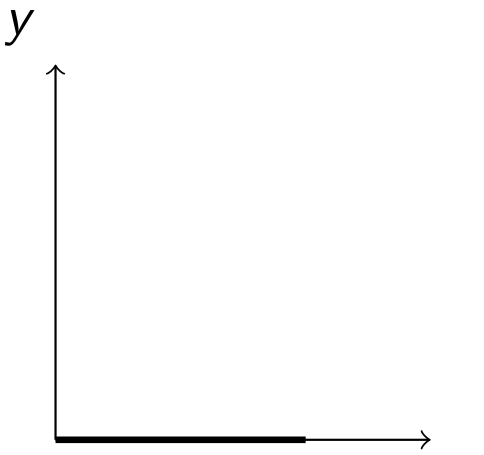
Symbolic States for PTA

- We do **not** want to **enumerate** all possible values of the parameters;
- We extend the classical **symbolic** approach to model-checking in dense-time:
  - Symbolic state (l, Z): location + clock zone - set of clock valuations defined by a clock constraint
  - Extension: parametric symbolic state (l, Z): location + polyhedron constraining **both** clocks and parameters

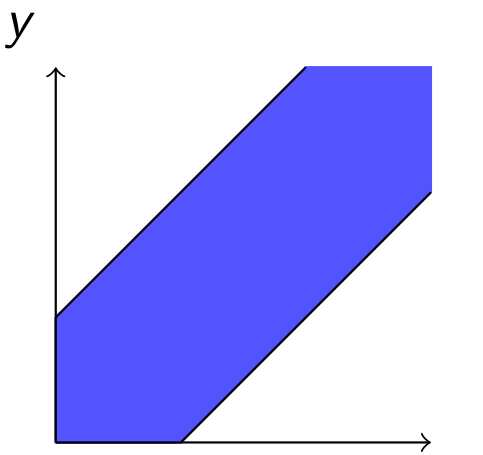


Symbolic States for PTA

- We do **not** want to **enumerate** all possible values of the parameters;
- We extend the classical **symbolic** approach to model-checking in dense-time:
  - Symbolic state (l, Z): location + clock zone - set of clock valuations defined by a clock constraint
  - Extension: parametric symbolic state (l, Z): location + polyhedron constraining **both** clocks and parameters



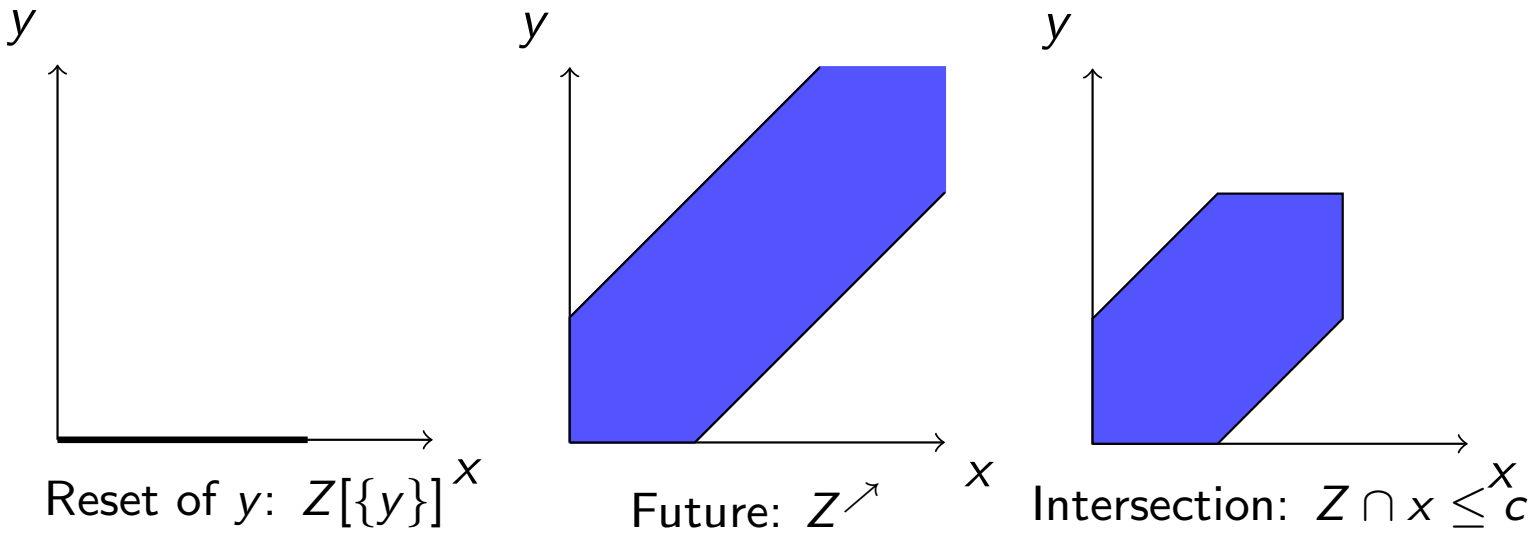
Reset of y: $Z[\{y\}]$



Future: $Z^{\nearrow}$

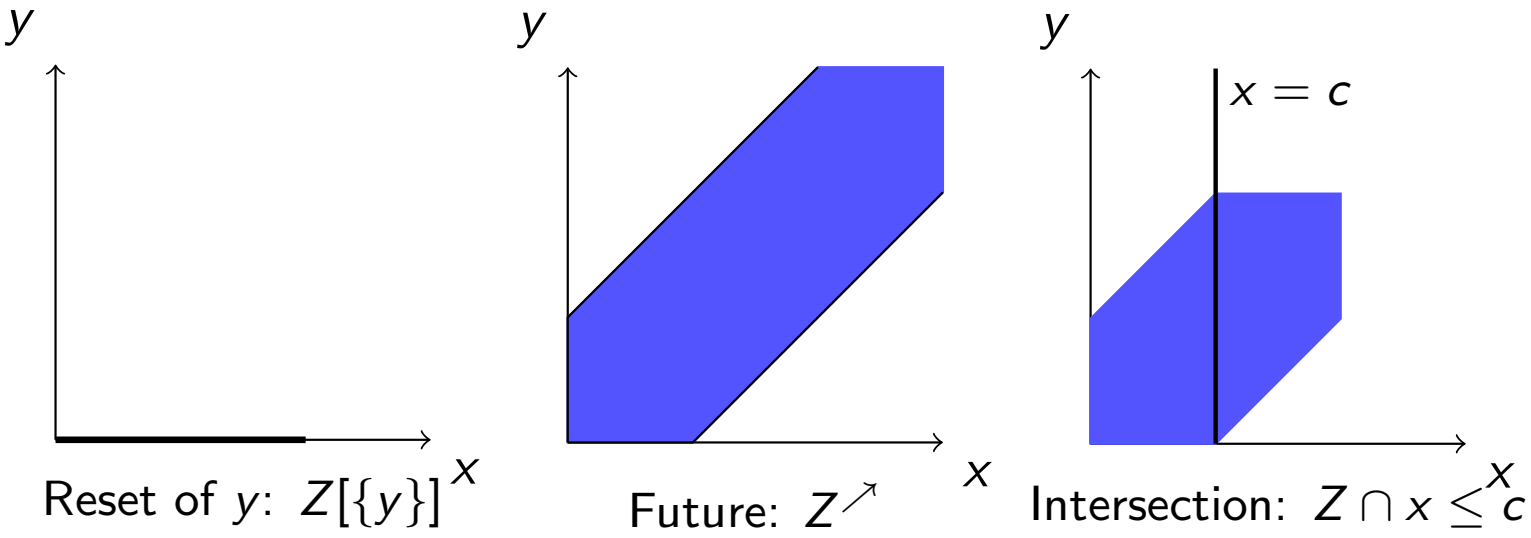
Symbolic States for PTA

- We do **not** want to **enumerate** all possible values of the parameters;
- We extend the classical **symbolic** approach to model-checking in dense-time:
  - Symbolic state (l, Z): location + clock zone - set of clock valuations defined by a clock constraint
  - Extension: parametric symbolic state (l, Z): location + polyhedron constraining **both** clocks and parameters



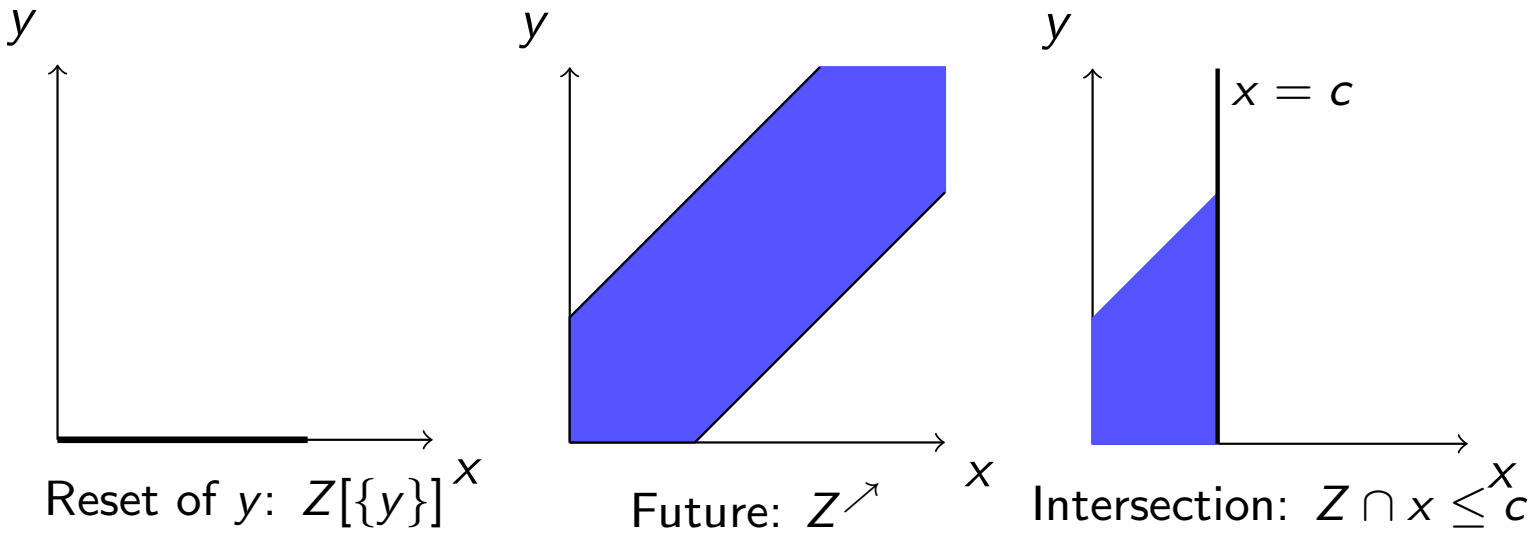
Symbolic States for PTA

- We do **not** want to **enumerate** all possible values of the parameters;
- We extend the classical **symbolic** approach to model-checking in dense-time:
  - Symbolic state (l, Z): location + clock zone - set of clock valuations defined by a clock constraint
  - Extension: parametric symbolic state (l, Z): location + polyhedron constraining **both** clocks and parameters



Symbolic States for PTA

- We do **not** want to **enumerate** all possible values of the parameters;
- We extend the classical **symbolic** approach to model-checking in dense-time:
  - Symbolic state (l, Z): location + clock zone - set of clock valuations defined by a clock constraint
  - Extension: parametric symbolic state (l, Z): location + polyhedron constraining **both** clocks and parameters



EF and AF semi-algorithms

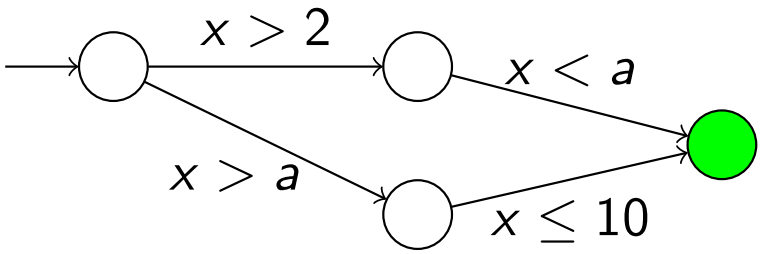
EF-synthesis algorithm:

- Aggregate all the values find on the paths that reach goal location

EF and AF semi-algorithms

EF-synthesis algorithm:

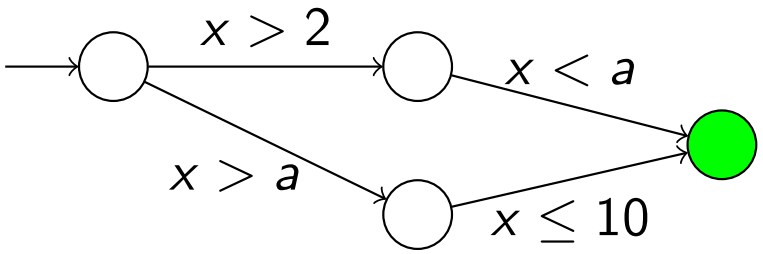
- Aggregate all the values find on the paths that reach goal location



EF and AF semi-algorithms

EF-synthesis algorithm:

- Aggregate all the values find on the paths that reach goal location

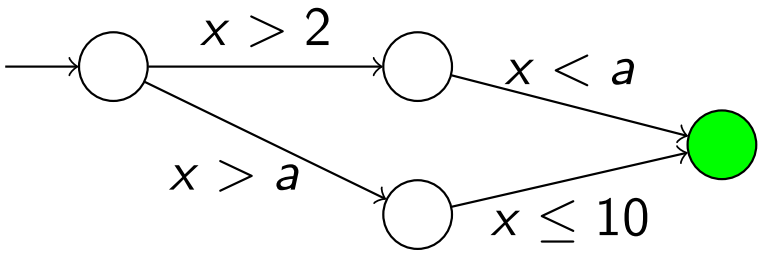


$\Rightarrow a > 2 \vee a \leq 10$

EF and AF semi-algorithms

EF-synthesis algorithm:

- Aggregate all the values find on the paths that reach goal location



$\Rightarrow a > 2 \vee a \leq 10$

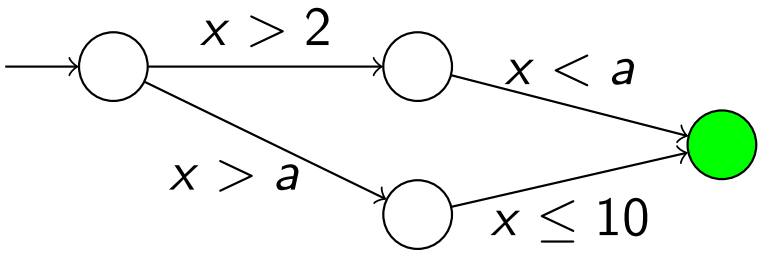
AF-synthesis algorithm:

- Cut the path that does not reach goal location

EF and AF semi-algorithms

EF-synthesis algorithm:

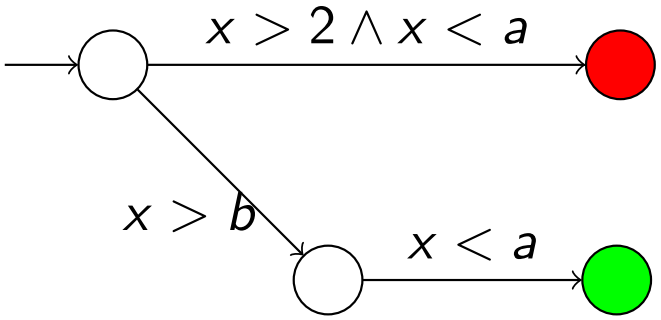
- Aggregate all the values find on the paths that reach goal location



$\Rightarrow a > 2 \vee a \leq 10$

AF-synthesis algorithm:

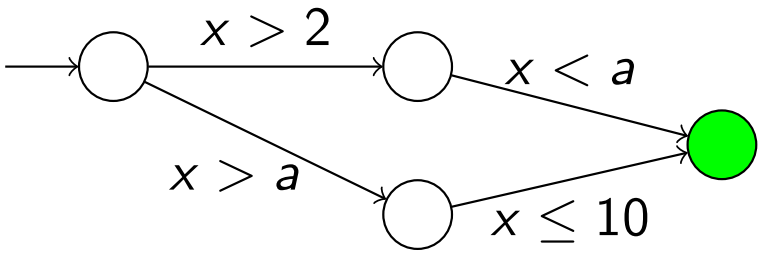
- Cut the path that does not reach goal location



EF and AF semi-algorithms

EF-synthesis algorithm:

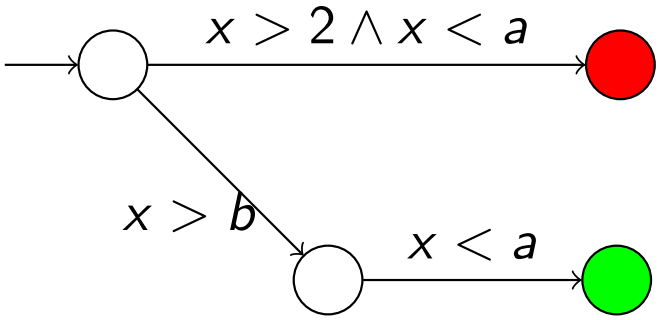
- Aggregate all the values find on the paths that reach goal location



$\Rightarrow a > 2 \vee a \leq 10$

AF-synthesis algorithm:

- Cut the path that does not reach goal location

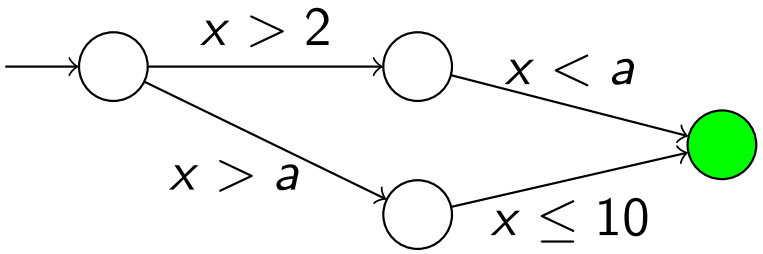


$\Rightarrow a \leq 2 \wedge a > b$

EF and AF semi-algorithms

EF-synthesis algorithm:

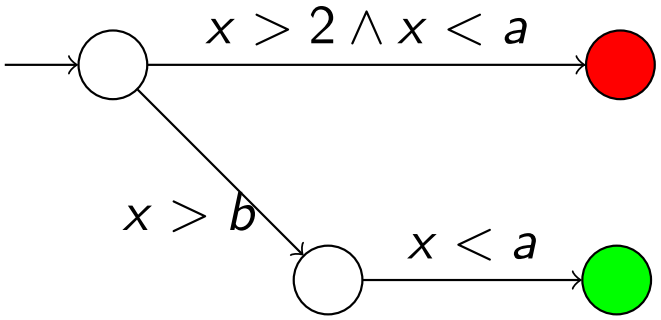
- Aggregate all the values find on the paths that reach goal location



$\Rightarrow a > 2 \vee a \leq 10$

AF-synthesis algorithm:

- Cut the path that does not reach goal location



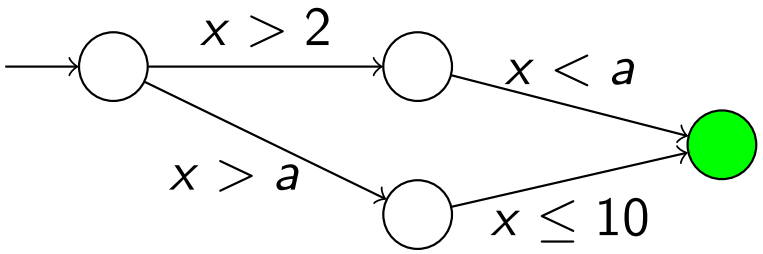
$\Rightarrow a \leq 2 \wedge a > b$

- Based on Succ (successor by edge) operator.

EF and AF semi-algorithms

EF-synthesis algorithm:

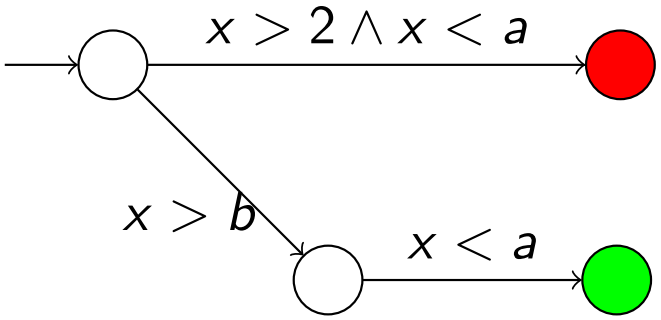
- Aggregate all the values find on the paths that reach goal location



$\Rightarrow a > 2 \vee a \leq 10$

AF-synthesis algorithm:

- Cut the path that does not reach goal location

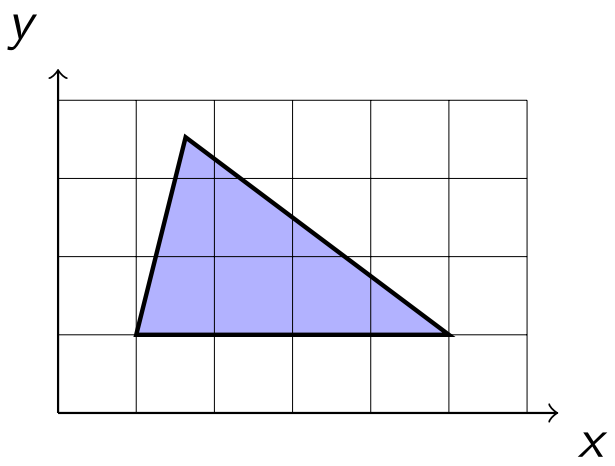


$\Rightarrow a \leq 2 \wedge a > b$

- Based on Succ (successor by edge) operator.
- Problem: **Termination** is **not** guaranteed.

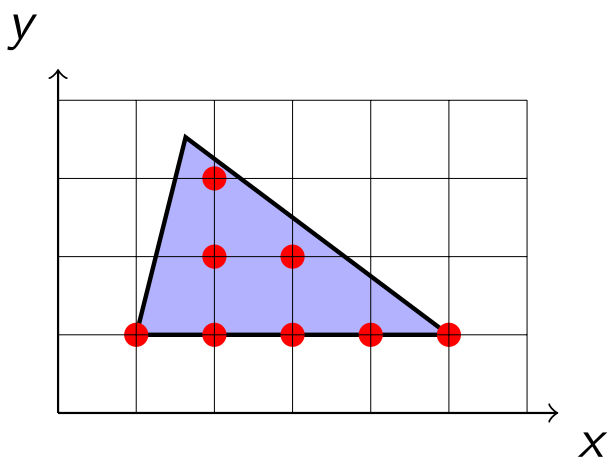
Computing the Integer Valuations: Integer Hulls

- We compute **integer hulls**: $\mathrm{IntHull}(Z) = Conv(IntVects(Z))$



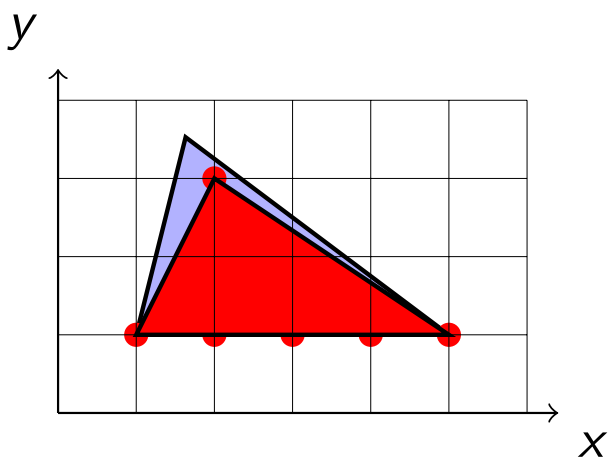
Computing the Integer Valuations: Integer Hulls

- We compute **integer hulls**: $\mathrm{IntHull}(Z) = Conv(IntVects(Z))$



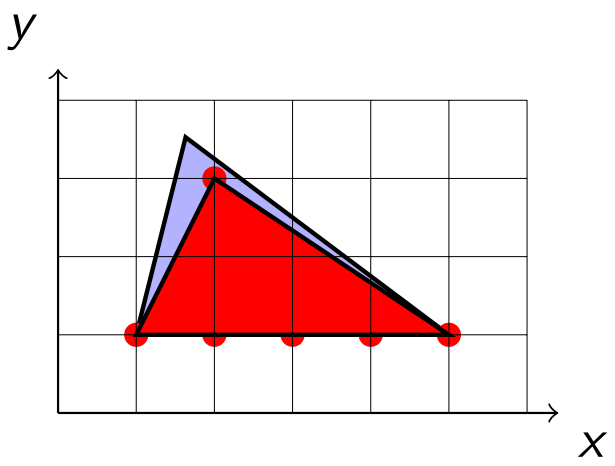
Computing the Integer Valuations: Integer Hulls

- We compute **integer hulls**: $\mathrm{IntHull}(Z) = Conv(IntVects(Z))$



Computing the Integer Valuations: Integer Hulls

- We compute **integer hulls**: $\mathsf{IntHull}(Z) = Conv(IntVects(Z))$



We use $\mathsf{ISucc}((l, Z), e) = \mathsf{IntHull}(\mathsf{Succ}((l, Z), e))$

Computing the Integer Valuations: Integer Hulls

- $\mathsf{ISucc}((I, Z), e) = \mathsf{IntHull}(\mathsf{Succ}((I, Z), e))$

Computing the Integer Valuations: Integer Hulls

- $\mathsf{ISucc}((l, Z), e) = \mathsf{IntHull}(\mathsf{Succ}((l, Z), e))$
- We use ISucc instead of Succ in semi-algortihms EF and AF to compute the **integer valuations**, giving semi-algorithms IEF and IAF.

Theorem [Soundness and completeness] [JLR13a]

For any PTA $\mathcal{A}$ and any subset of its locations G, upon termination, $\mathsf{IEF}(\mathsf{Init}(\mathcal{A}), G, \emptyset)$ (resp. $\mathsf{IAF}(\mathsf{Init}(\mathcal{A}), G, \emptyset)$) is the solution the integer EF-synthesis problem (resp. AF-synthesis problem) for PTA $\mathcal{A}$ and set of locations to reach G.

Computing the Integer Valuations: Integer Hulls

- $\mathsf{ISucc}((l, Z), e) = \mathsf{IntHull}(\mathsf{Succ}((l, Z), e))$
- We use ISucc instead of Succ in semi-algortihms EF and AF to compute the **integer valuations**, giving semi-algorithms IEF and IAF.

Theorem [Soundness and completeness] [JLR13a]

For any PTA $\mathcal{A}$ and any subset of its locations G, upon termination, $\mathsf{IEF}(\mathsf{Init}(\mathcal{A}), G, \emptyset)$ (resp. $\mathsf{IAF}(\mathsf{Init}(\mathcal{A}), G, \emptyset)$) is the solution the integer EF-synthesis problem (resp. AF-synthesis problem) for PTA $\mathcal{A}$ and set of locations to reach G.

- If the algorithm terminates, the solution is obtained as a **set of symbolic constraints** on parameters.

Computing the Integer Valuations: Integer Hulls

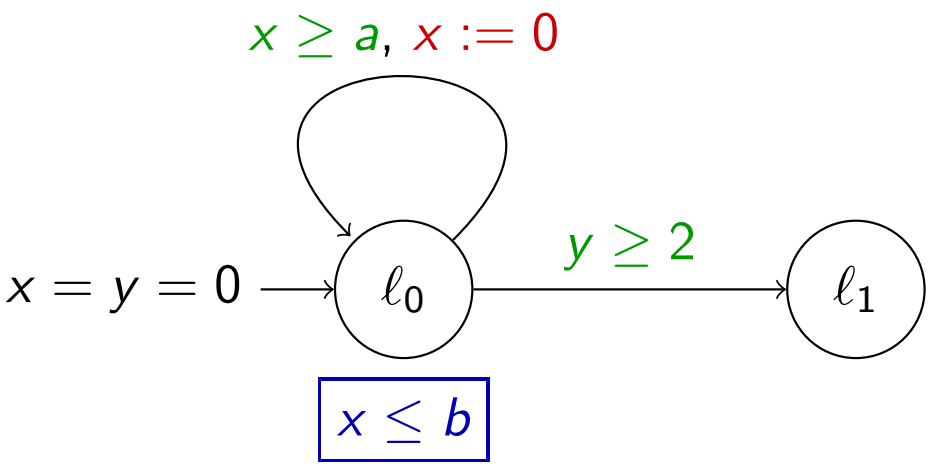
- $\mathsf{ISucc}((l, Z), e) = \mathsf{IntHull}(\mathsf{Succ}((l, Z), e))$
- We use ISucc instead of Succ in semi-algortihms EF and AF to compute the **integer valuations**, giving semi-algorithms IEF and IAF.

Theorem [Soundness and completeness] [JLR13a]

For any PTA $\mathcal{A}$ and any subset of its locations G, upon termination, $\mathsf{IEF}(\mathsf{Init}(\mathcal{A}), G, \emptyset)$ (resp. $\mathsf{IAF}(\mathsf{Init}(\mathcal{A}), G, \emptyset)$) is the solution the integer EF-synthesis problem (resp. AF-synthesis problem) for PTA $\mathcal{A}$ and set of locations to reach G.

- If the algorithm terminates, the solution is obtained as a **set of symbolic constraints** on parameters.
- Termination?

Termination in the Bounded Case

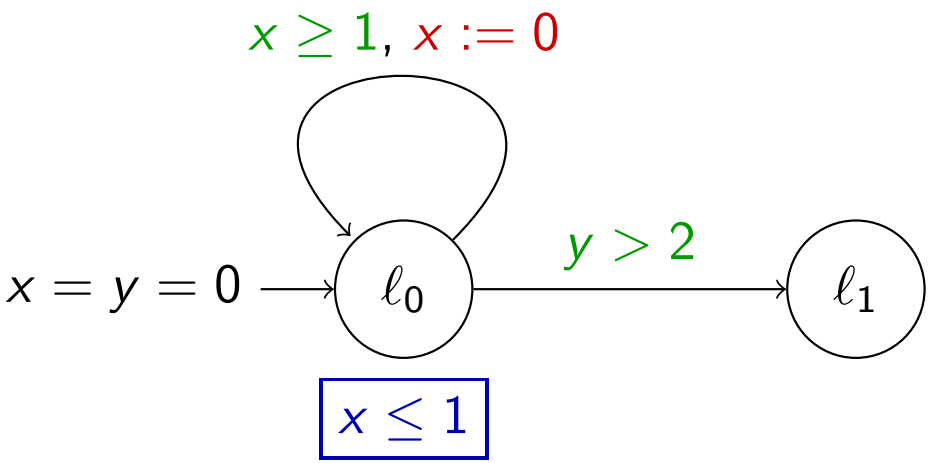


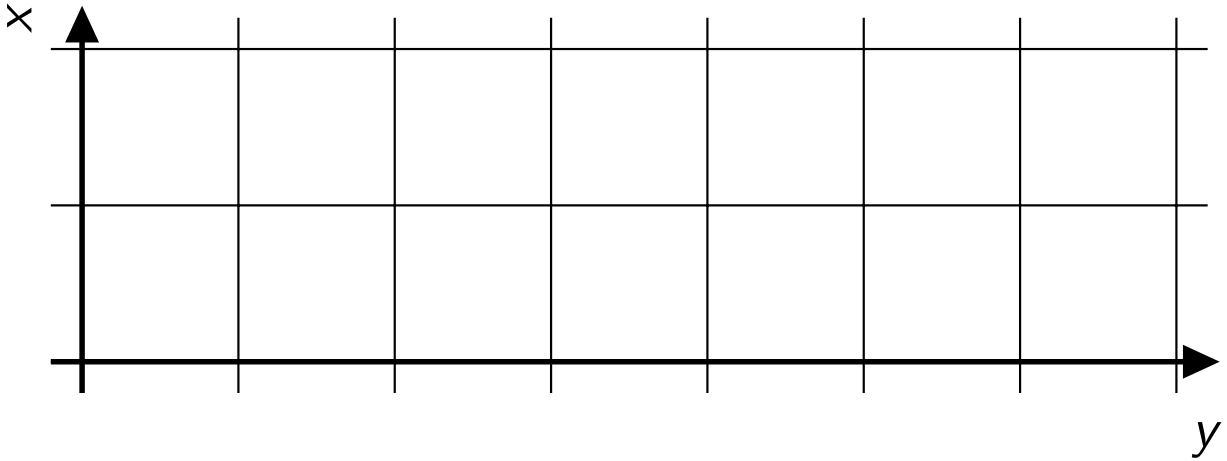
- Initial symbolic state: (ℓ_0, Z_0) with $Z_0 = \{x = y, x \leq b\}$;
- After n loops: (ℓ_0, Z_n) with **still**

$$Z_n = \{0 \leq x \leq b, 0 \leq na \leq y - x \leq (n+1)b\}$$

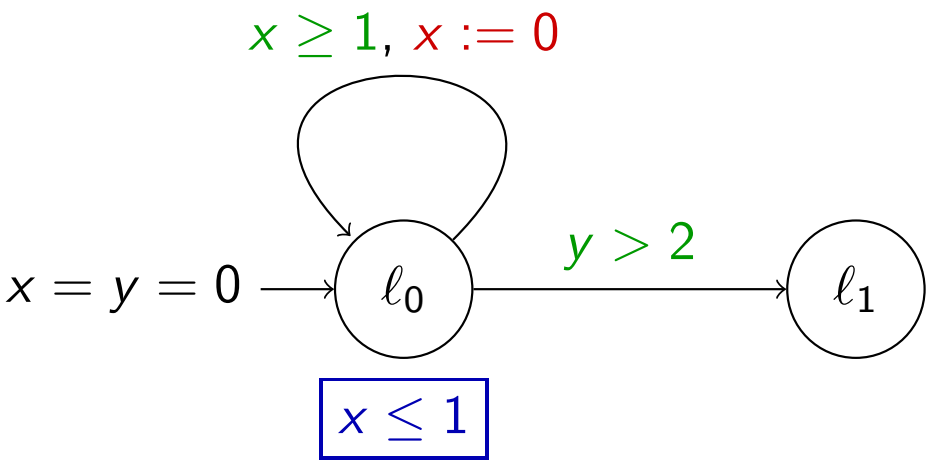
- Actually, $Z_n = \mathrm{IntHull}(Z_n)$

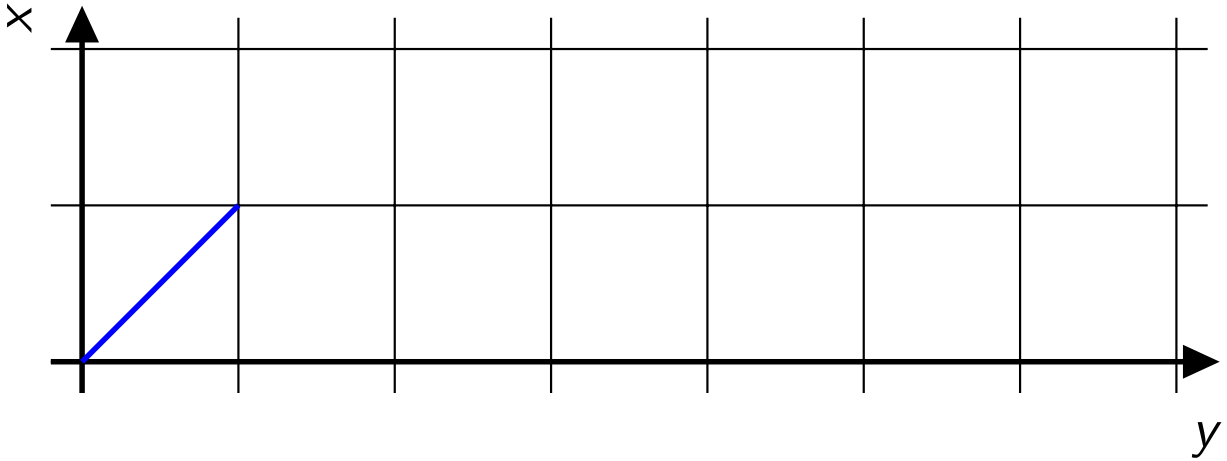
Termination in the Bounded Case



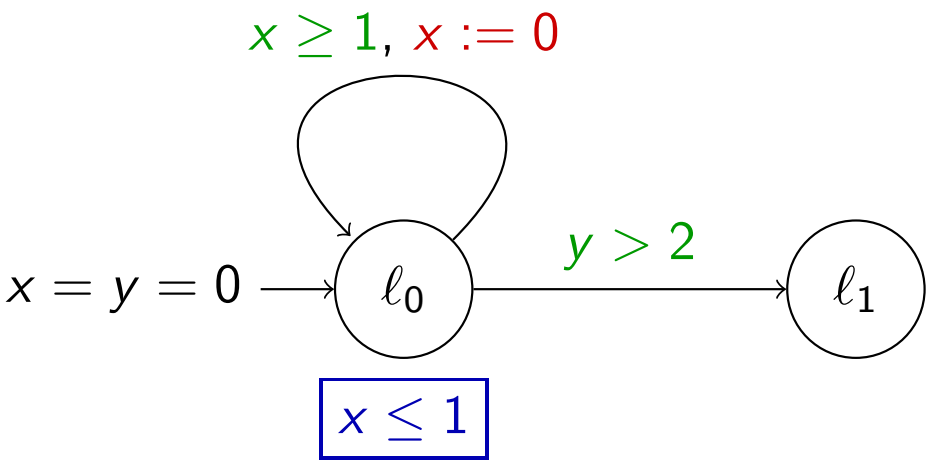


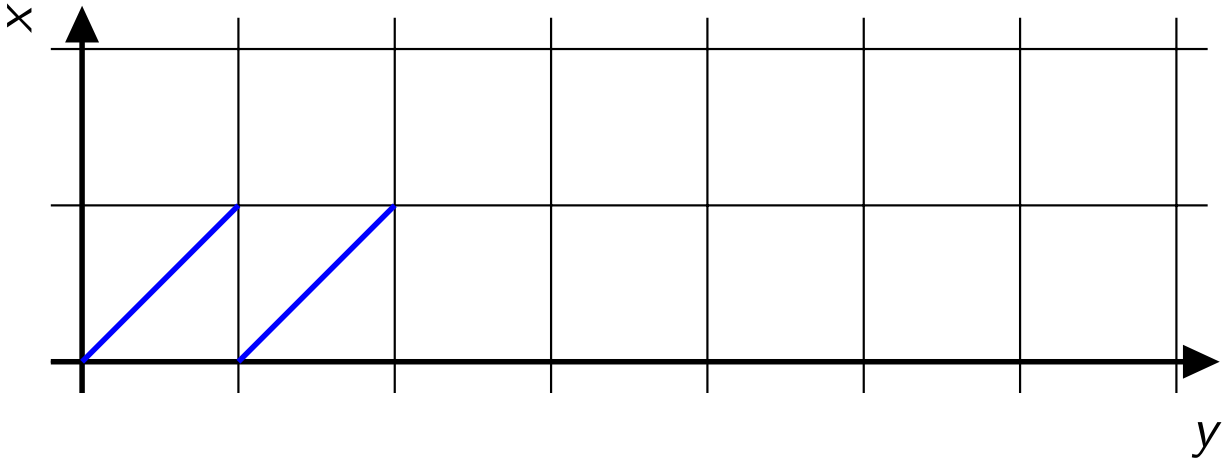
Termination in the Bounded Case



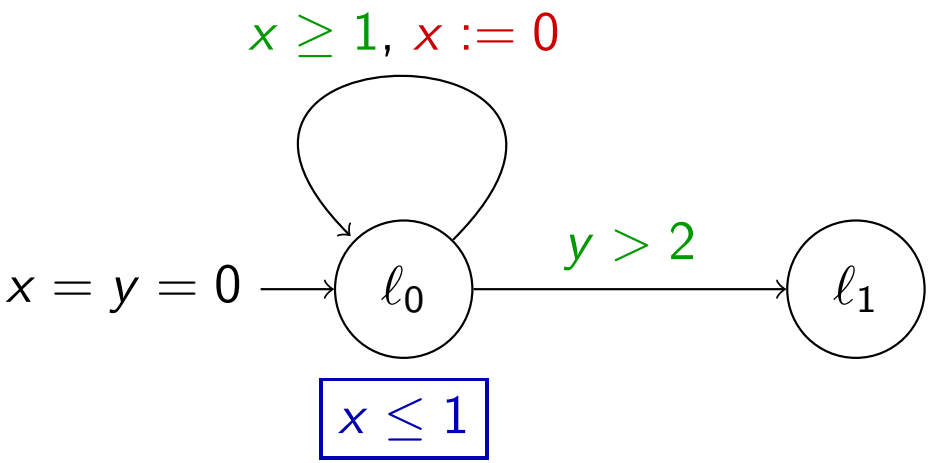


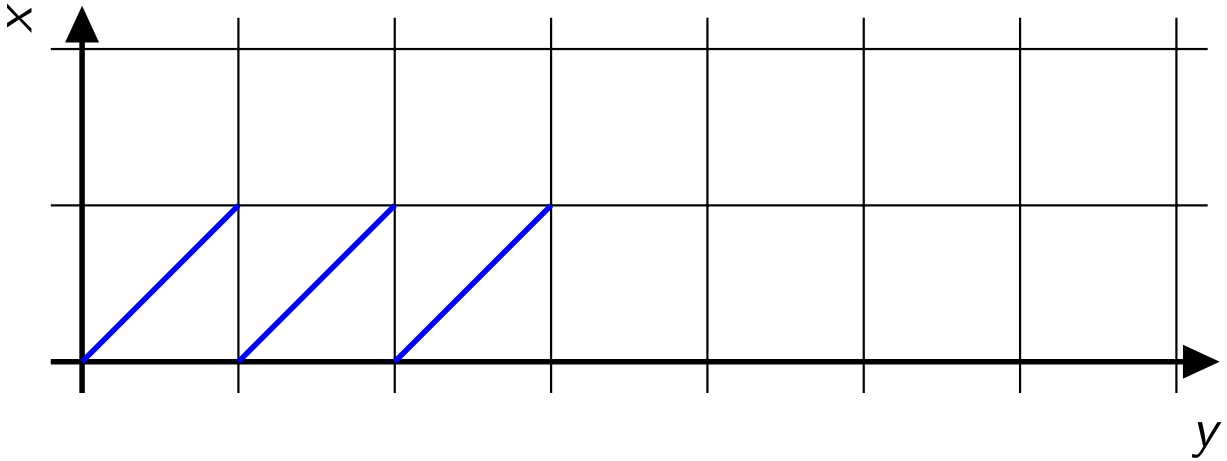
Termination in the Bounded Case



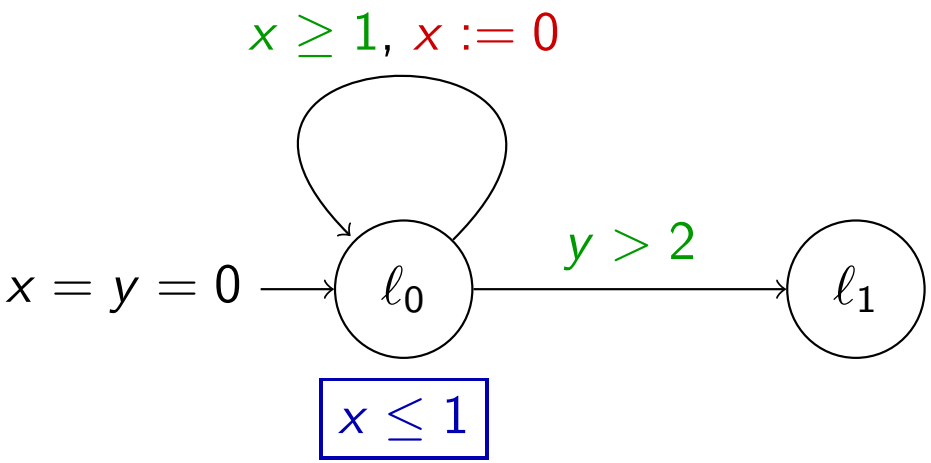


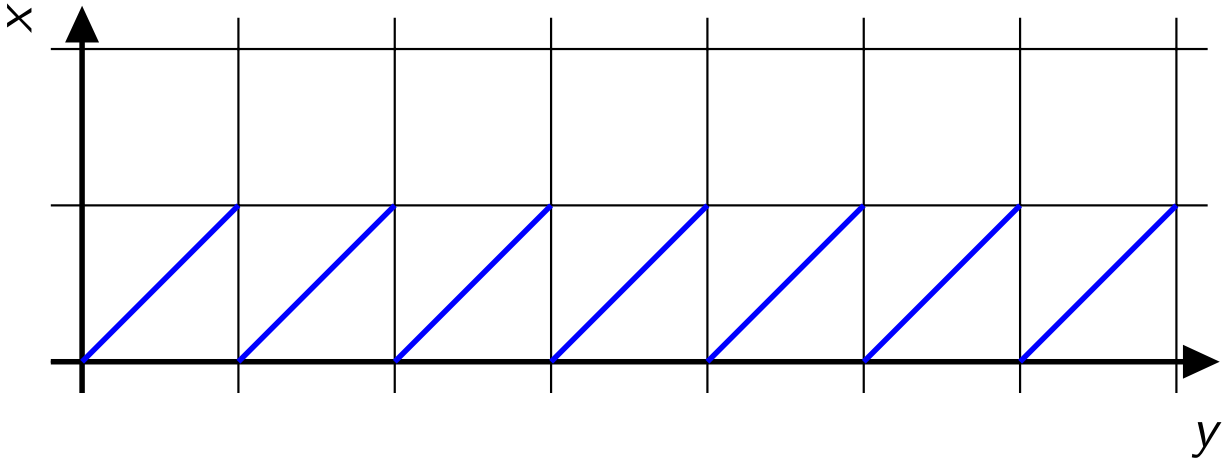
Termination in the Bounded Case



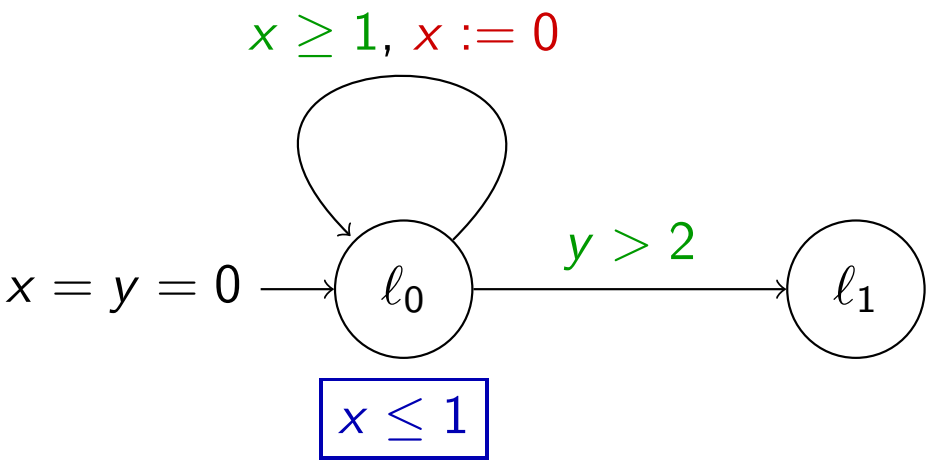


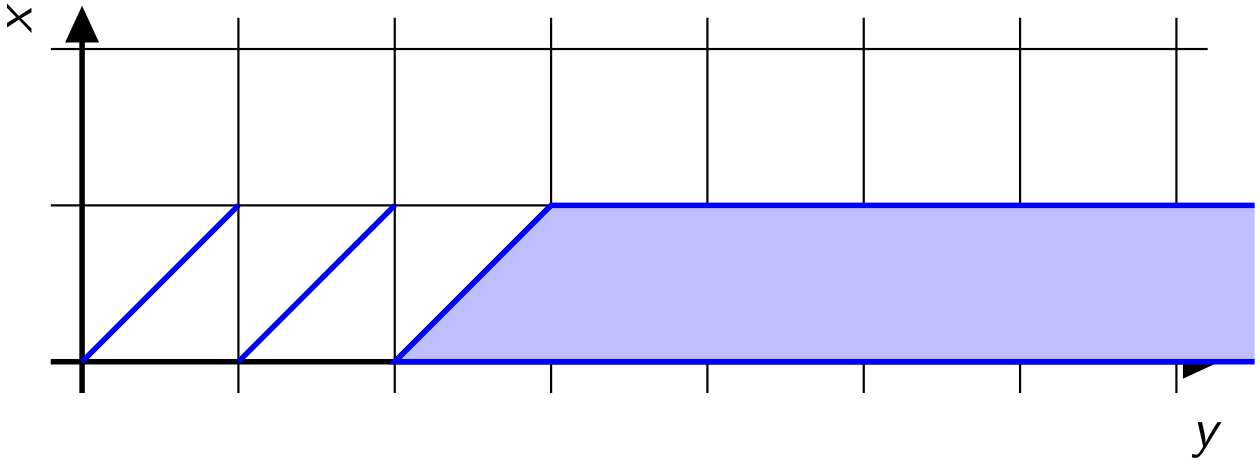
Termination in the Bounded Case





Termination in the Bounded Case





Termination in the Bounded Case

- Abstraction technique similar to k-extrapolation [DT98]:

Lemma ([JLR13a])

For any symbolic state (l, Z) of the PTA $\mathcal{A}$:

$$\mathsf{IntHull}(Z) = \mathsf{Conv}\Big(\bigcup_{v \in \mathsf{IntVects}(Z_{|P})} v(Z)\Big)$$

Termination in the Bounded Case

- Abstraction technique similar to k-extrapolation [DT98]:

Lemma ([JLR13a])

For any symbolic state (l, Z) of the PTA $\mathcal{A}$:

$$\mathsf{IntHull}(Z) = \mathsf{Conv}\Big(\bigcup_{v \in \mathsf{IntVects}(Z_{|P})} v(Z)\Big)$$

- Let K be the maximal constant appearing in PTA, $R(K)$ the number of regions and $|L|$ the number of locations

Termination in the Bounded Case

- Abstraction technique similar to k-extrapolation [DT98]:

Lemma ([JLR13a])

For any symbolic state (l, Z) of the PTA $\mathcal{A}$:

$$\mathsf{IntHull}(Z) = \mathsf{Conv}\Big(\bigcup_{v \in \mathsf{IntVects}(Z_{|P})} v(Z)\Big)$$

- Let K be the maximal constant appearing in PTA, $R(K)$ the number of regions and $|L|$ the number of locations
- Add invariant $x \leq |L| \times R(K)$ to all locations

Termination in the Bounded Case

- Abstraction technique similar to k-extrapolation [DT98]:

Lemma ([JLR13a])

For any symbolic state (l, Z) of the PTA $\mathcal{A}$:

$$\mathsf{IntHull}(Z) = \mathsf{Conv}\Big(\bigcup_{v \in \mathsf{IntVects}(Z_{|P})} v(Z)\Big)$$

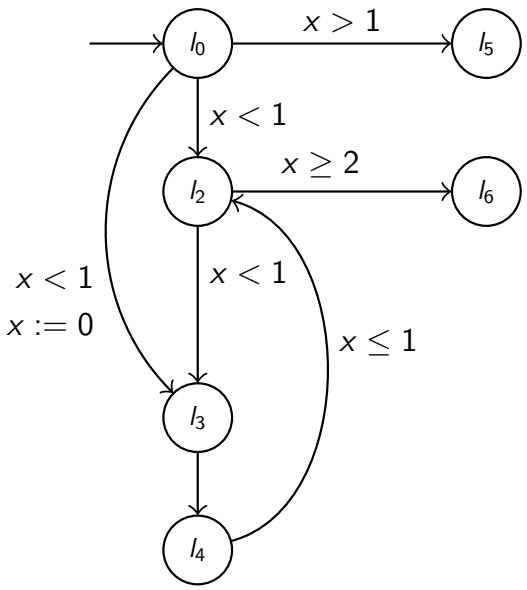
- Let K be the maximal constant appearing in PTA, $R(K)$ the number of regions and $|L|$ the number of locations
- Add invariant $x \leq |L| \times R(K)$ to all locations
- Reachability and unavoidability are preserved.
- **Finite** number of zones $\implies$ **termination** guaranteed.

Outline

Real-Timed Control: Timed Game Automata [MPS95]

Control problem:

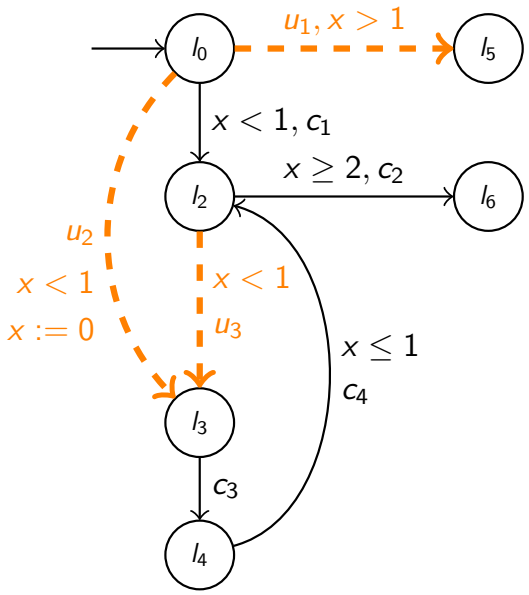
- system is modeled as a timed automaton



Real-Timed Control: Timed Game Automata [MPS95]

Control problem:

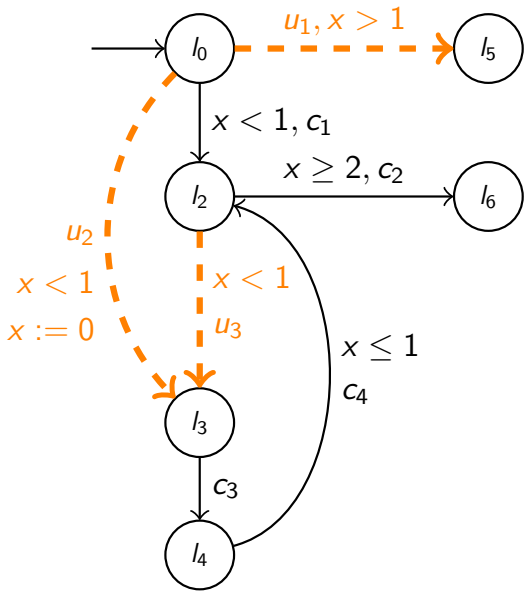
- system is modeled as a timed automaton
- some of the transitions are *uncontrollable*



Real-Timed Control: Timed Game Automata [MPS95]

Control problem:

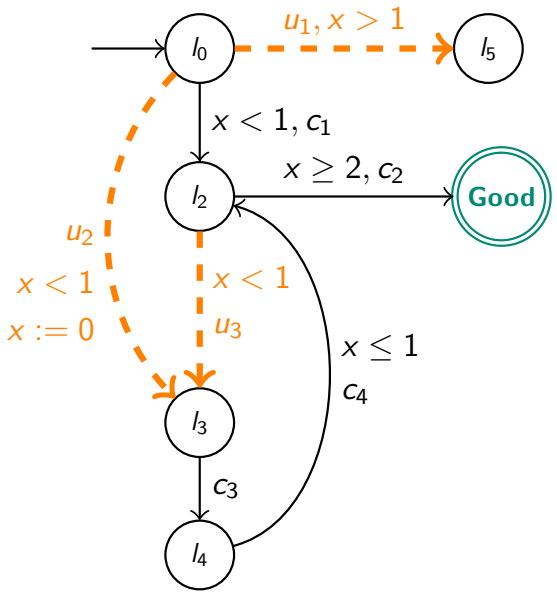
- system is modeled as a timed automaton
- some of the transitions are *uncontrollable*
- *controller*: inhibits certain transitions



Real-Timed Control: Timed Game Automata [MPS95]

Control problem:

- system is modeled as a timed automaton
- some of the transitions are *uncontrollable*
- *controller*: inhibits certain transitions
- control problem reduces to the reachability game: controller vs. environment



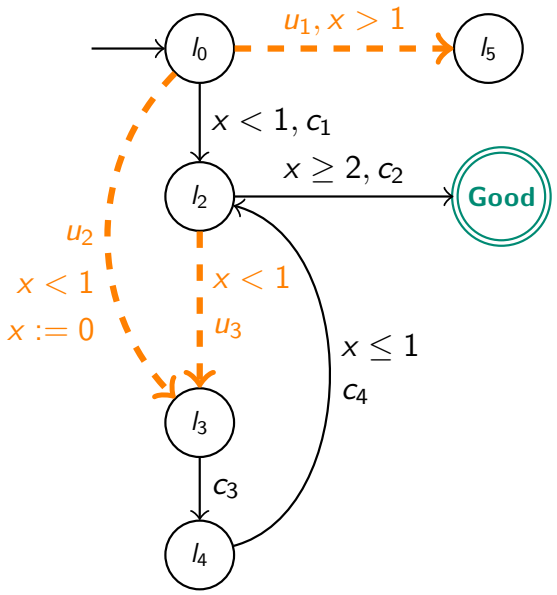
Real-Timed Control: Timed Game Automata [MPS95]

Control problem:

- system is modeled as a timed automaton
- some of the transitions are *uncontrollable*
- *controller*: inhibits certain transitions
- control problem reduces to the reachability game: controller vs. environment

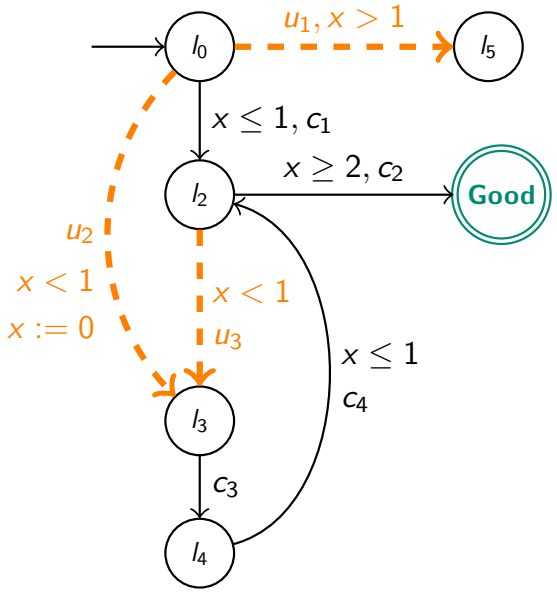
Controller synthesis:

- find a **winning strategy** to reach Good



Parametric Timed Games

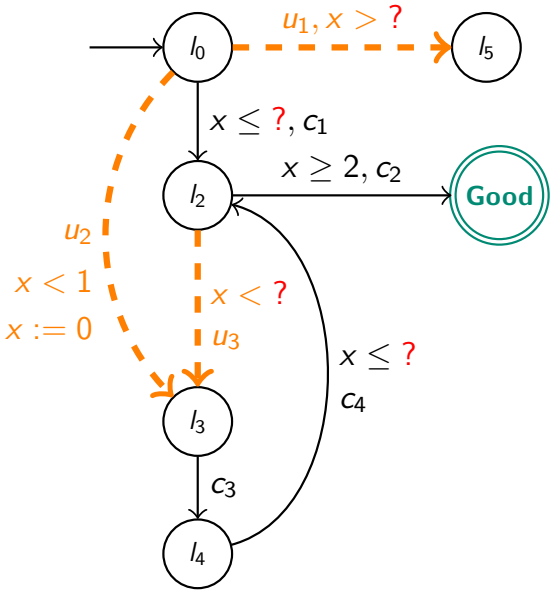
Motivation:



Parametric Timed Games

Motivation:

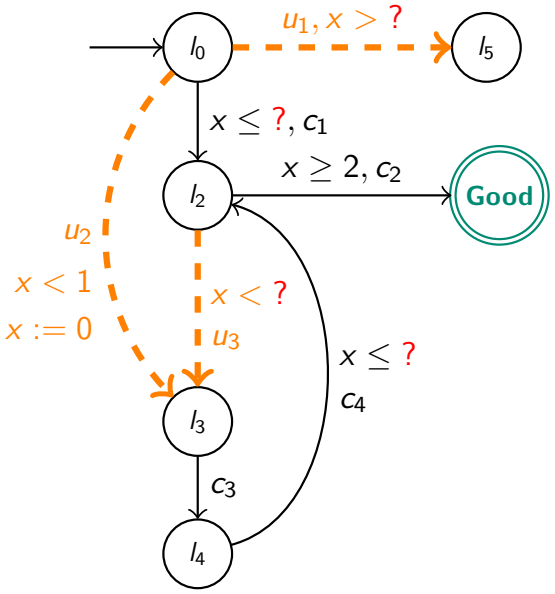
- unknown environment



Parametric Timed Games

Motivation:

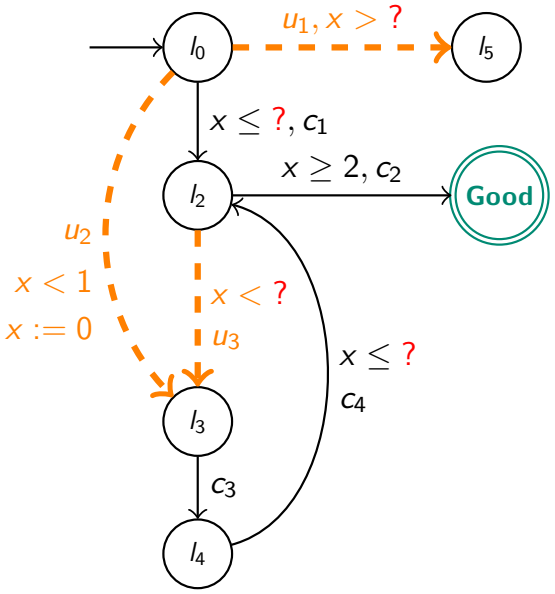
- unknown environment
- parametrized specifications



Parametric Timed Games

Motivation:

- unknown environment
- parametrized specifications
- solution: parameters in clock constraints



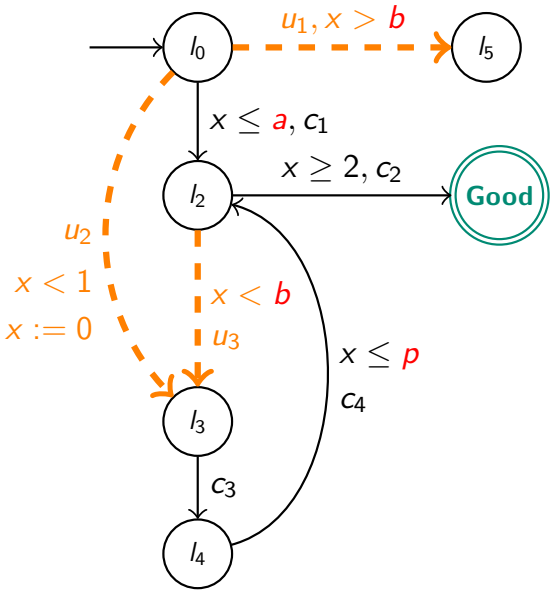
Parametric Timed Games

Motivation:

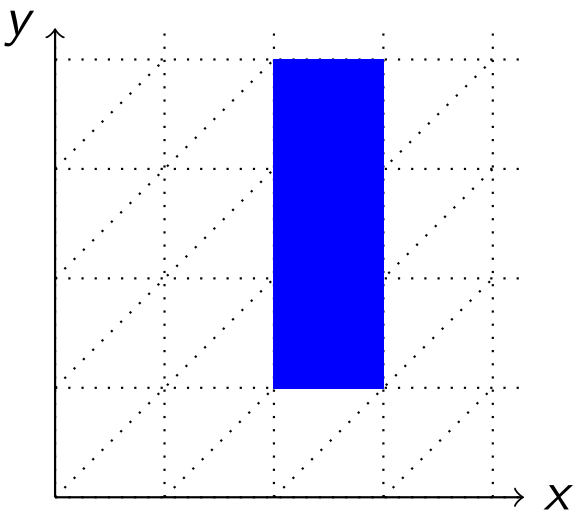
- unknown environment
- parametrized specifications
- solution: parameters in clock constraints

Parametric model:

- **Parametric timed game automata** (PGA) [JFLR12]
- *control problem* on PGA: compute parameter valuations s.t. there exists the winning strategy.

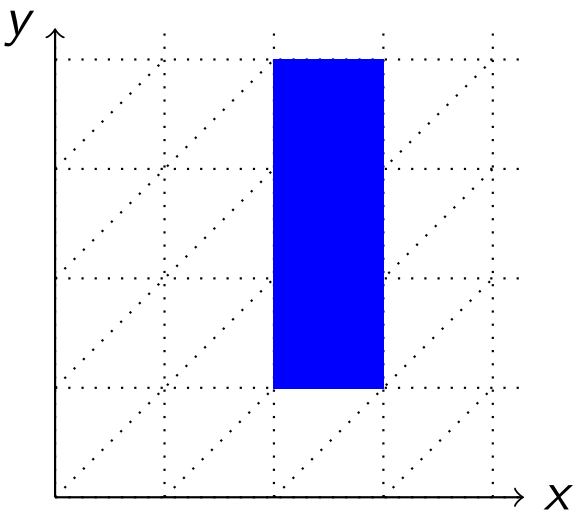


Backward Computation of Winning States in PGA [MPS95]



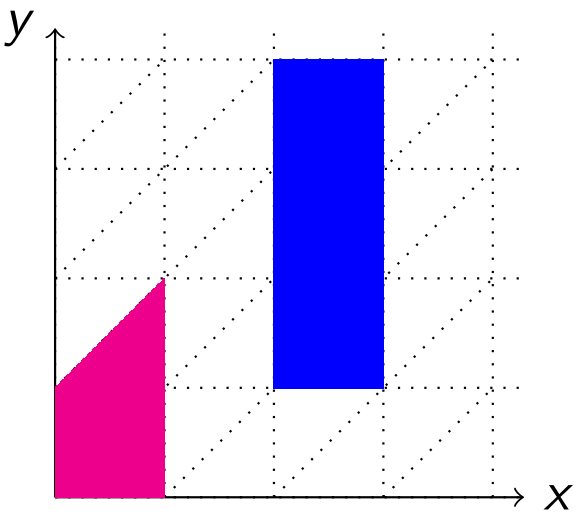
- zone X

Backward Computation of Winning States in PGA [MPS95]



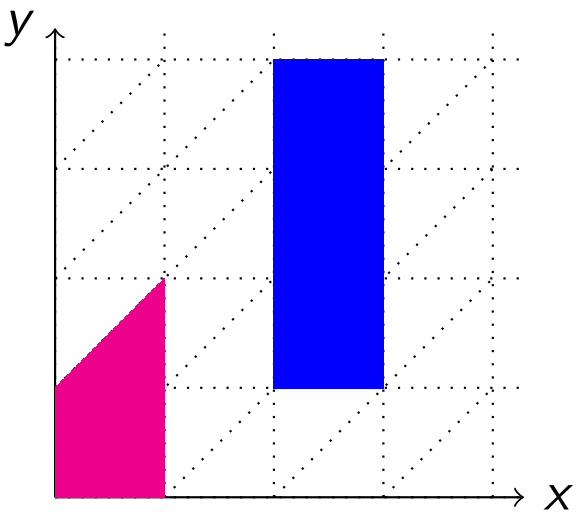
- zone X
- $\mathrm{Pred}^a(X) = \{\bullet \mid \bullet \xrightarrow{a} \bullet \in X\}$
- $\mathrm{uPred}(X) = \bigcup_{a \in uncont} \mathrm{Pred}^a(X)$
- $\mathrm{cPred}(X) = \bigcup_{a \in cont} \mathrm{Pred}^a(X)$

Backward Computation of Winning States in PGA [MPS95]



- zone X
- $\mathrm{Pred}^a(X) = \{\bullet \mid \bullet \xrightarrow{a} \bullet \in X\}$
- $\mathrm{uPred}(X) = \bigcup_{a \in uncont} \mathrm{Pred}^a(X)$
- $\mathrm{cPred}(X) = \bigcup_{a \in cont} \mathrm{Pred}^a(X)$

Backward Computation of Winning States in PGA [MPS95]

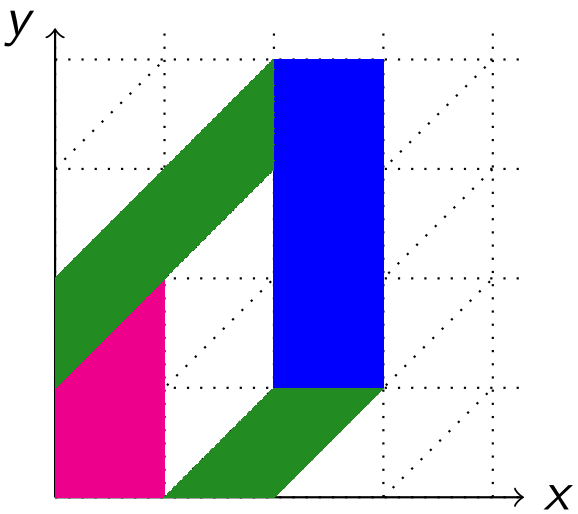


- zone X
- $\mathrm{Pred}^a(X) = \{\bullet \mid \bullet \xrightarrow{a} \bullet \in X\}$
- $\mathrm{uPred}(X) = \bigcup_{a \in uncont} \mathrm{Pred}^a(X)$
- $\mathrm{cPred}(X) = \bigcup_{a \in cont} \mathrm{Pred}^a(X)$

Safe-timed predecessors:

$$\mathrm{Pred_t}(\mathrm{Good}, \mathrm{Bad}) = \{\bullet \mid \exists t \geq 0, \bullet \xrightarrow{t} \bullet \text{ and } \forall 0 \leq t' \leq t, \bullet \xrightarrow{t'} \bullet \in \neg\, \mathrm{Bad}\}$$

Backward Computation of Winning States in PGA [MPS95]

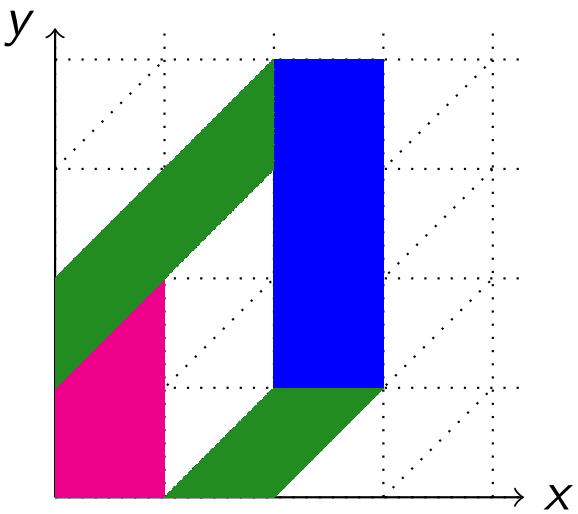


- zone X
- $\mathrm{Pred}^a(X) = \{\bullet \mid \bullet \xrightarrow{a} \bullet \in X\}$
- $\mathrm{uPred}(X) = \bigcup_{a \in uncont} \mathrm{Pred}^a(X)$
- $\mathrm{cPred}(X) = \bigcup_{a \in cont} \mathrm{Pred}^a(X)$

Safe-timed predecessors:

$$\mathrm{Pred_t}(\mathrm{Good}, \mathrm{Bad}) = \{\bullet \mid \exists t \geq 0, \bullet \xrightarrow{t} \bullet \text{ and } \forall 0 \leq t' \leq t, \bullet \xrightarrow{t'} \bullet \in \neg\, \mathrm{Bad}\}$$

Backward Computation of Winning States in PGA [MPS95]



- zone X
- $\mathrm{Pred}^a(X) = \{\bullet \mid \bullet \xrightarrow{a} \bullet \in X\}$
- $\mathrm{uPred}(X) = \bigcup_{a \in uncont} \mathrm{Pred}^a(X)$
- $\mathrm{cPred}(X) = \bigcup_{a \in cont} \mathrm{Pred}^a(X)$

Safe-timed predecessors:

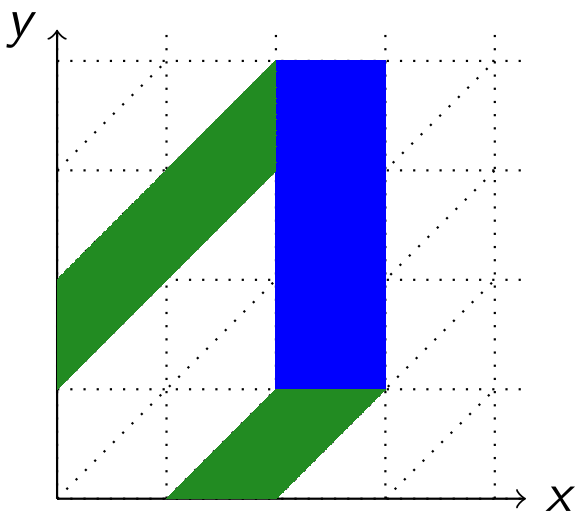
$$\mathrm{Pred_t}(\mathrm{Good}, \mathrm{Bad}) = \{\bullet \mid \exists t \geq 0, \bullet \xrightarrow{t} \bullet \text{ and } \forall 0 \leq t' \leq t, \bullet \xrightarrow{t'} \bullet \in \neg\, \mathrm{Bad}\}$$

Computation of the winning states: *controllable predecessors*

$$\mathrm{Pred_t}(X \cup \mathrm{cPred}(X), \mathrm{uPred}(\neg X))$$

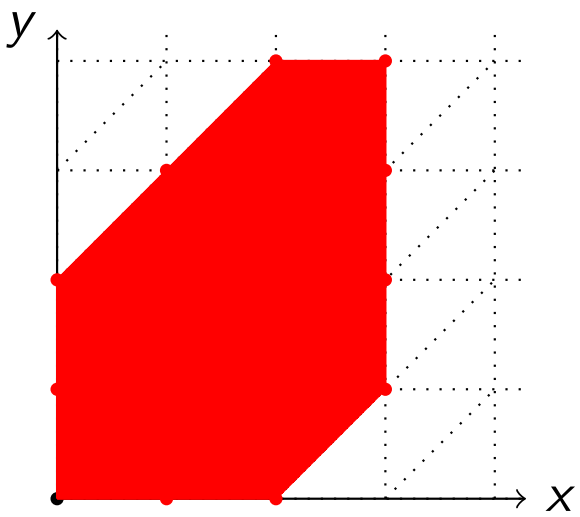
Synthesis of Integer Parameters: Integer Shape

Problem: in the parametric domain the computation might not terminate.



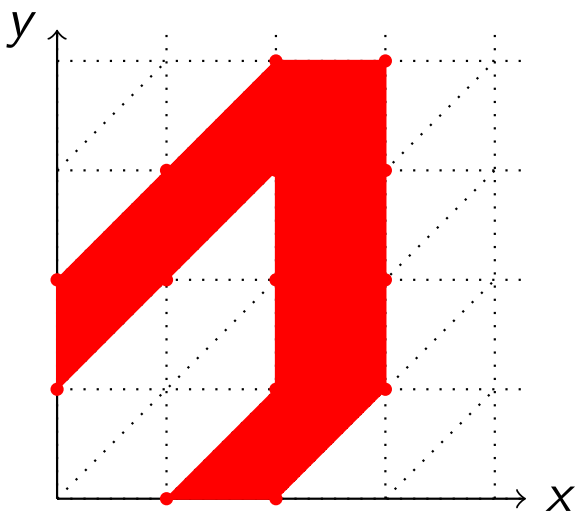
Synthesis of Integer Parameters: Integer Shape

Problem: in the parametric domain the computation might not terminate.



Synthesis of Integer Parameters: Integer Shape

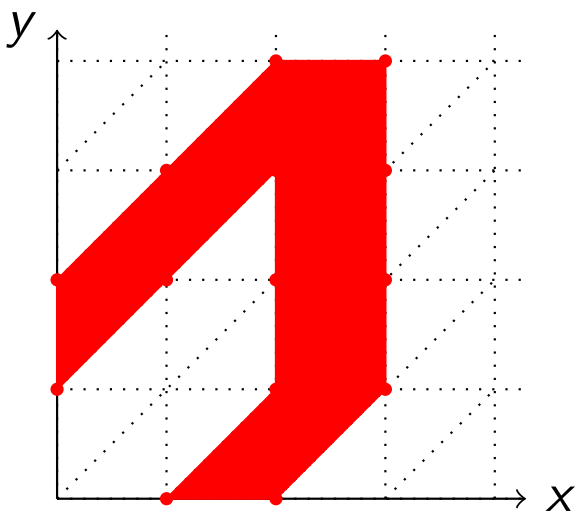
Problem: in the parametric domain the computation might not terminate.



- $\mathsf{IntShape}(S) = \bigcup_i \mathsf{IntHull}(S_i)$
- Apply IntShape in the backwards computation.

Synthesis of Integer Parameters: Integer Shape

Problem: in the parametric domain the computation might not terminate.



- $\mathsf{IntShape}(S) = \bigcup_i \mathsf{IntHull}(S_i)$
- Apply IntShape in the backwards computation.

- Our algorithm: bounded integer computation (forward: IntHull, backward: IntShape) [JLR13b].

Example

Backward computation starts from:
$(Goal, x \geq 2)$.

The set of **winning states**:

- $(\ell_3, (x \geq 0 \wedge a \geq b))$,
- $(\ell_2, (x \geq b) \vee (x \geq 0 \wedge a \geq b))$,
- $(\ell_1, (x \leq a \wedge a \geq b))$,
- $(\ell_0, (a \geq b) \wedge \Big((a > b+1) \vee ((a \leq b+1) \wedge (a > 0))\Big) \wedge (x \geq 0))$.

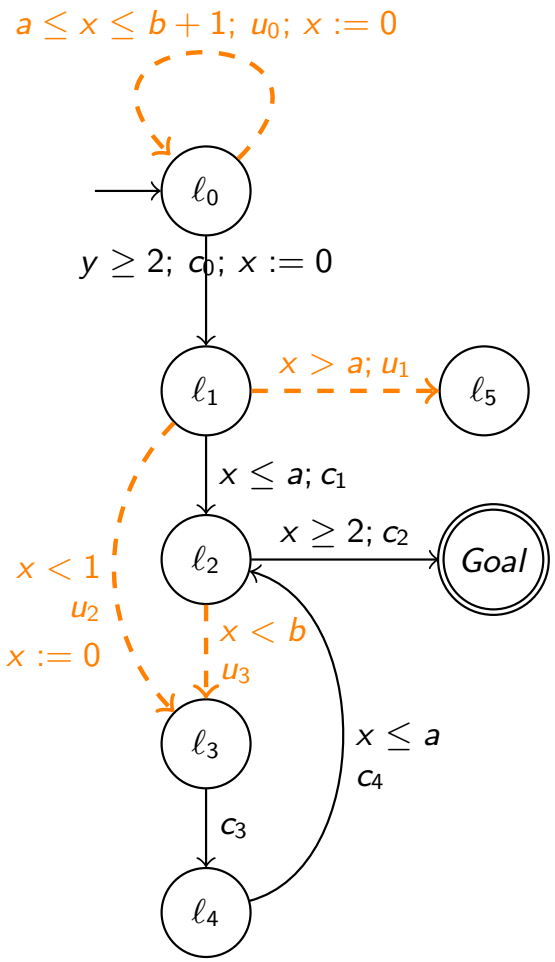


Figure : PGA

Example

Backward computation starts from:
$(Goal, x \geq 2)$.

The set of **winning states**:

- $(\ell_3, (x \geq 0 \wedge a \geq b))$,
- $(\ell_2, (x \geq b) \vee (x \geq 0 \wedge a \geq b))$,
- $(\ell_1, (x \leq a \wedge a \geq b))$,
- $(\ell_0, (a \geq b) \wedge \Big((a > b+1) \vee ((a \leq b+1) \wedge (a > 0))\Big) \wedge (x \geq 0))$.

The **set of constraints** obtained:

- $(a \geq b) \wedge ((a > b+1) \vee (a > 0))$

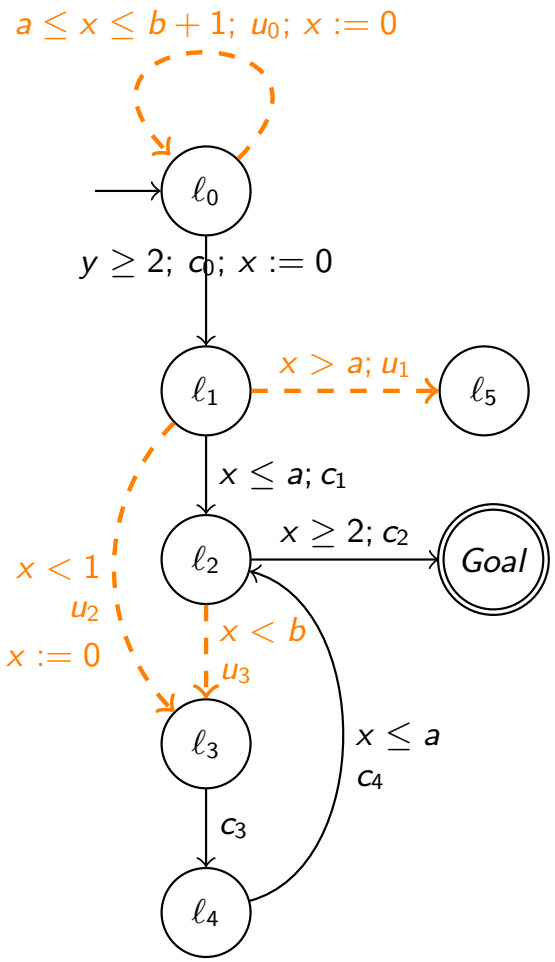


Figure : PGA

Example

The **winning strategy** consists in:

- delaying in all states $(\ell_0, \{y < 2\})$,
- doing c_0 in all states $(\ell_0, \{y \geq 2\})$,
- doing c_1 in all states $(\ell_1, \{x \leq a\})$,
- delaying in all states $(\ell_2, \{x < 2\})$,
- doing c_2 in all states $(\ell_2, \{x \geq 2\})$,
- doing c_3 in all states $(\ell_3, \{x \geq 0\})$,
- delaying in all states $(\ell_4, \{x < b\})$,
- doing c_4 in all states $(\ell_4, \{x \geq b \wedge x \leq a\})$.

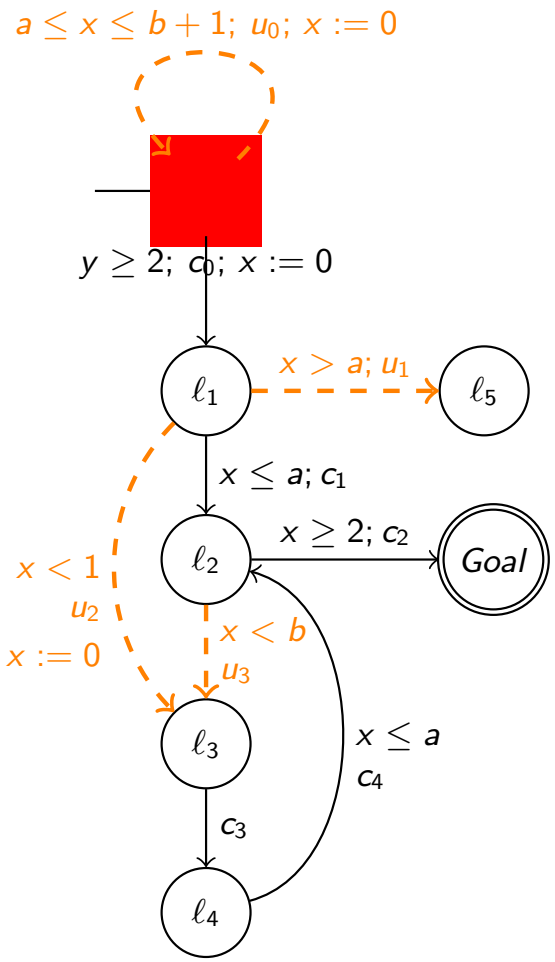


Figure : PGA

Example

The **winning strategy** consists in:

- delaying in all states $(\ell_0, \{y < 2\})$,
- doing c_0 in all states $(\ell_0, \{y \geq 2\})$,
- doing c_1 in all states $(\ell_1, \{x \leq a\})$,
- delaying in all states $(\ell_2, \{x < 2\})$,
- doing c_2 in all states $(\ell_2, \{x \geq 2\})$,
- doing c_3 in all states $(\ell_3, \{x \geq 0\})$,
- delaying in all states $(\ell_4, \{x < b\})$,
- doing c_4 in all states $(\ell_4, \{x \geq b \wedge x \leq a\})$.

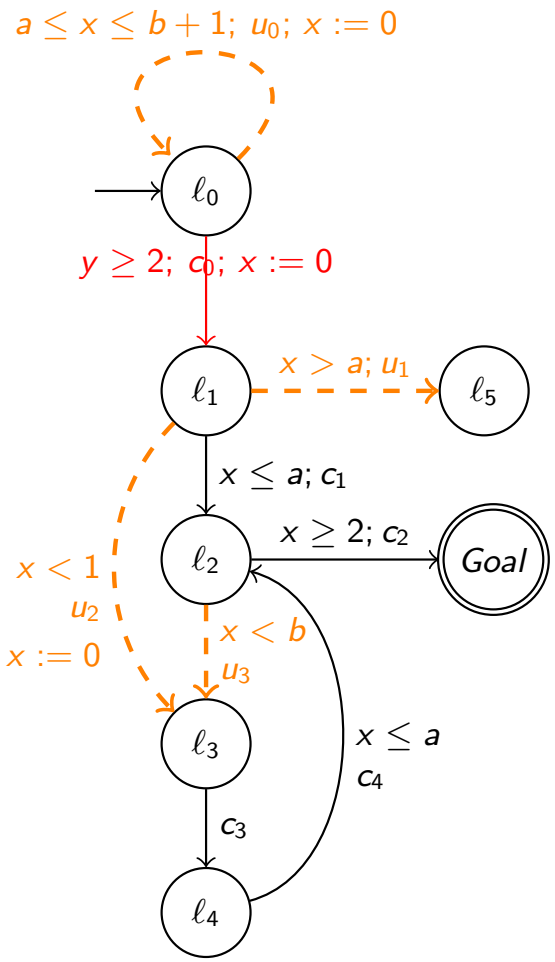


Figure : PGA

Example

The **winning strategy** consists in:

- delaying in all states $(\ell_0, \{y < 2\})$,
- doing c_0 in all states $(\ell_0, \{y \geq 2\})$,
- doing c_1 in all states $(\ell_1, \{x \leq a\})$,
- delaying in all states $(\ell_2, \{x < 2\})$,
- doing c_2 in all states $(\ell_2, \{x \geq 2\})$,
- doing c_3 in all states $(\ell_3, \{x \geq 0\})$,
- delaying in all states $(\ell_4, \{x < b\})$,
- doing c_4 in all states $(\ell_4, \{x \geq b \wedge x \leq a\})$.

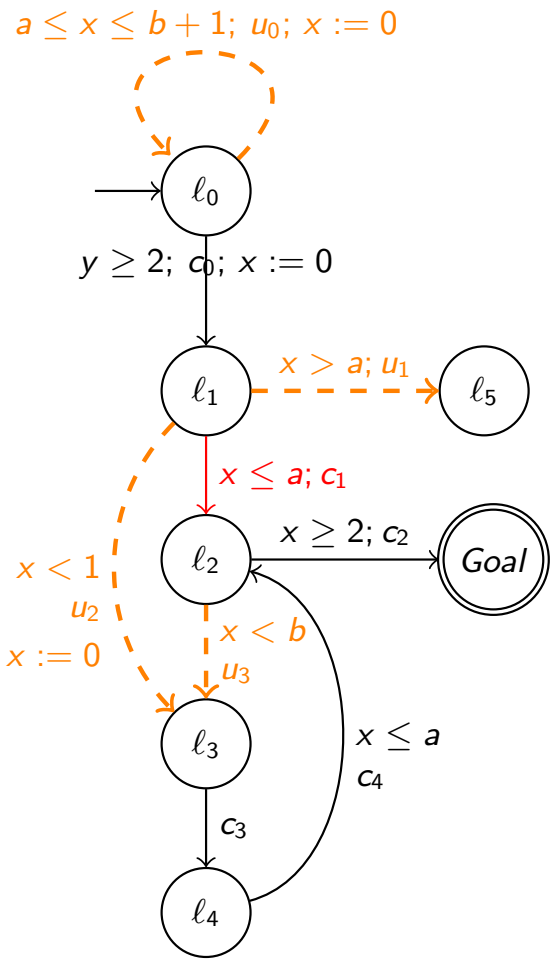


Figure : PGA

Example

The **winning strategy** consists in:

- delaying in all states $(\ell_0, \{y < 2\})$,
- doing c_0 in all states $(\ell_0, \{y \geq 2\})$,
- doing c_1 in all states $(\ell_1, \{x \leq a\})$,
- delaying in all states $(\ell_2, \{x < 2\})$,
- doing c_2 in all states $(\ell_2, \{x \geq 2\})$,
- doing c_3 in all states $(\ell_3, \{x \geq 0\})$,
- delaying in all states $(\ell_4, \{x < b\})$,
- doing c_4 in all states $(\ell_4, \{x \geq b \wedge x \leq a\})$.

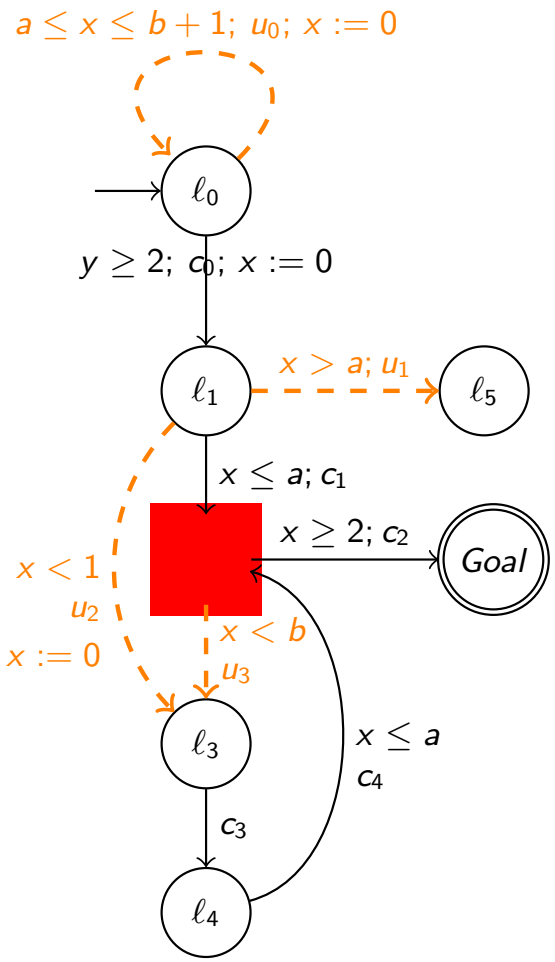


Figure : PGA

Example

The **winning strategy** consists in:

- delaying in all states $(\ell_0, \{y < 2\})$,
- doing c_0 in all states $(\ell_0, \{y \geq 2\})$,
- doing c_1 in all states $(\ell_1, \{x \leq a\})$,
- delaying in all states $(\ell_2, \{x < 2\})$,
- doing c_2 in all states $(\ell_2, \{x \geq 2\})$,
- doing c_3 in all states $(\ell_3, \{x \geq 0\})$,
- delaying in all states $(\ell_4, \{x < b\})$,
- doing c_4 in all states $(\ell_4, \{x \geq b \wedge x \leq a\})$.

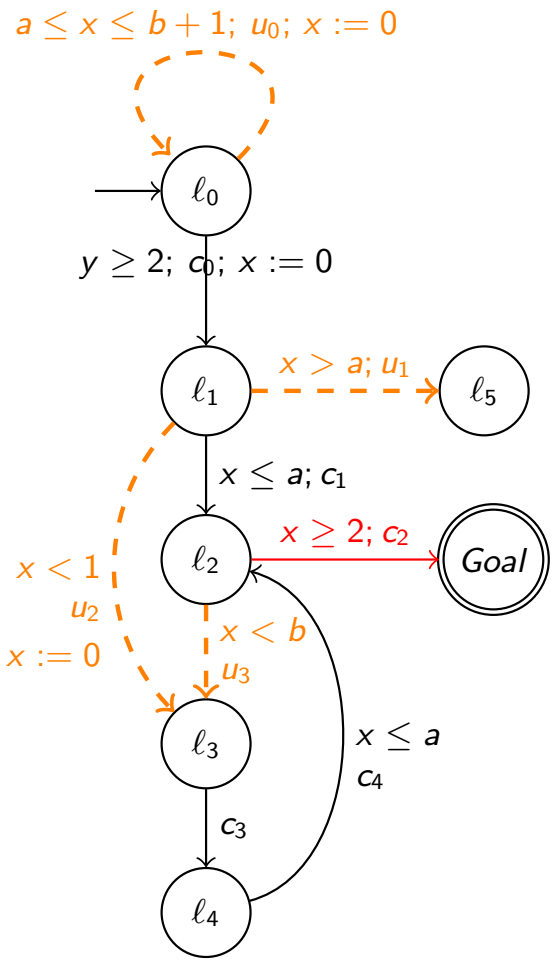


Figure : PGA

Example

The **winning strategy** consists in:

- delaying in all states $(\ell_0, \{y < 2\})$,
- doing c_0 in all states $(\ell_0, \{y \geq 2\})$,
- doing c_1 in all states $(\ell_1, \{x \leq a\})$,
- delaying in all states $(\ell_2, \{x < 2\})$,
- doing c_2 in all states $(\ell_2, \{x \geq 2\})$,
- doing c_3 in all states $(\ell_3, \{x \geq 0\})$,
- delaying in all states $(\ell_4, \{x < b\})$,
- doing c_4 in all states $(\ell_4, \{x \geq b \wedge x \leq a\})$.

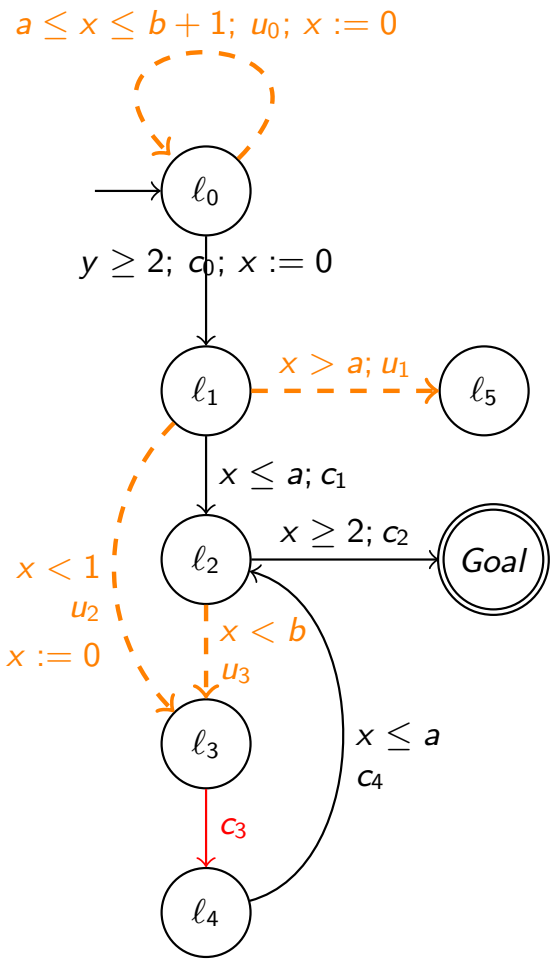


Figure : PGA

Example

The **winning strategy** consists in:

- delaying in all states $(\ell_0, \{y < 2\})$,
- doing c_0 in all states $(\ell_0, \{y \geq 2\})$,
- doing c_1 in all states $(\ell_1, \{x \leq a\})$,
- delaying in all states $(\ell_2, \{x < 2\})$,
- doing c_2 in all states $(\ell_2, \{x \geq 2\})$,
- doing c_3 in all states $(\ell_3, \{x \geq 0\})$,
- delaying in all states $(\ell_4, \{x < b\})$,
- doing c_4 in all states $(\ell_4, \{x \geq b \wedge x \leq a\})$.

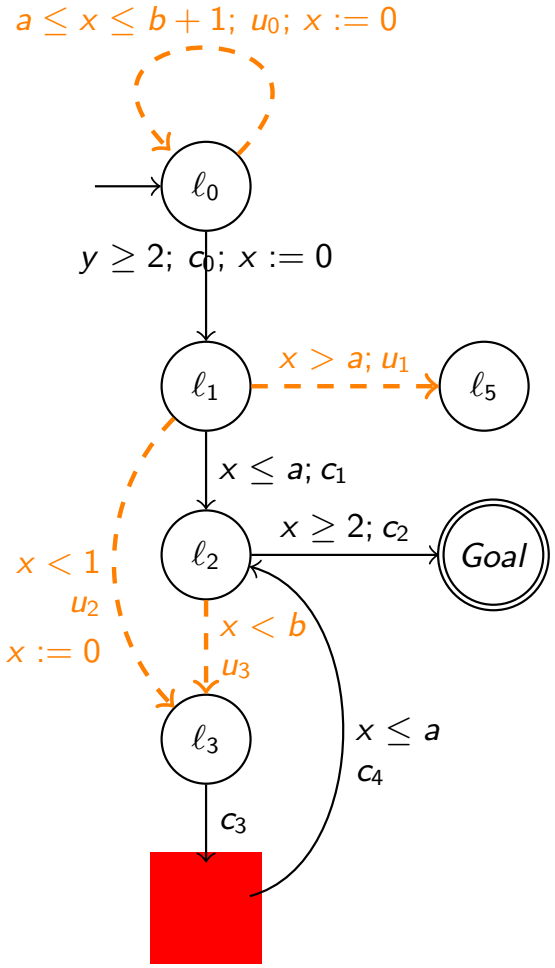


Figure : PGA

Example

The **winning strategy** consists in:

- delaying in all states $(\ell_0, \{y < 2\})$,
- doing c_0 in all states $(\ell_0, \{y \geq 2\})$,
- doing c_1 in all states $(\ell_1, \{x \leq a\})$,
- delaying in all states $(\ell_2, \{x < 2\})$,
- doing c_2 in all states $(\ell_2, \{x \geq 2\})$,
- doing c_3 in all states $(\ell_3, \{x \geq 0\})$,
- delaying in all states $(\ell_4, \{x < b\})$,
- doing c_4 in all states $(\ell_4, \{x \geq b \wedge x \leq a\})$.

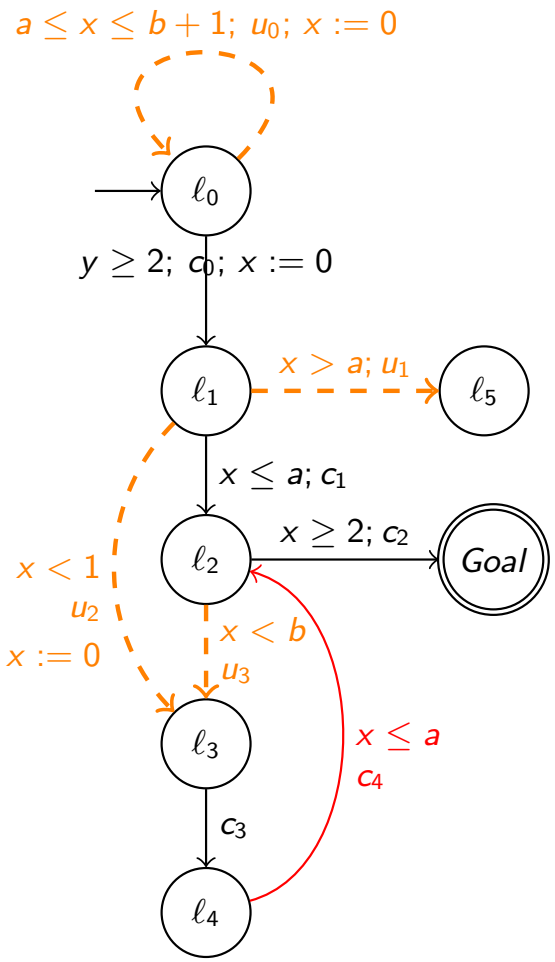
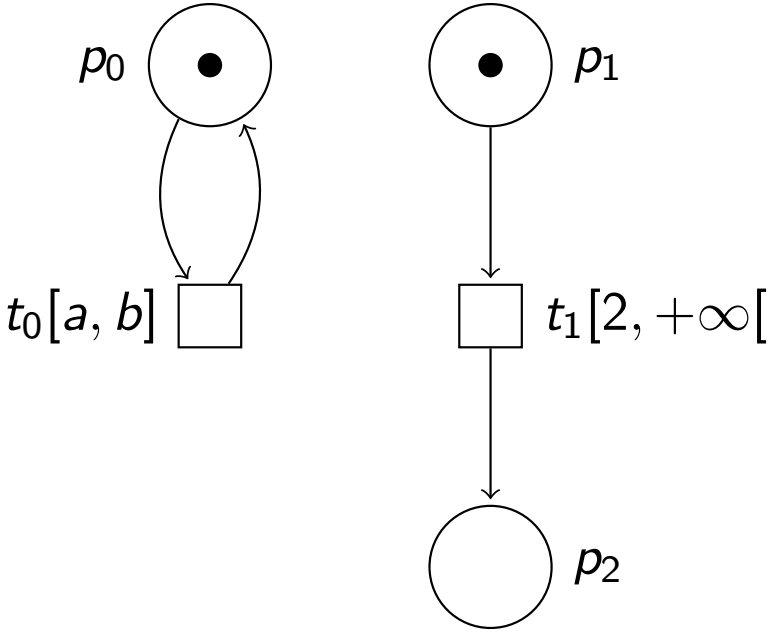


Figure : PGA

Outline

Parametric Time Petri Nets



Results for Parametric TPNs

- There are several time-bisimulation preserving **transformations** between TA and TPNs allowing to transfer (un)decidability results (e.g. [BCH+05, CR06]);

Results for Parametric TPNs

- There are several time-bisimulation preserving **transformations** between TA and TPNs allowing to transfer (un)decidability results (e.g. [BCH$^+$05, CR06]);
- There are two main symbolic states abstractions for TPN:

Results for Parametric TPNs

- There are several time-bisimulation preserving **transformations** between TA and TPNs allowing to transfer (un)decidability results (e.g. [BCH$^+$05, CR06]);
- There are two main symbolic states abstractions for TPN:
  - The zone-based approach is **directly** applicable [GRR06];

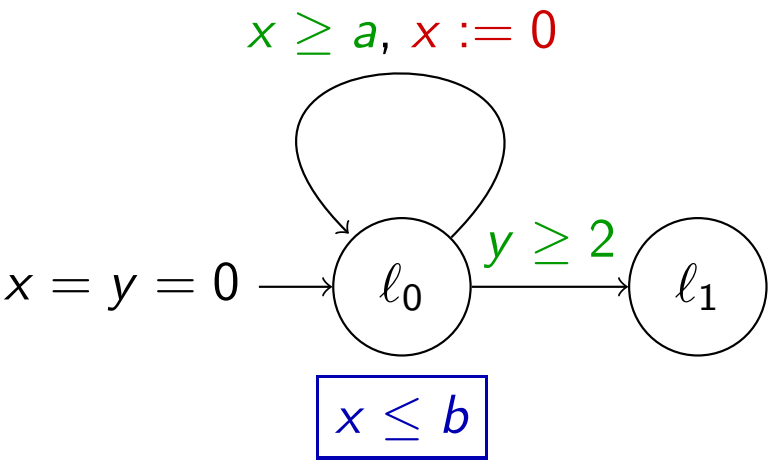
Results for Parametric TPNs

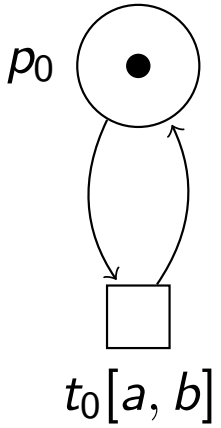
- There are several time-bisimulation preserving **transformations** between TA and TPNs allowing to transfer (un)decidability results (e.g. [BCH$^+$05, CR06]);
- There are two main symbolic states abstractions for TPN:
  - The zone-based approach is **directly** applicable [GRR06];
  - The historical approach is **state classes** [BD91]: they have **all** the good properties required before!

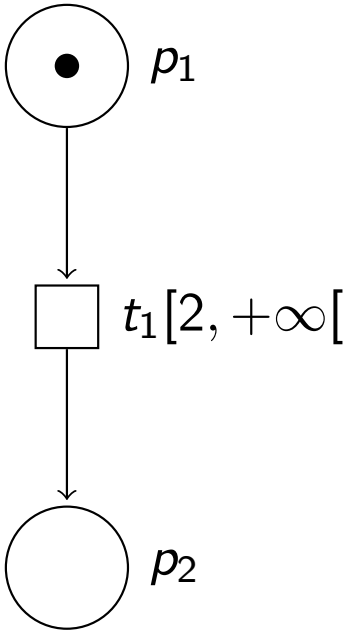
Results for Parametric TPNs

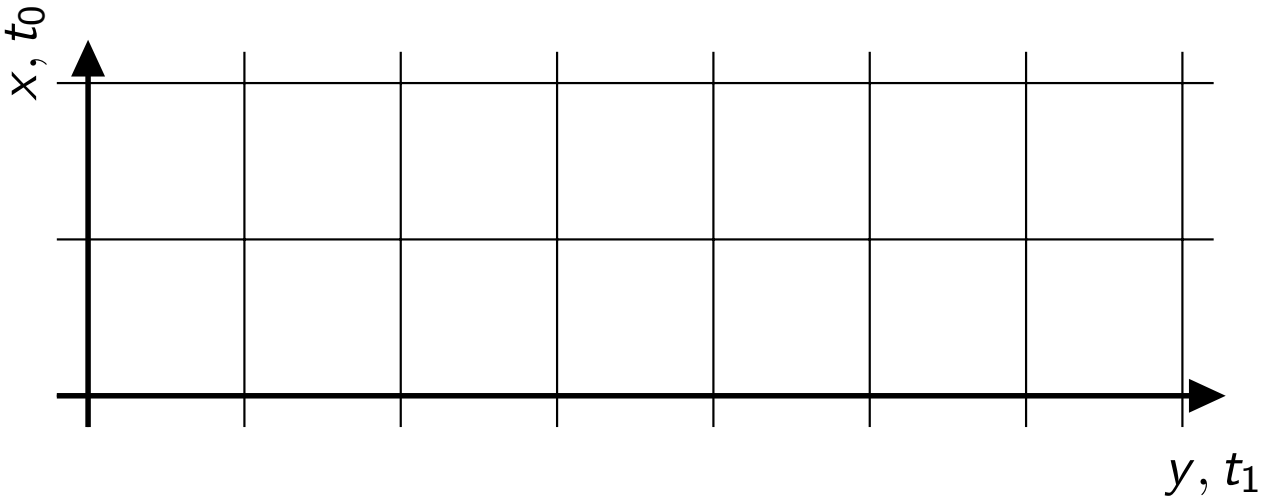
- There are several time-bisimulation preserving **transformations** between TA and TPNs allowing to transfer (un)decidability results (e.g. [BCH+05, CR06]);
- There are two main symbolic states abstractions for TPN:
  - The zone-based approach is **directly** applicable [GRR06];
  - The historical approach is **state classes** [BD91]: they have **all** the good properties required before!
- In both cases, we can use the integer parameters approach.

Termination in the Bounded Case

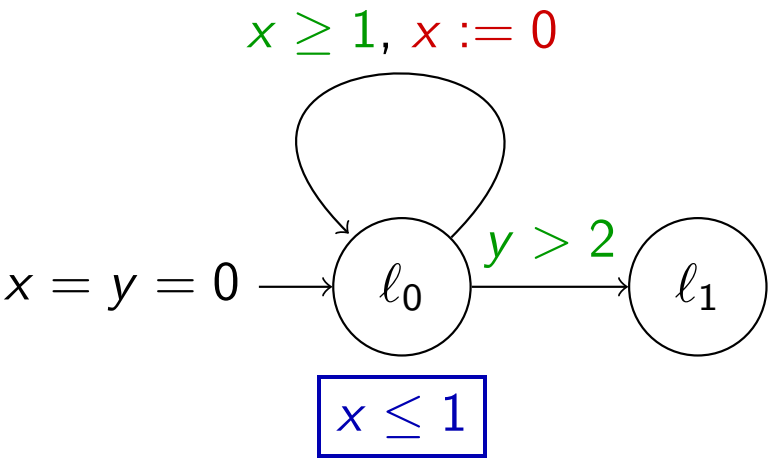


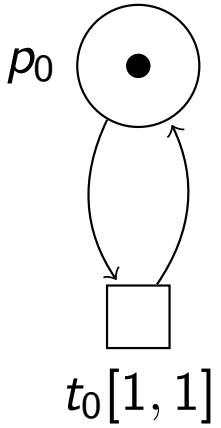


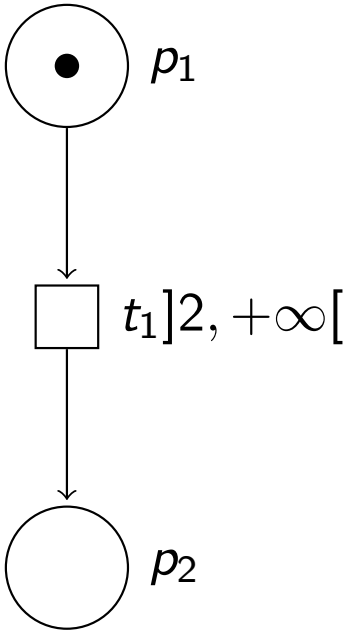


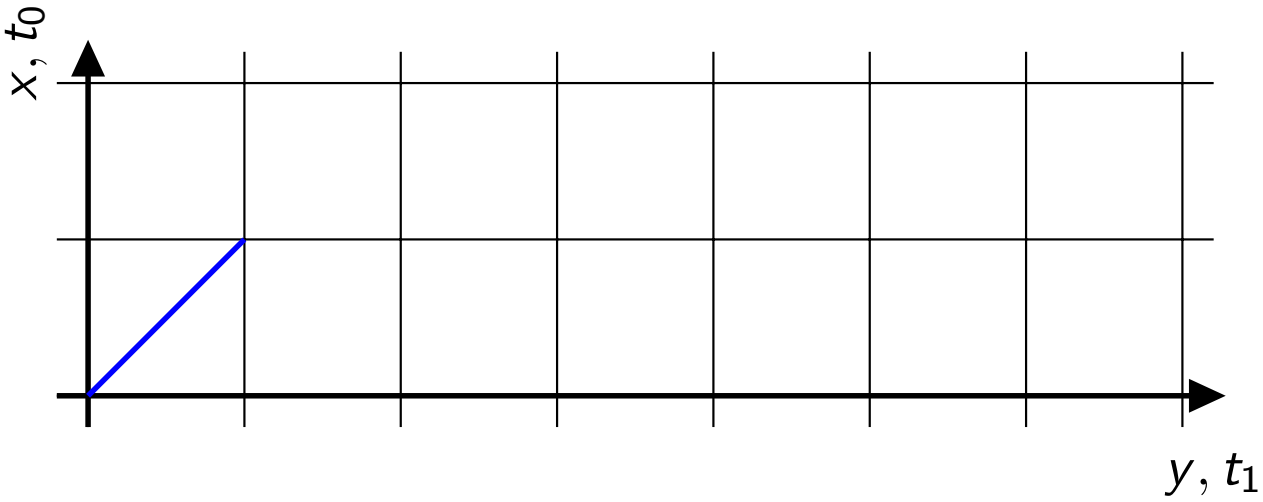


Termination in the Bounded Case

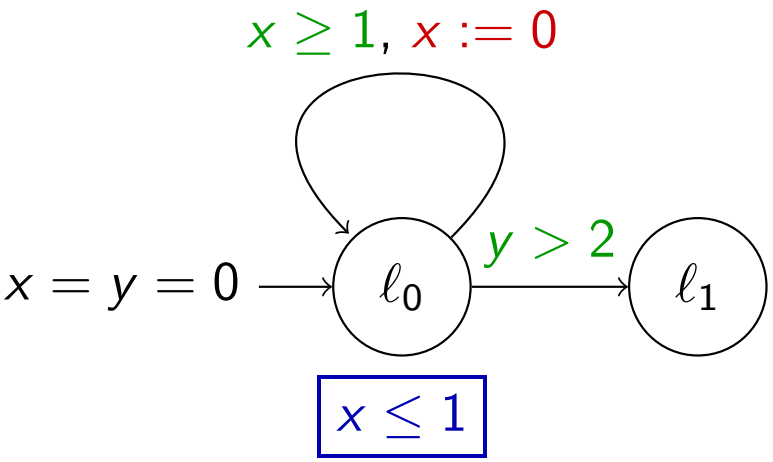


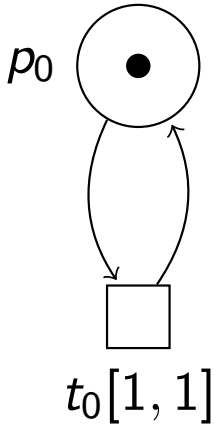


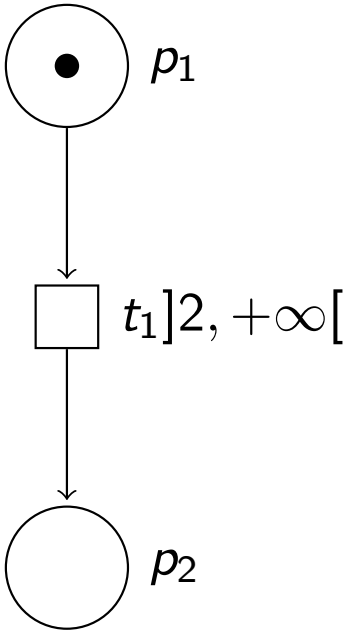


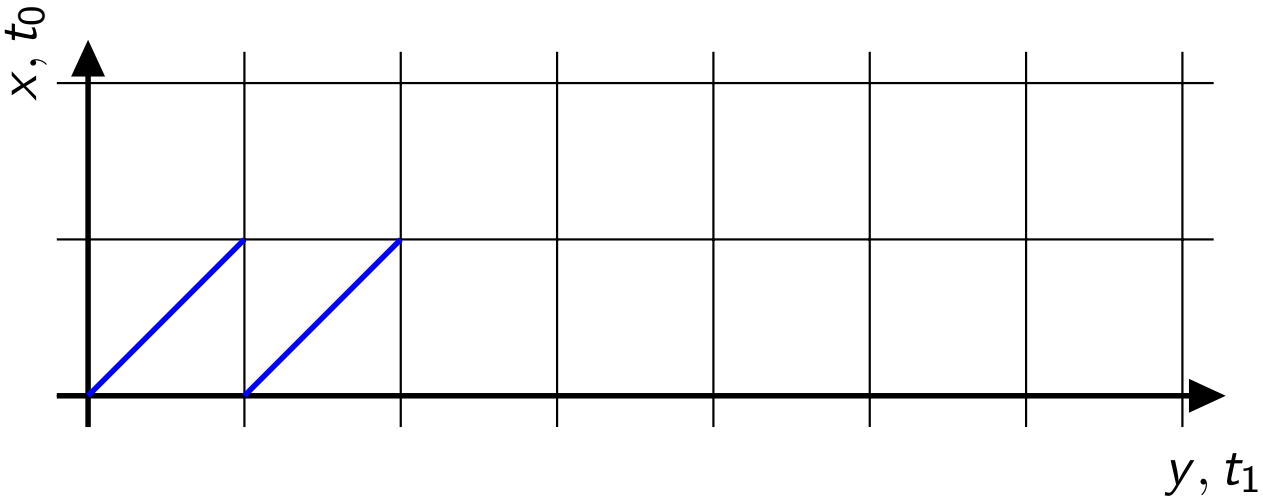


Termination in the Bounded Case

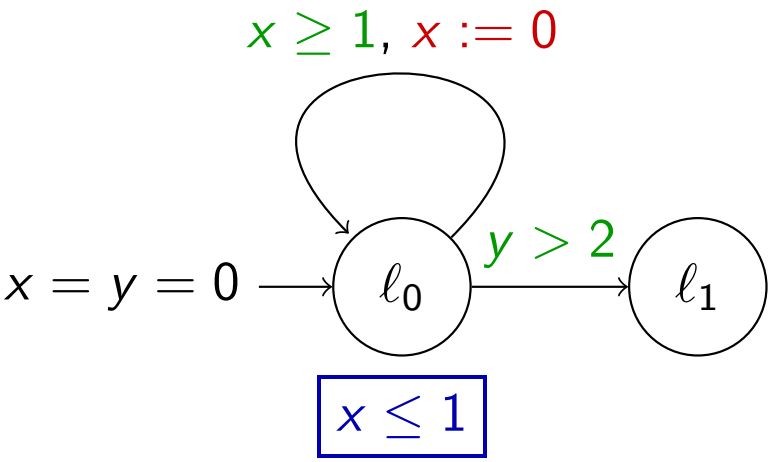


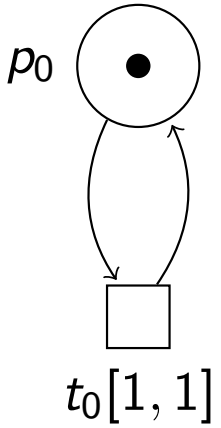


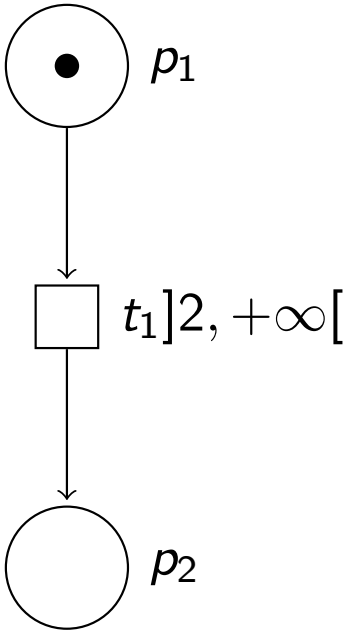


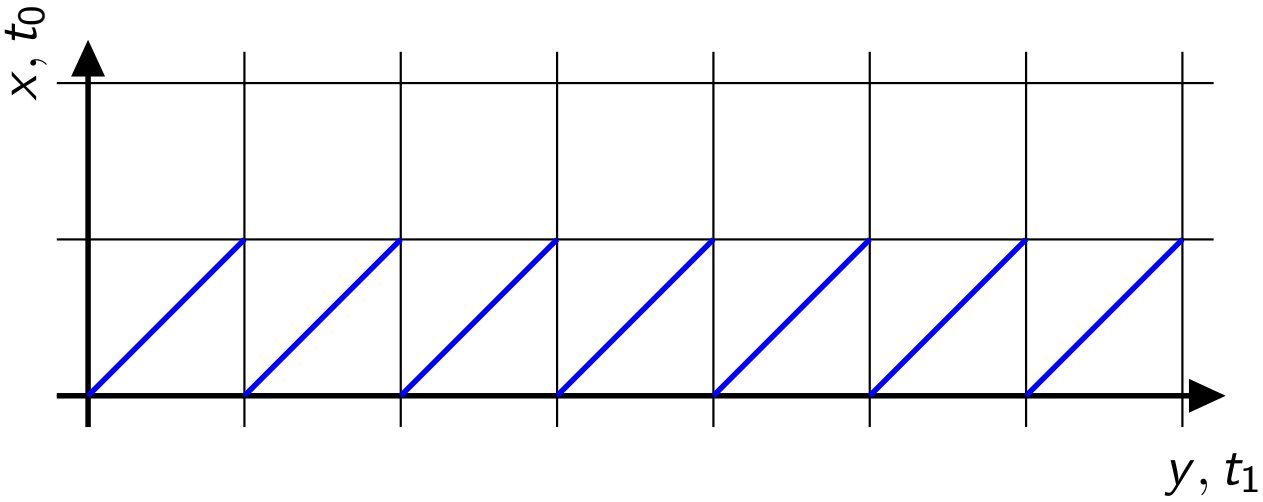


Termination in the Bounded Case

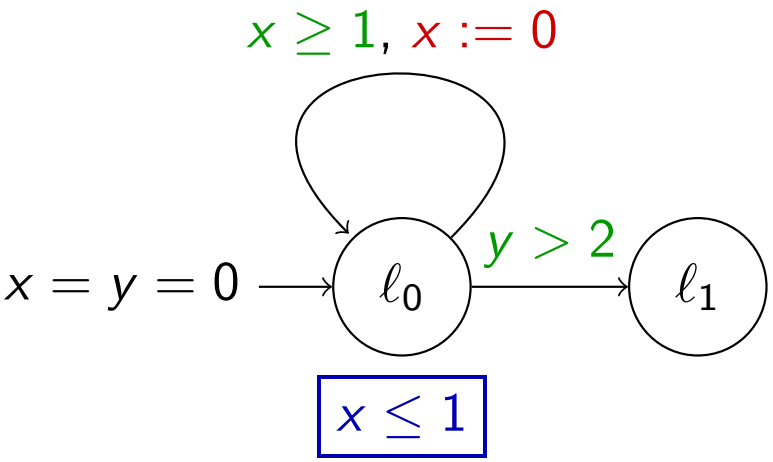


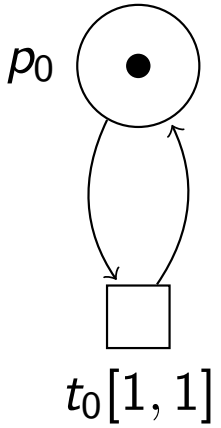


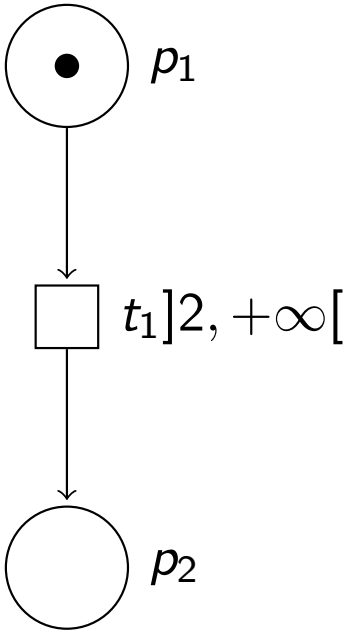


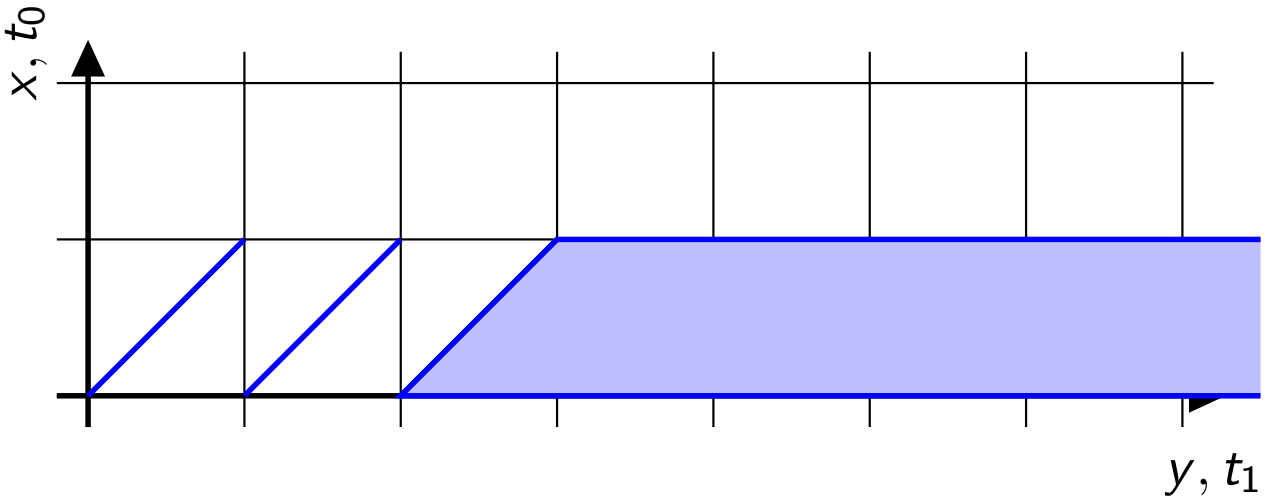


Termination in the Bounded Case

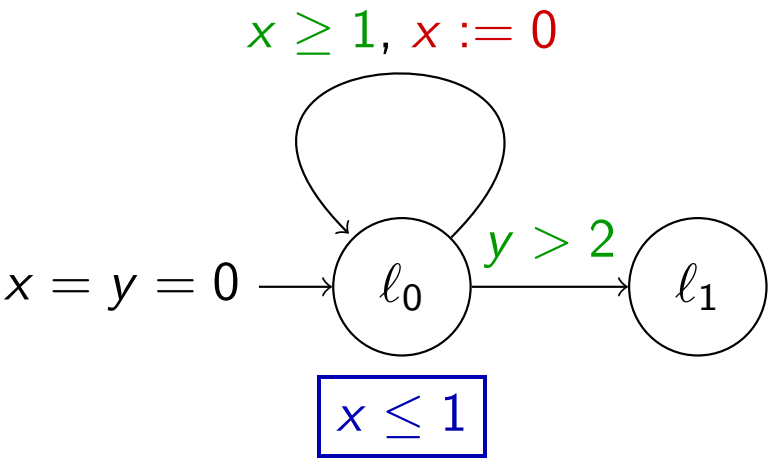


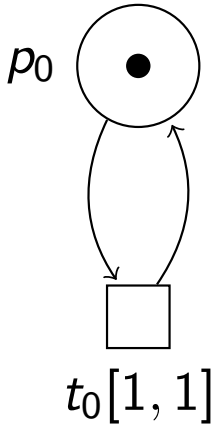


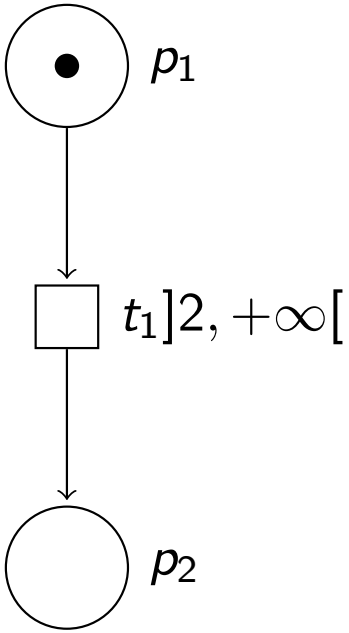


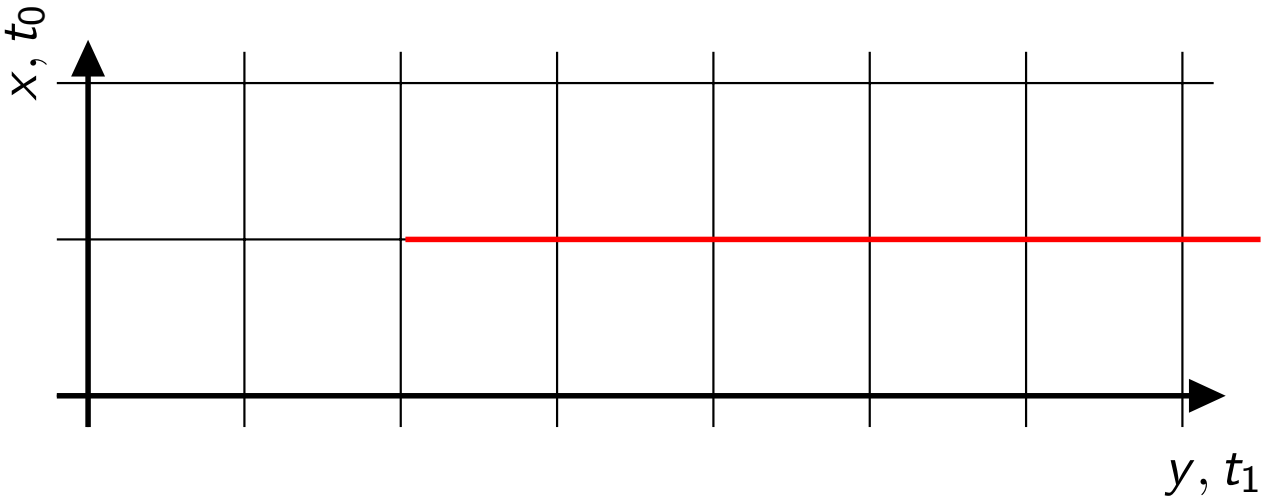


Termination in the Bounded Case

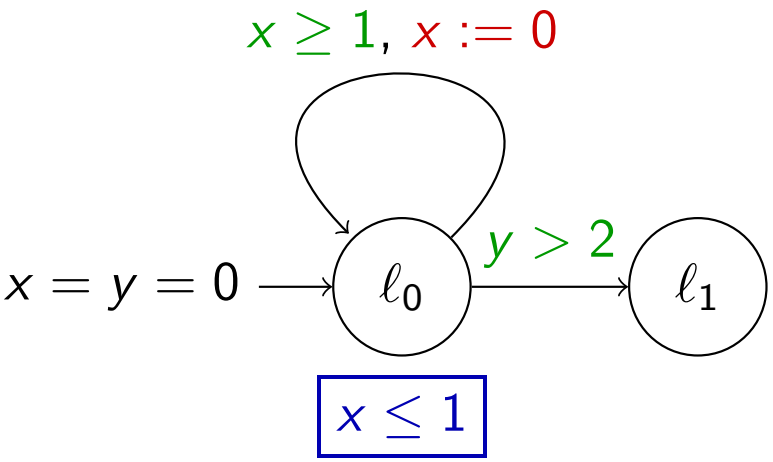


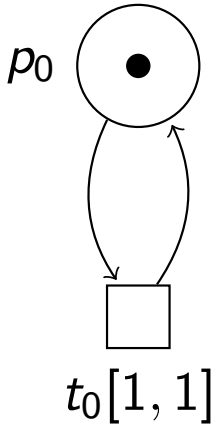


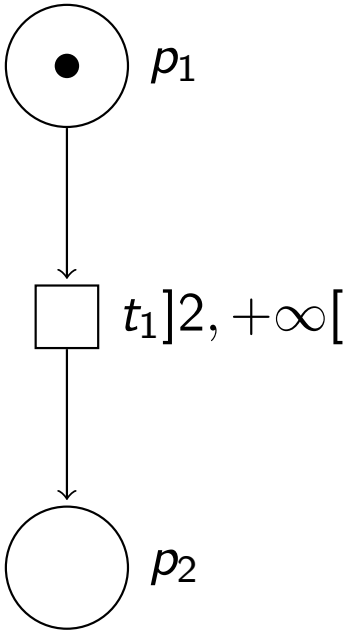


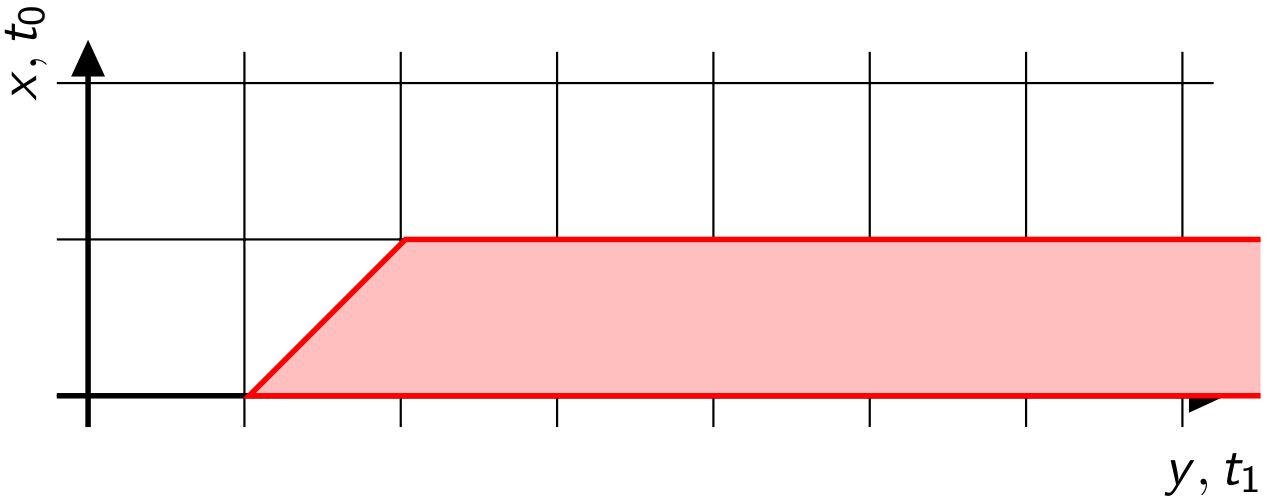


Termination in the Bounded Case

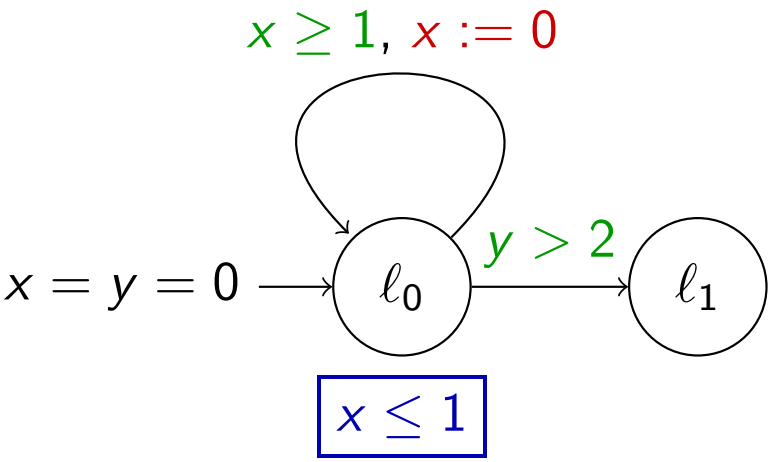


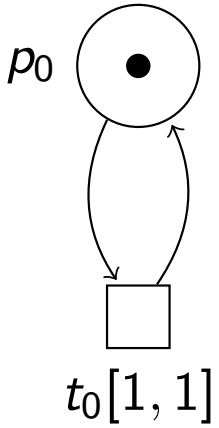


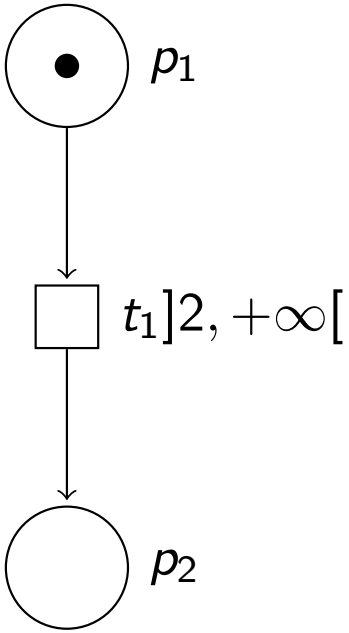


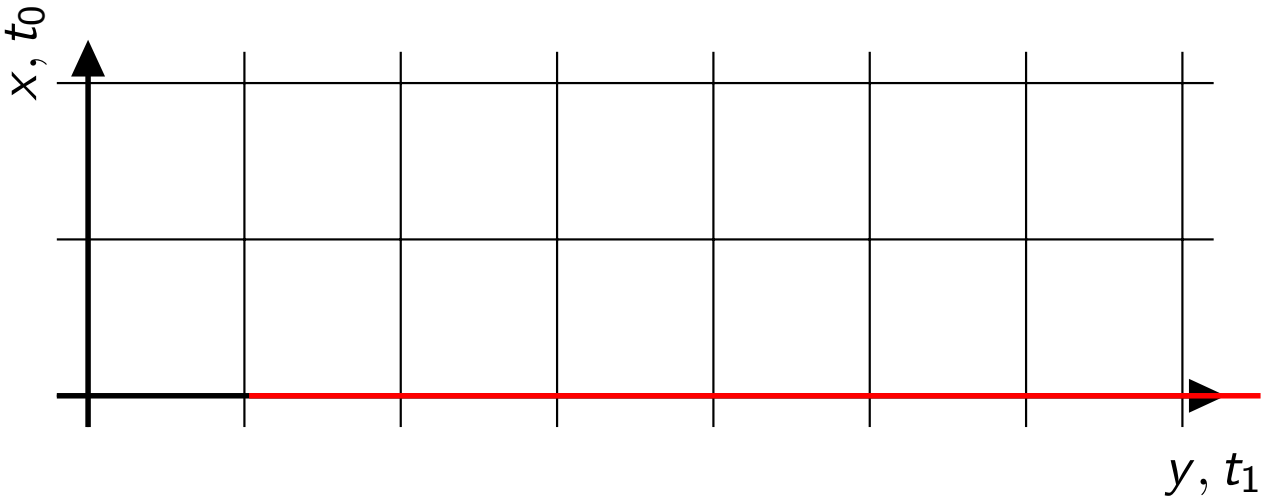


Termination in the Bounded Case

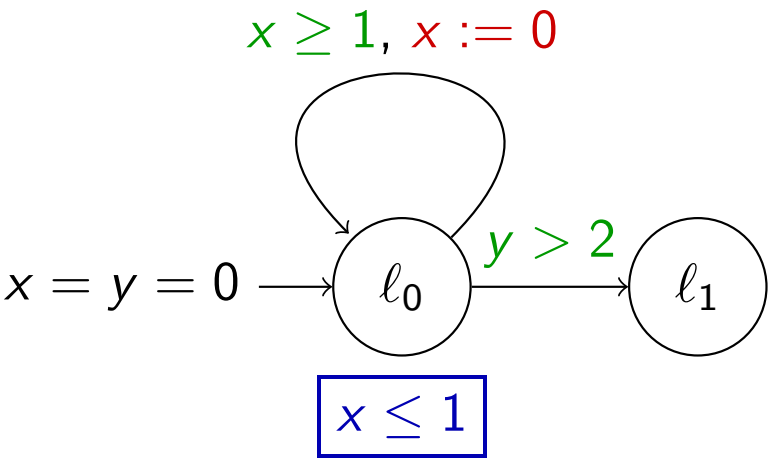


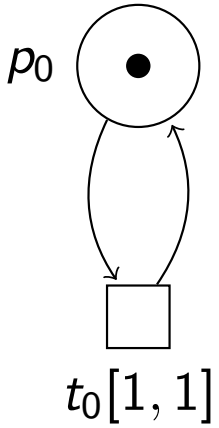


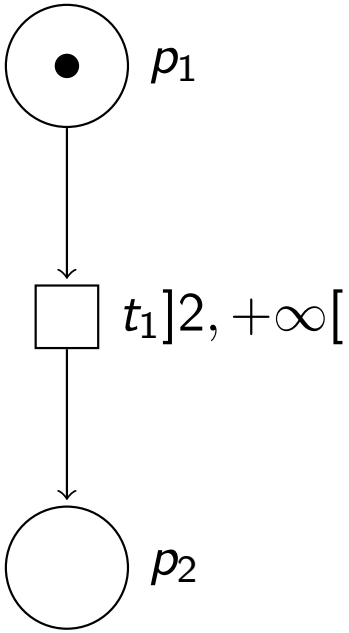


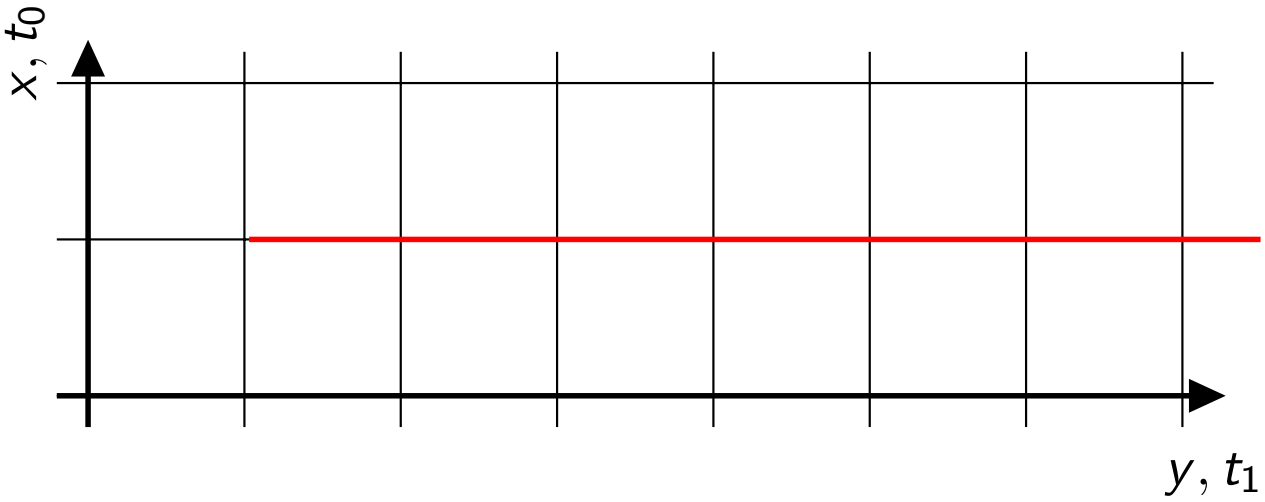


Termination in the Bounded Case

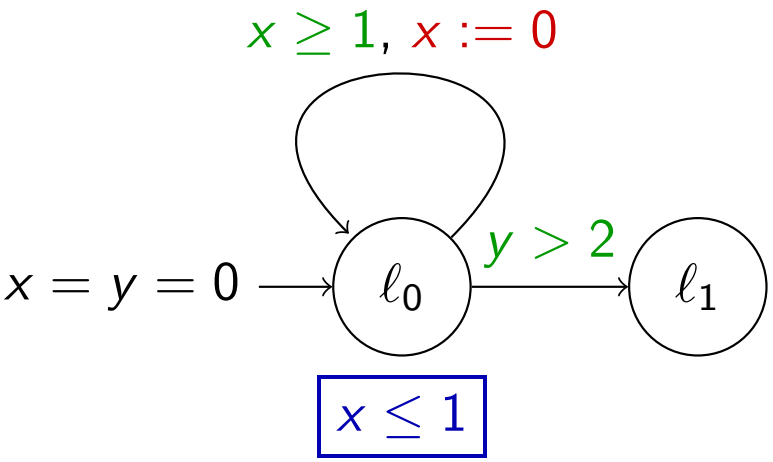


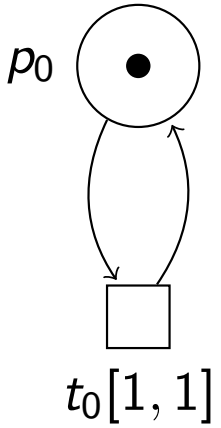


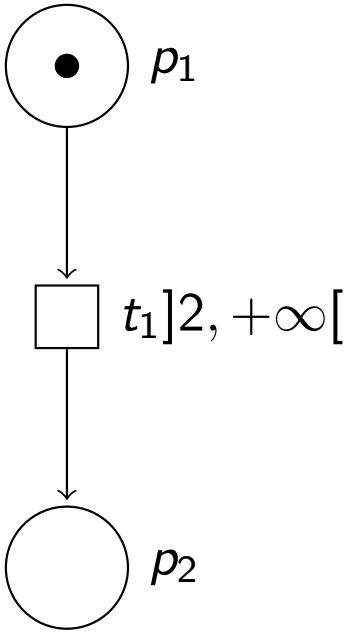


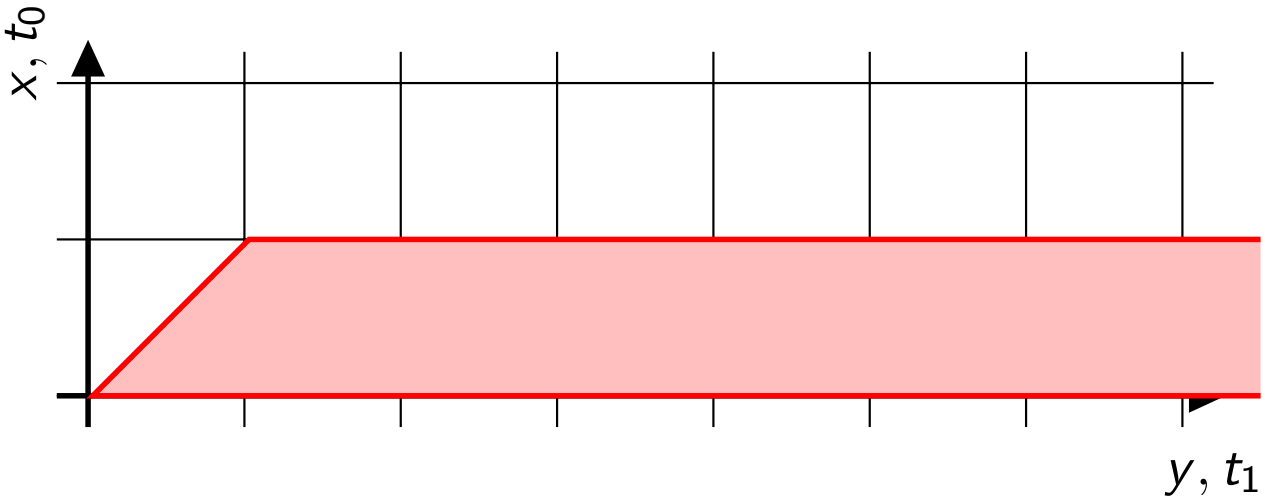


Termination in the Bounded Case

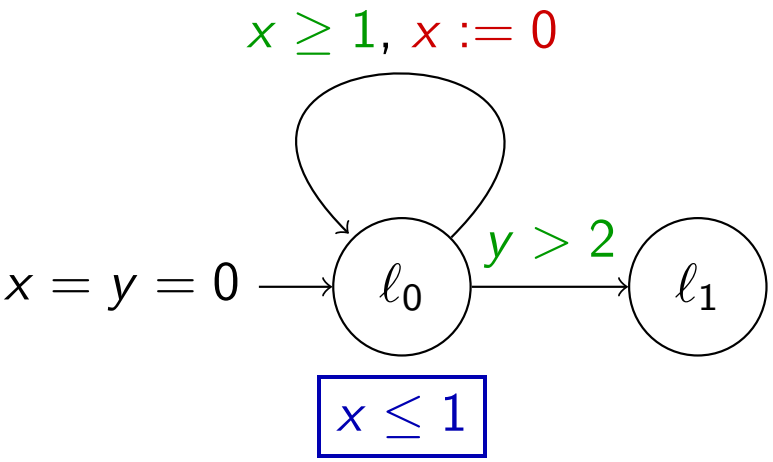


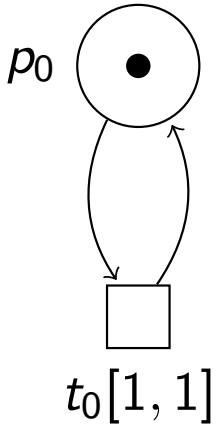


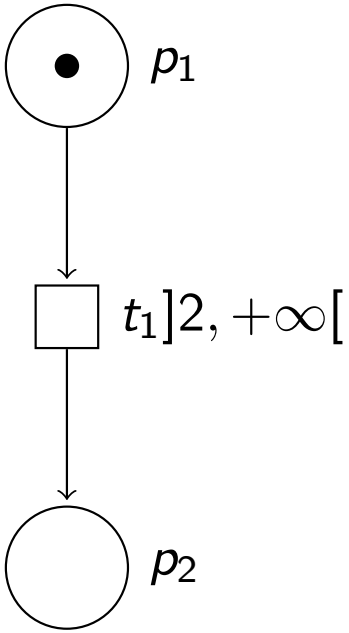


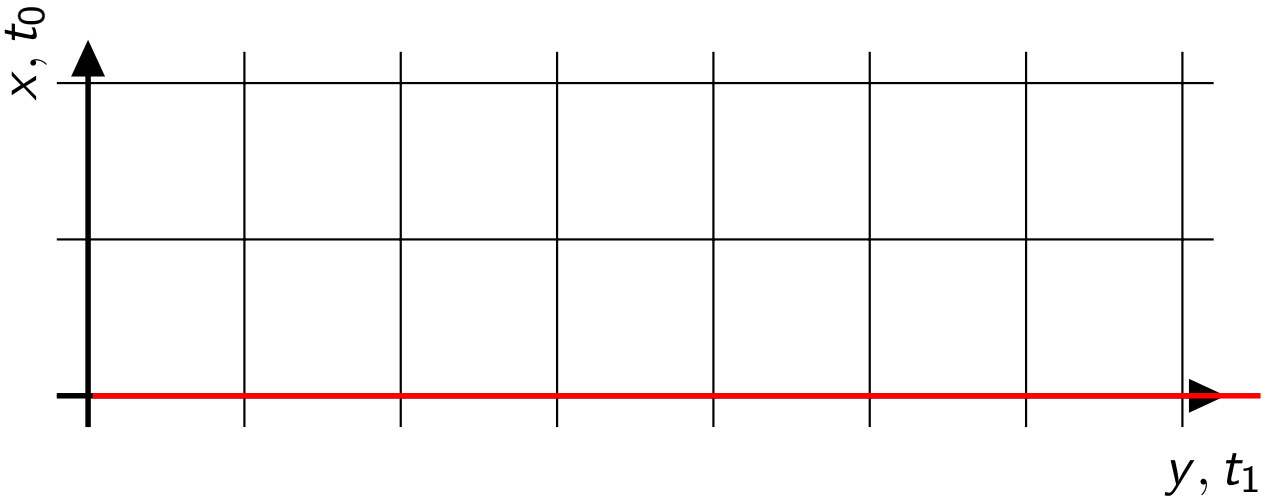


Termination in the Bounded Case

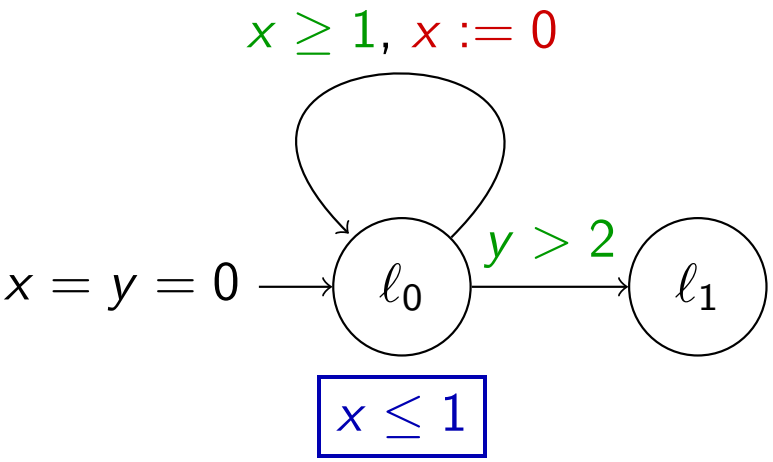


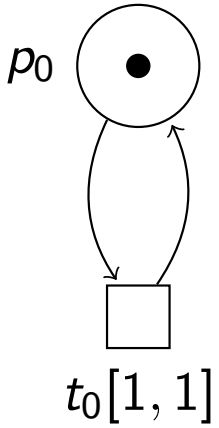


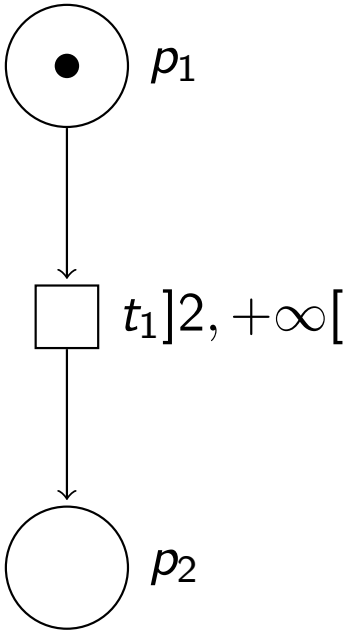


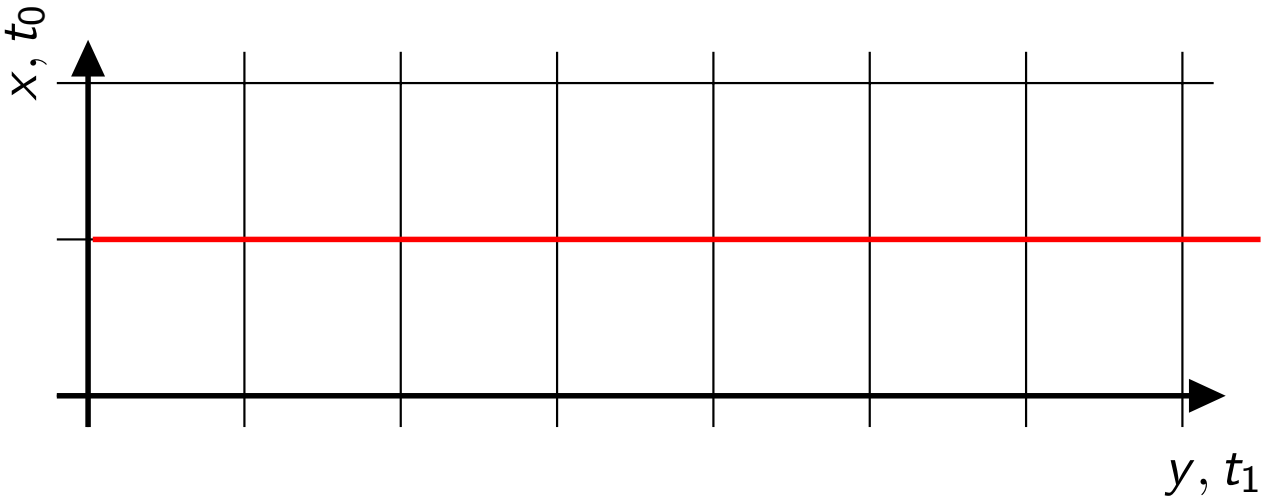


Termination in the Bounded Case

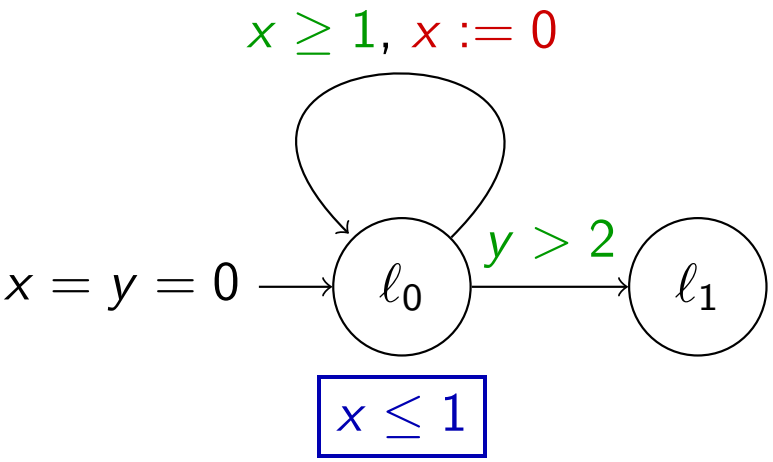


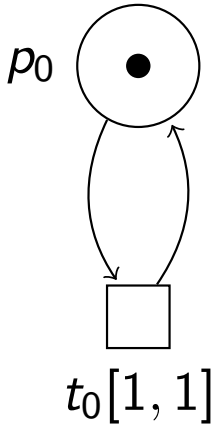


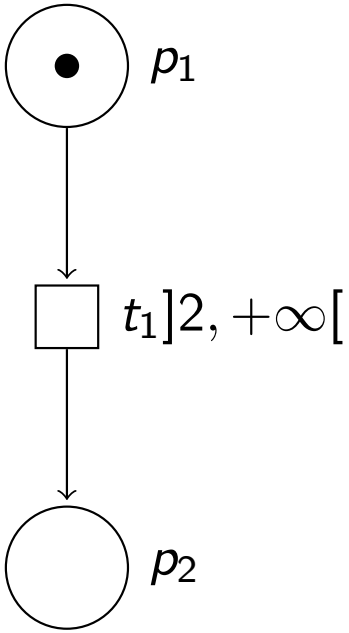


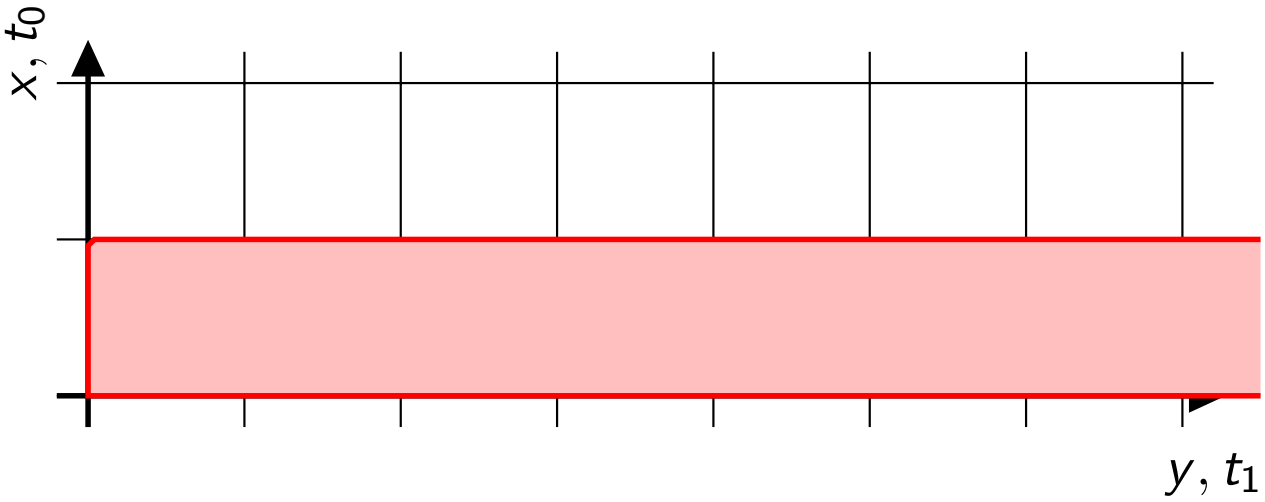


Termination in the Bounded Case









A Scheduling Problem (adapted from [BFSV04])

- Three real-time tasks τ_1, τ_2 and τ_3:
  - τ_1 is periodic with period 50 and execution time $C_1 \in [10, 20]$;
  - τ_2 is sporadic with min. delay 100 and execution time $C_2 \in [18, 28]$;
  - τ_3 is periodic with period 150 and execution time $C_3 \in [20, 28]$;
- Scheduling policy: priority with $\tau_1 > \tau_2 > \tau_3$ (non preemptive);
- Is it **schedulable**? Each task always has at most one instance running

A Scheduling Problem (adapted from [BFSV04])

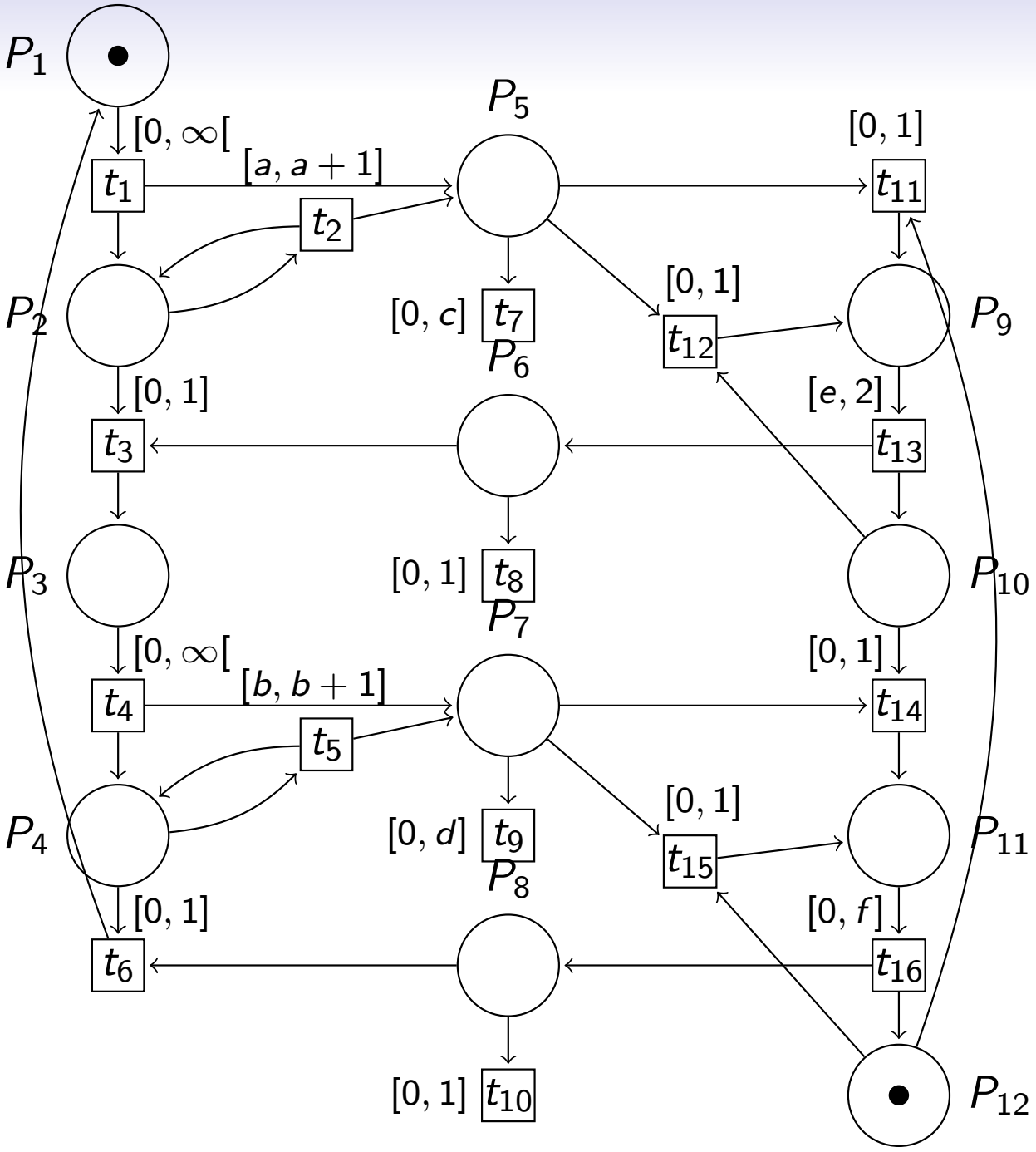
- Three real-time tasks τ_1, τ_2 and τ_3:
  - τ_1 is periodic with period 50 and execution time $C_1 \in [10, 20]$;
  - τ_2 is sporadic with min. delay 100 and execution time $C_2 \in [18, 28]$;
  - τ_3 is periodic with period 150 and execution time $C_3 \in [20, 28]$;
- Scheduling policy: priority with $\tau_1 > \tau_2 > \tau_3$ (non preemptive);
- Is it **schedulable**? Each task always has at most one instance running

A Scheduling Problem (adapted from [BFSV04])

- Three real-time tasks τ_1, τ_2 and τ_3:
  - τ_1 is periodic with period a and execution time $C_1 \in [10, 20]$;
  - τ_2 is sporadic with min. delay $2a$ and execution time $C_2 \in [18, 28]$;
  - τ_3 is periodic with period $3a$ and execution time $C_3 \in [20, 28]$;
- Scheduling policy: priority with $\tau_1 > \tau_2 > \tau_3$ (non preemptive);
- Is it **schedulable**? Each task always has at most one instance running

A Scheduling Problem (adapted from [BFSV04])

- Three real-time tasks τ_1, τ_2 and τ_3:
  - τ_1 is periodic with period a and execution time $C_1 \in [10, 20]$;
  - τ_2 is sporadic with min. delay $2a$ and execution time $C_2 \in [18, 28]$;
  - τ_3 is periodic with period $3a$ and execution time $C_3 \in [20, 28]$;
- Scheduling policy: priority with $\tau_1 > \tau_2 > \tau_3$ (non preemptive);
- Is it **schedulable**? Each task always has at most one instance running
- What is the minimum value of a such that the system is schedulable?


P_1
$[0, \infty[$
t_1
$[a, a+1]$
t_2
P_5
$[0, 1]$
t_{11}
P_2
$[0, c]$
t_7
P_6
$[0, 1]$
t_{12}
P_9
$[0, 1]$
t_3
$[e, 2]$
t_{13}
P_3
$[0, 1]$
t_8
P_7
P_{10}
$[0, \infty[$
t_4
$[b, b+1]$
t_5
$[0, 1]$
t_{14}
P_4
$[0, d]$
t_9
P_8
$[0, 1]$
t_{15}
P_{11}
$[0, 1]$
t_6
$[0, f]$
t_{16}
$[0, 1]$
t_{10}
P_{12}

For the alternating bit protocol, we have obtained the following constraint with all six parameters as non-negative integers:

$$\begin{cases} c = 0 \\ a \geq 4 \\ e \leq 2 \\ b - f \geq 3 \end{cases} \text{ or } \begin{cases} e \leq 2 \\ b - f \geq 3 \\ a \geq 5 \end{cases} \text{ or } \begin{cases} e \leq 2 \\ -b + d + f \leq -2 \\ a \geq 5 \end{cases}$$

$$\text{or } \begin{cases} e = 0 \\ a - c \geq 4 \\ b - f \geq 3 \end{cases} \text{ or } \begin{cases} e = 0 \\ a - c \geq 4 \\ -b + d + f \leq -2 \end{cases} \text{ or } \begin{cases} c = 0 \\ a \geq 4 \\ e \leq 2 \\ -b + d + f \leq -2 \end{cases}$$

	a, b	a, b, c	a, b, c, d	a, b, c, d, e	a, b, c, d, e, f
EF Time	0 s	0.2 s	0.7 s	DNF	DNF
EF Mem.	1.6 Mo	2.4 Mo	4.7 Mo	DNF	DNF
IEF Time	0 s	0.2 s	0.6 s	44.7 s	152.7 s
IEF Mem.	1.6 Mo	2.5 Mo	4.4 Mo	75.2 Mo	106.9 Mo

Table : Scaling up the number of parameters for the ABP problem

	$a \leq 10$	$a \leq 100$	$a \leq 1000$	$a \leq 10000$
IEF Time	152.8 s	152.6 s	154 s	153.2 s
Int. Hull	2.3 s (%)	2.3 s (%)	2.3 s (%)	2.3 s (%)
IEF Mem.	108.2 Mo	108.2 Mo	108.2 Mo	108.2 Mo

Table : Scaling up a's upper bound for ABP problem

Outline

Summary

Summary

- Decidable model-checking of parametrized timed systems

Summary

- Decidable model-checking of parametrized timed systems
- Control problems: extension to timed games

Conclusion and Perspectives

Conclusion and Perspectives

Bounded integer parameter synthesis:

- explore other timed models (priced timed automata, stopwatch automata,...)

Conclusion and Perspectives

Bounded integer parameter synthesis:

- explore other timed models (priced timed automata, stopwatch automata,...)

Parametric timed games:

- state class graph

Merci de votre attention.

References I

R. Alur and D. Dill.
A Theory of Timed Automata.
Theoretical Computer Science, 126(2):183–235, 1994.

Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi.
Parametric real-time reasoning.
In *ACM Symposium on Theory of Computing*, pages 592–601, 1993.

Béatrice Bérard, Franck Cassez, Serge Haddad, Didier Lime, and Olivier H. Roux.
Comparison of the expressiveness of timed automata and time Petri nets.
In Paul Pettersson and Wang Yi, editors, *3rd International Conference on Formal Modelling and Analysis of Timed Systems (FORMATS 2005)*, volume 3829 of *LNCS*, pages 211–225, Uppsala, Sweden, September 2005. Springer-Verlag.

References II

B. Berthomieu and M. Diaz.
Modeling and verification of time dependent systems using time Petri nets.
IEEE trans. on soft. eng., 17(3):259–273, 1991.

G. Bucci, A. Fedeli, L. Sassoli, and E. Vicario.
Time state space analysis of real-time preemptive systems.
IEEE Trans. on Soft. Eng., 30(2):97–111, February 2004.

Laura Bozzelli and Salvatore La Torre.
Decision problems for lower/upper bound parametric timed automata.
Formal Methods in System Design, 35(2):121–151, 2009.

Franck Cassez and Olivier H. Roux.
Structural translation from Time Petri Nets to Timed Automata – Model-Checking Time Petri Nets via Timed Automata.
The journal of Systems and Software, 79(10):1456–1468, 2006.

References III

Conrado Daws and Stavros Tripakis.
Model checking of real-time reachability properties using abstractions, 1998.

Guillaume Gardey, Olivier H. Roux, and Olivier F. Roux.
State space computation and analysis of time Petri nets.
Theory and Practice of Logic Programming (TPLP). Special Issue on Specification Analysis and Verification of Reactive Systems, 6(3):301–320, 2006.

Thomas Hune, Judi Romijn, Marielle Stoelinga, and Frits Vaandrager.
Linear parametric model checking of timed automata.
Journal of Logic and Algebraic Programming, 52-53:183–220, 2002.

References IV

A. Jovanović, S. Faucou, D. Lime, and Olivier H. Roux.
Real-time control with parametric timed reachability games.
In *11th International Workshop on Discrete Event Systems (WODES'12)*, pages 323–330, Guadalajara, Mexico, October 2012. IFAC.

A. Jovanović, D. Lime, and Olivier H. Roux.
Integer parameter synthesis for timed automata.
In *19th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2013)*, volume 7795 of *Lecture Notes in Computer Science*, pages 391–405, Rome, Italy, March 2013. Springer.

References V

A. Jovanović, D. Lime, and Olivier H. Roux.
Synthesis of bounded integer parameters for parametric timed reachability games.
In *11th International Symposium on Automated Technology for Verification and Analysis (ATVA 2013)*, volume 8172 of *Lecture Notes in Computer Science*, pages 87–101, Hanoi, Vietnam, October 2013. Springer.

Oded Maler, Amir Pnueli, and Joseph Sifakis.
On the synthesis of discrete controllers for timed systems.
In *in E.W. Mayr and C. Puech (Eds), Proc. STACS'95, LNCS 900*, pages 229–242. Springer, 1995.

Louis-Marie Traonouez, Didier Lime, and Olivier (H.) Roux.
Parametric model-checking of stopwatch Petri nets.
Journal of Universal Computer Science, 15(17):3273–3304, 2009.