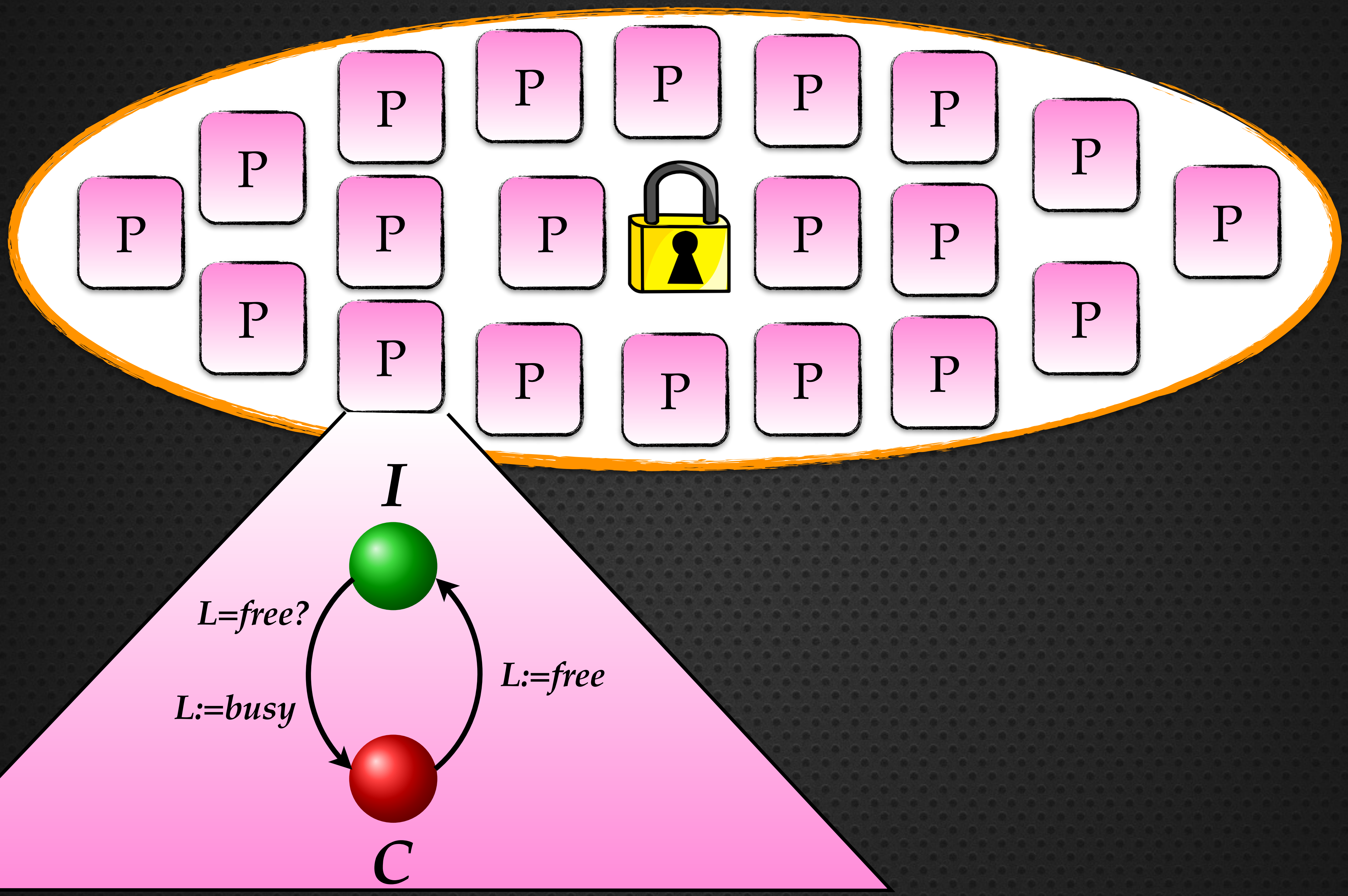
- Background
- Parameterized Systems
- Petri Nets
- Lossy Channel Systems
- Timed Petri Nets

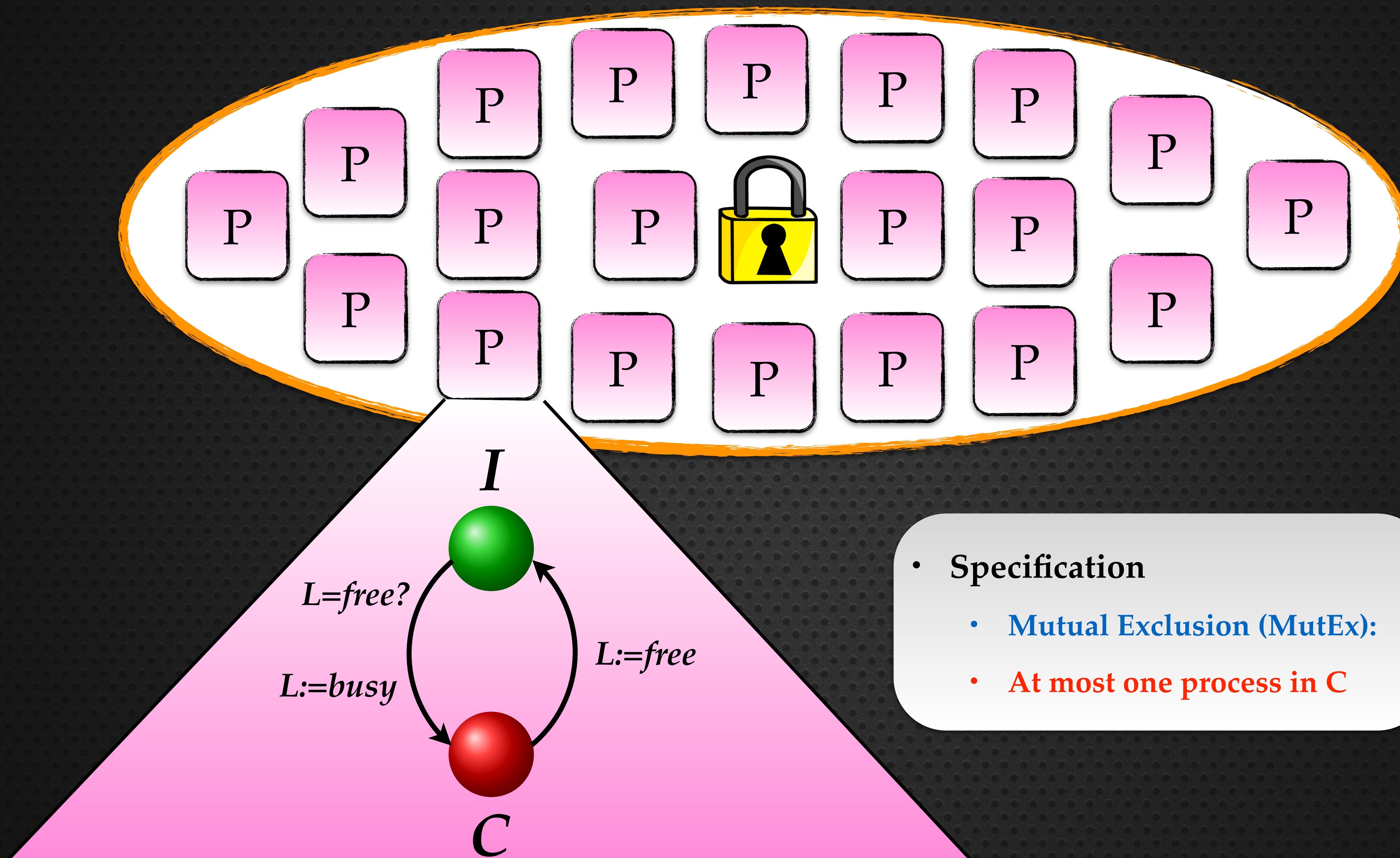
Parameterized Systems

Parameterized Systems

Parameterized Systems



Parameterized Systems



- **Specification**
  - **Mutual Exclusion (MutEx):**
  - **At most one process in C**

Parameterized Systems

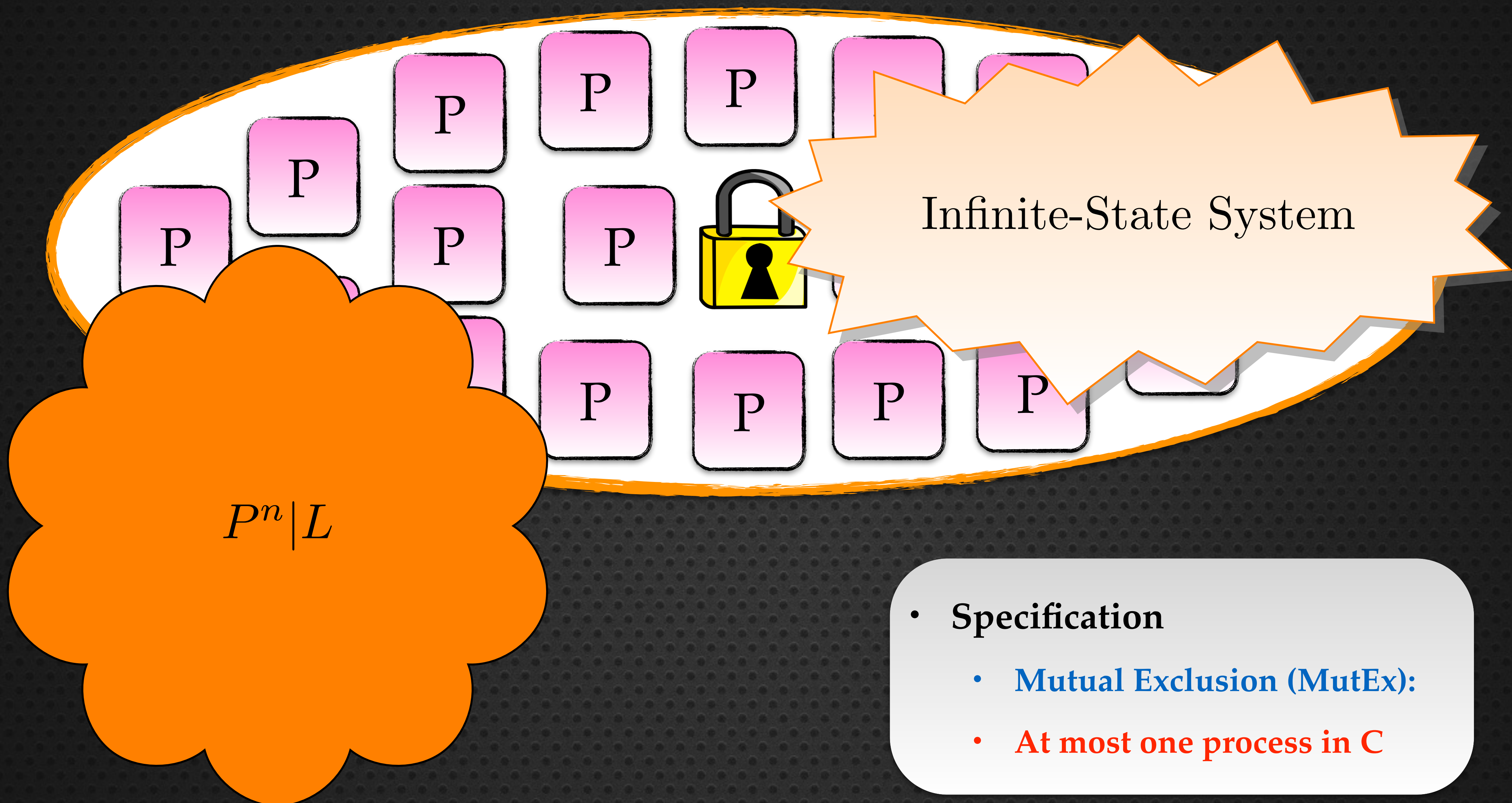
- **Specification**
  - **Mutual Exclusion (MutEx):**
  - **At most one process in C**

Parameterized Systems

- **Specification**
  - **Mutual Exclusion (MutEx):**
  - **At most one process in C**

- **Task = Parameterized Verification**
  - **Verify correctness regardless of the number of processes**
  - $\forall n. (P^n \mid L) \models \mathit{MutEx}$

Parameterized Systems



- **Specification**
  - **Mutual Exclusion (MutEx):**
  - **At most one process in C**

- **Task = Parameterized Verification**
  - **Verify correctness regardless of the number of processes**
  - $\forall n. (P^n \mid L) \models \mathit{MutEx}$

Background

Parameterized Systems

Petri Nets

Lossy Channel Systems

Timed Petri Nets

Petri Nets

Petri Nets

Model

Configurations

Transitions

Ordering

Monotonicity

Upward Closed Sets

Computing Predecessors

Backward Reachability

Petri Nets

Model

Configurations

Transitions

Ordering

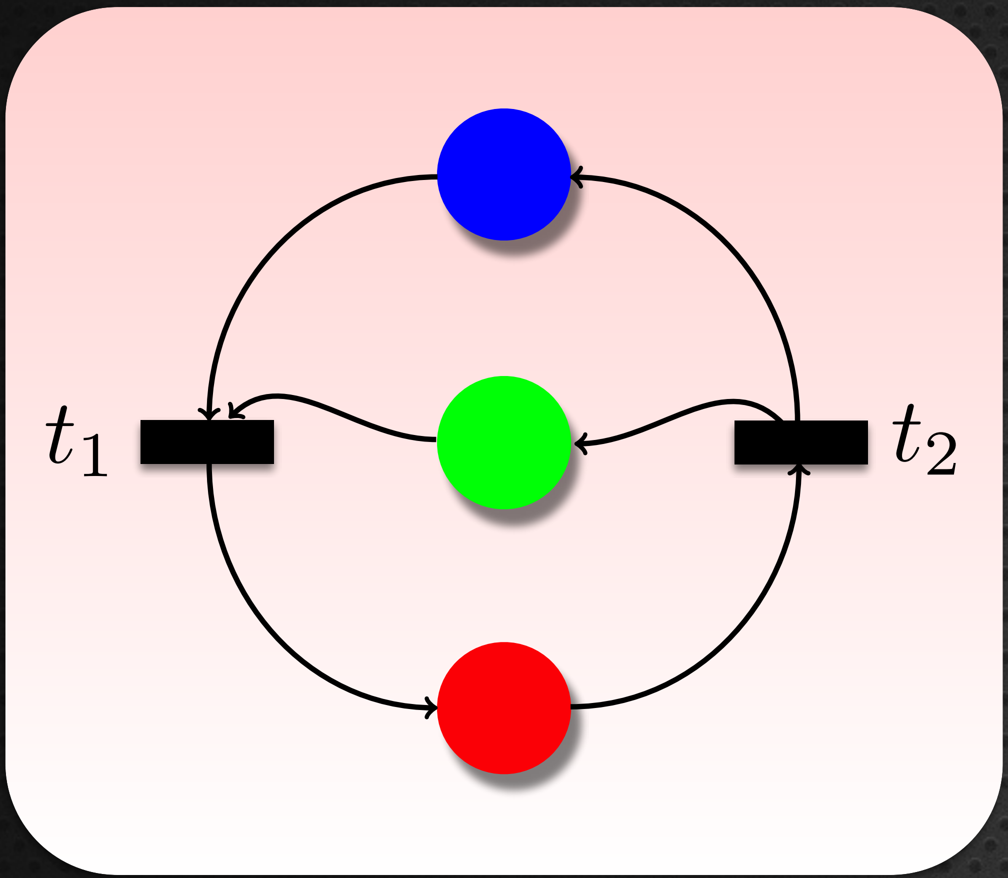
Monotonicity

Upward Closed Sets

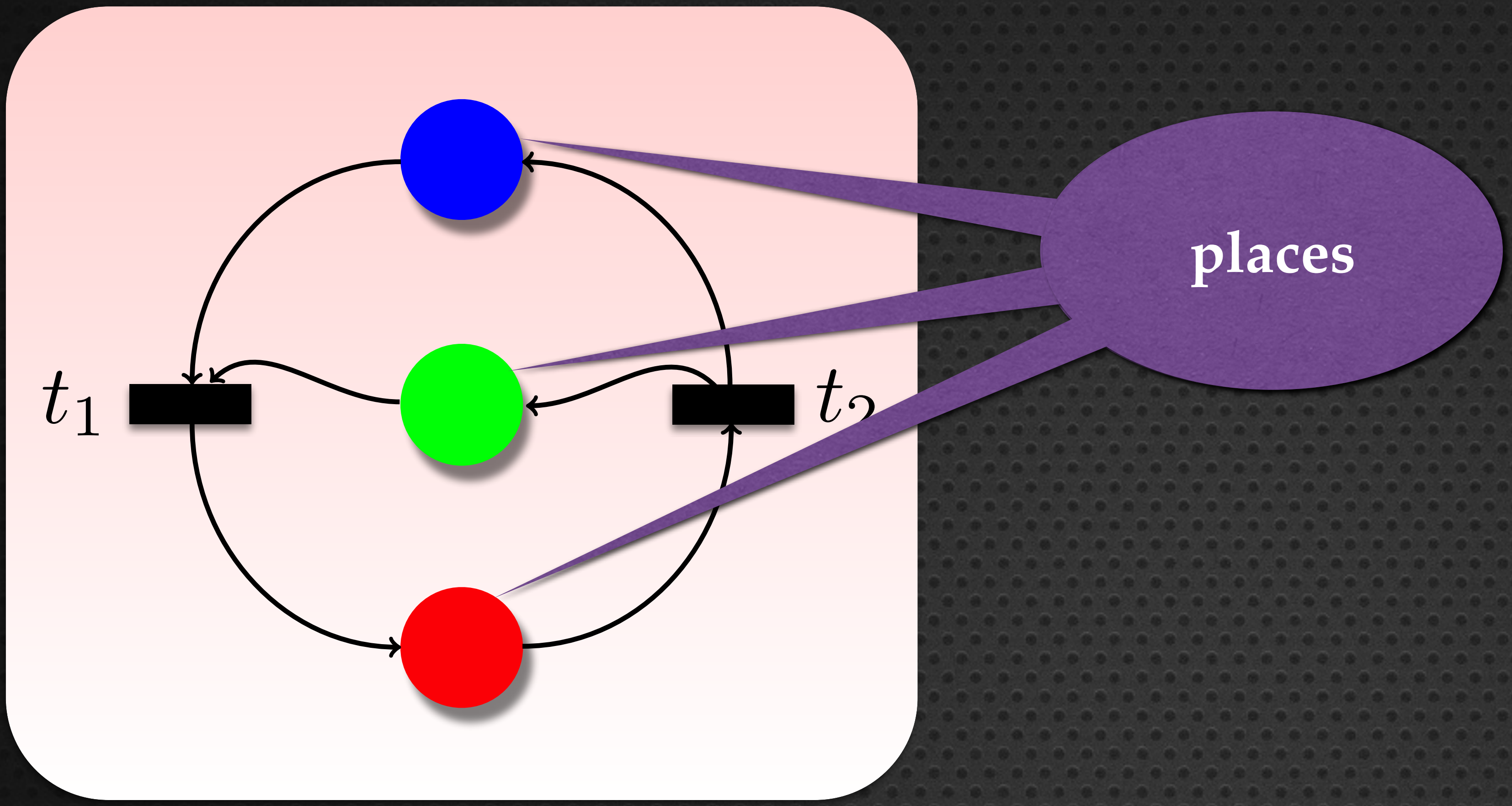
Computing Predecessors

Backward Reachability

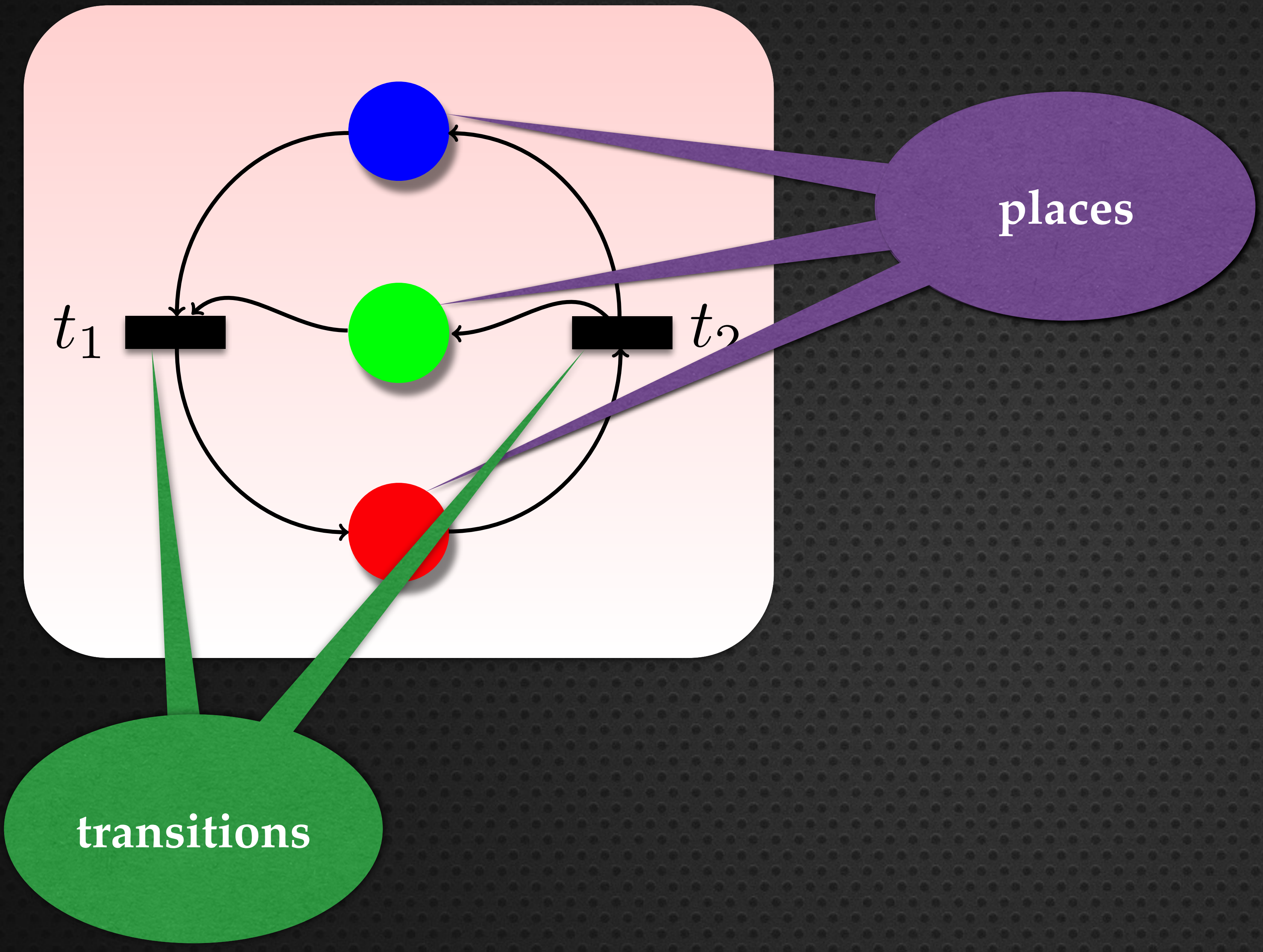
Petri Nets



Petri Nets



Petri Nets



Petri Nets

Model

Configurations

Transitions

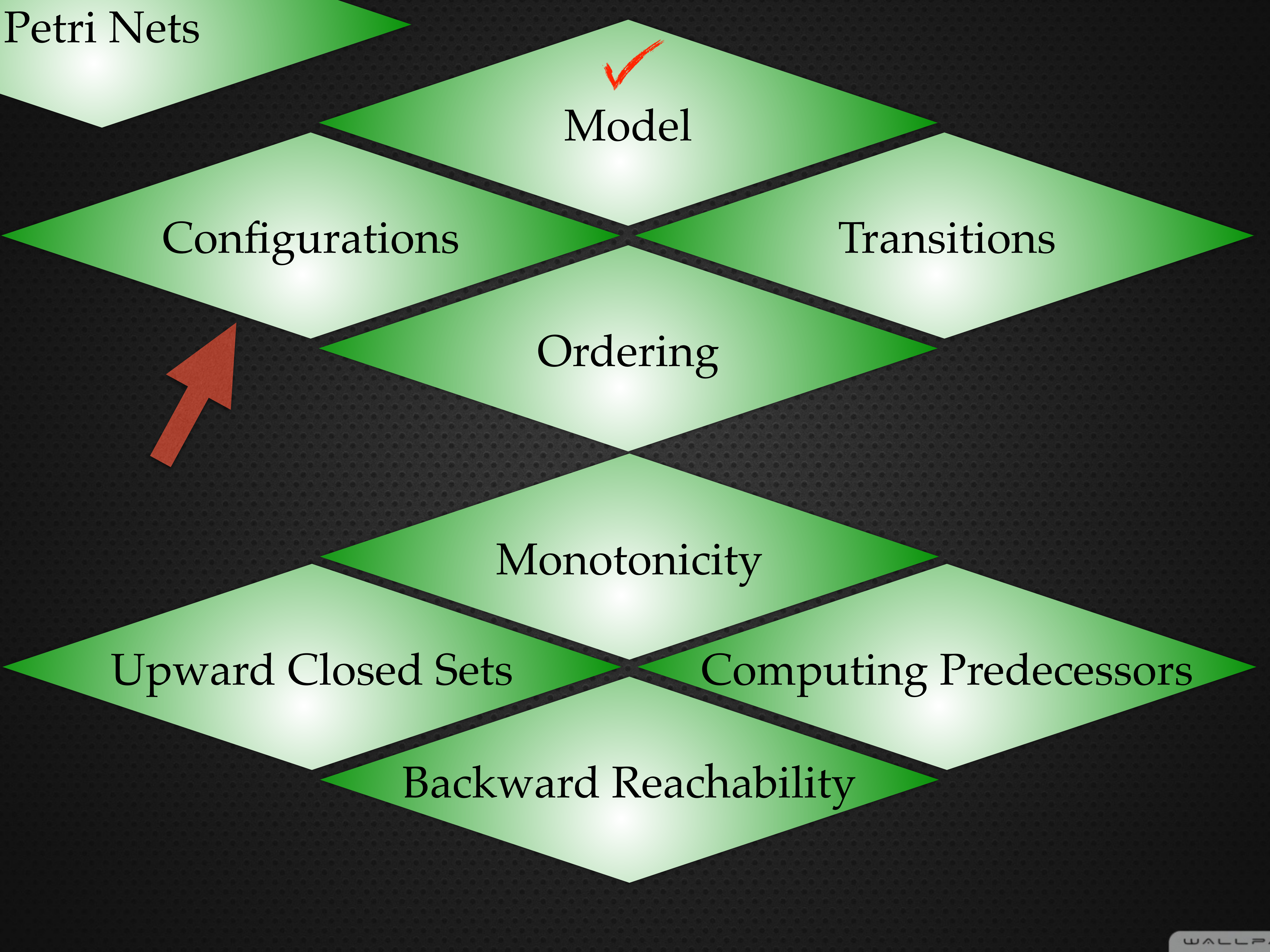
Ordering

Monotonicity

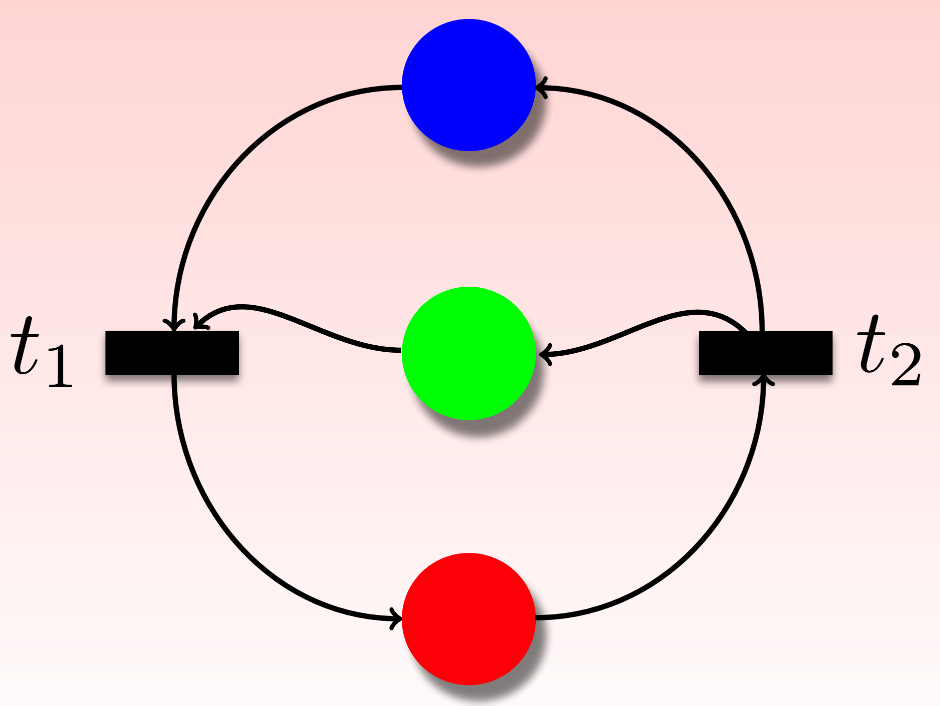
Upward Closed Sets

Computing Predecessors

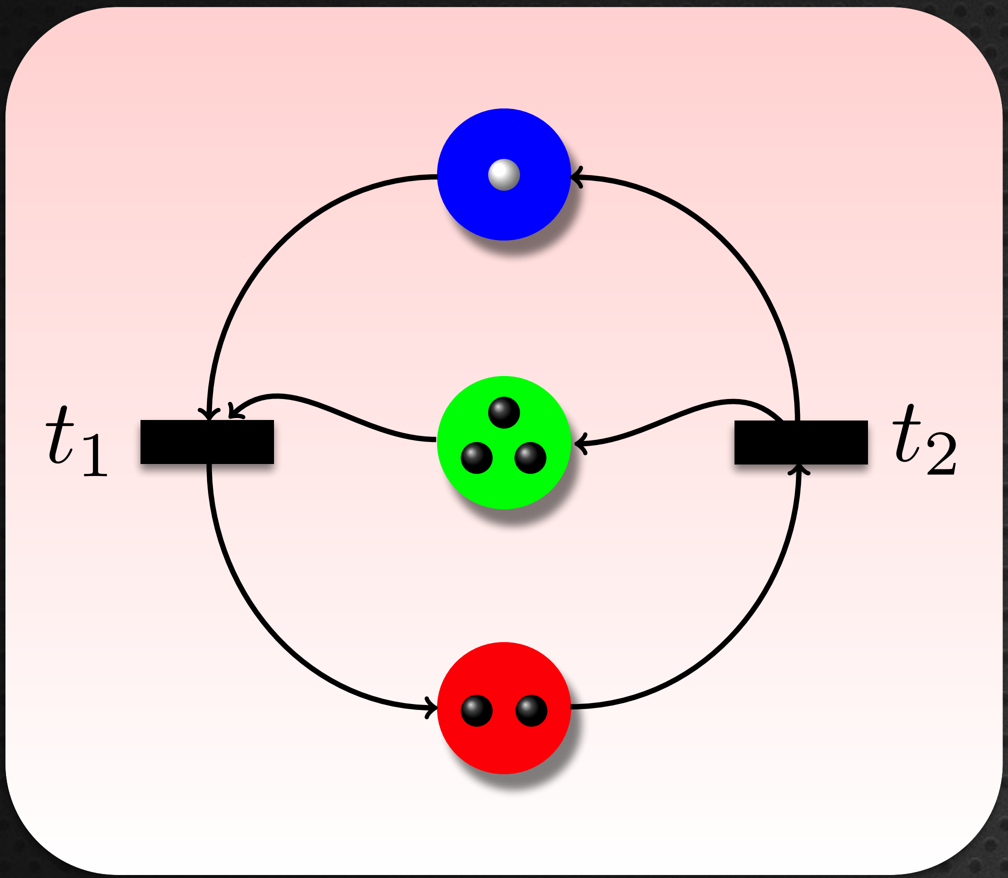
Backward Reachability


Petri Nets
Model
Configurations
Transitions
Ordering
Monotonicity
Upward Closed Sets
Computing Predecessors
Backward Reachability

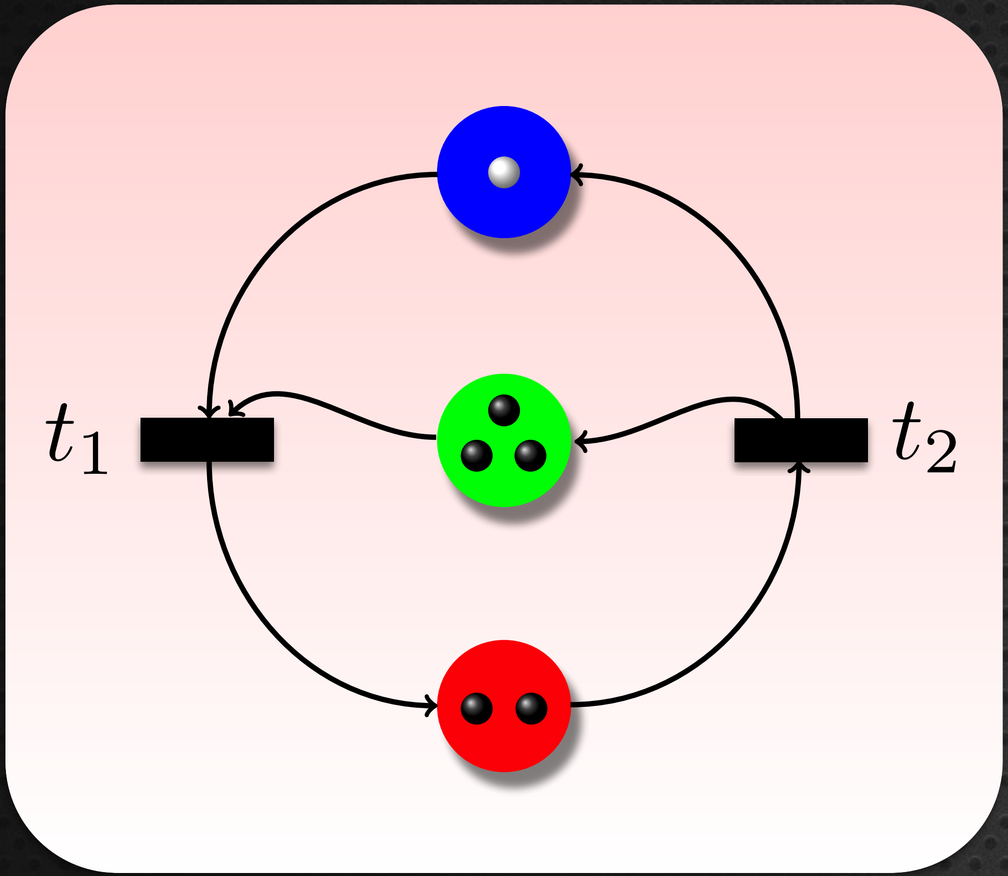
Markings

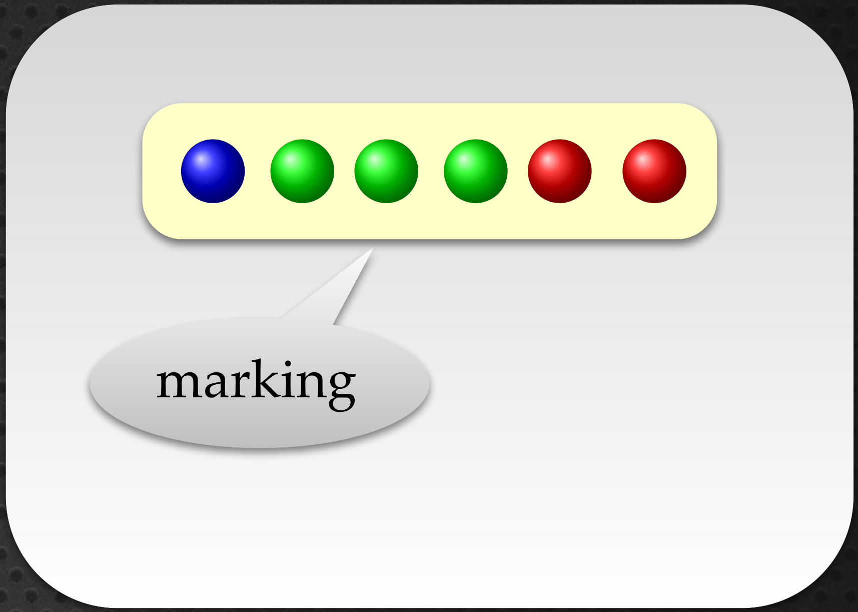


Markings

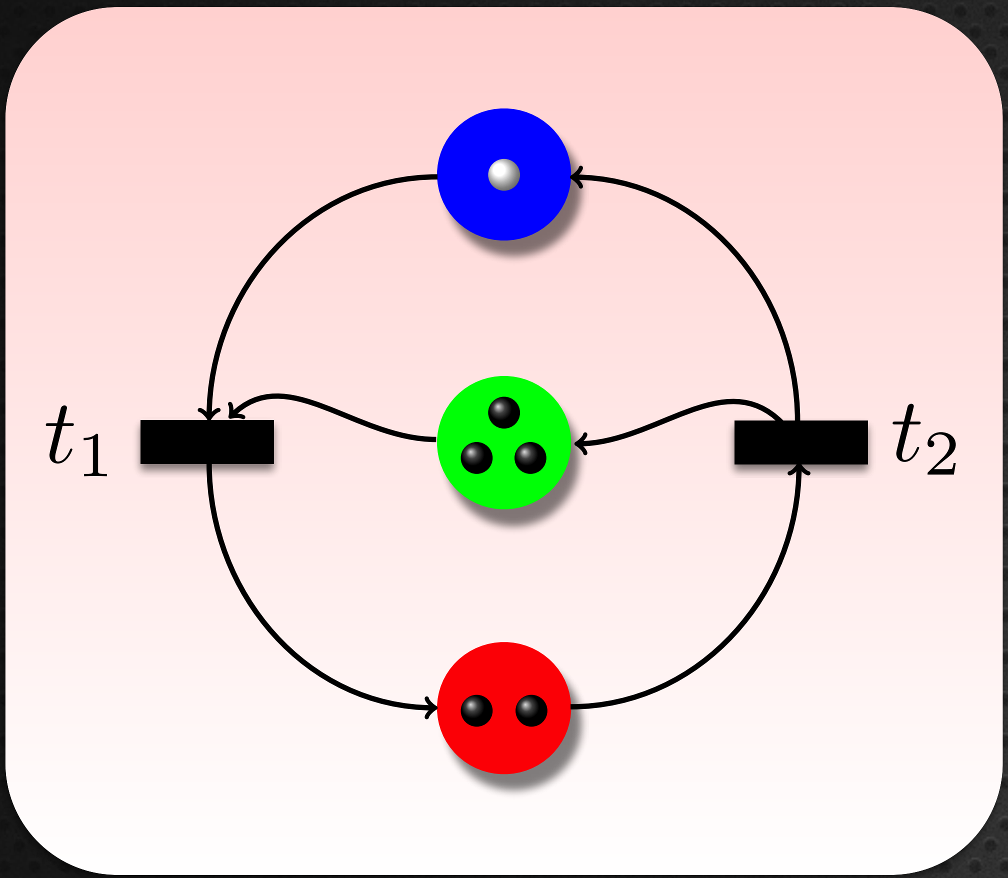


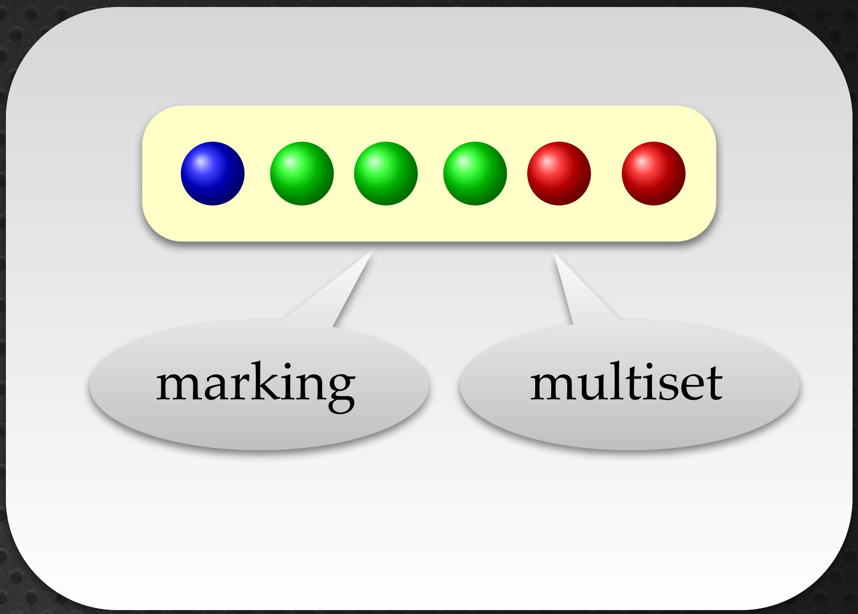
Markings



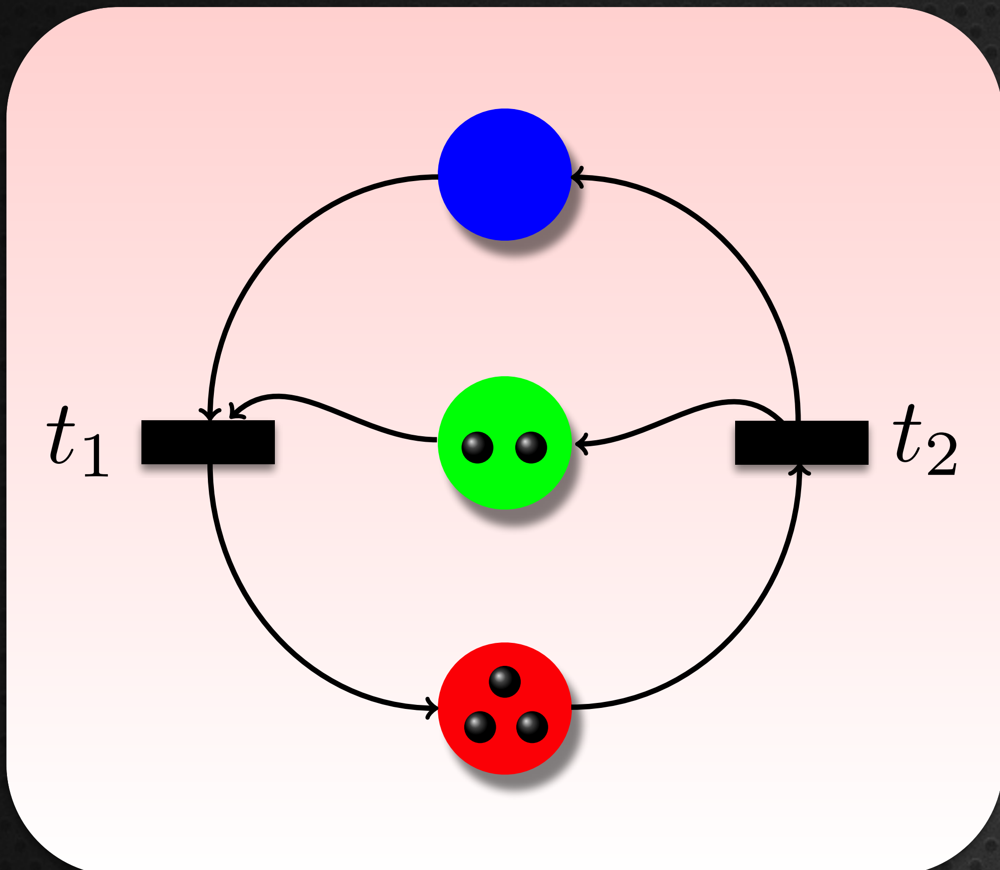


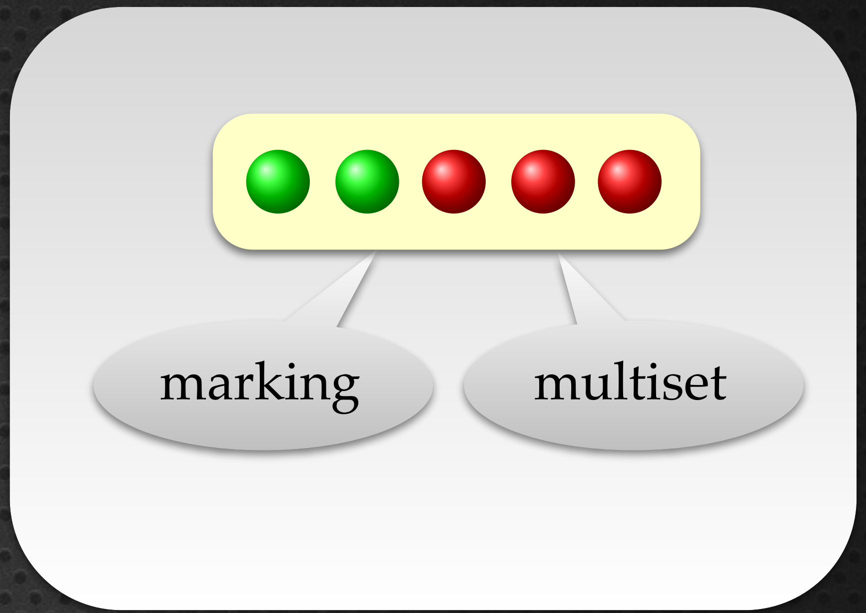
Markings

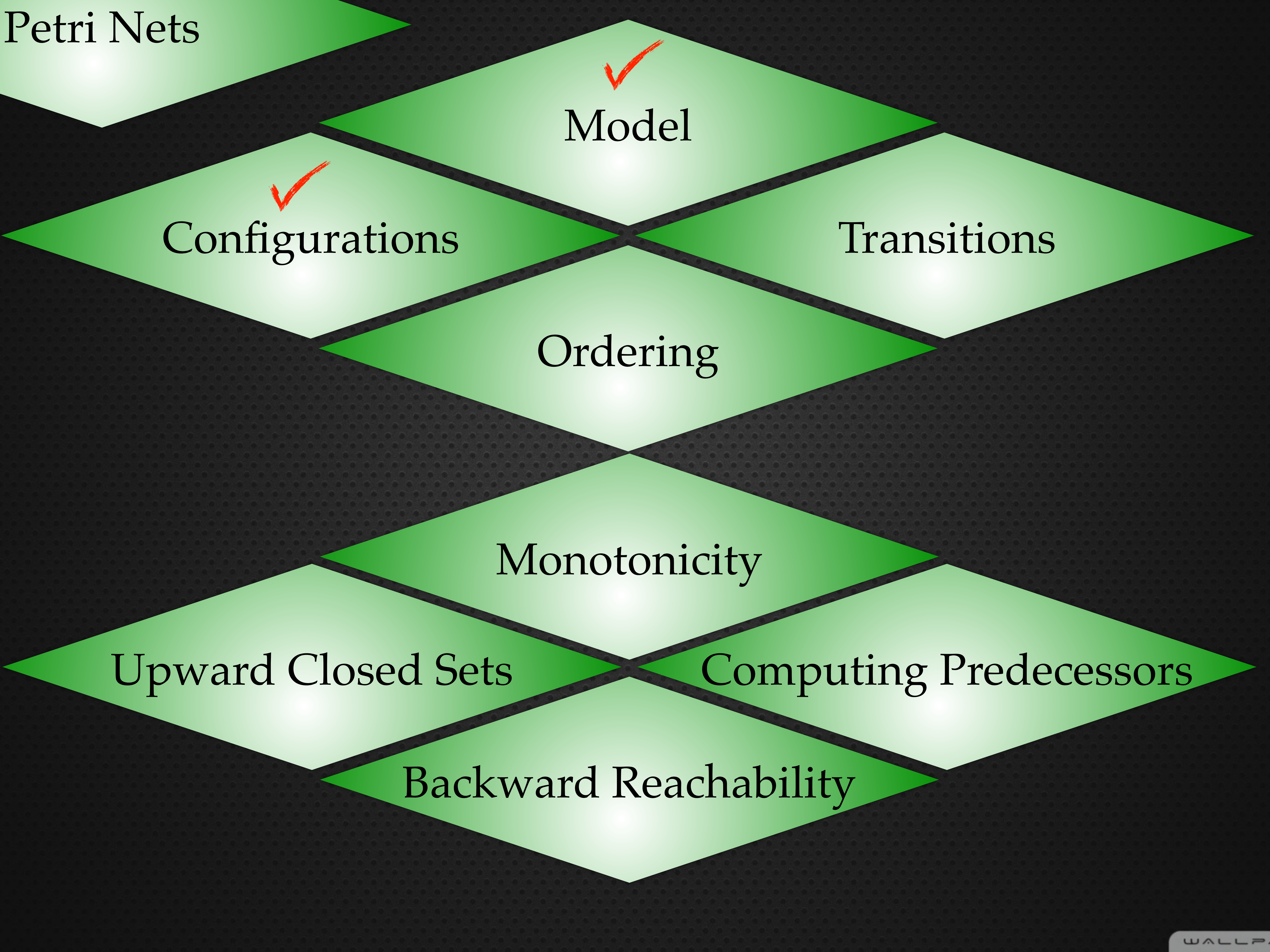




Markings







Petri Nets
Model
Configurations
Transitions
Ordering
Monotonicity
Upward Closed Sets
Computing Predecessors
Backward Reachability

Petri Nets

Model

Configurations

Transitions

Ordering

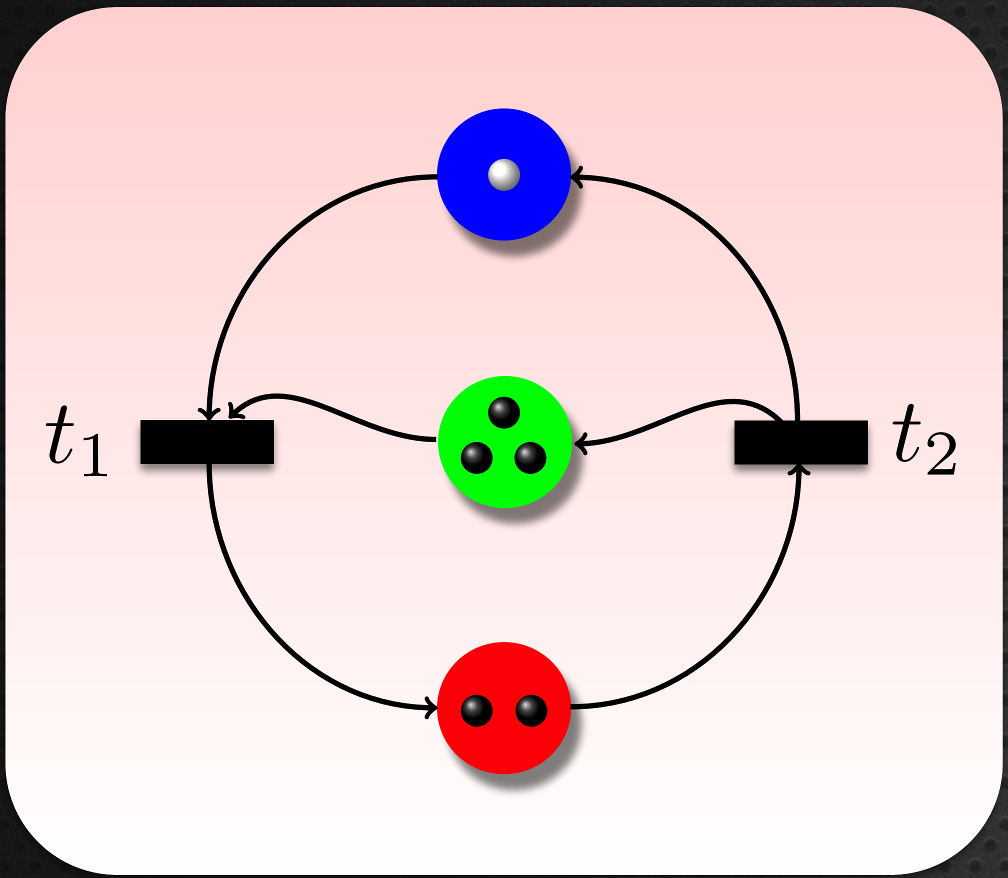
Monotonicity

Upward Closed Sets

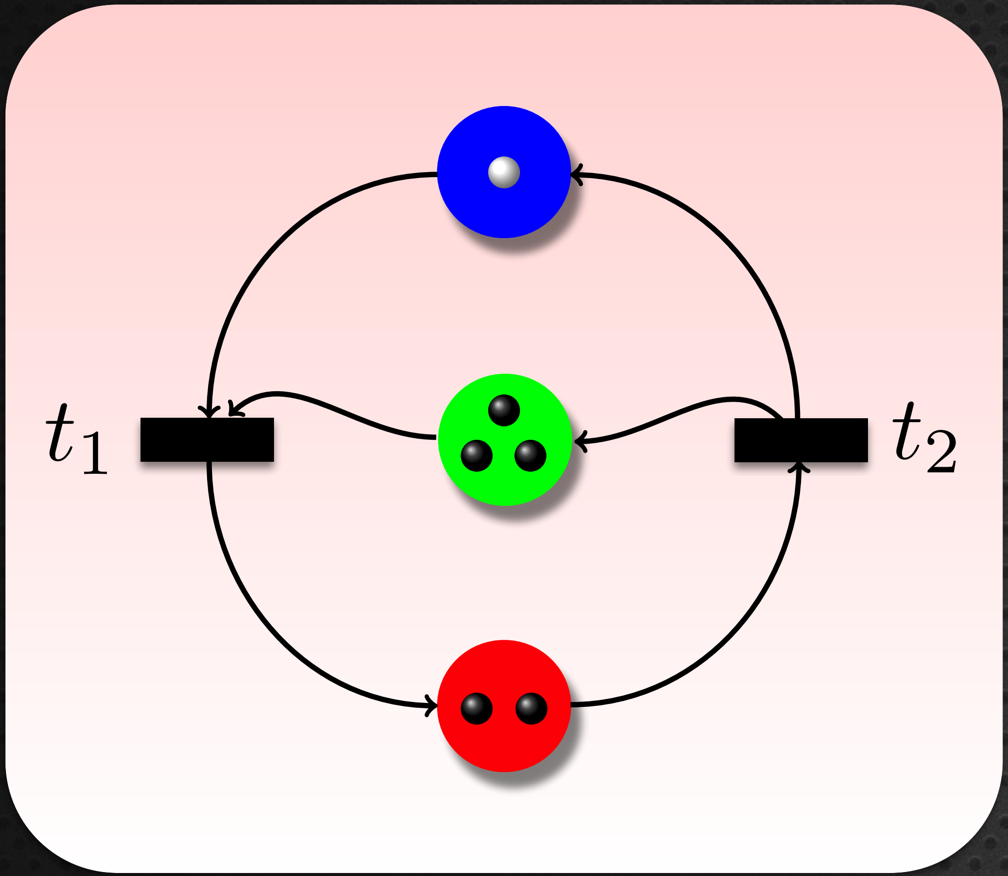
Computing Predecessors

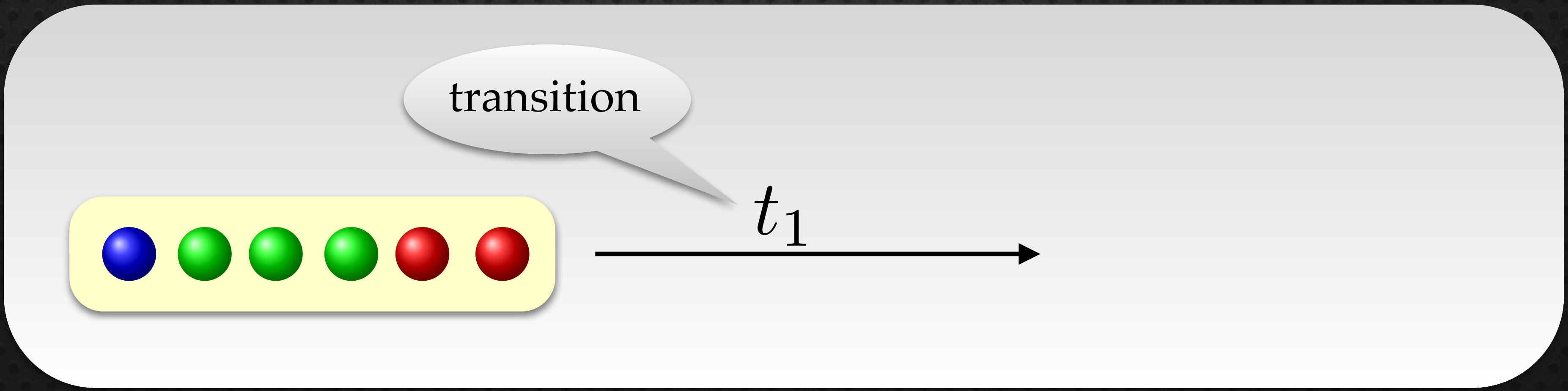
Backward Reachability

Transitions

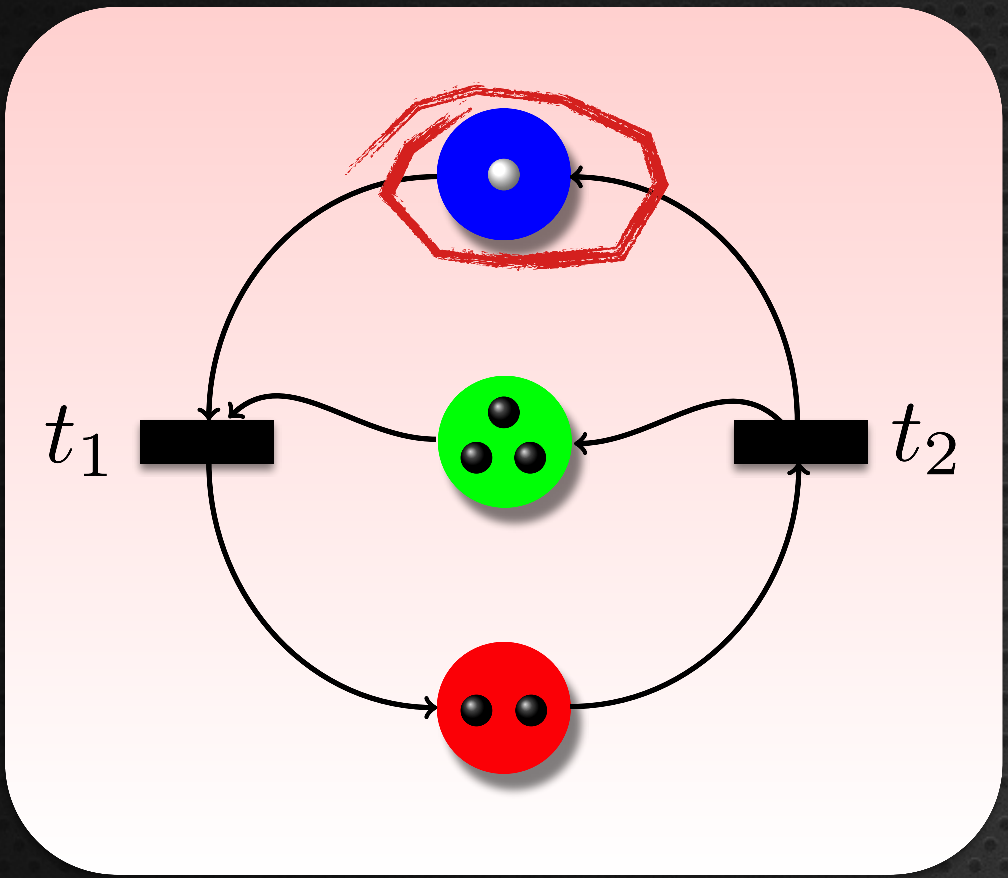


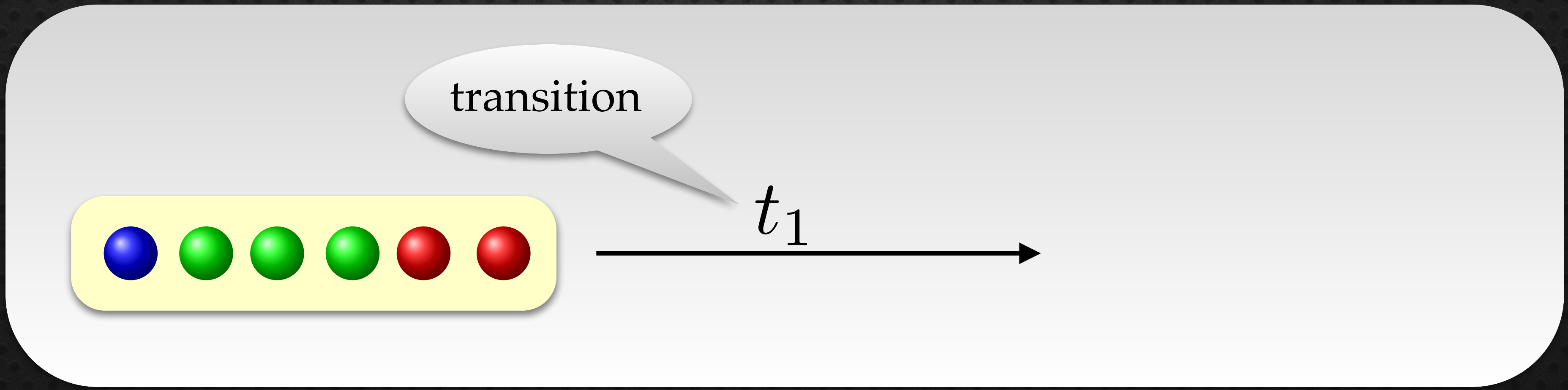
Transitions



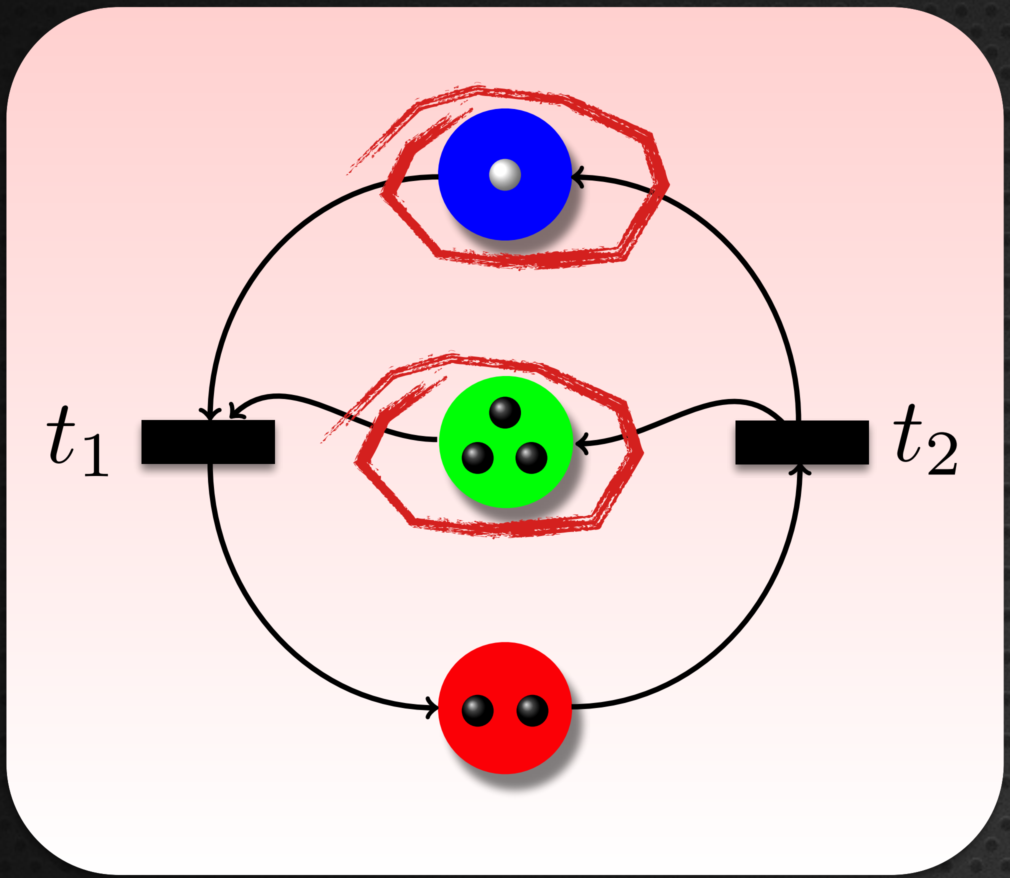


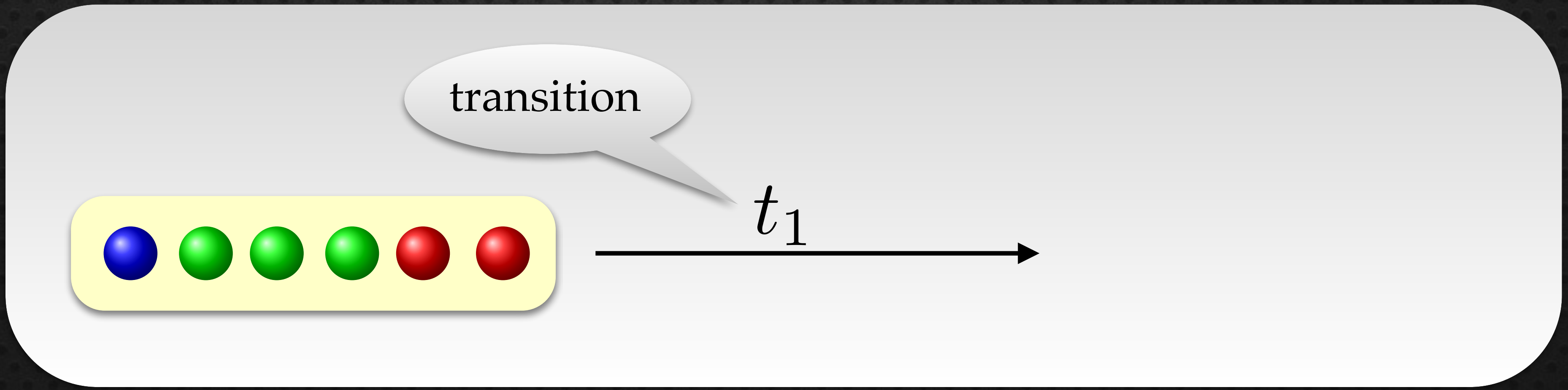
Transitions



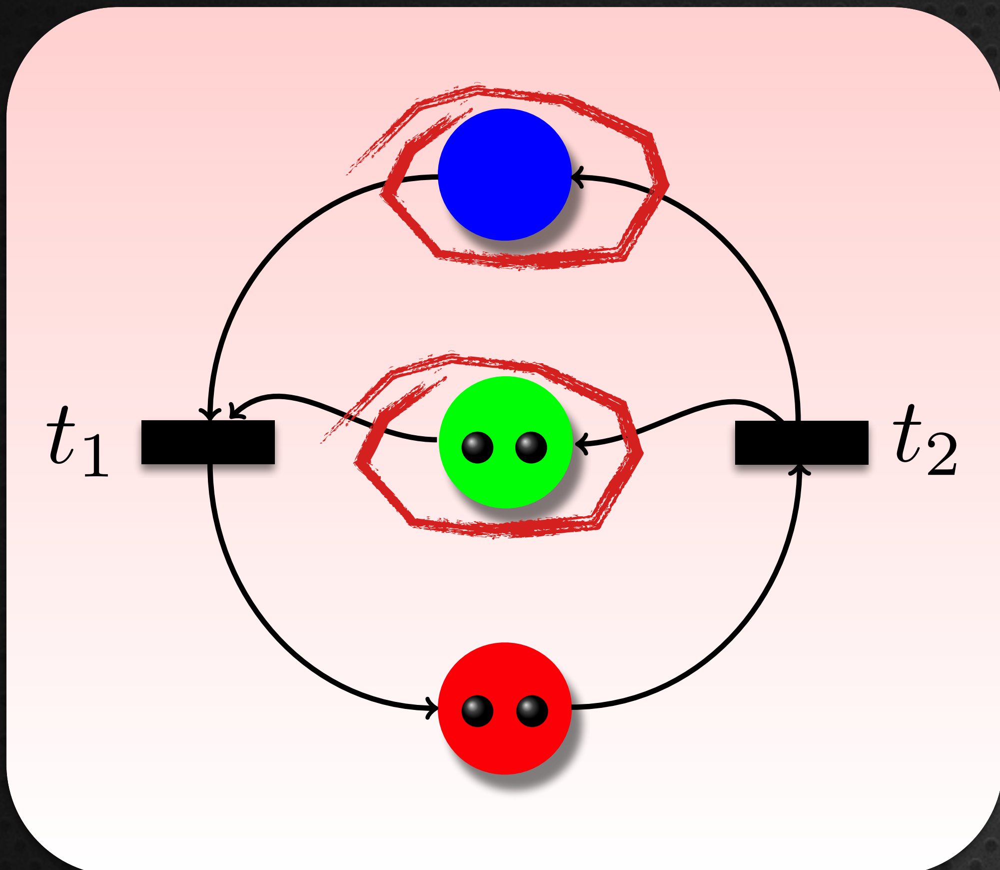


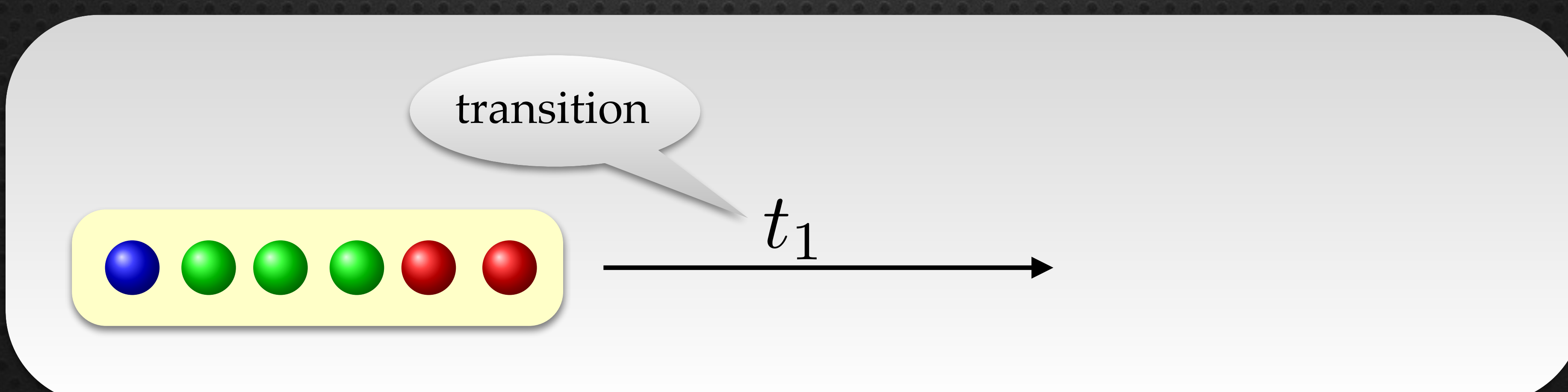
Transitions



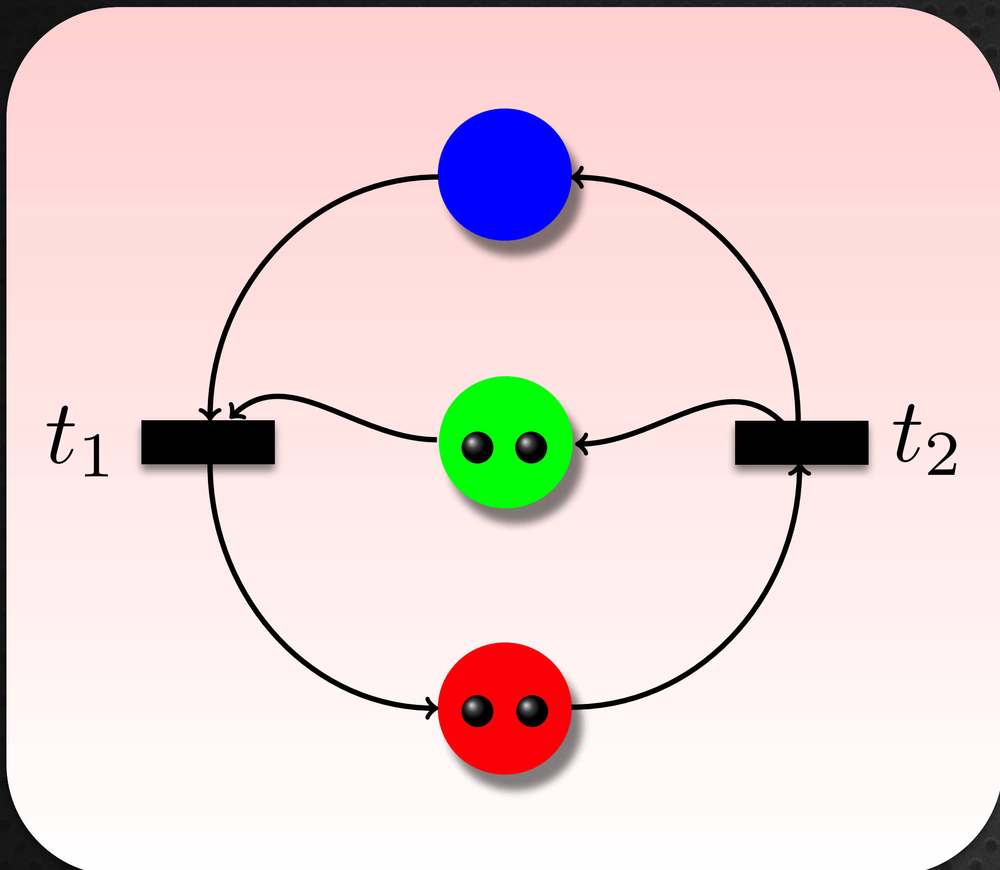


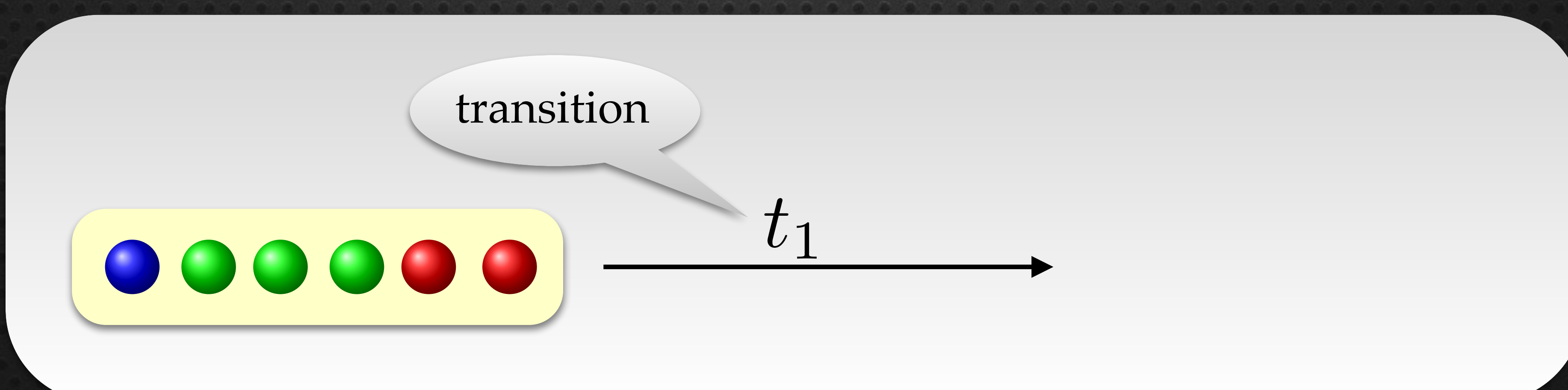
Transitions



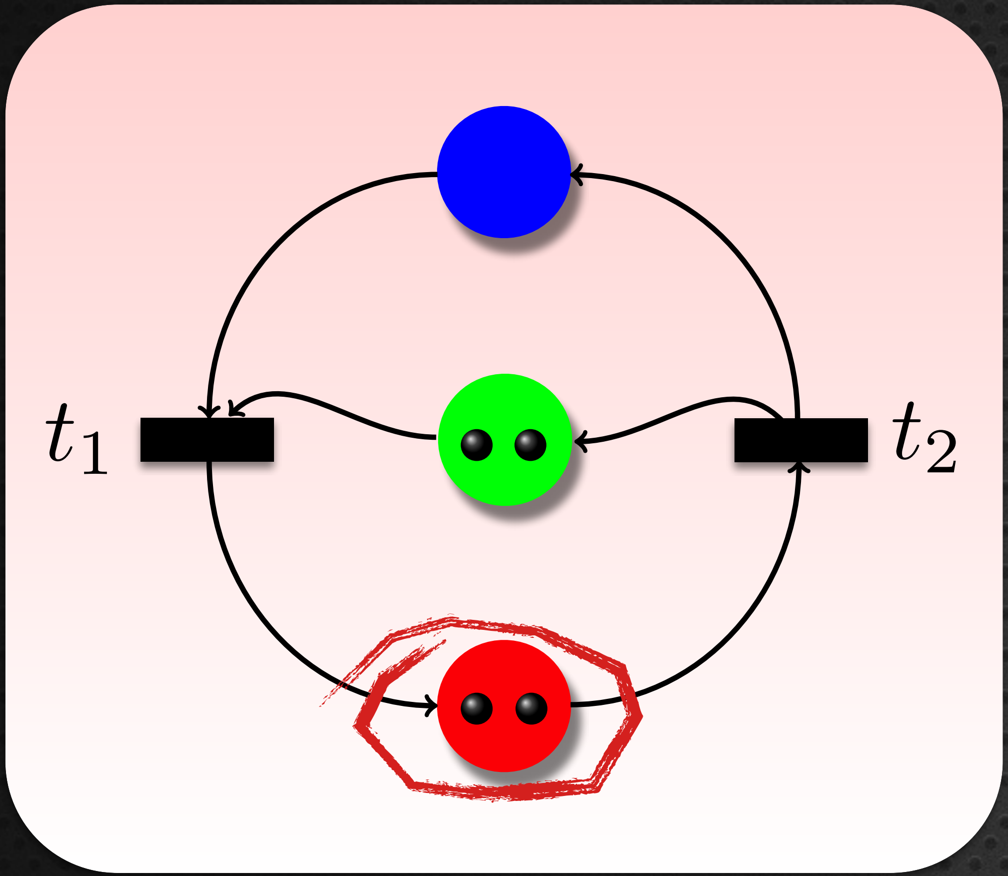


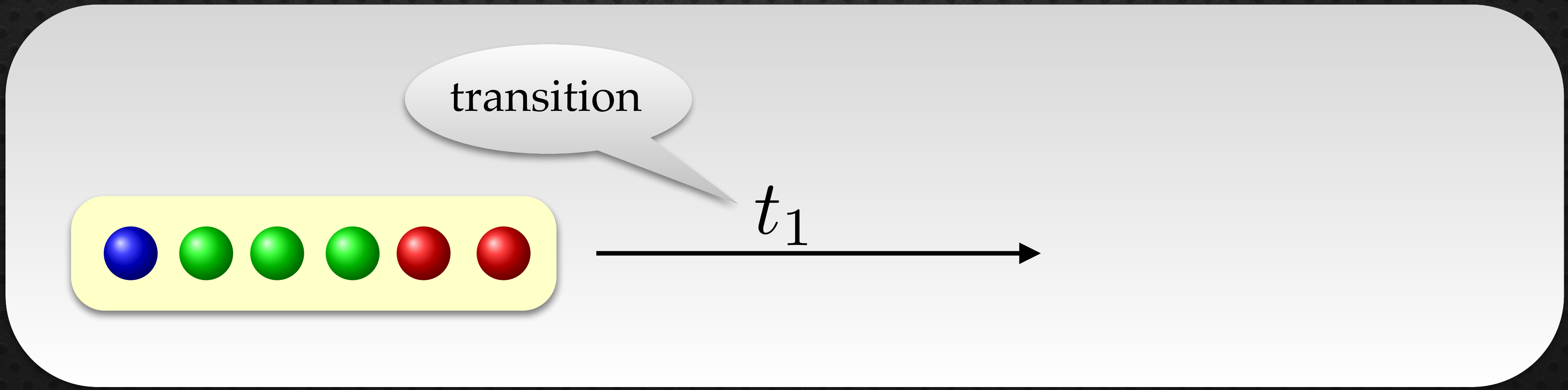
Transitions



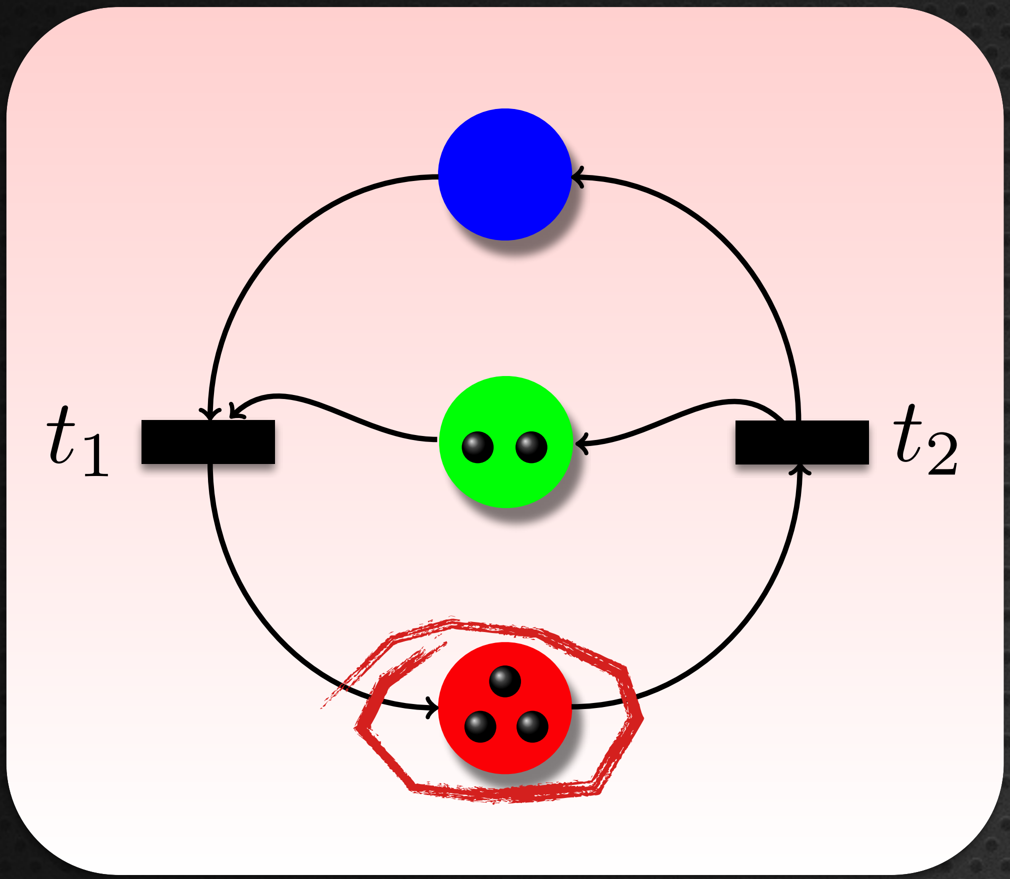


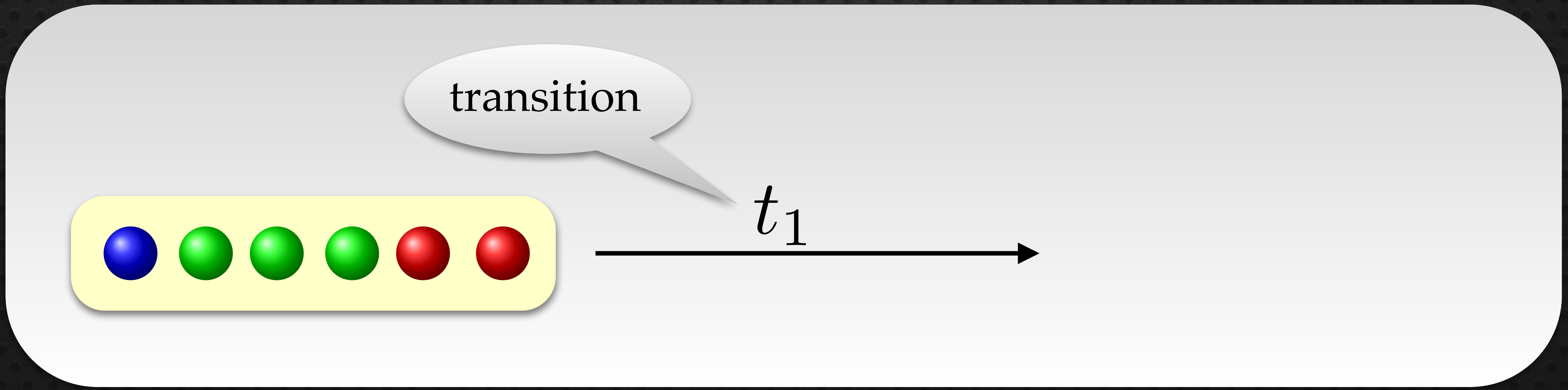
Transitions



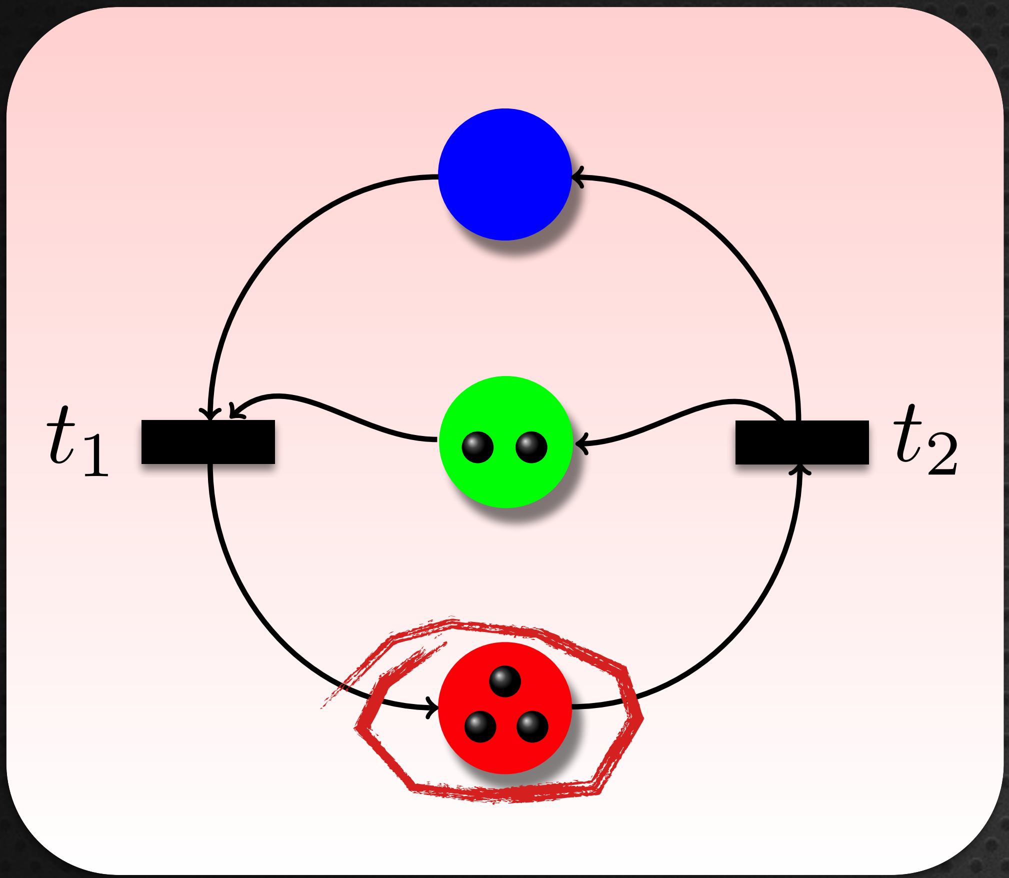


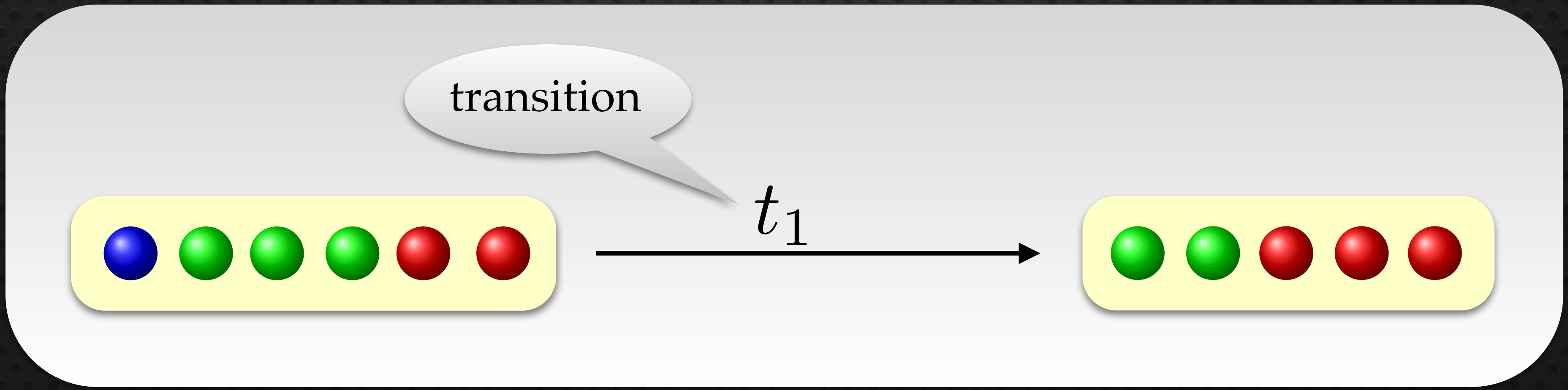
Transitions



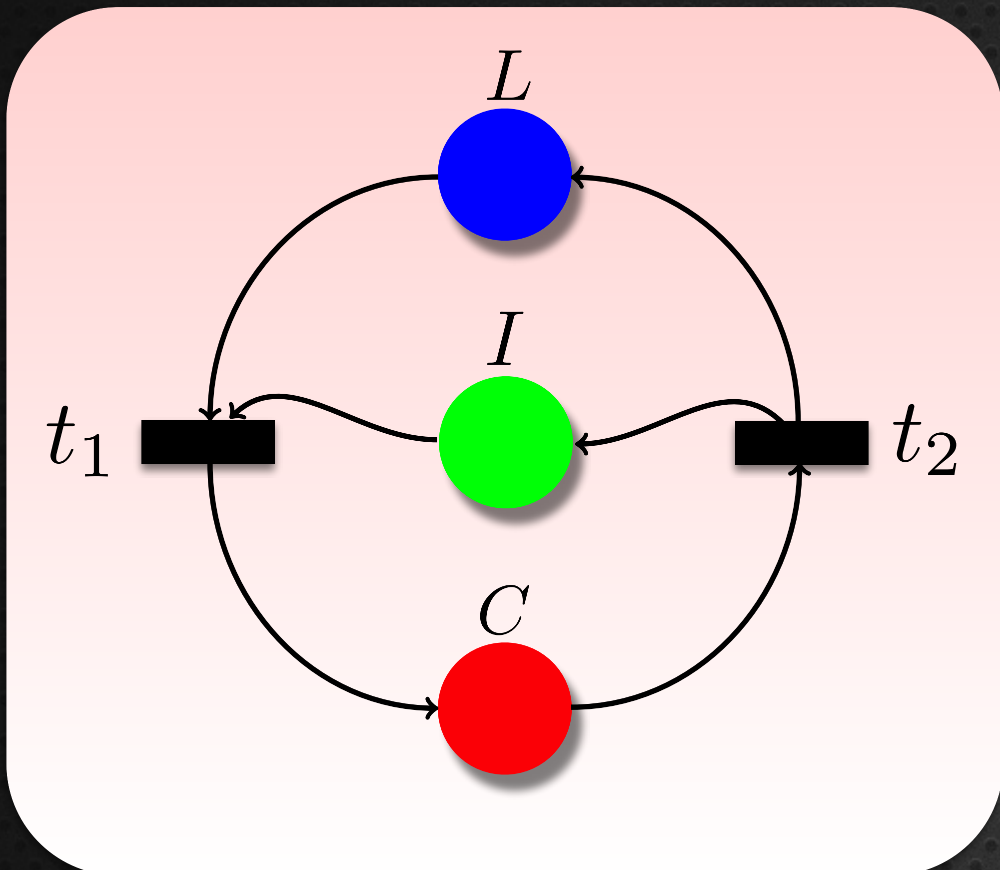


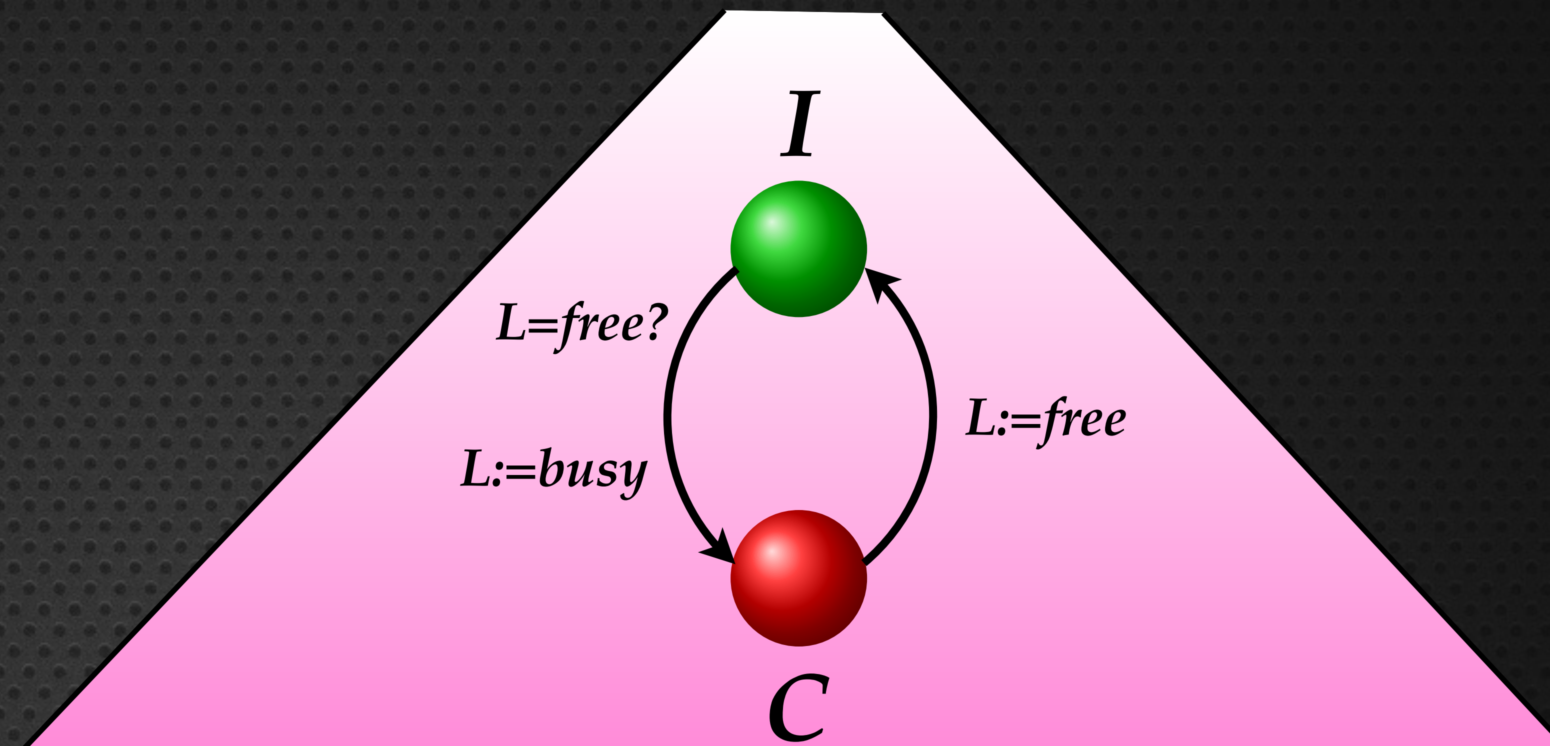
Transitions





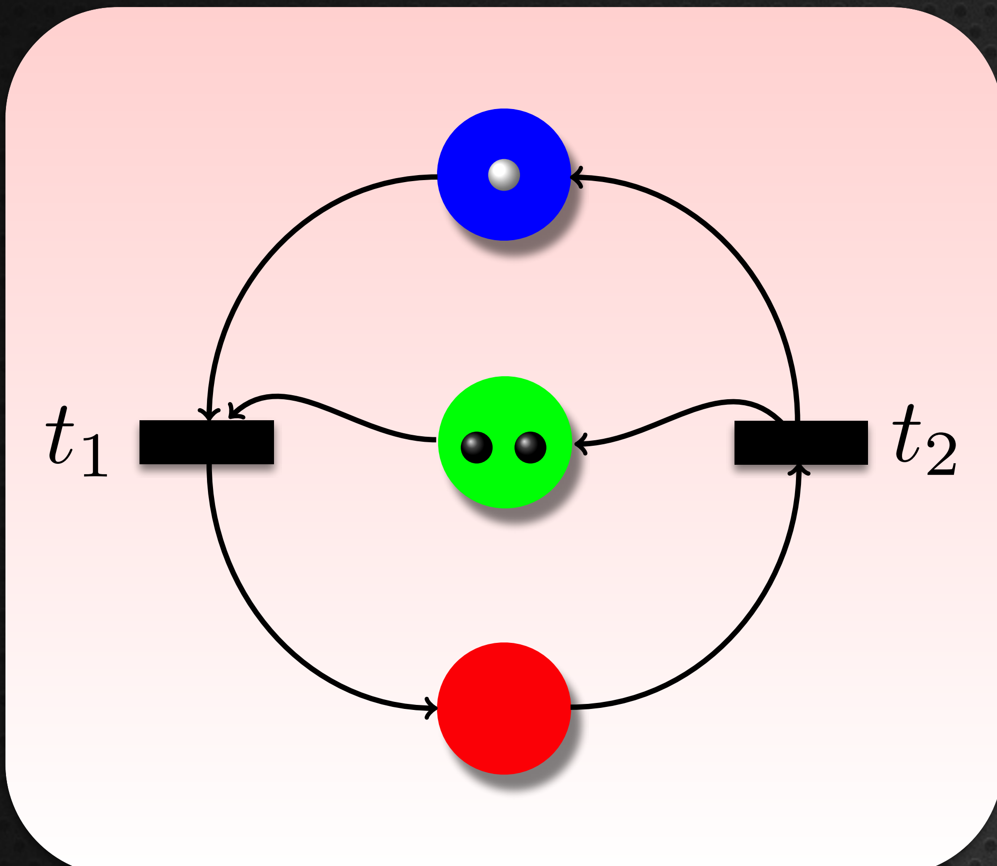
Modeling

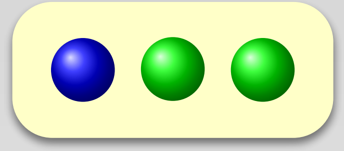




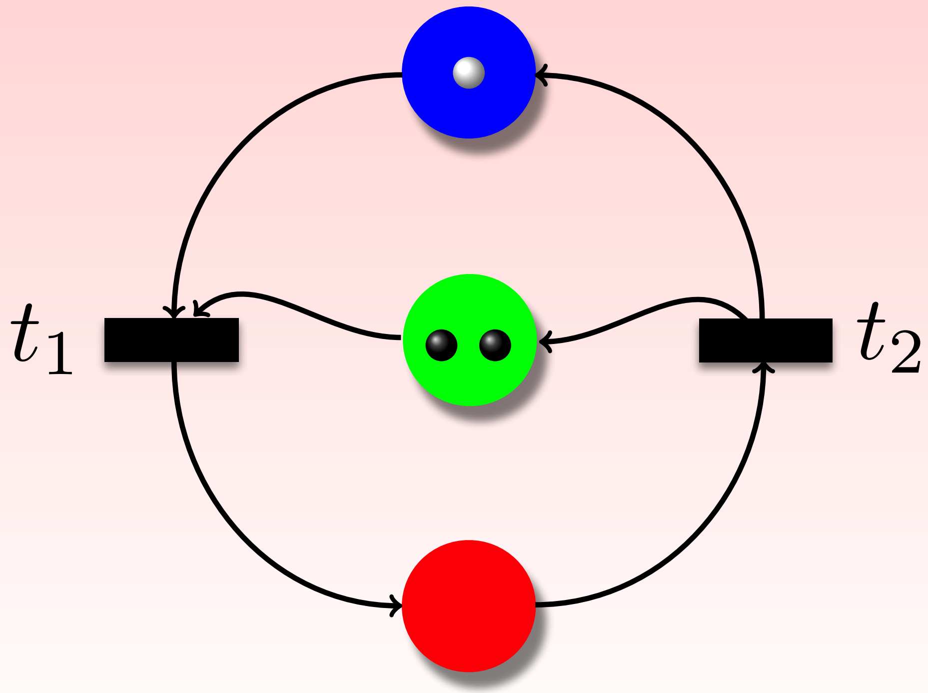
- **Encoding (counter abstraction)**
  - # tokens in = # processes in
  - # tokens in = # processes in
  - one/no token in = lock free/busy

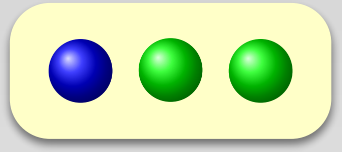
Transitions



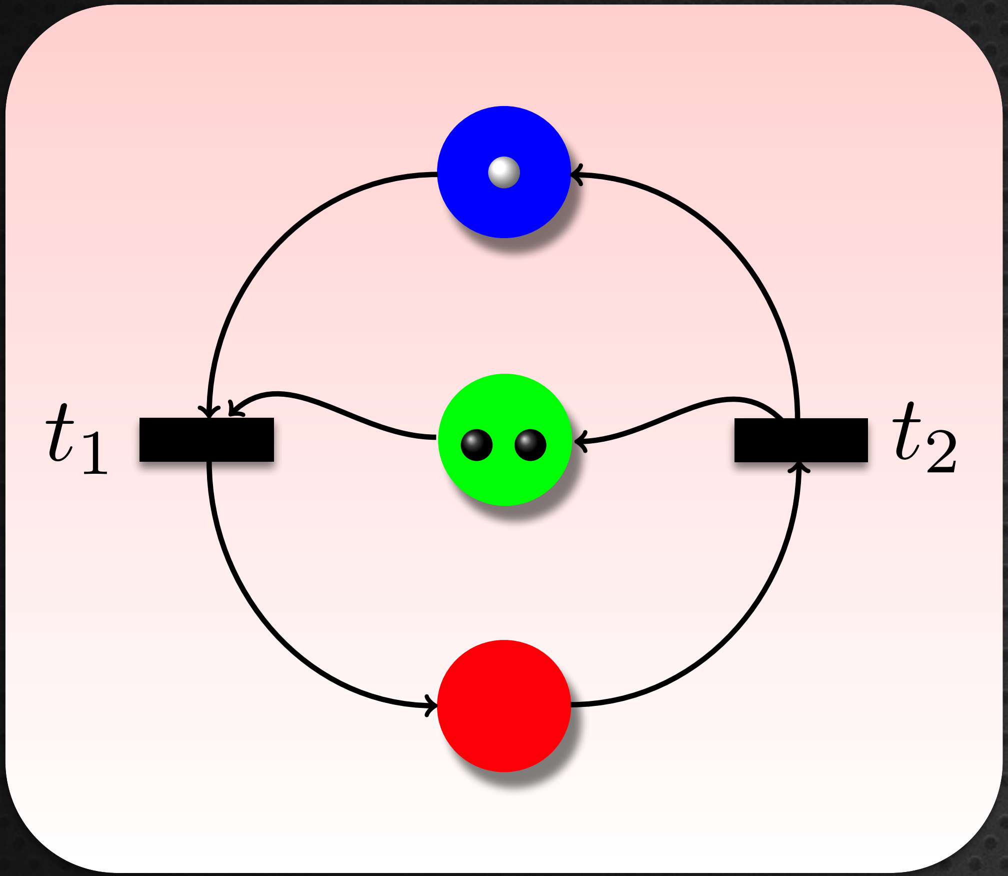


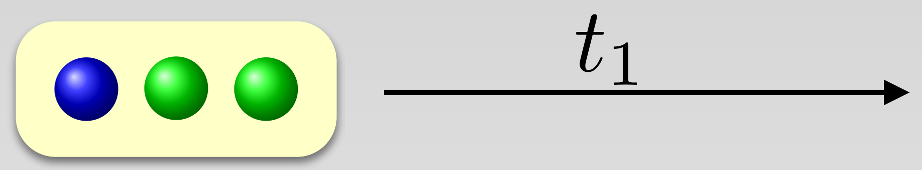
Transitions



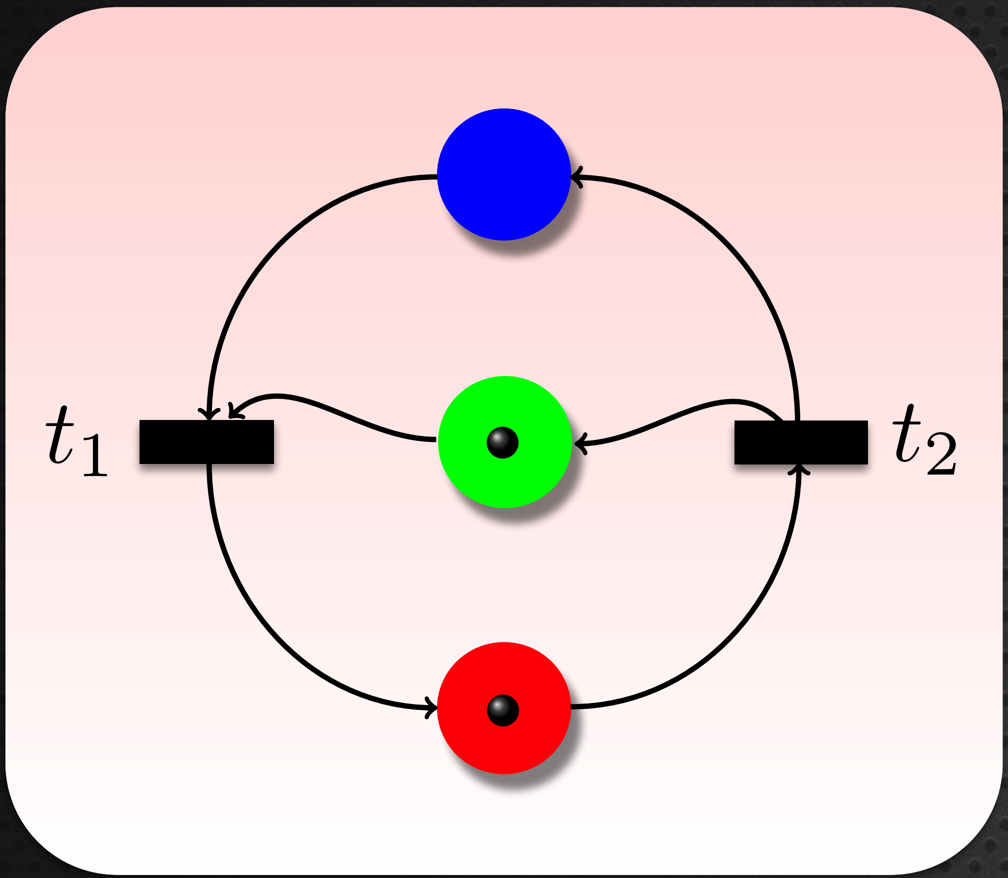


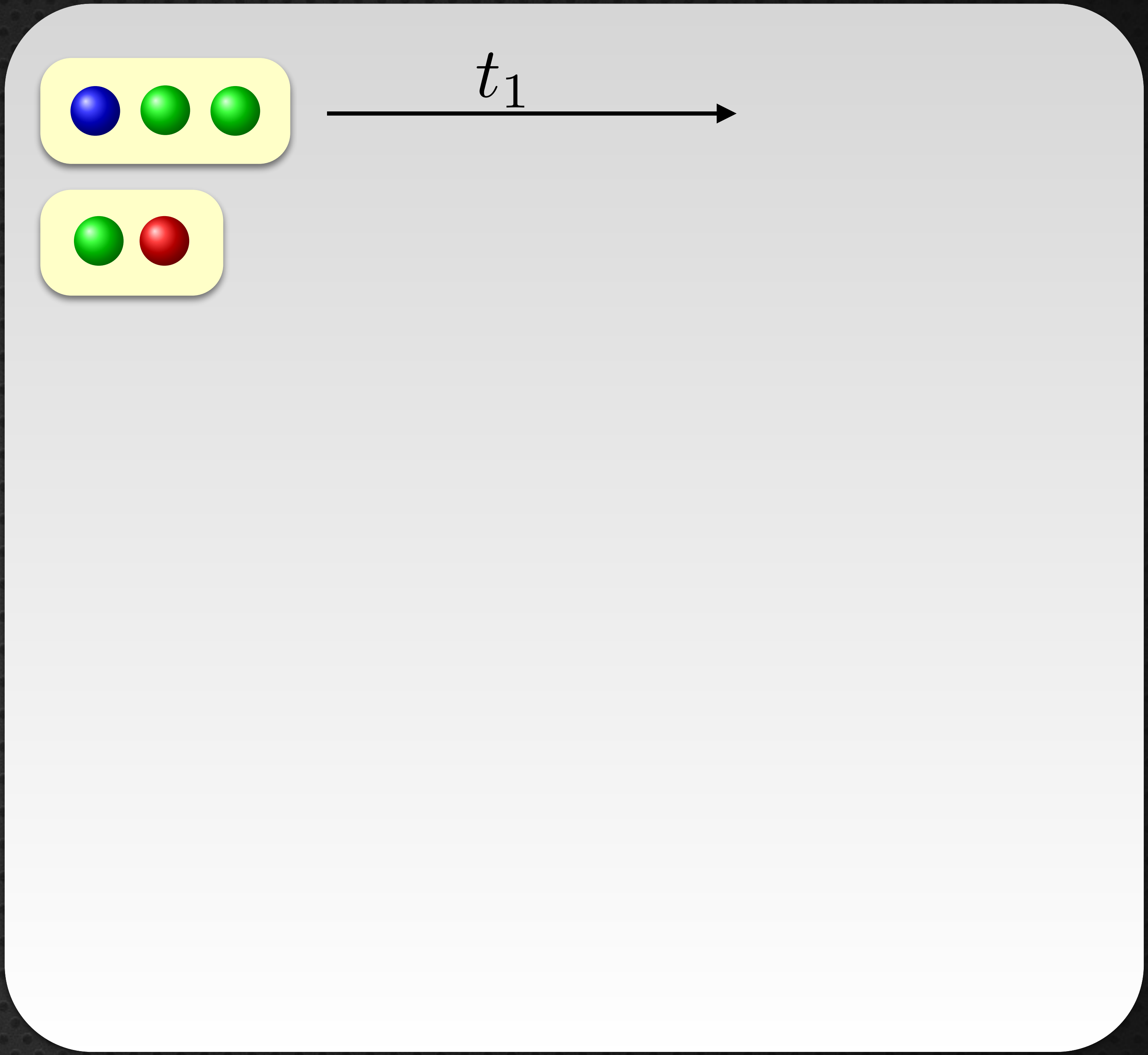
Transitions



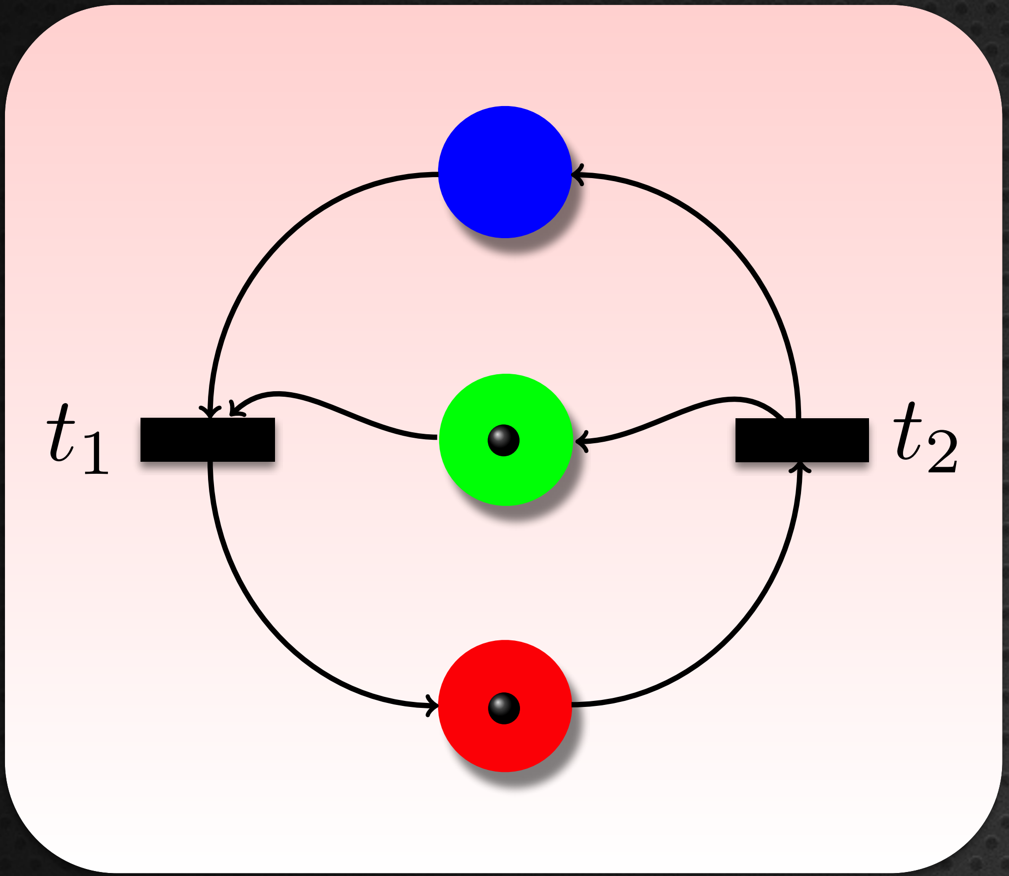


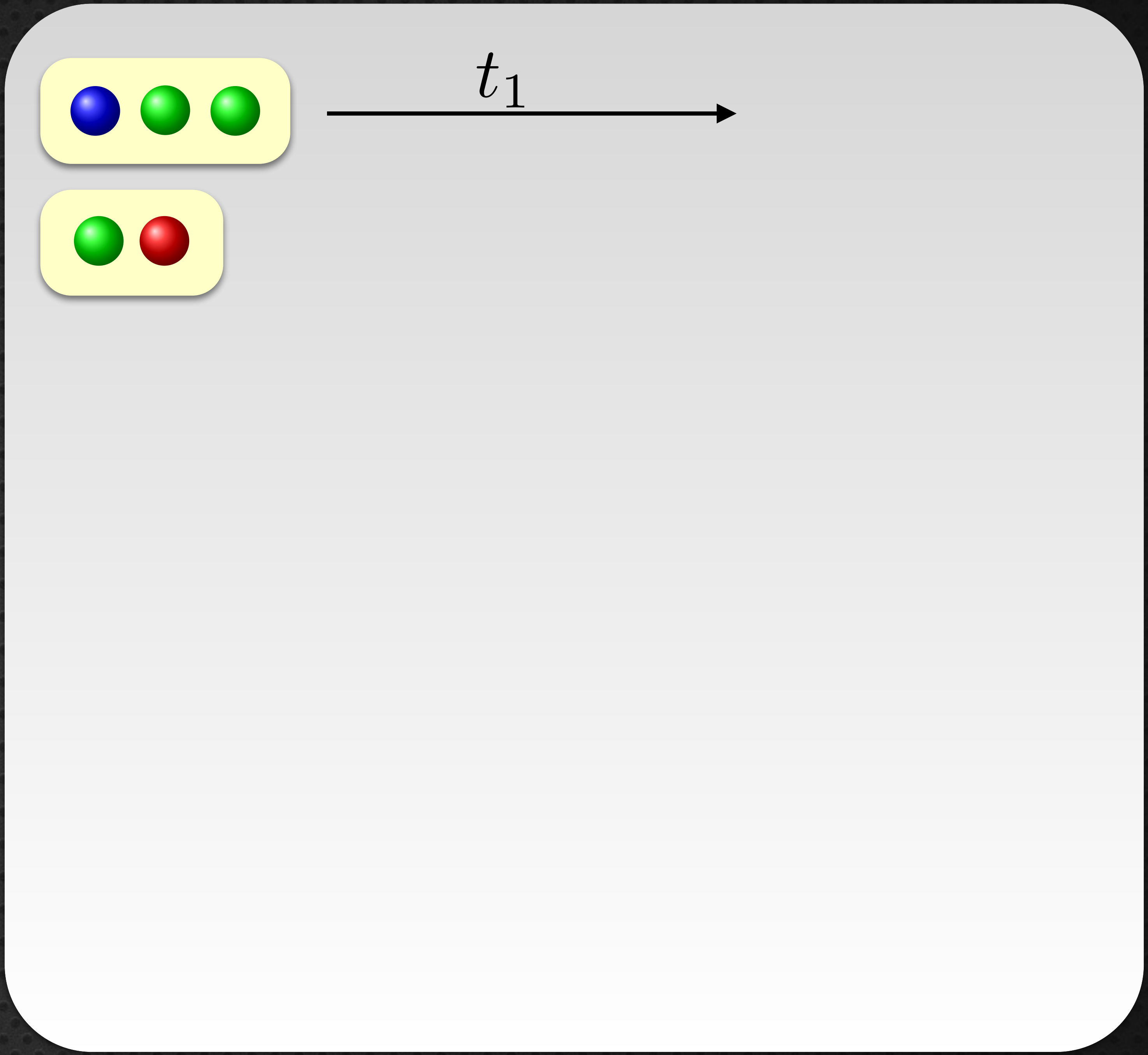
Transitions



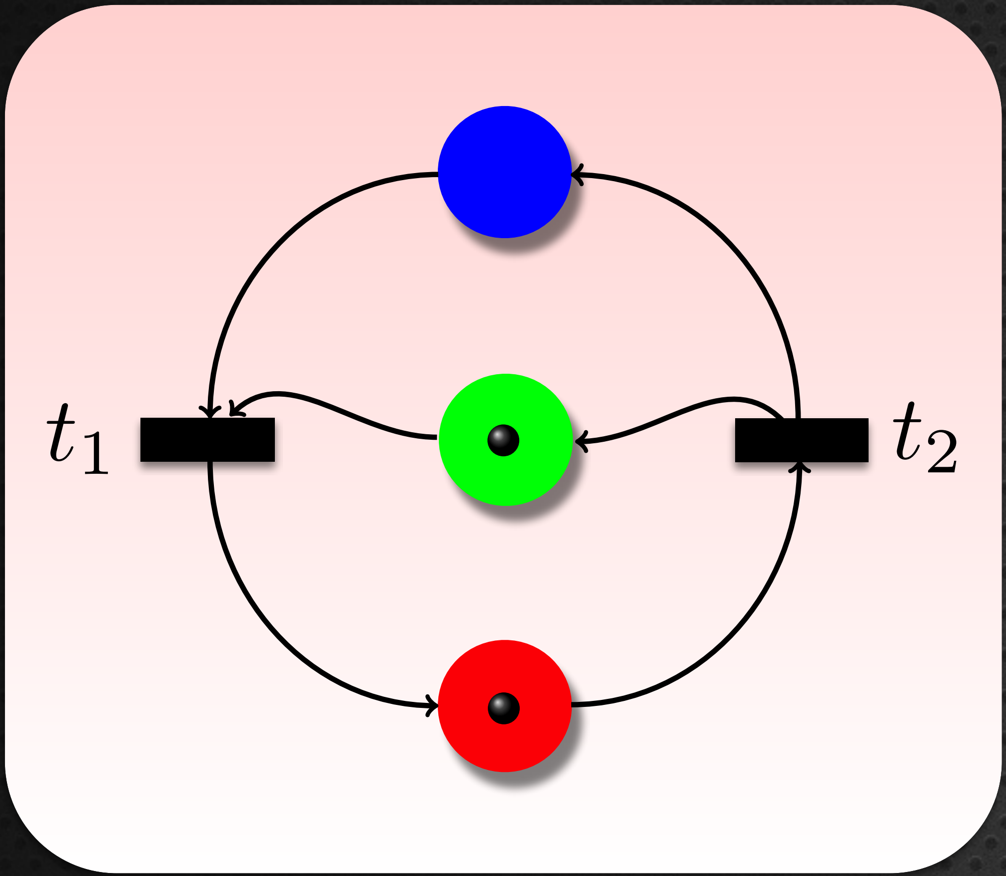


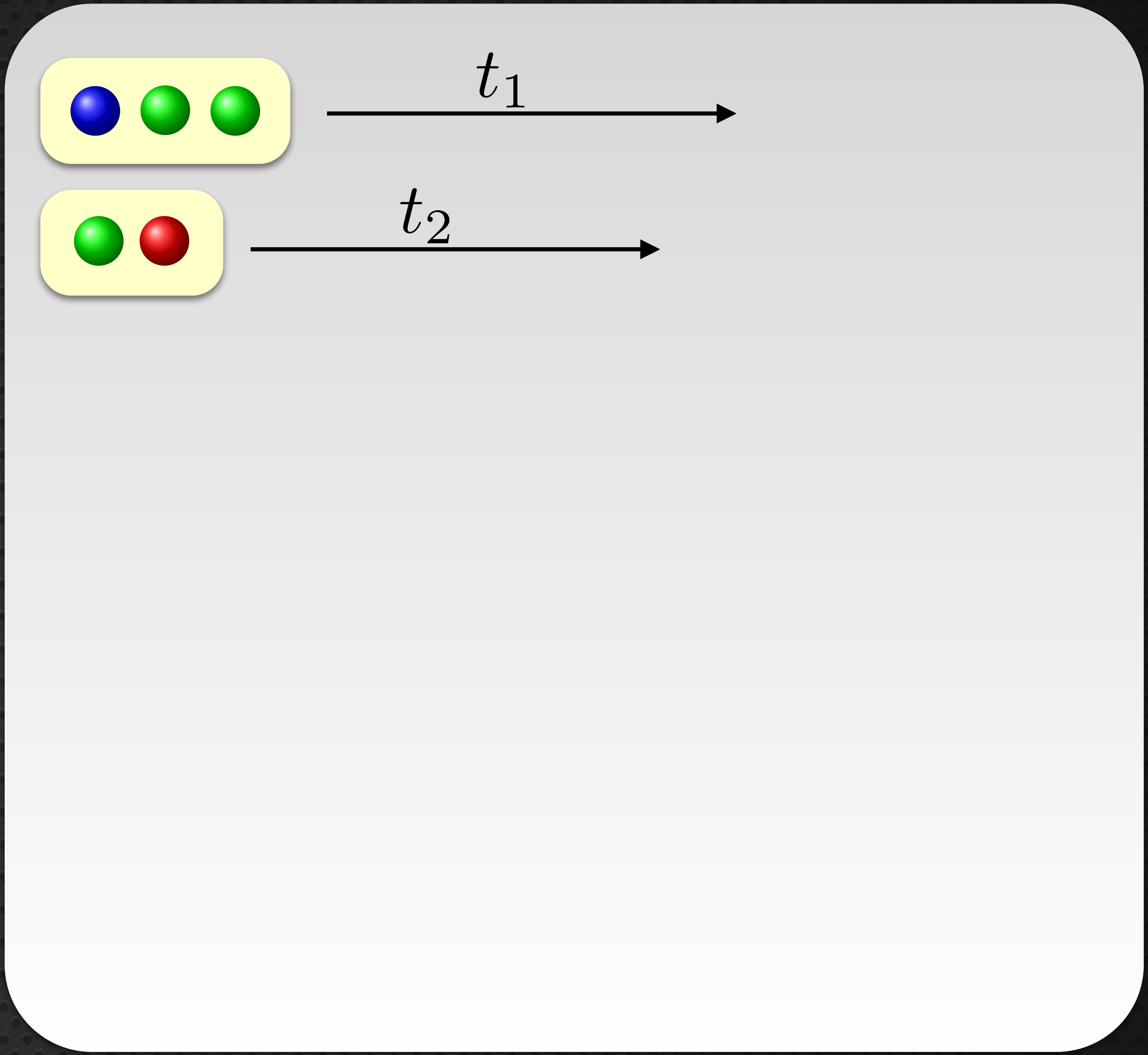
Transitions



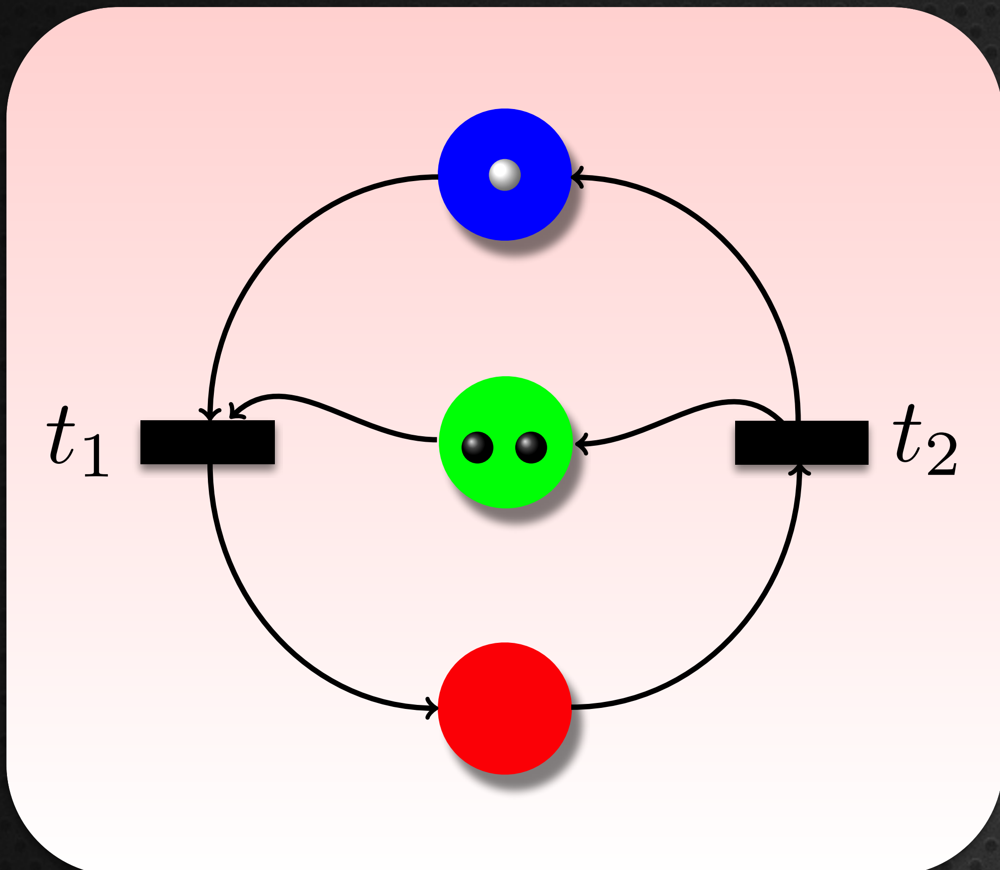


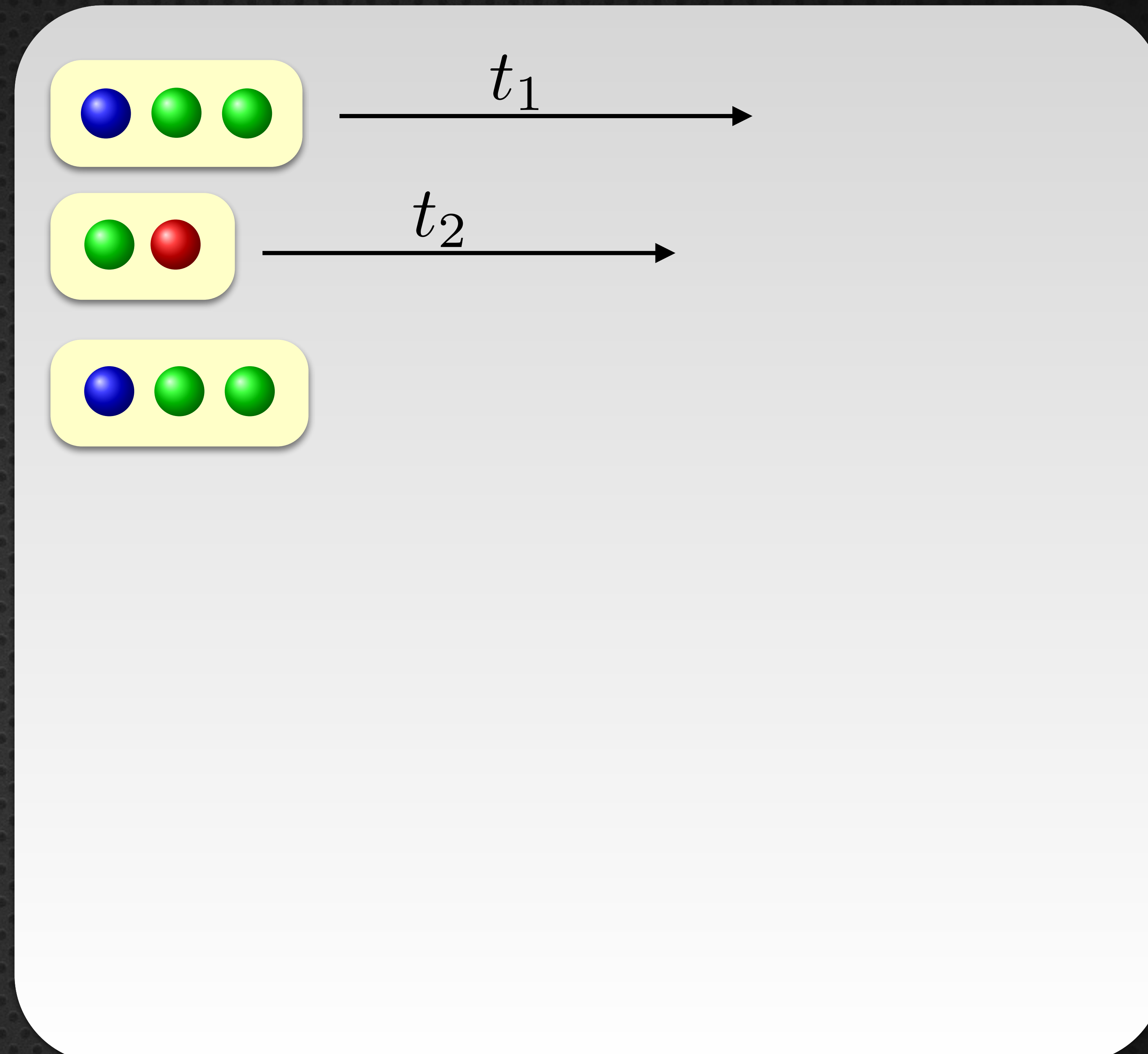
Transitions



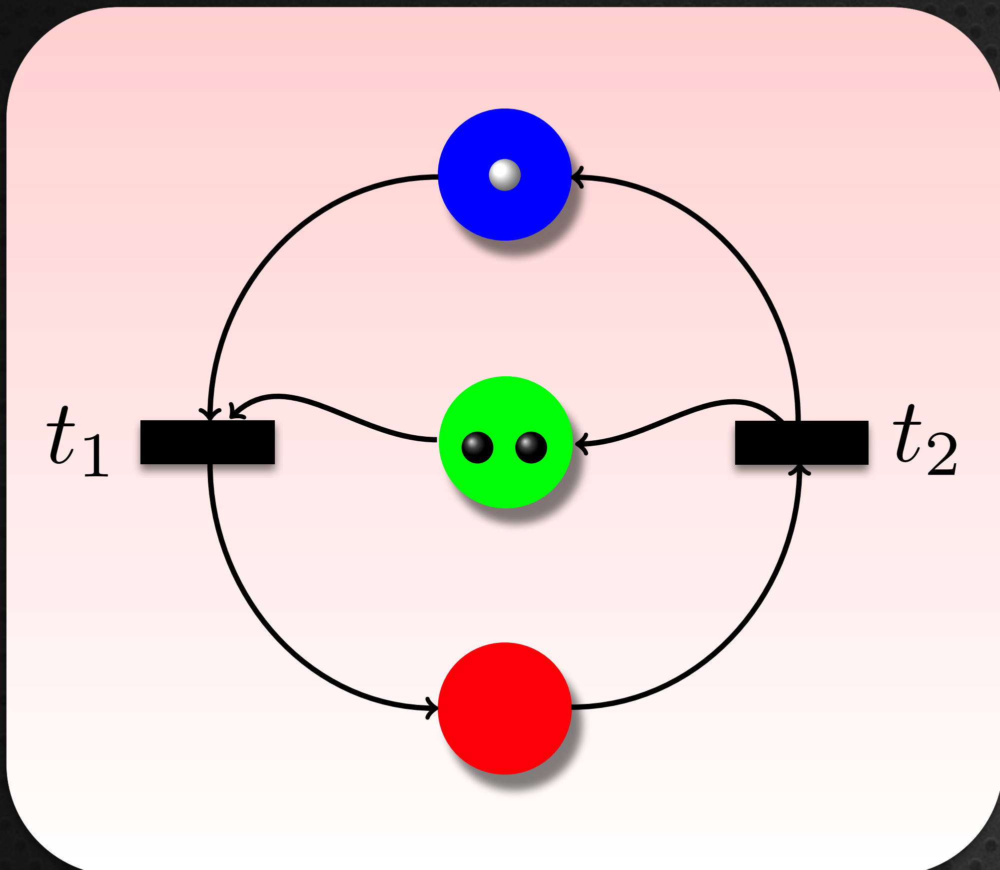


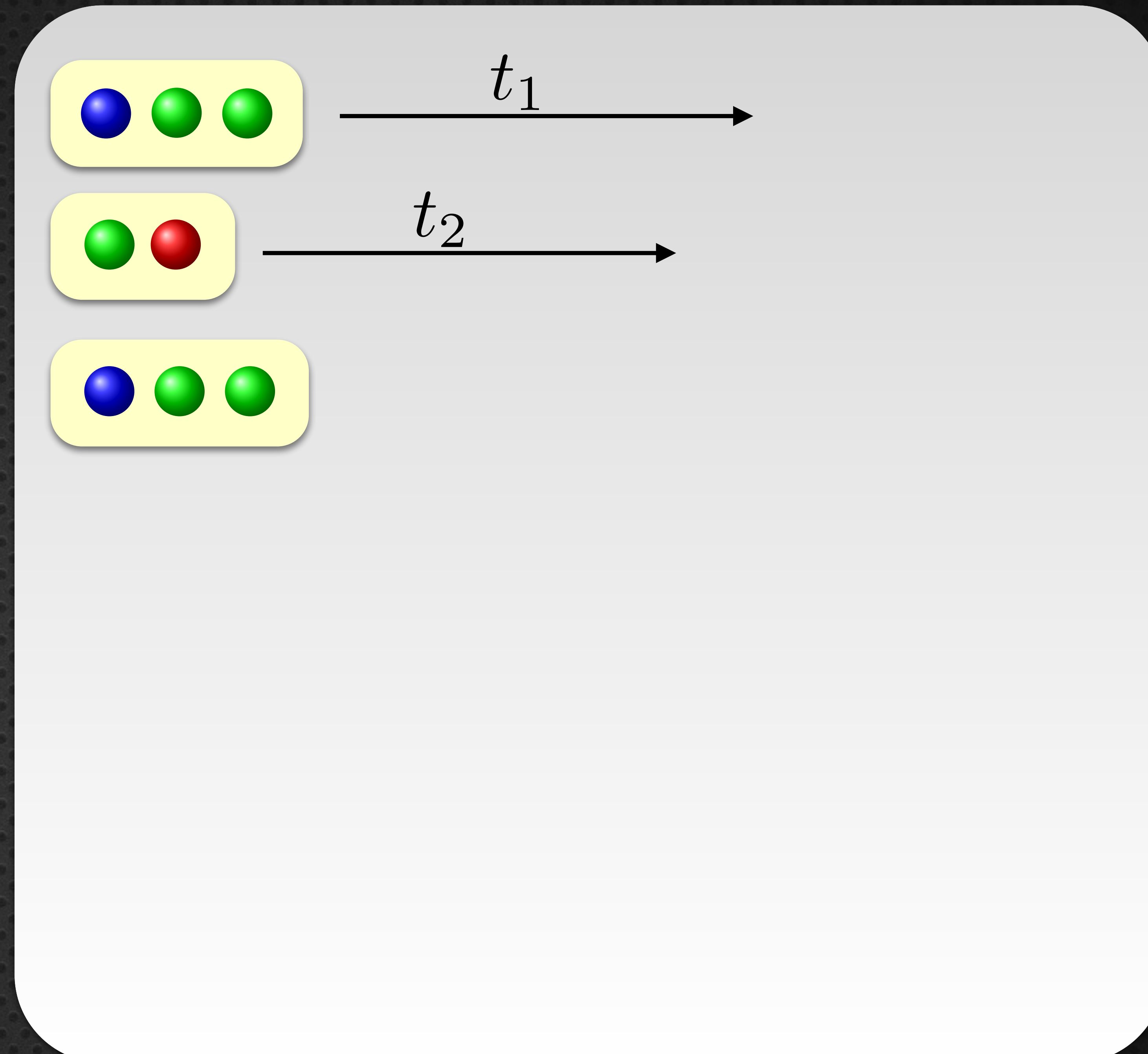
Transitions



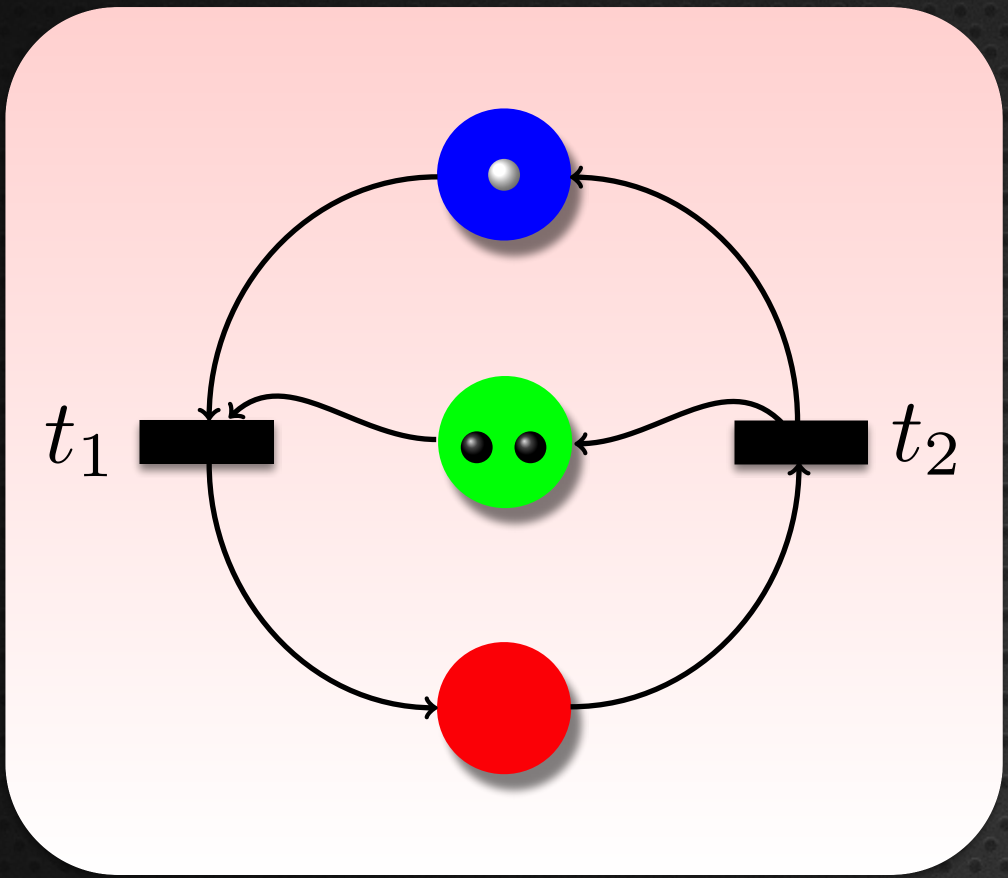


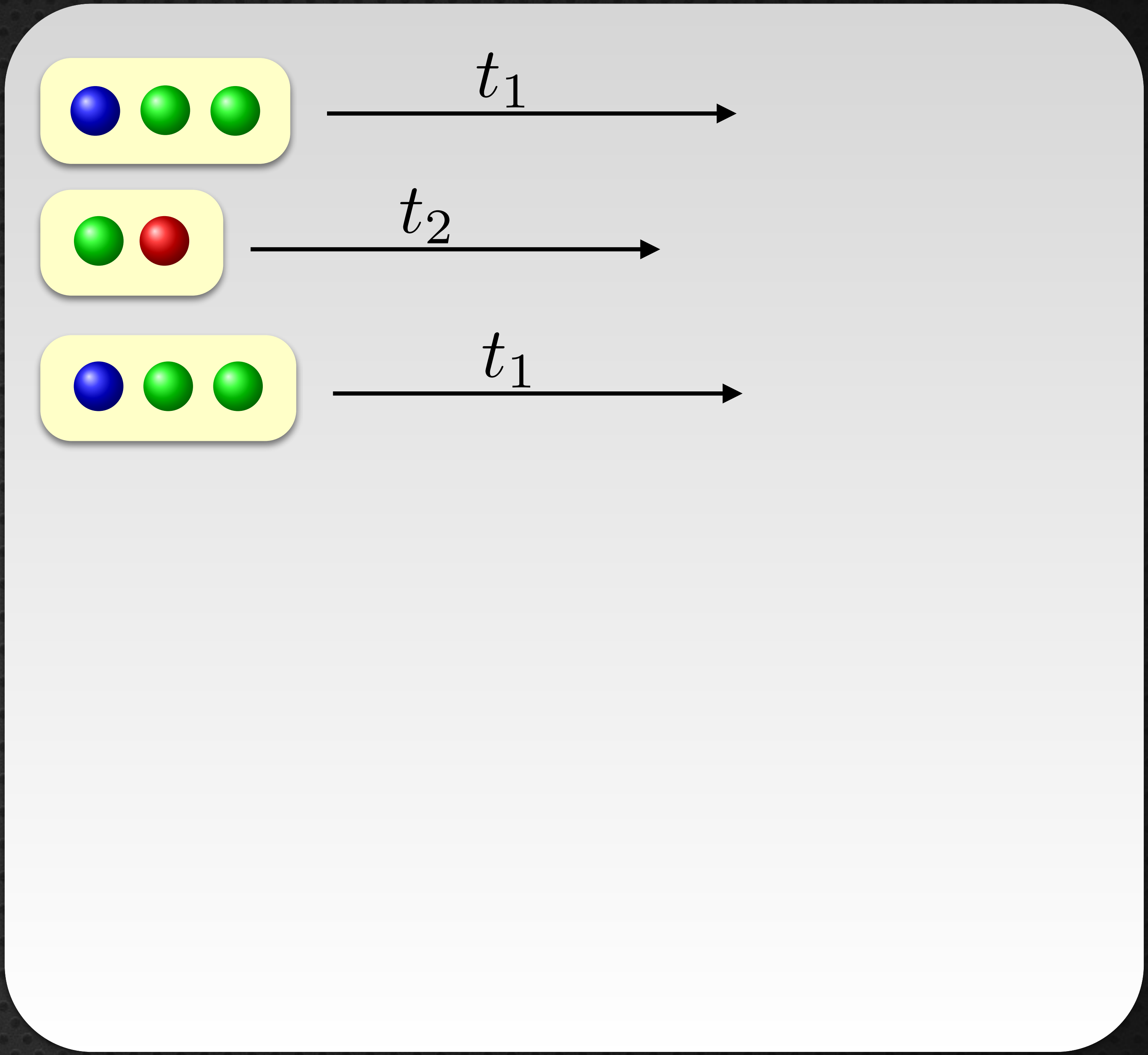
Transitions



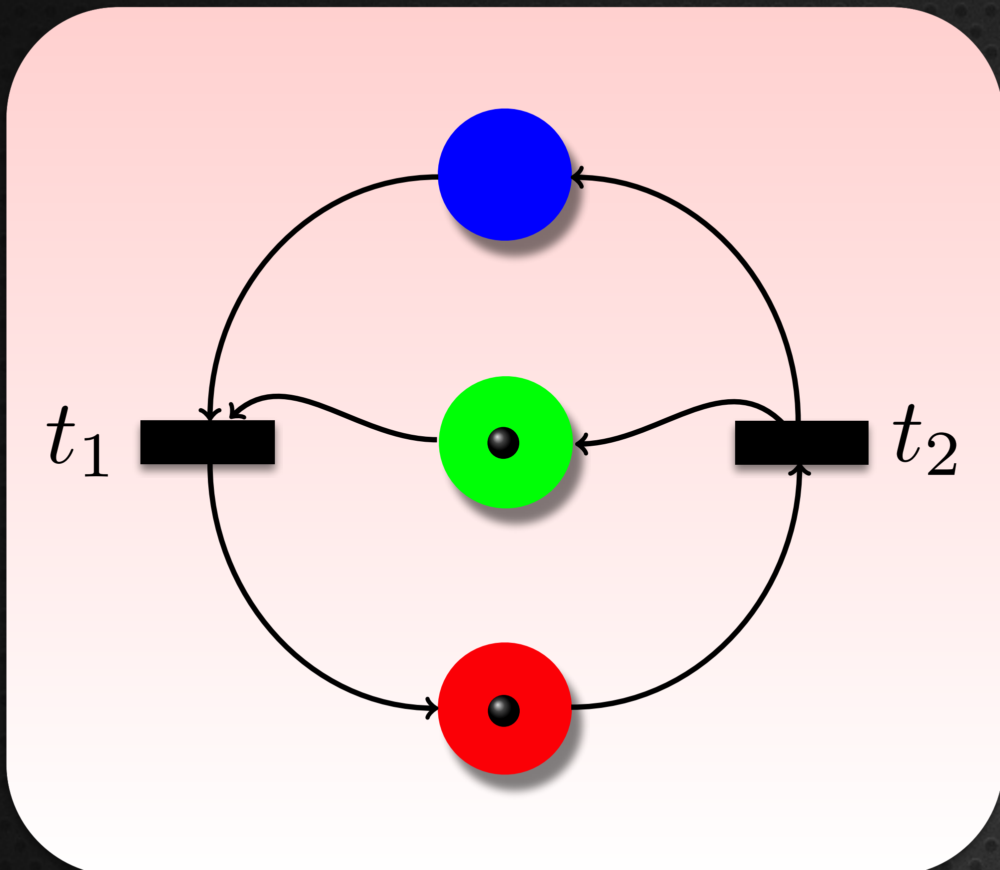


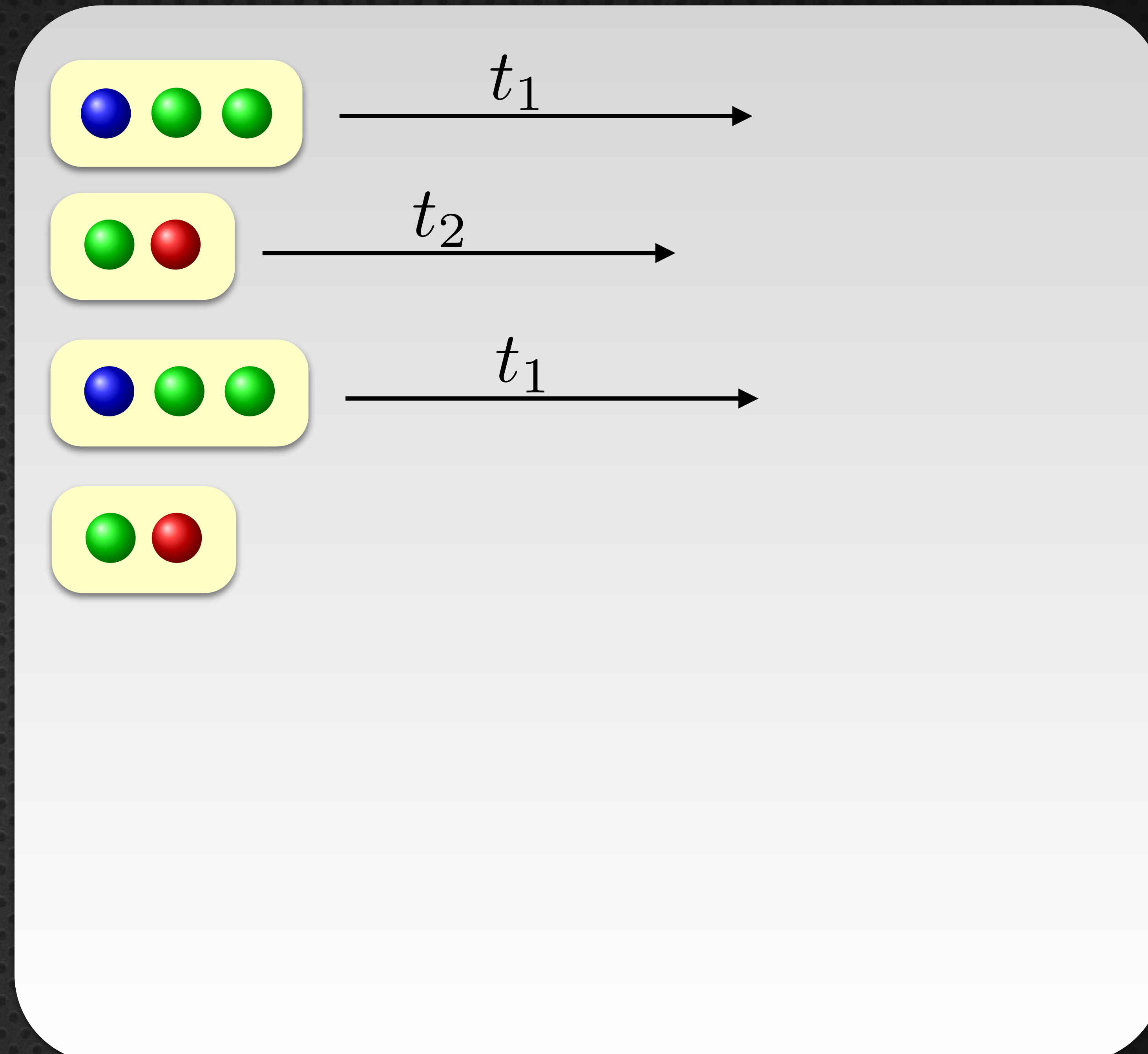
Transitions



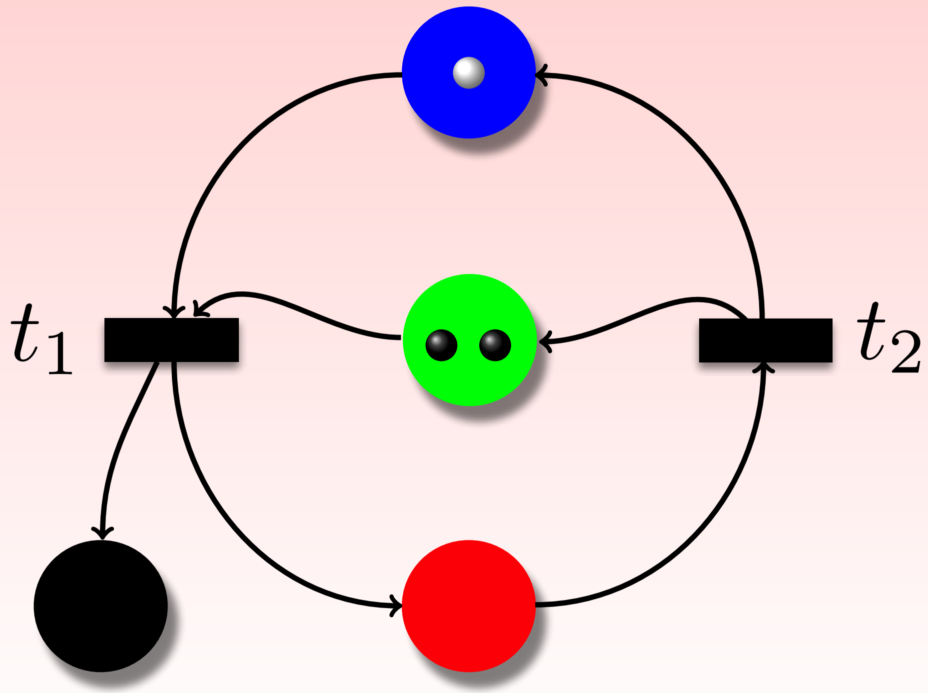


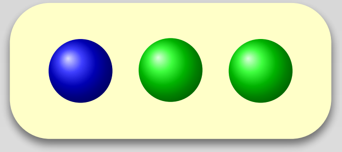
Transitions



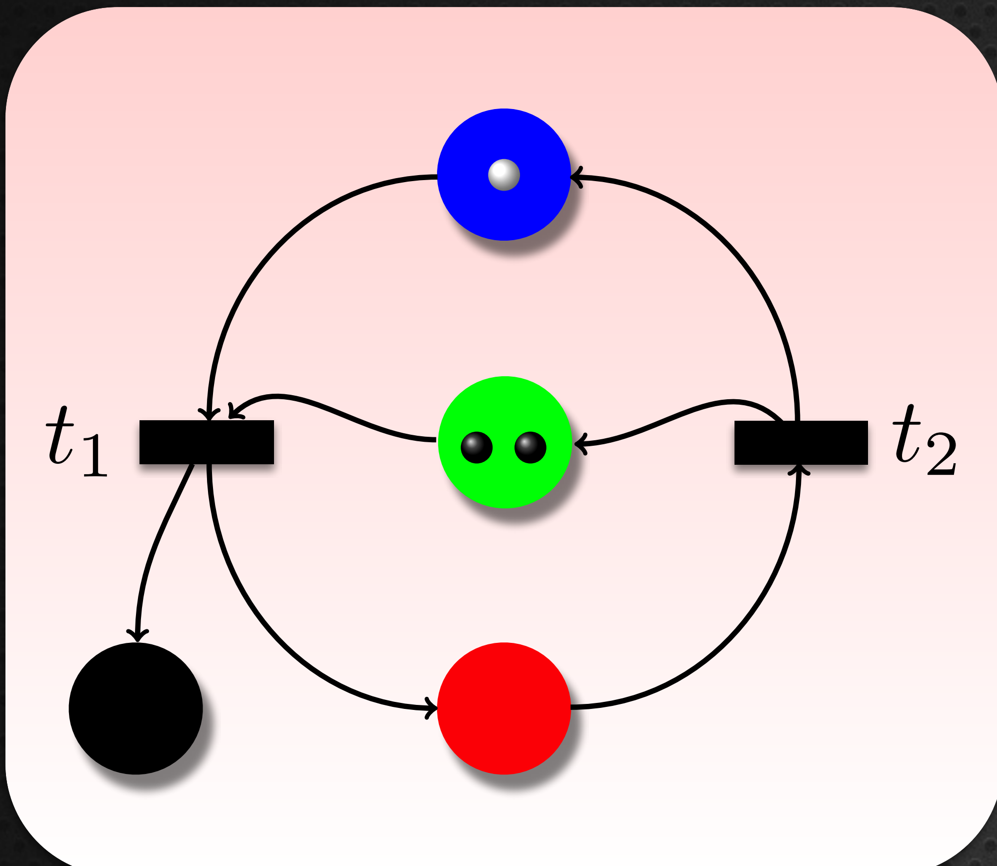


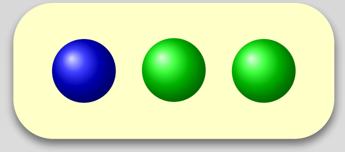
Transitions



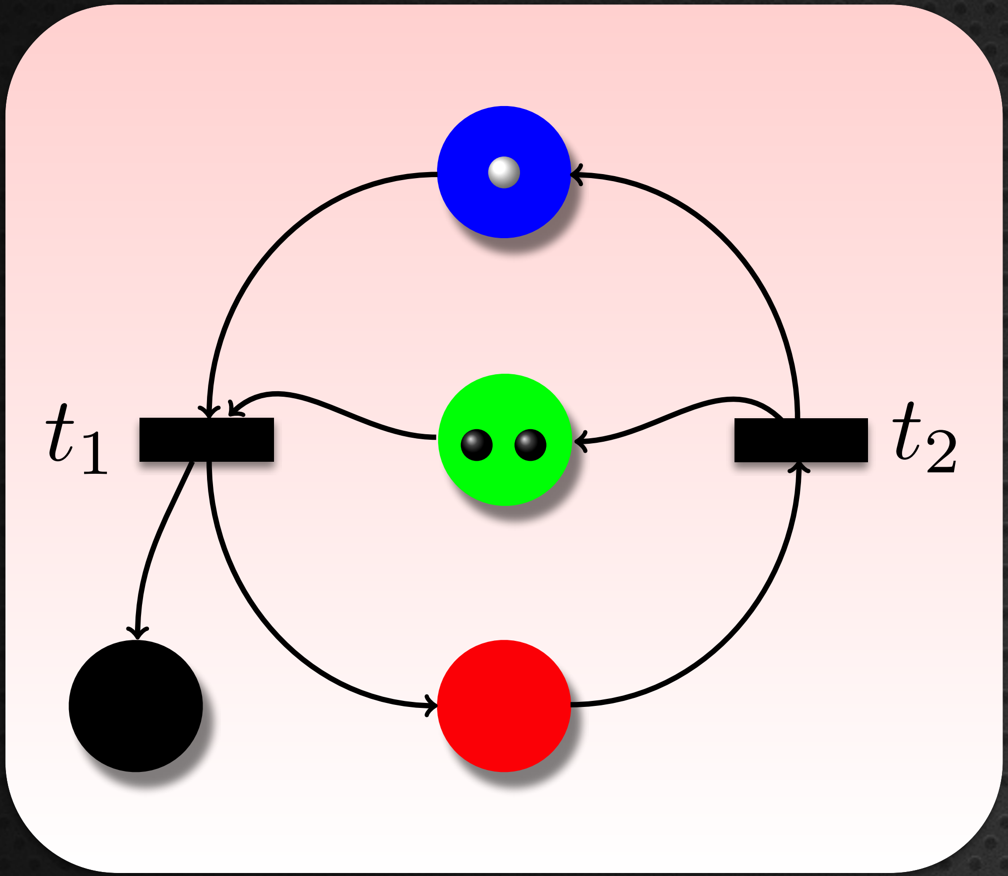


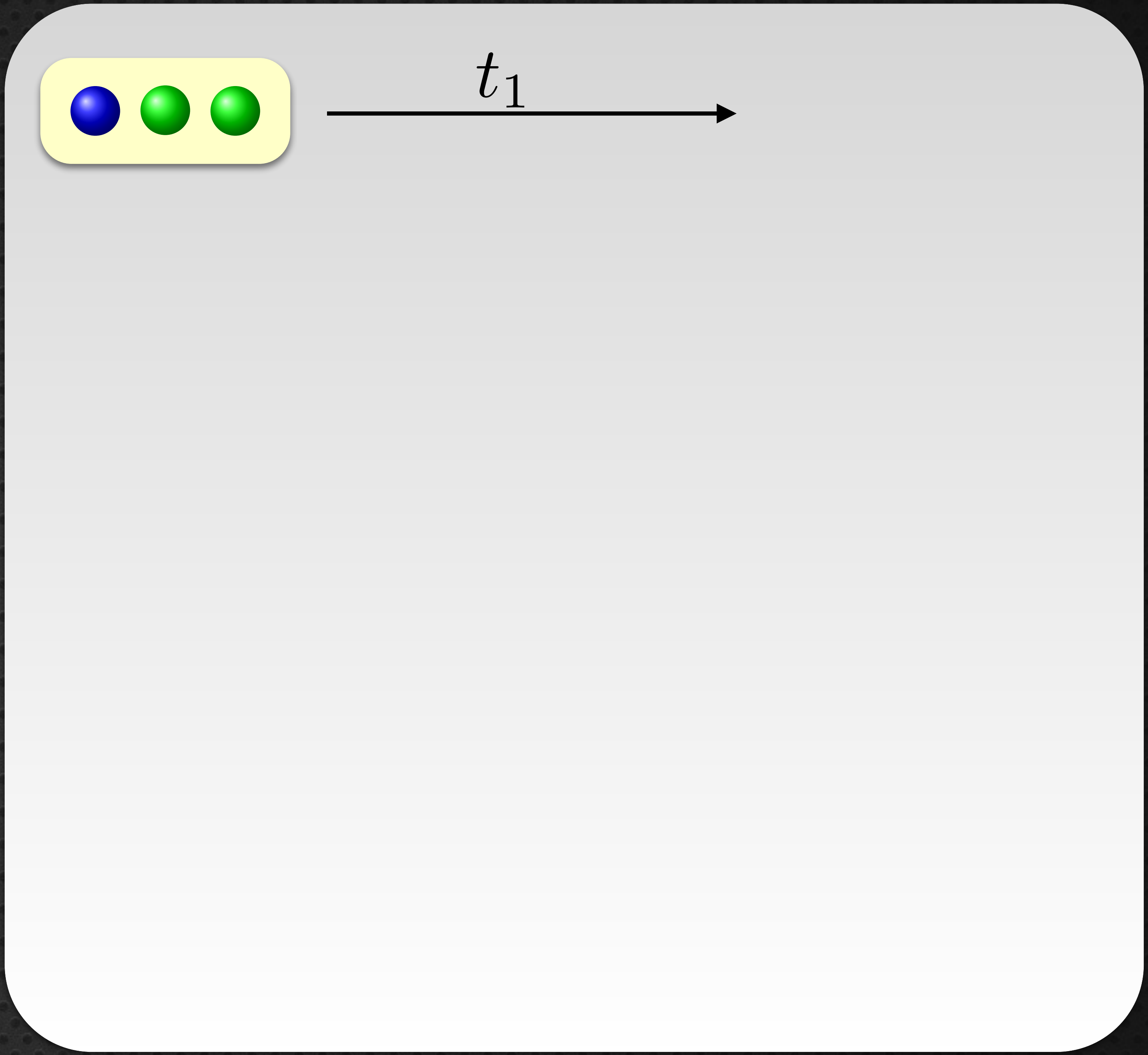
Transitions



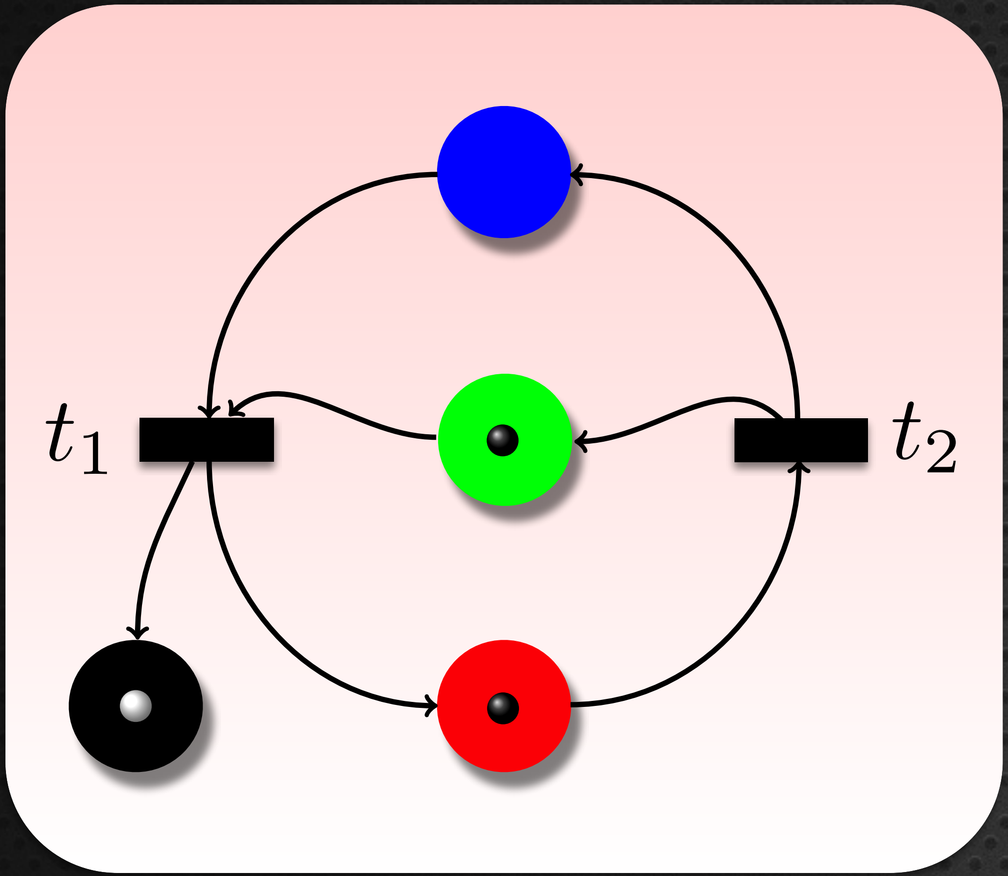


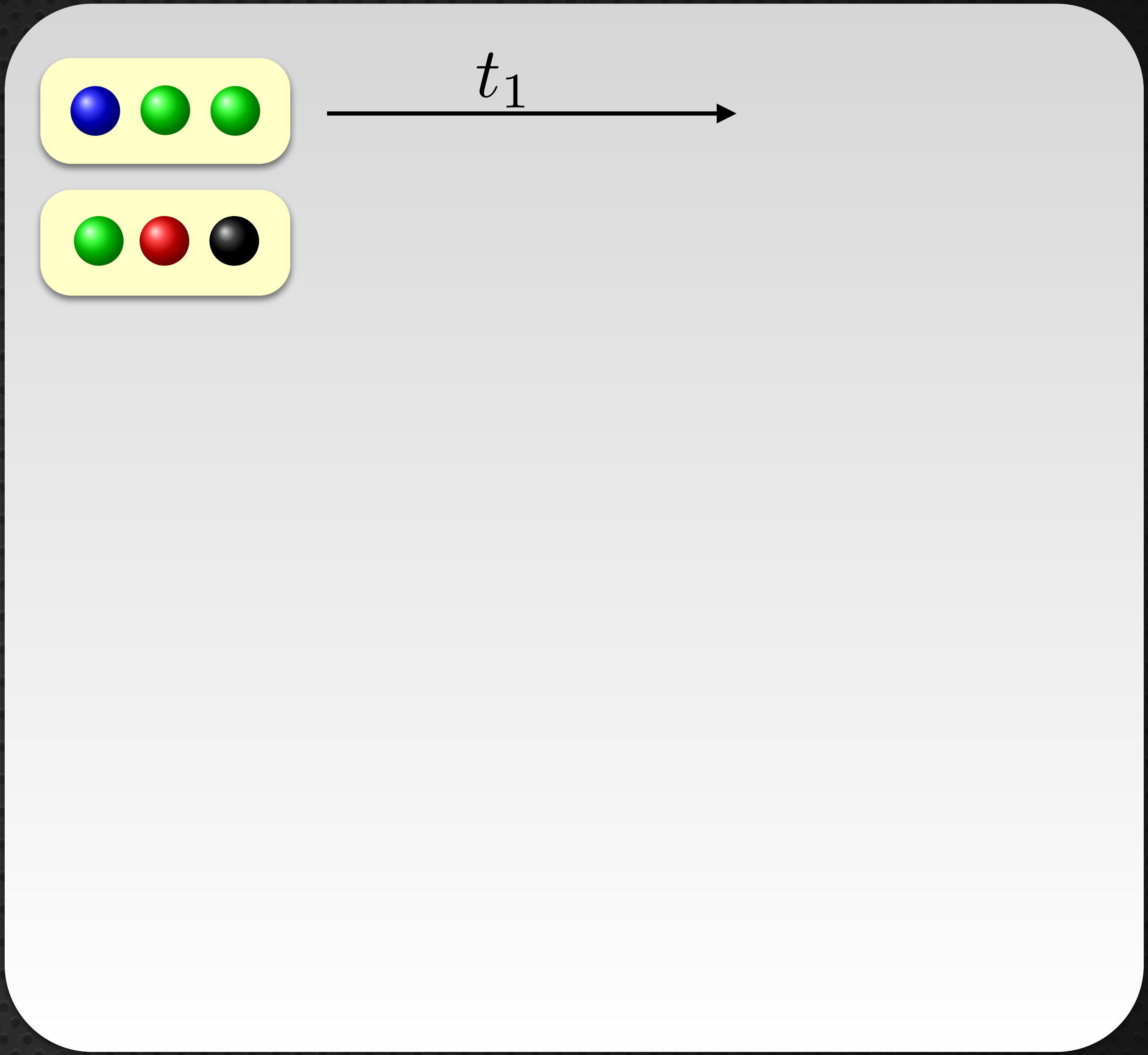
Transitions



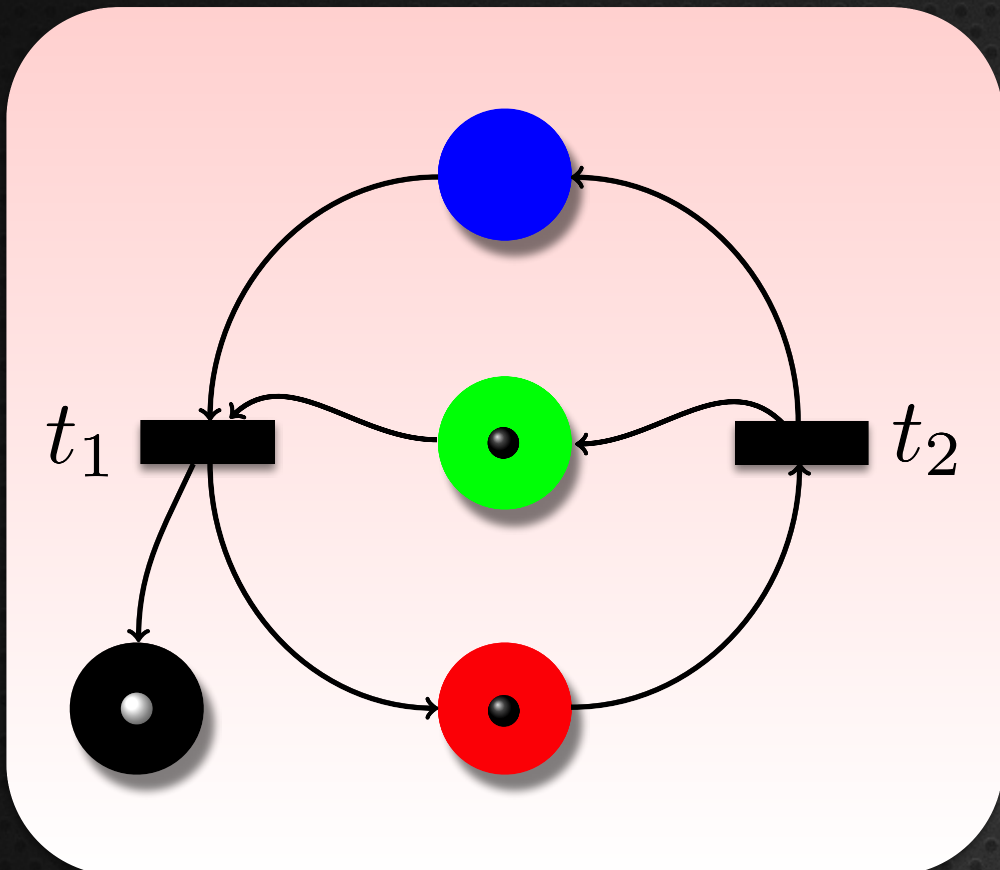


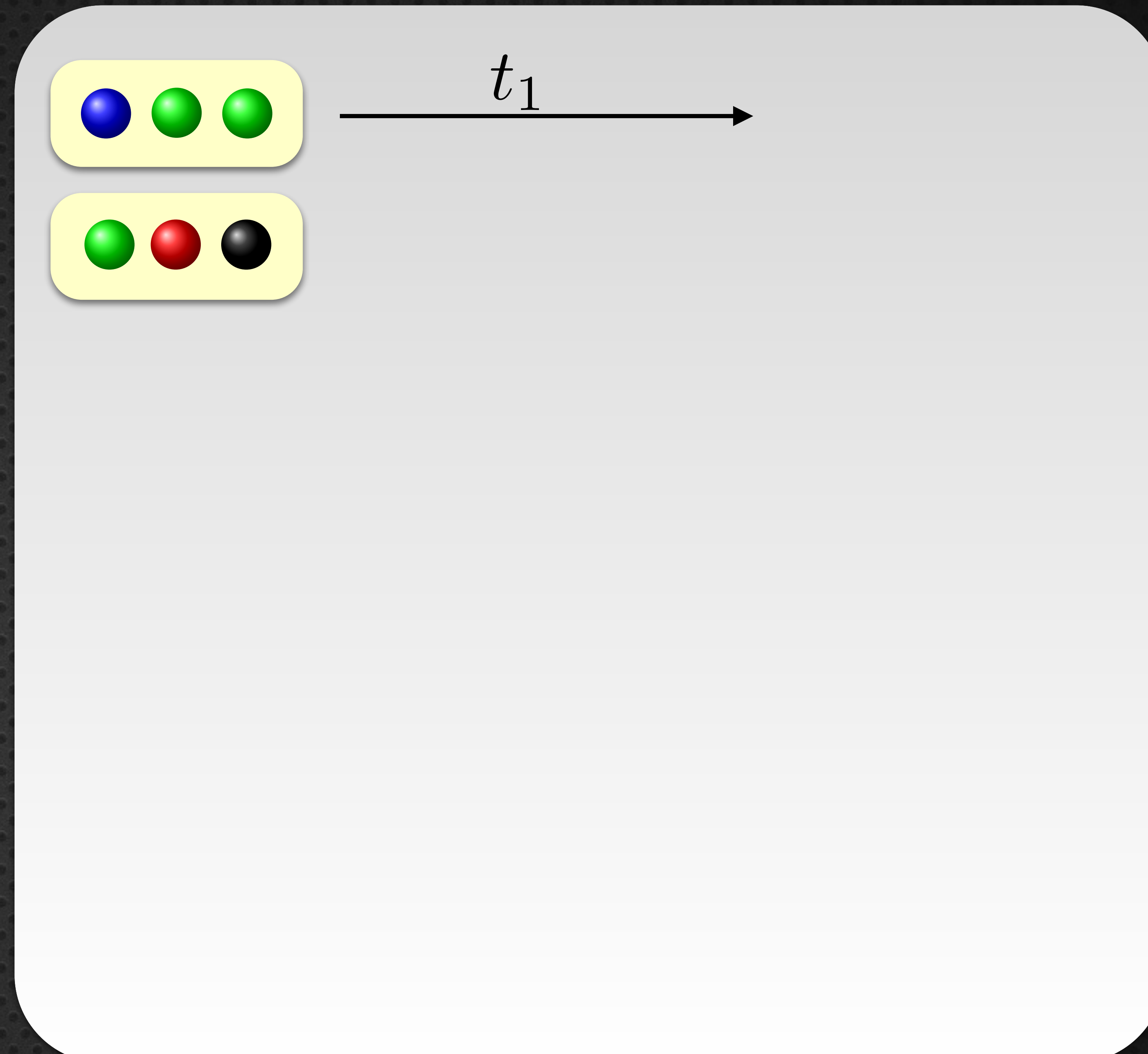
Transitions



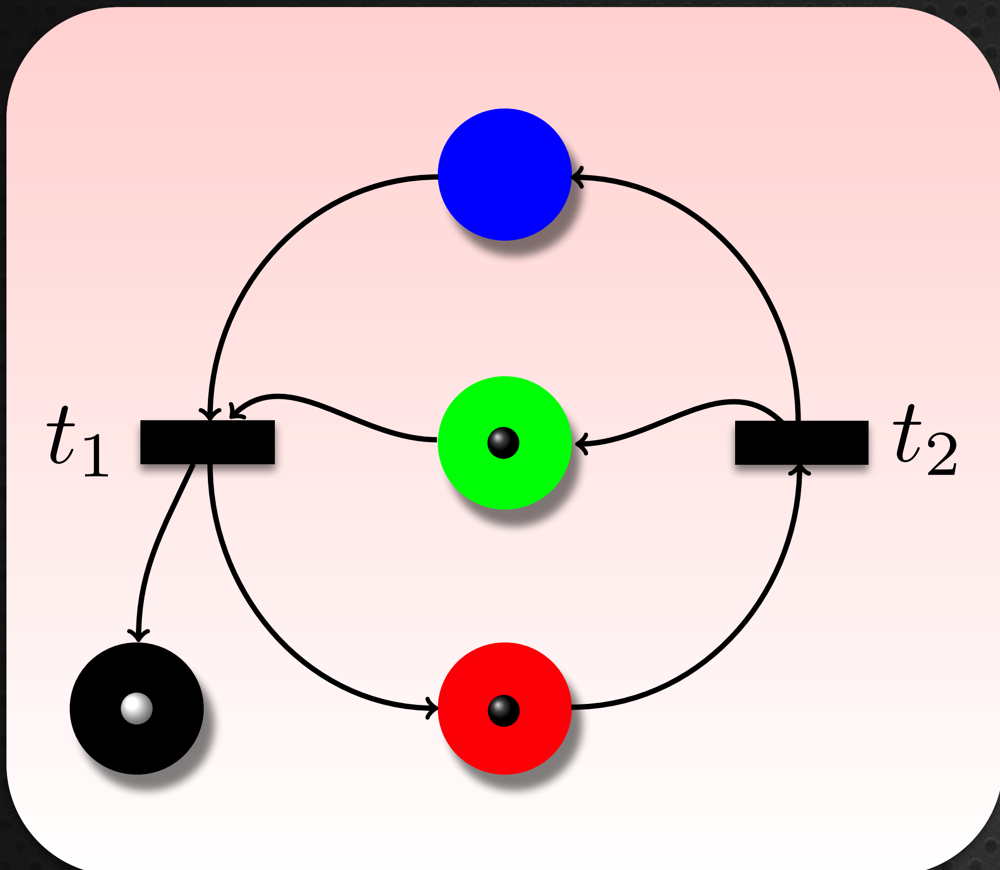


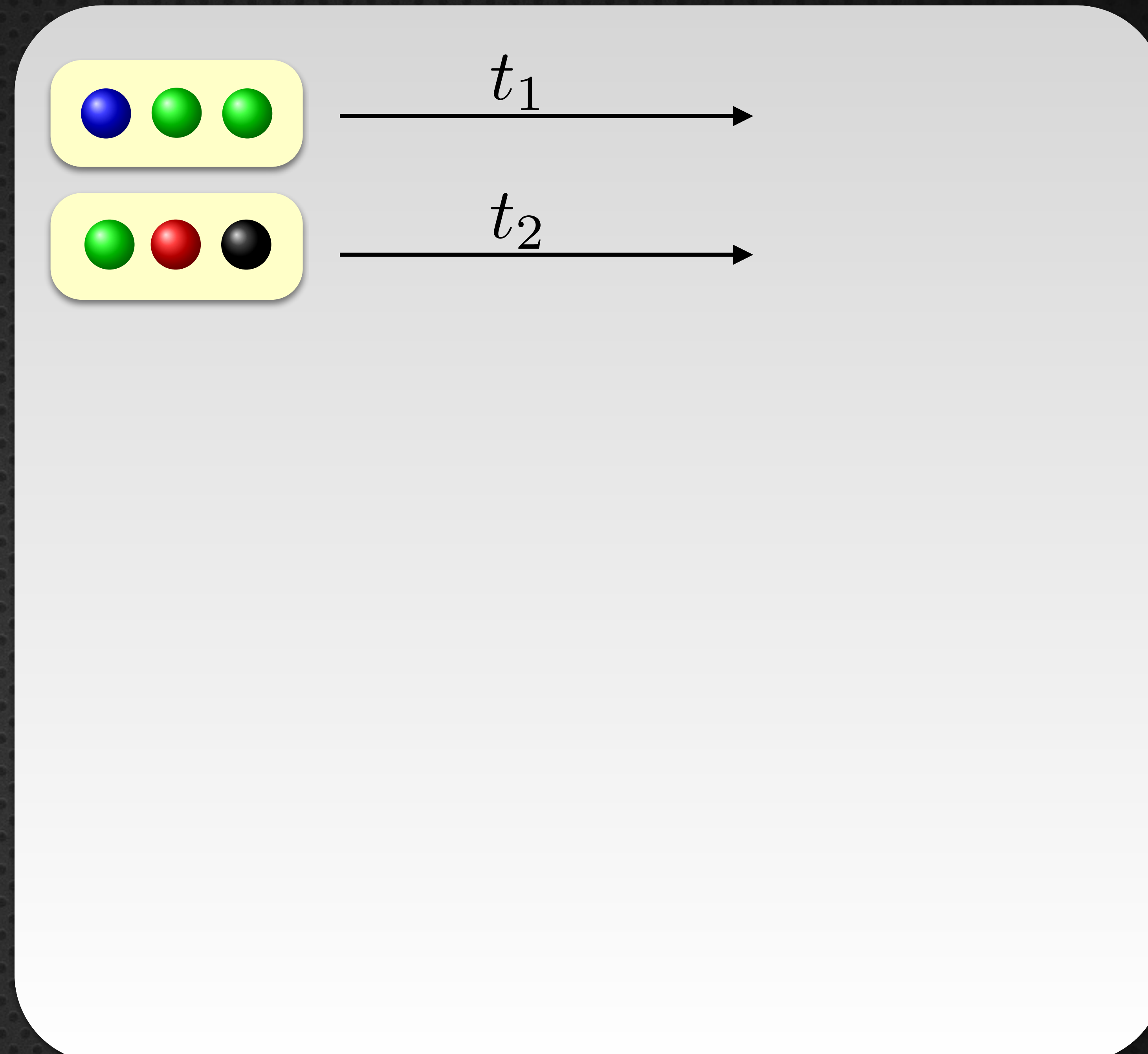
Transitions



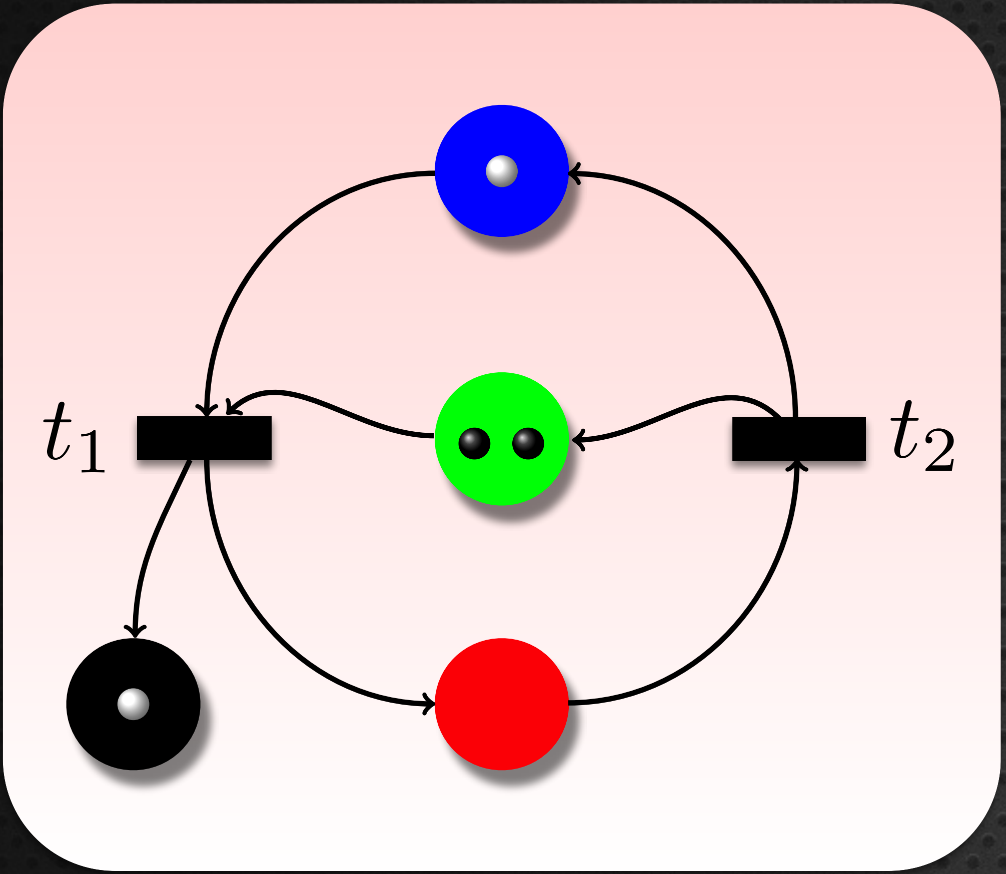


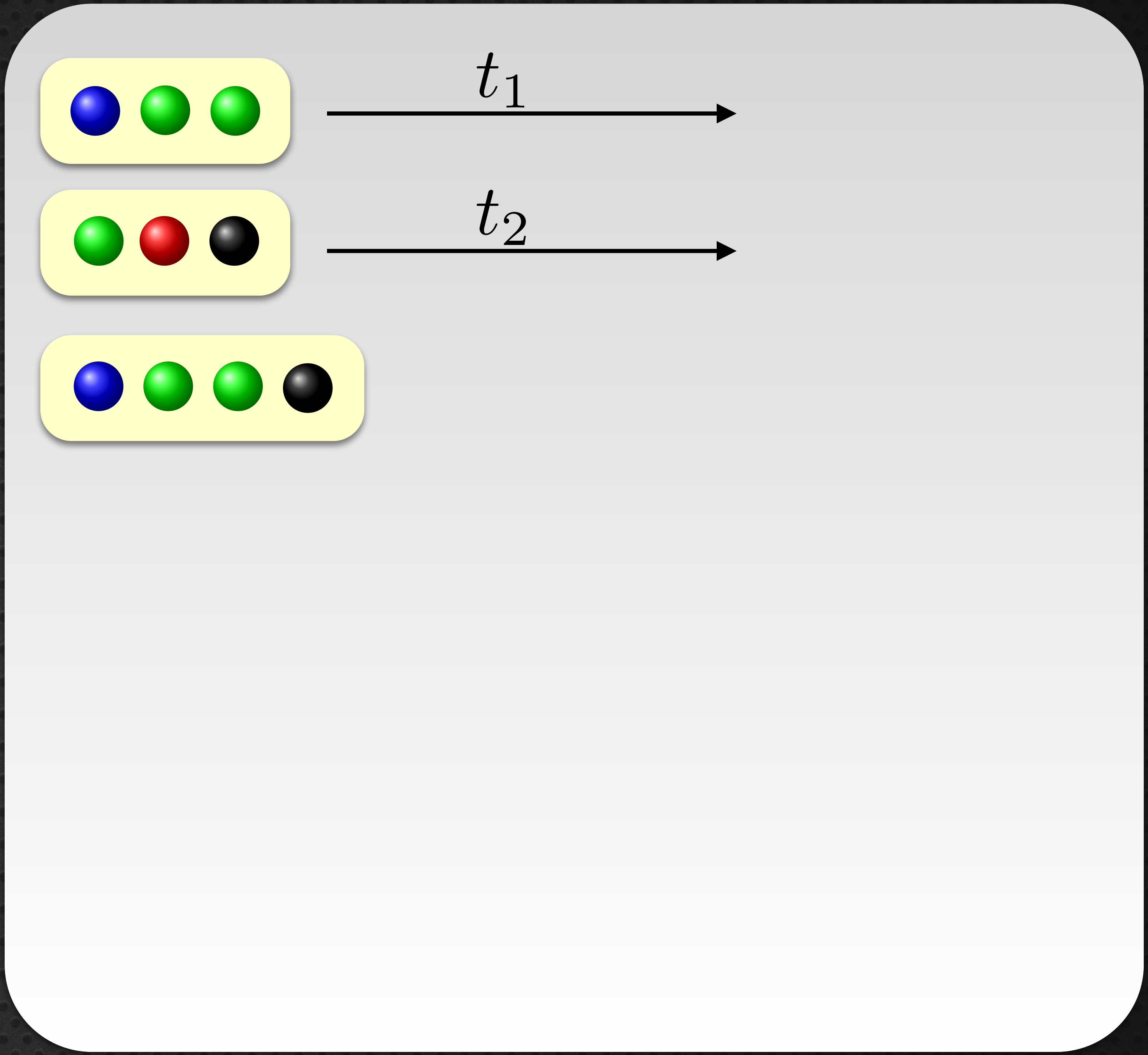
Transitions



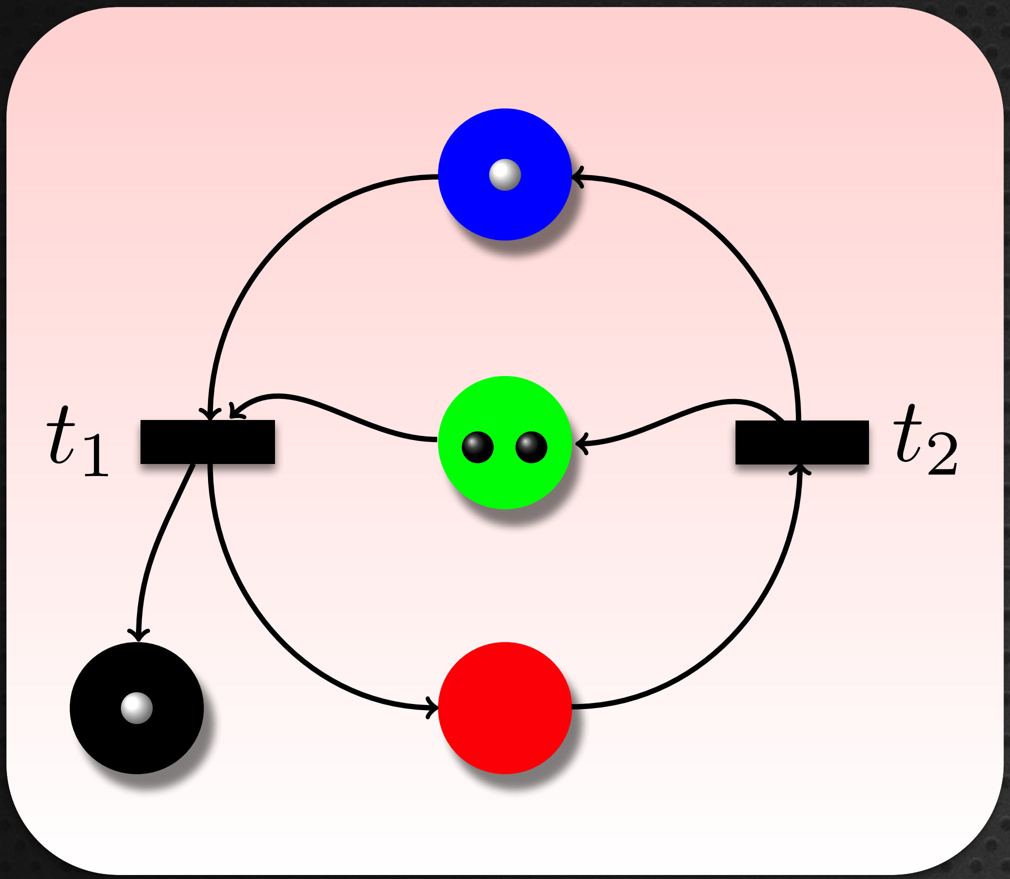


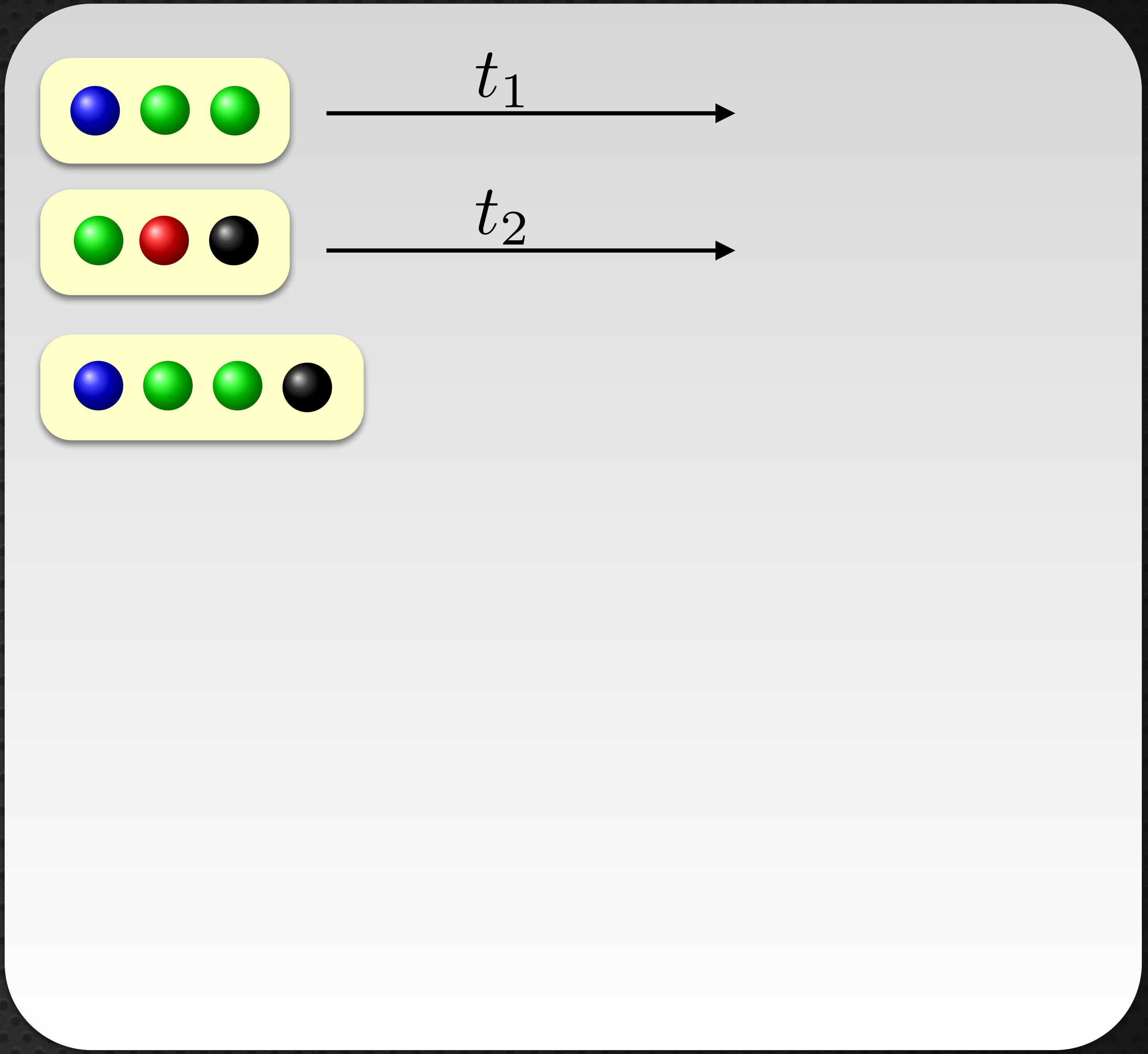
Transitions



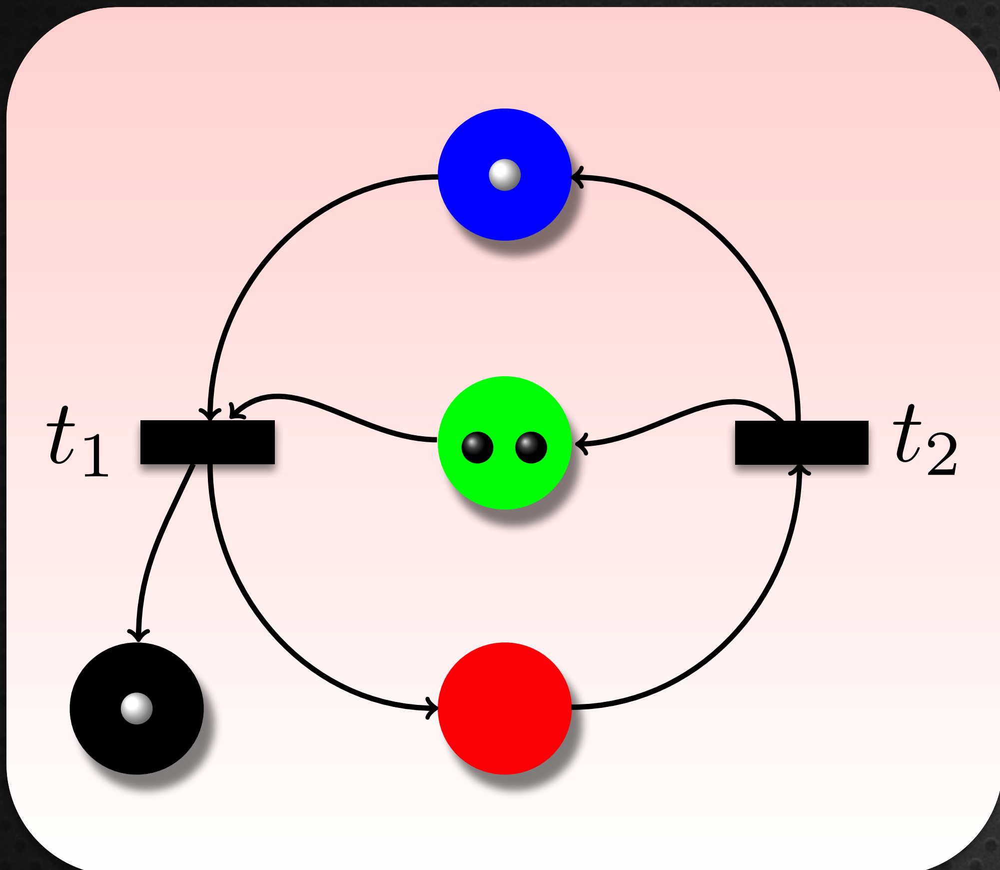


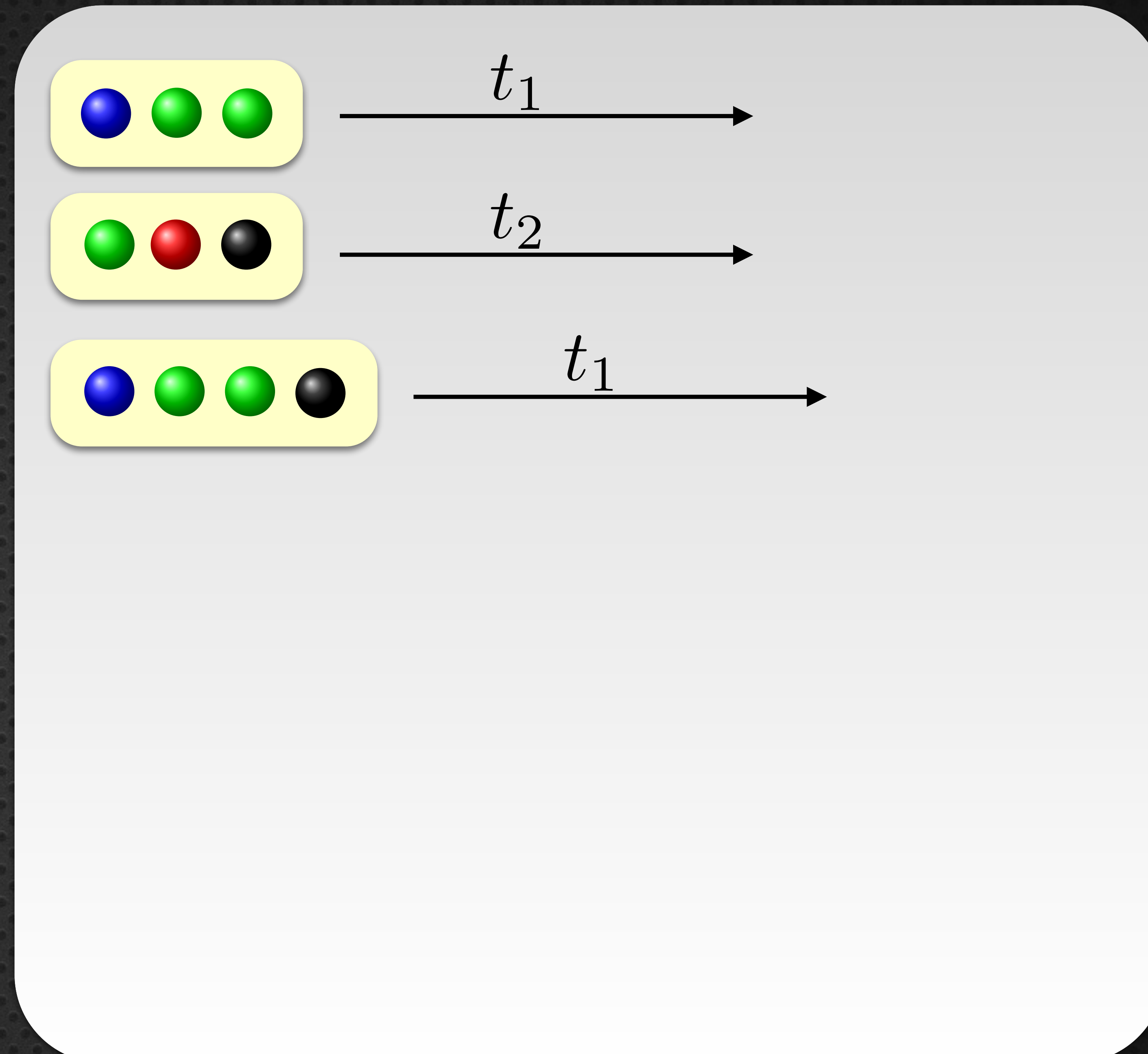
Transitions



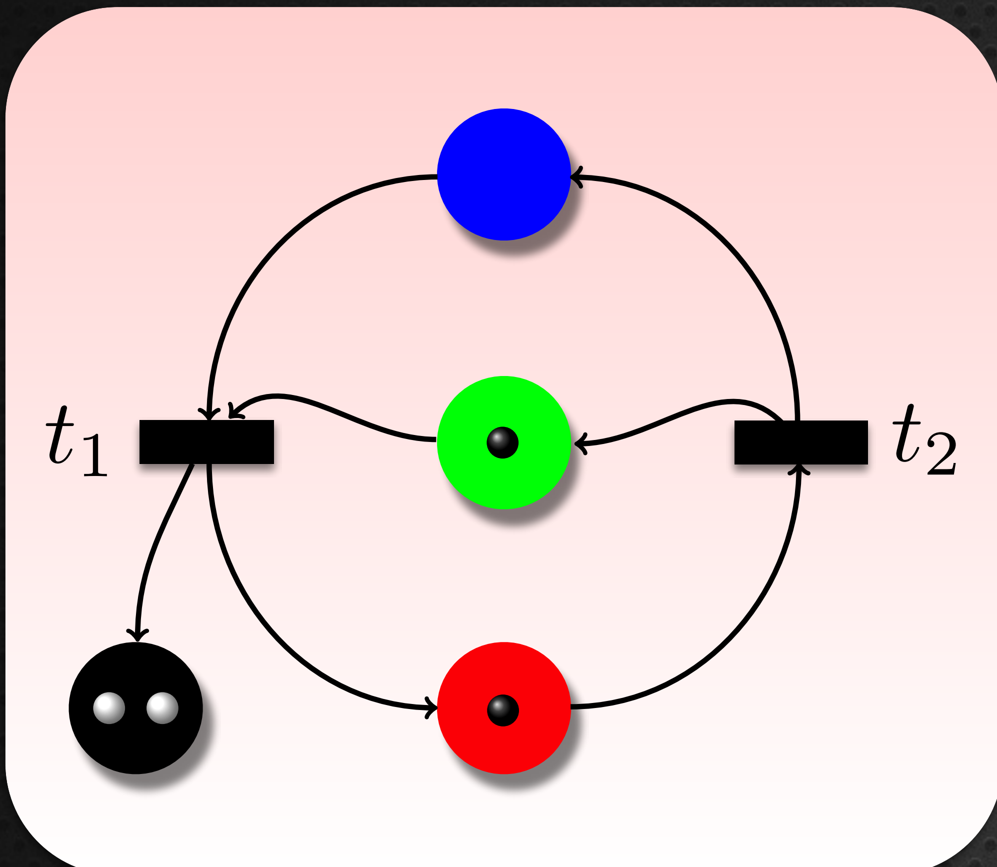


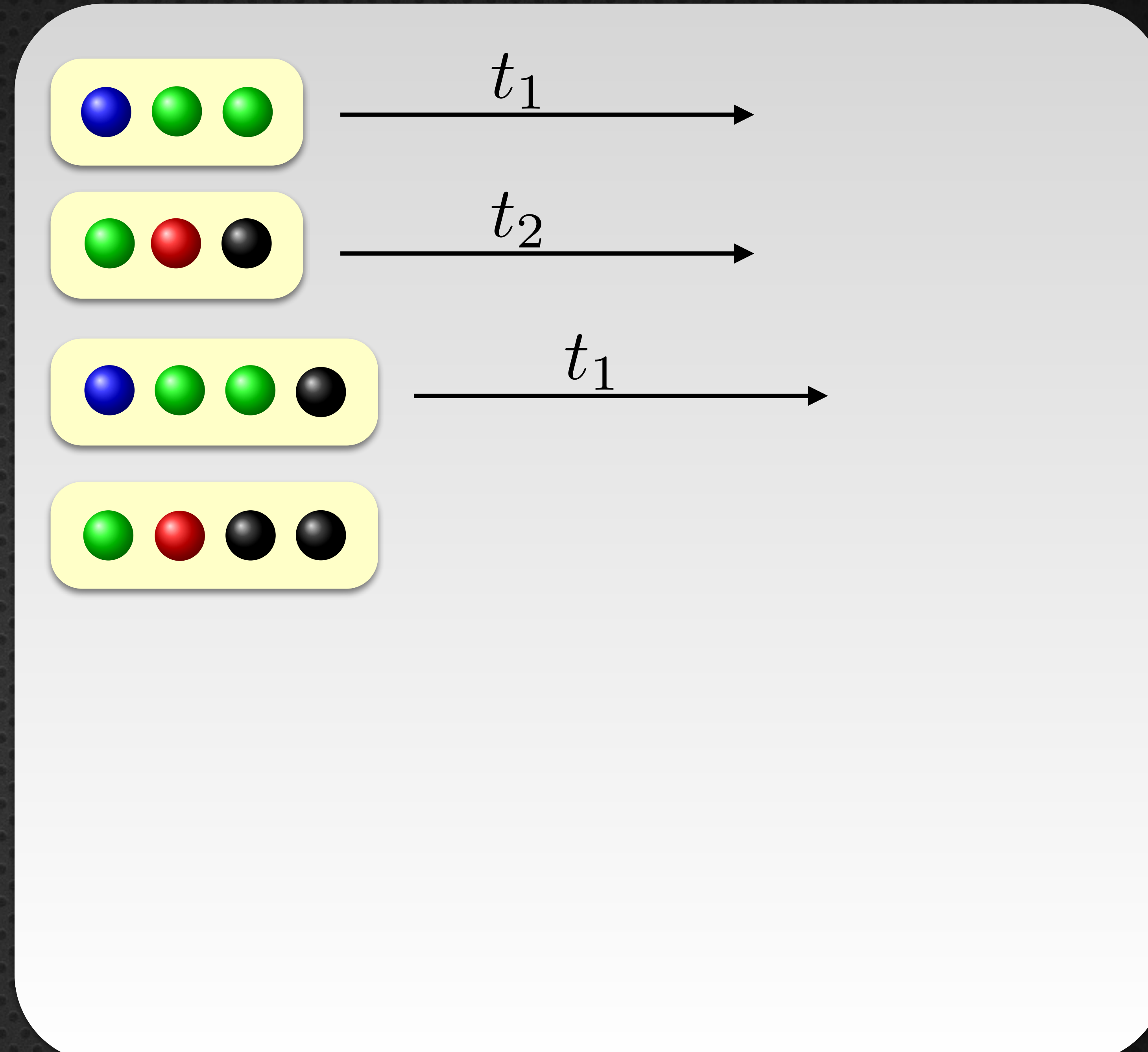
Transitions



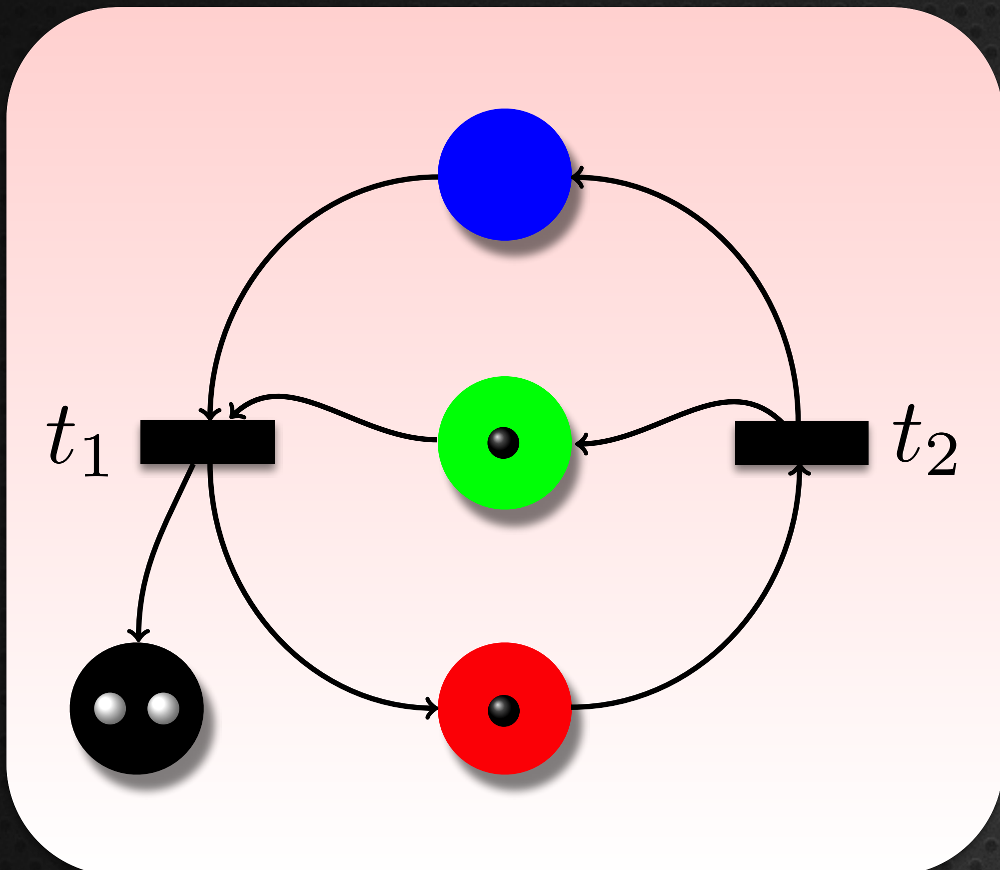


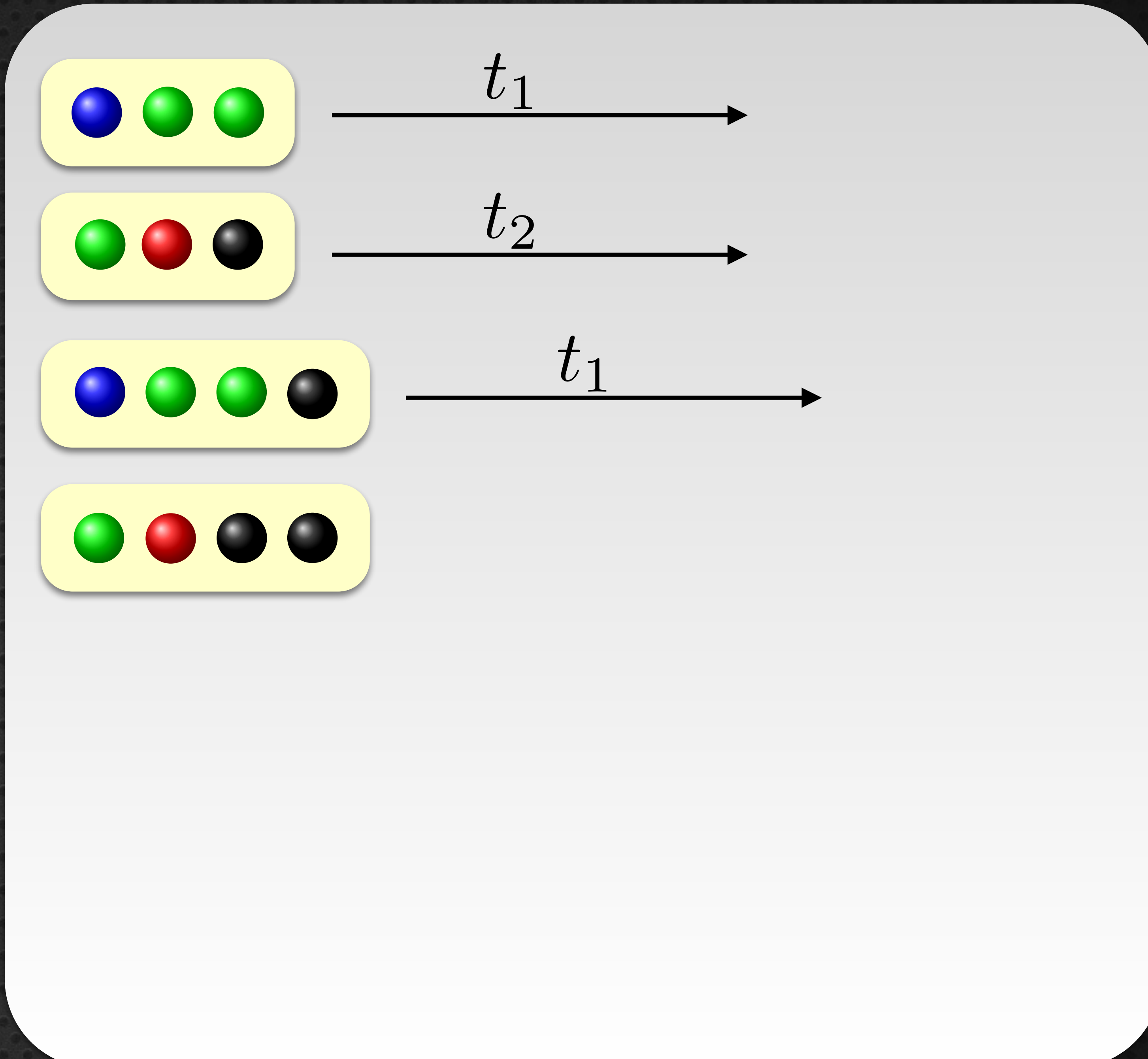
Transitions



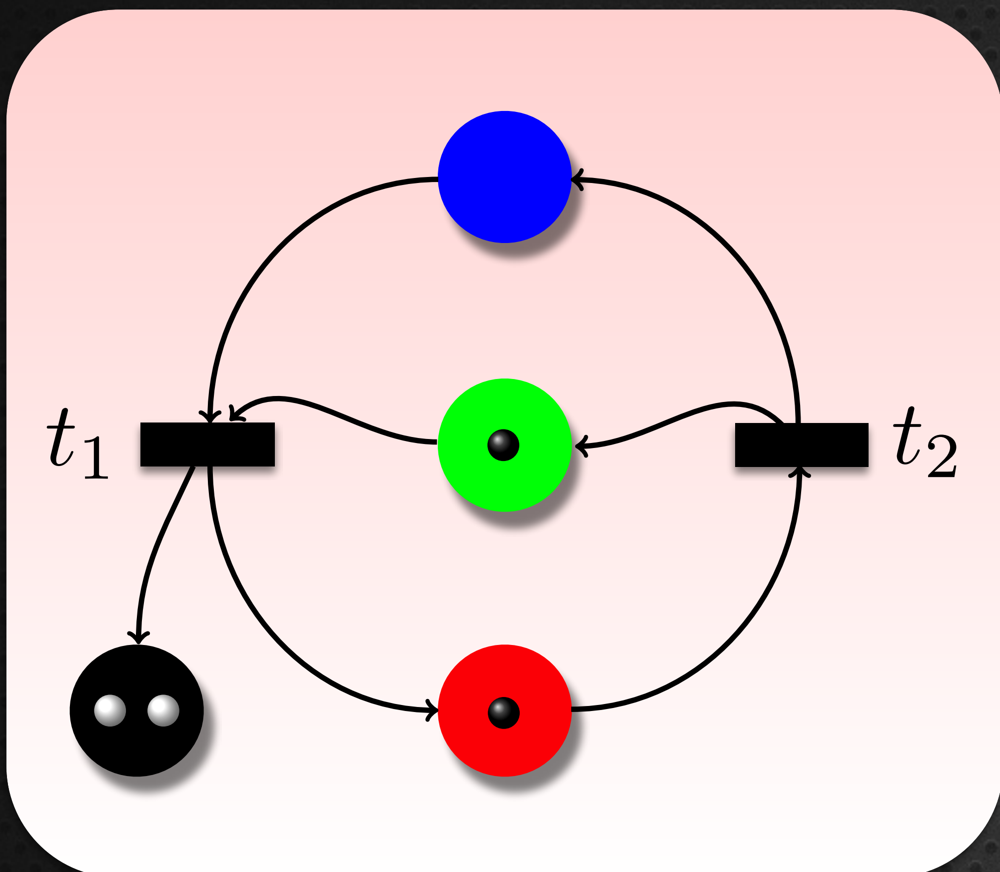


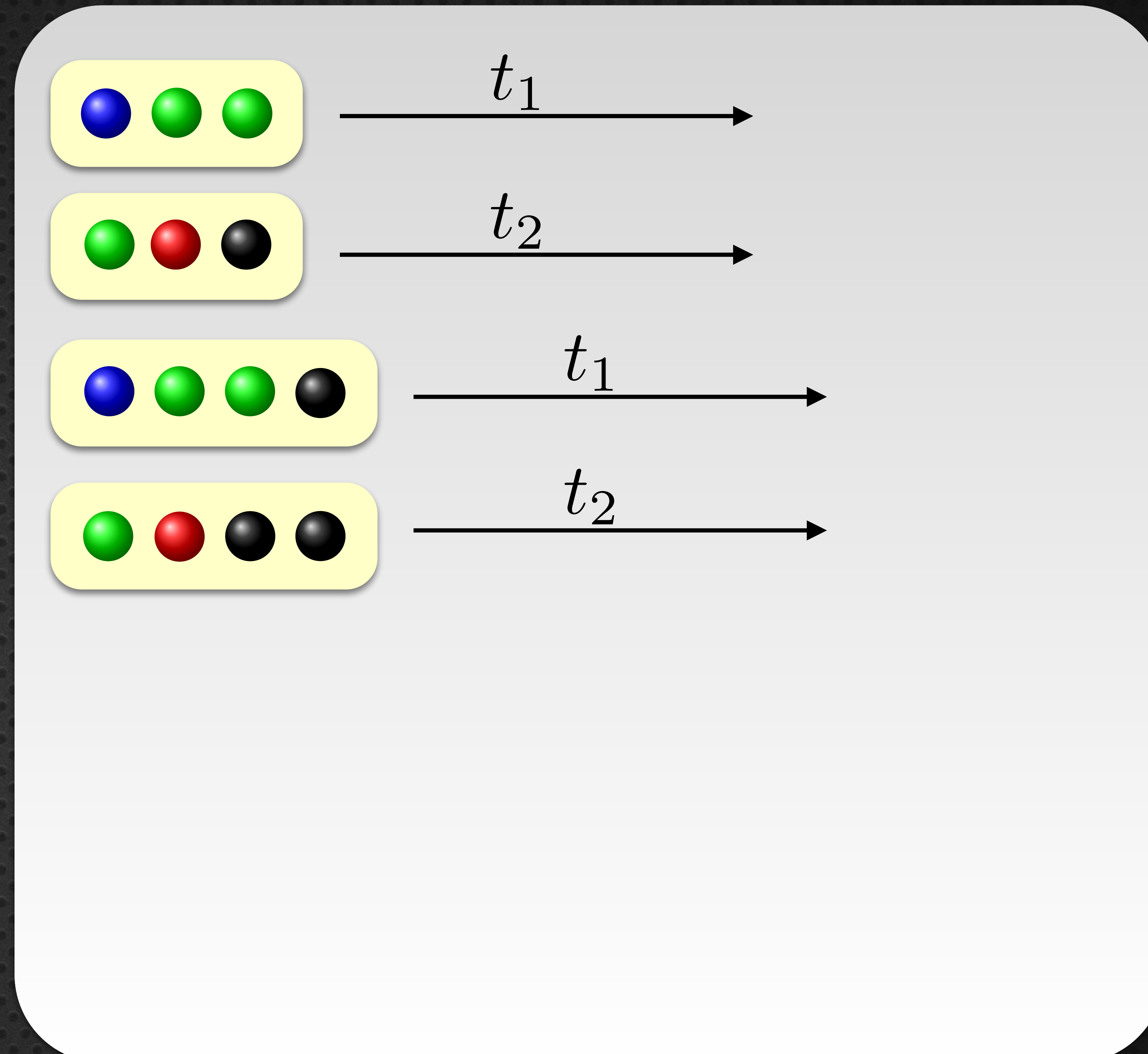
Transitions



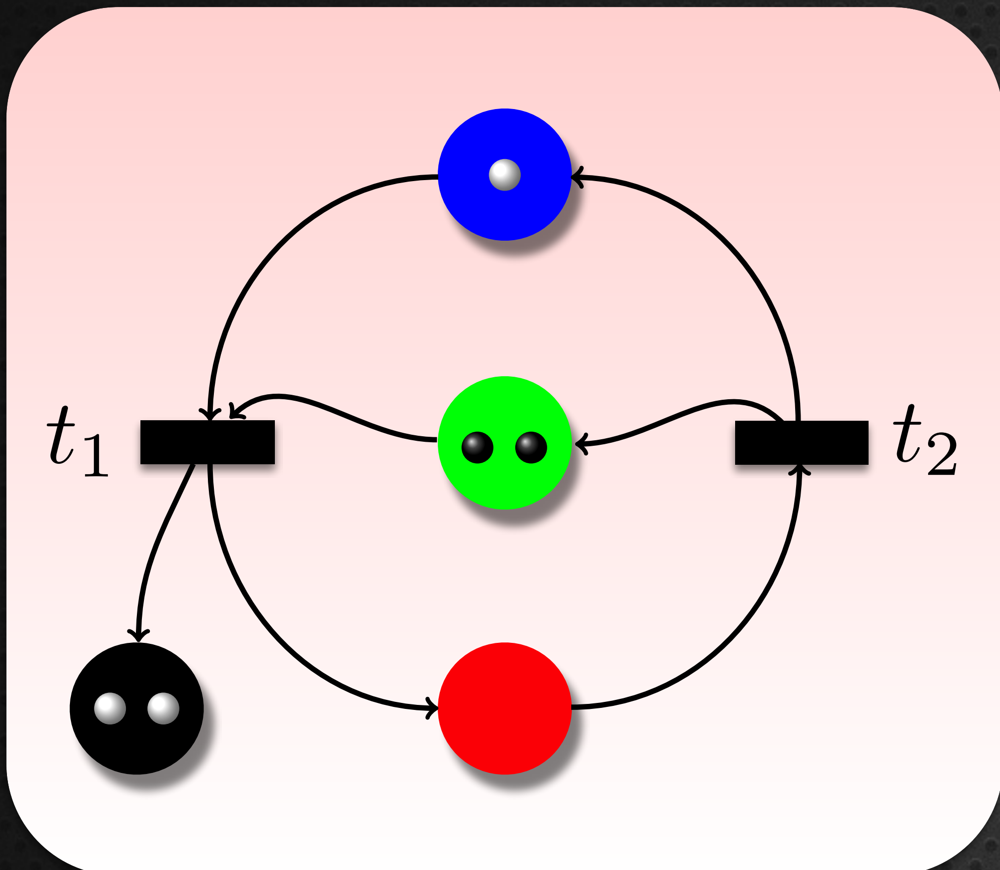


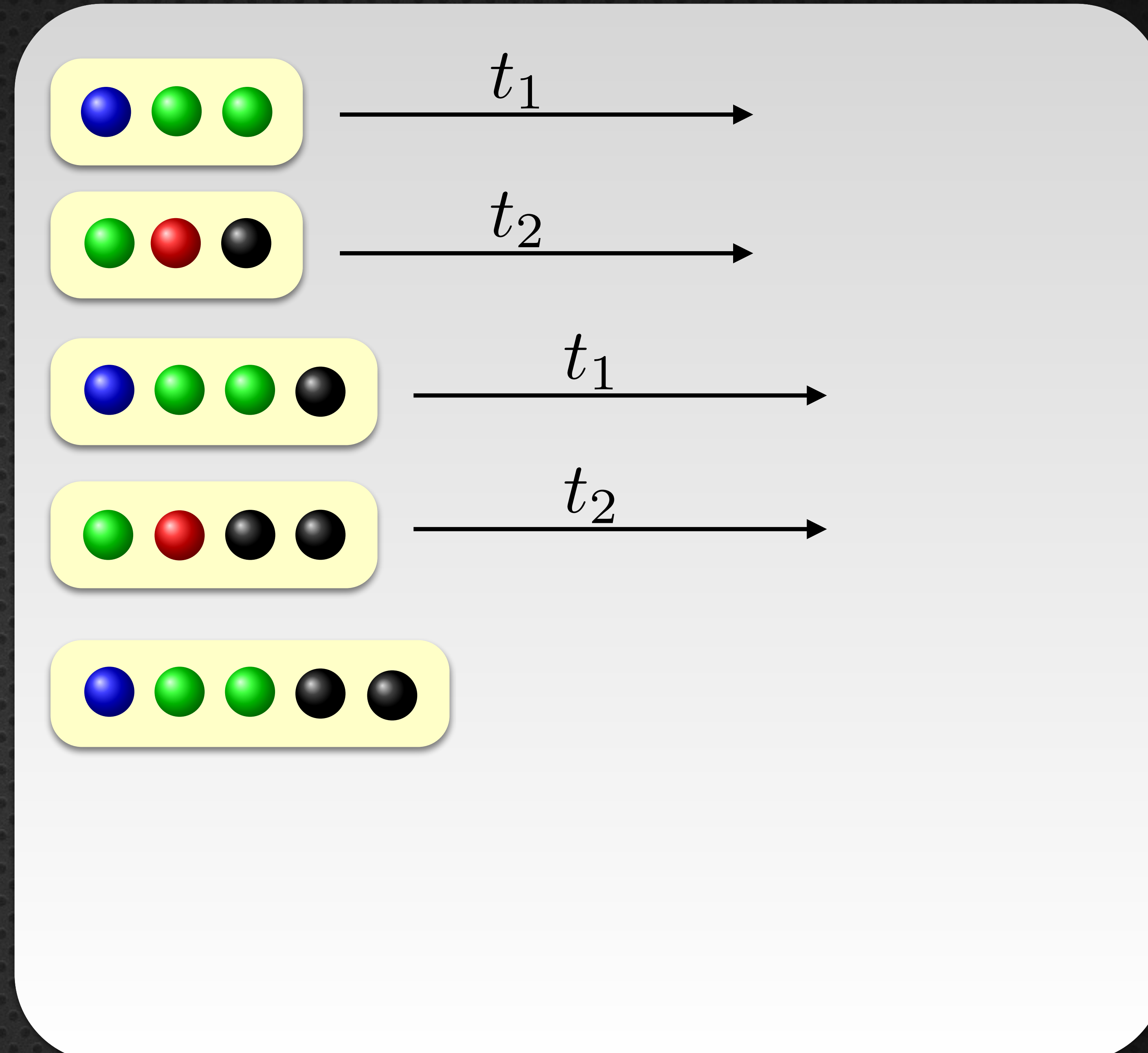
Transitions



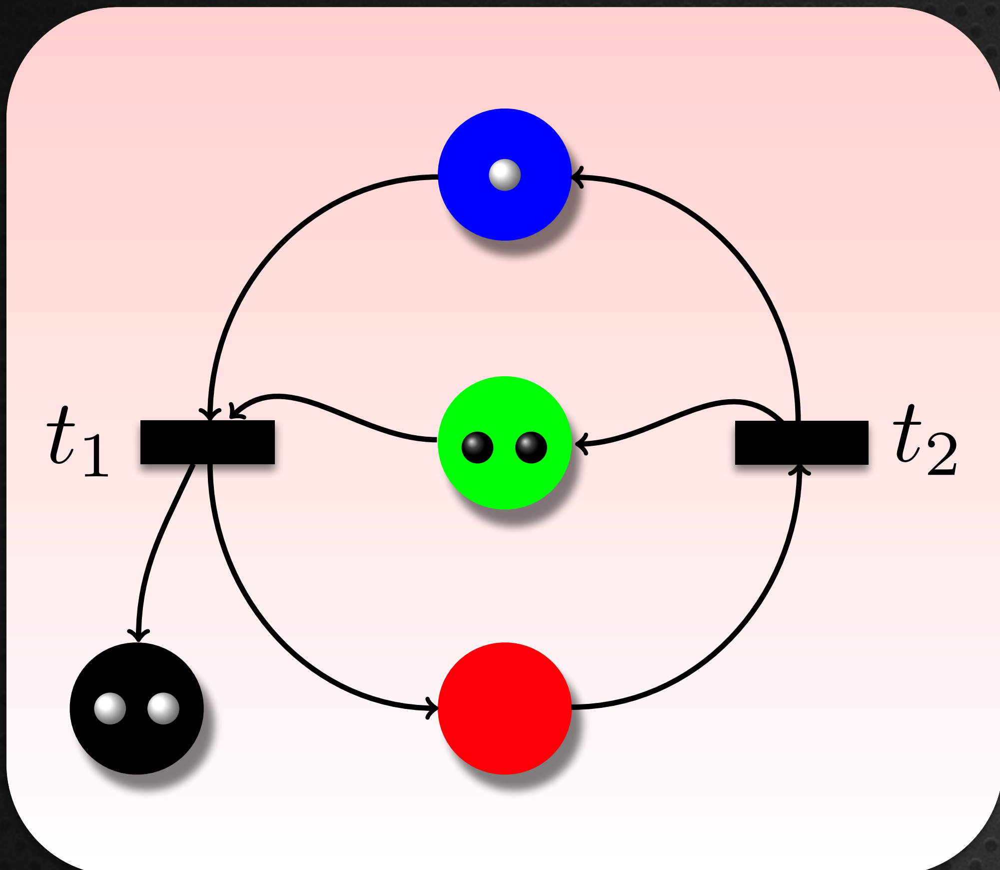


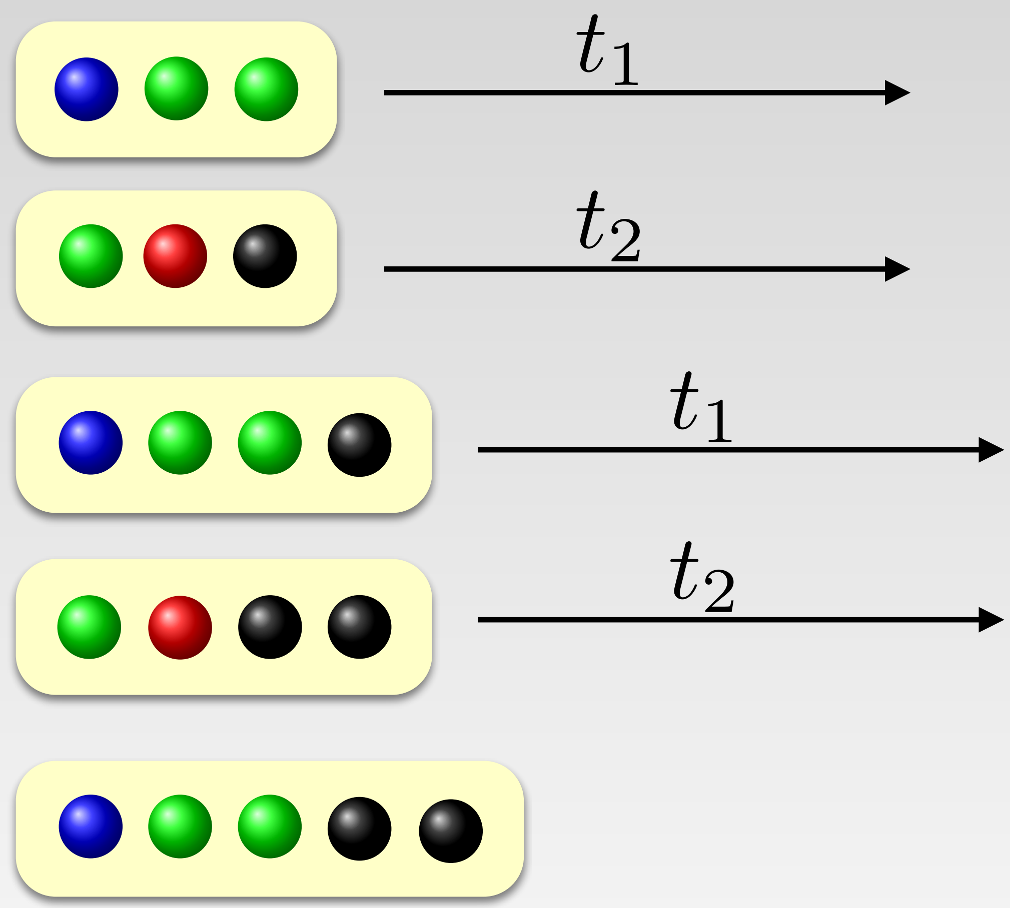
Transitions



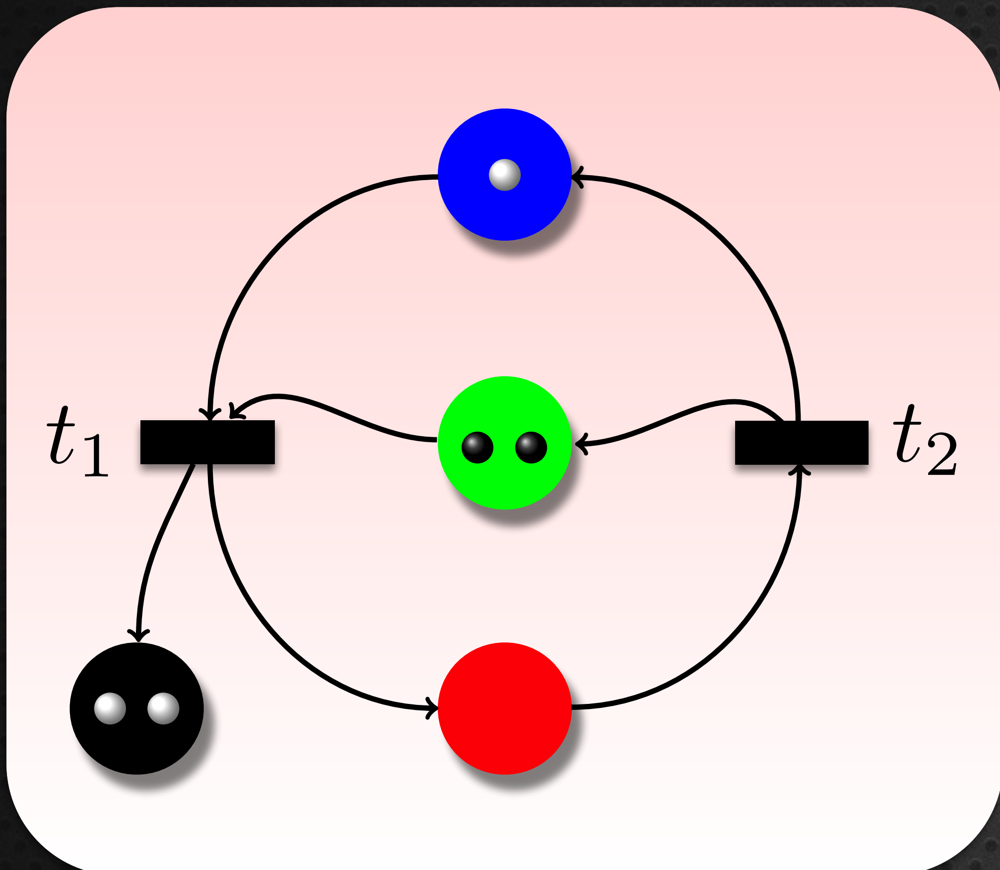


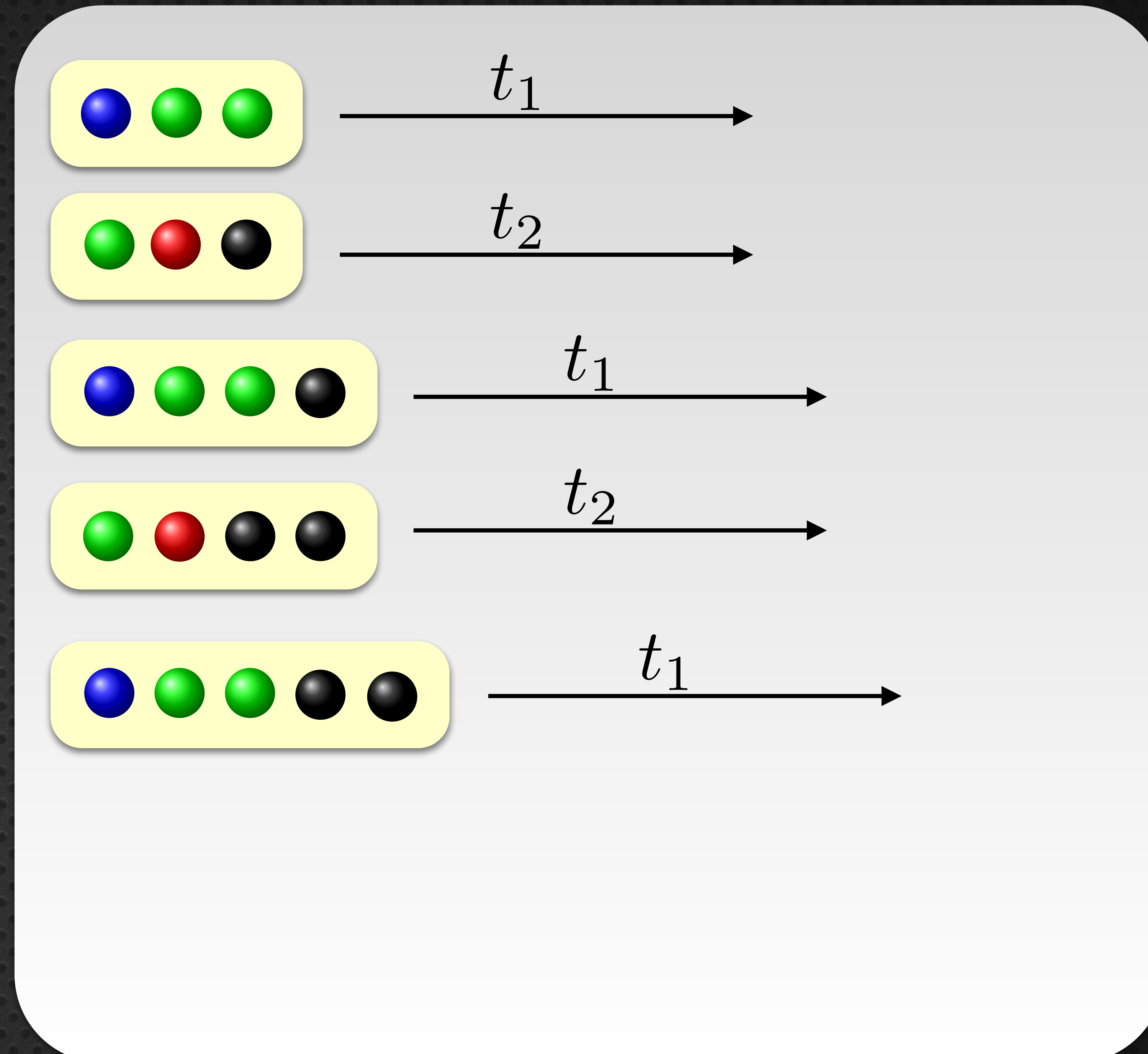
Transitions



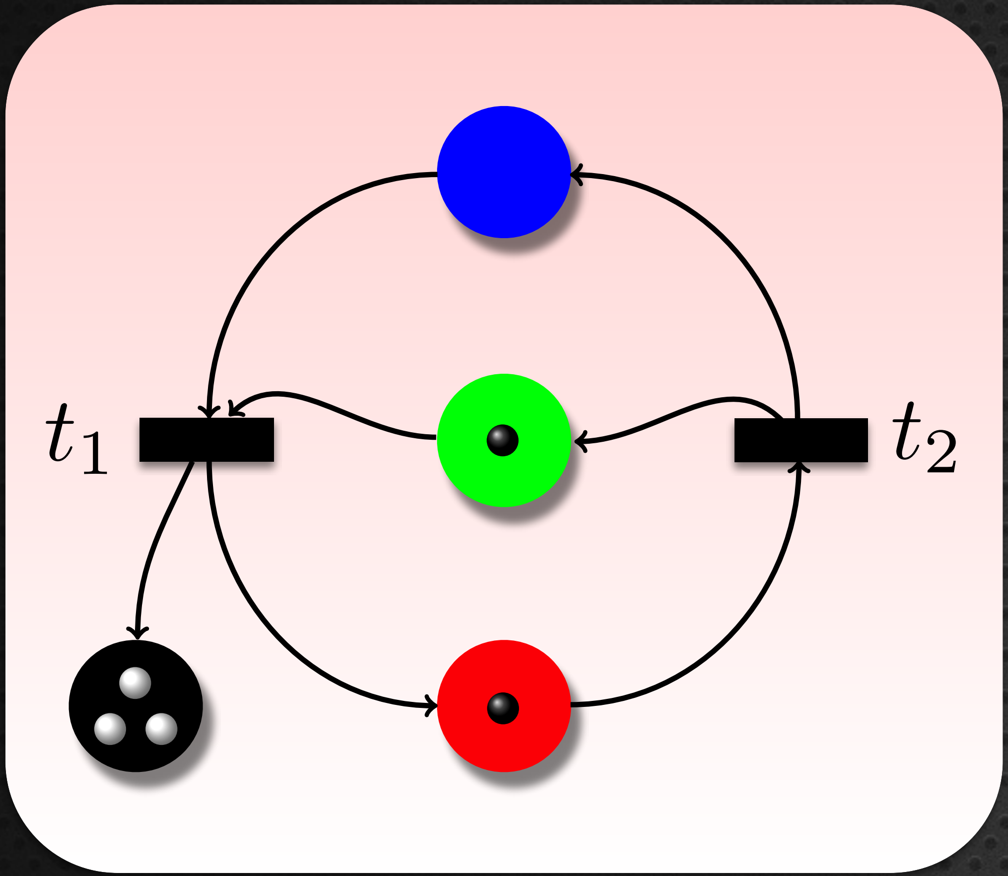


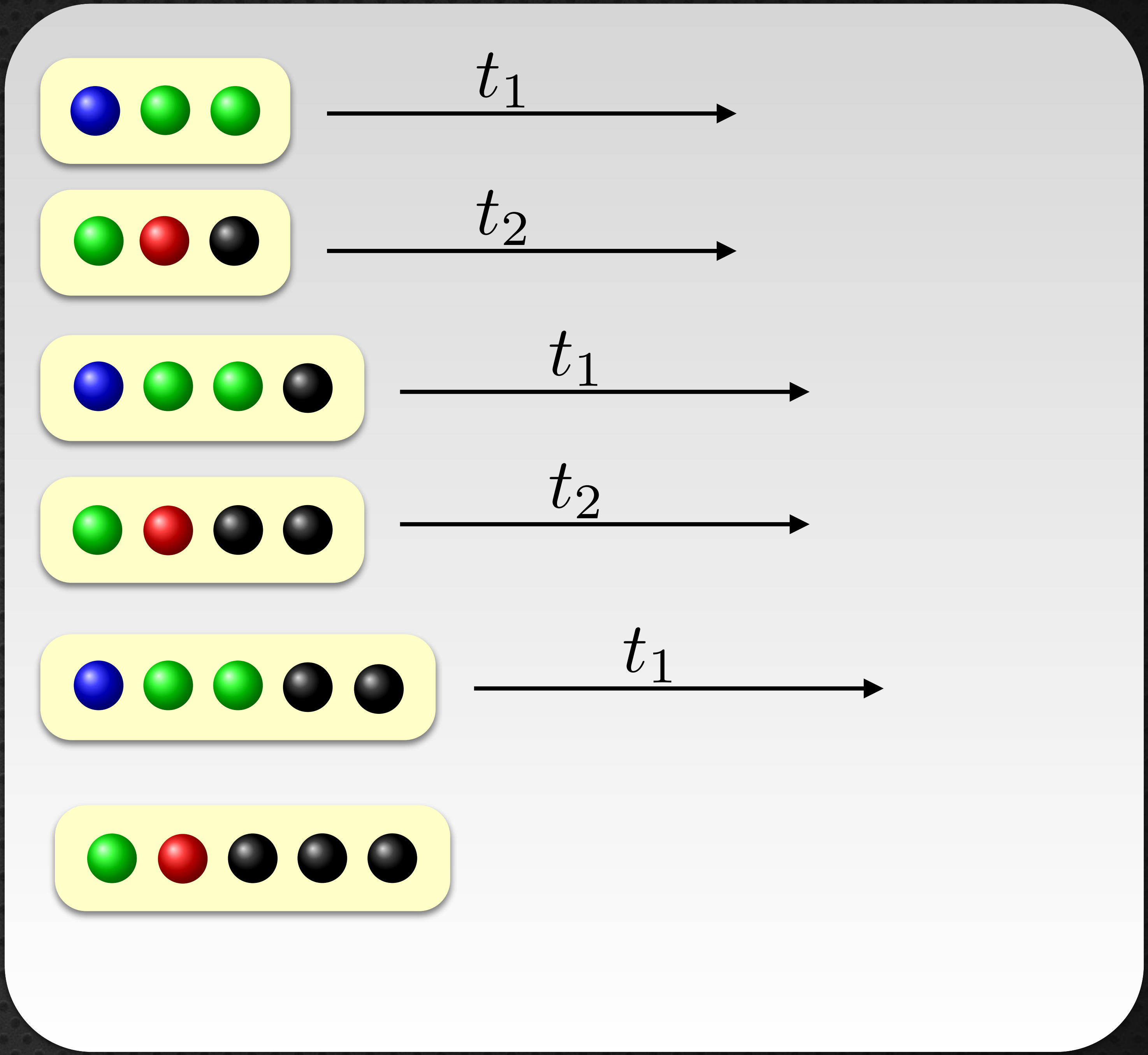
Transitions



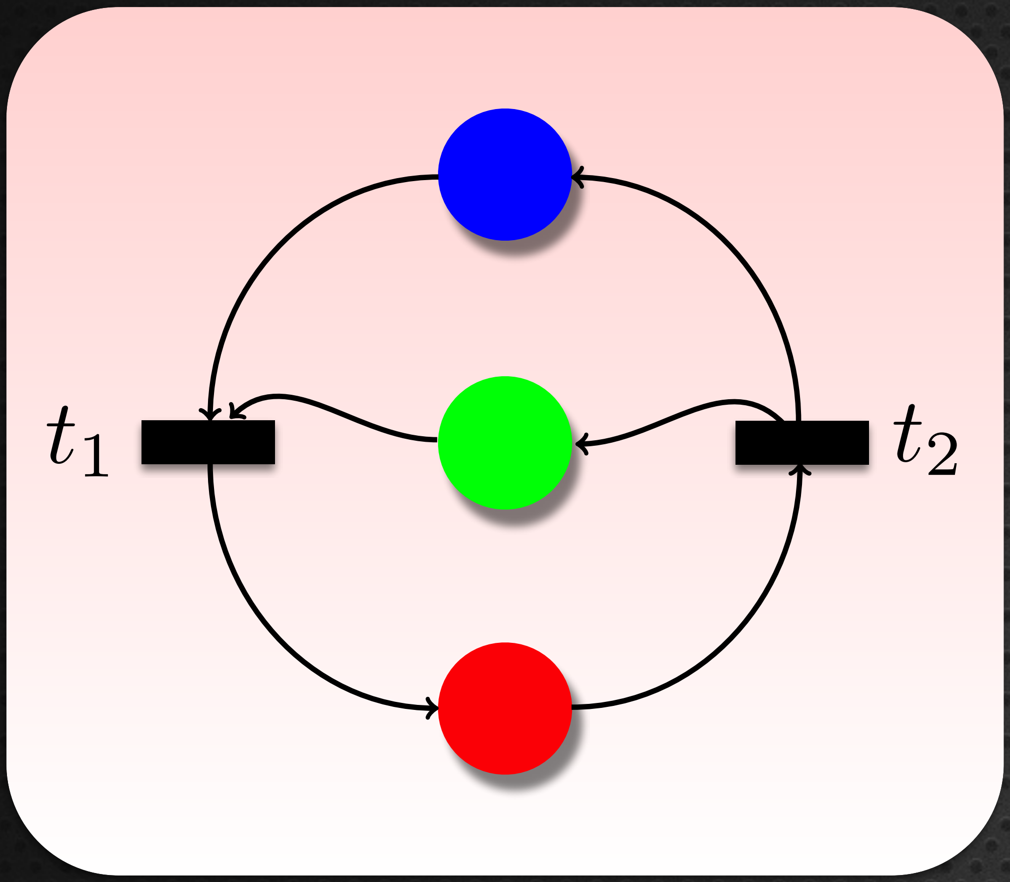


Transitions

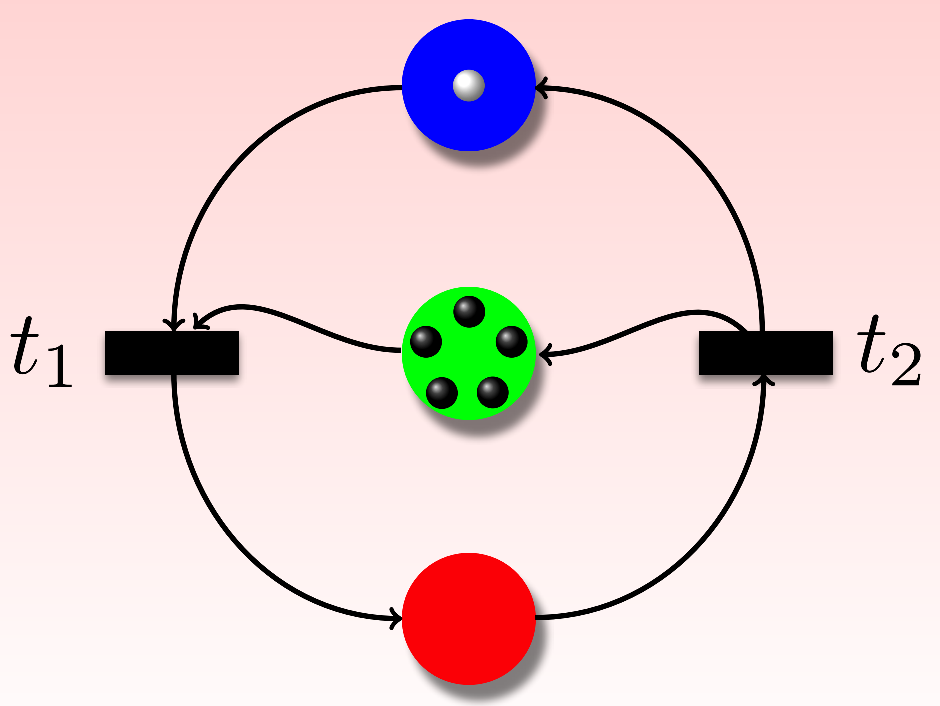




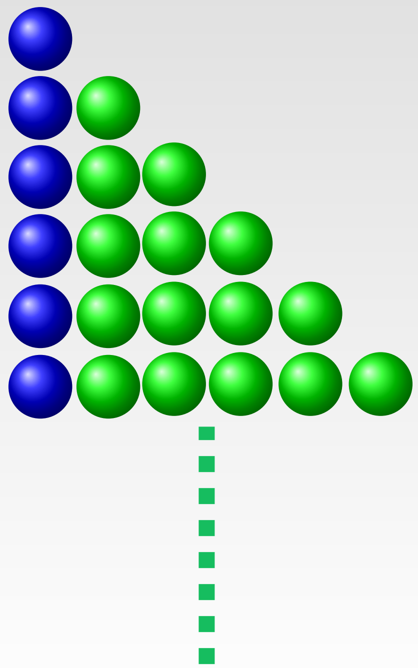
Safety Properties



Safety Properties

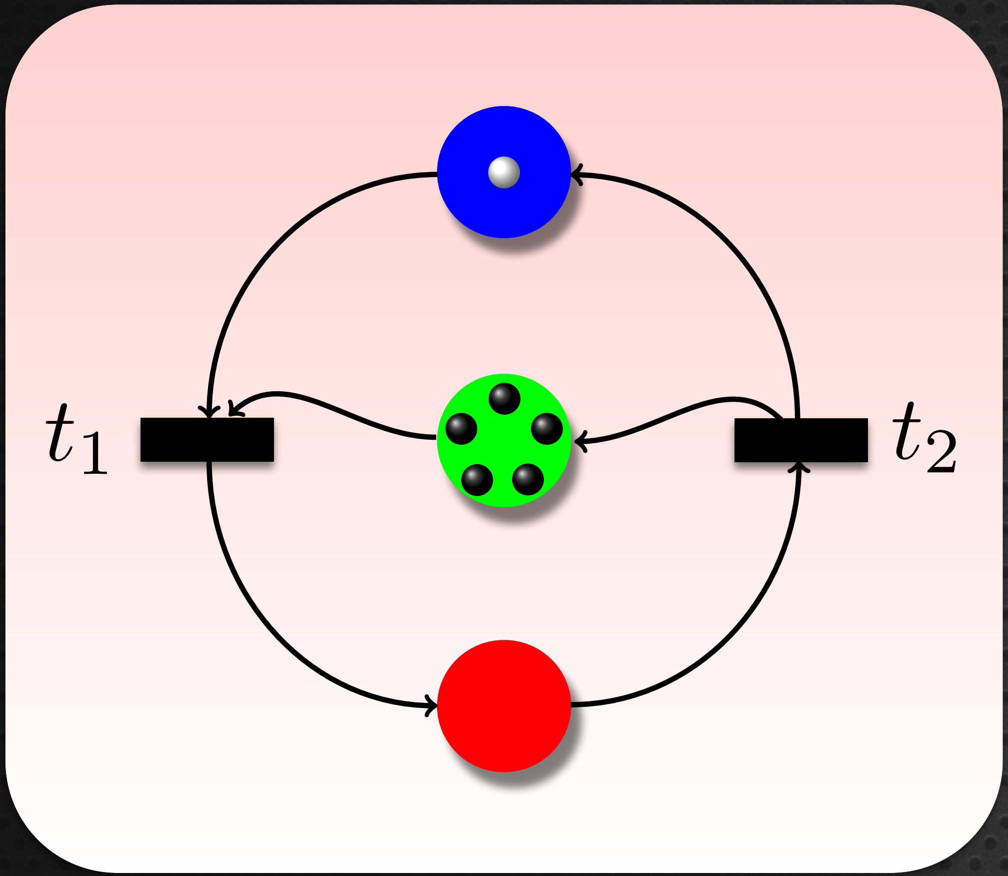


Initial Markings (*Init*)

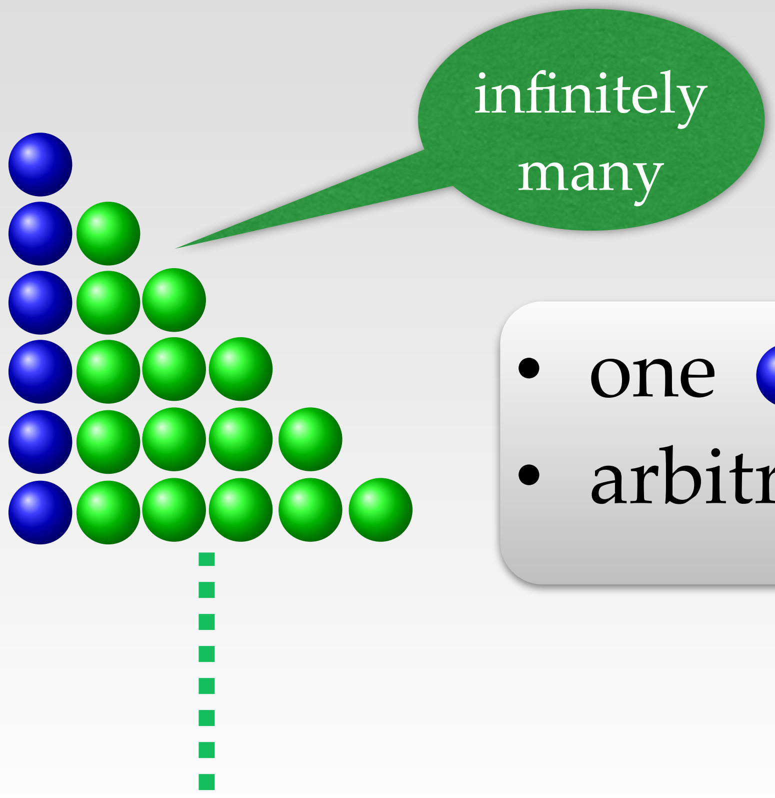


- one
- arbitrarily many

Safety Properties

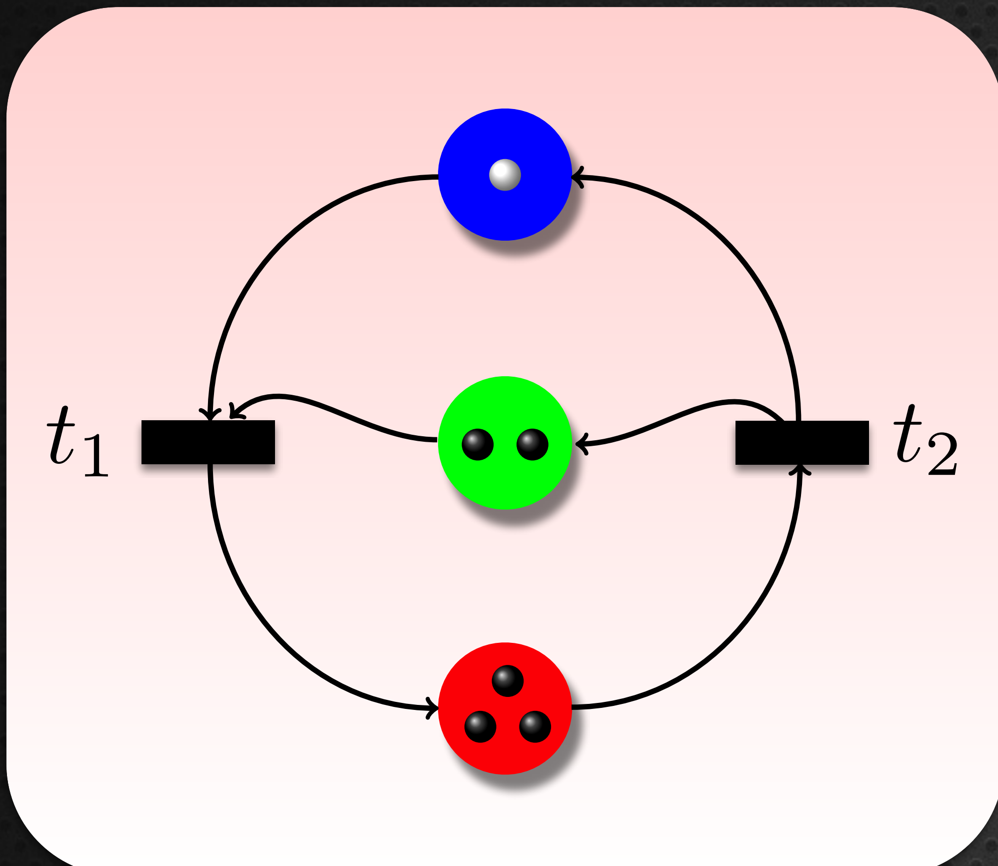


Initial Markings (*Init*)

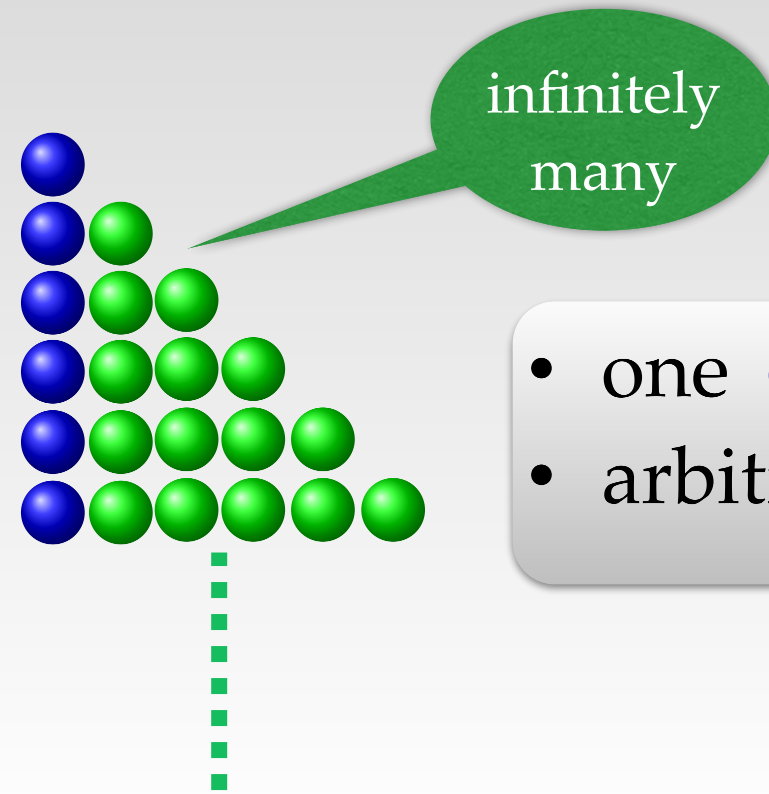


- one
- arbitrarily many

Safety Properties

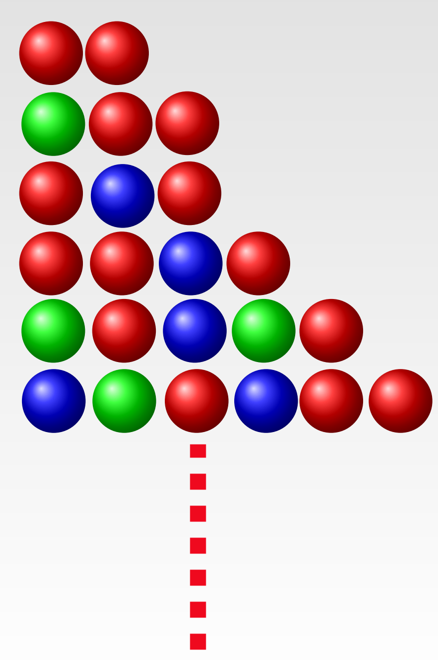


Initial Markings (*Init*)



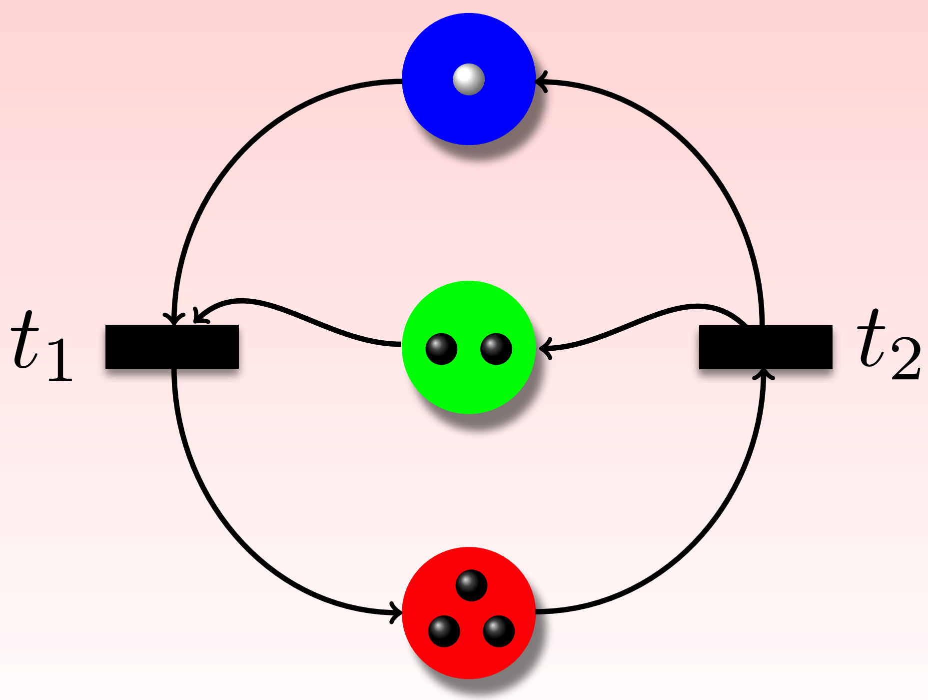
- one
- arbitrarily many

Bad Markings (*Bad*)

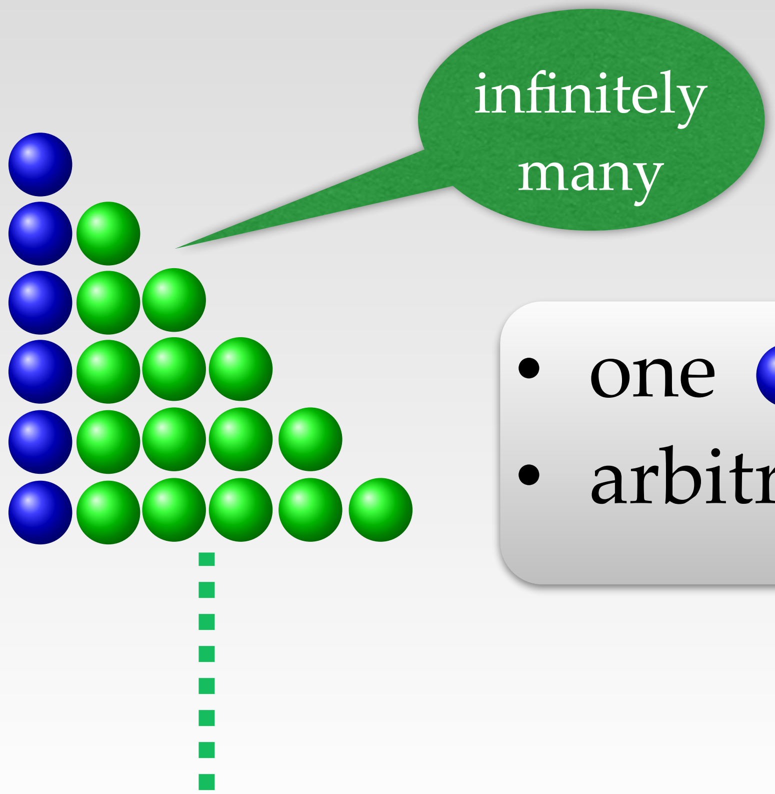


- at least two

Safety Properties

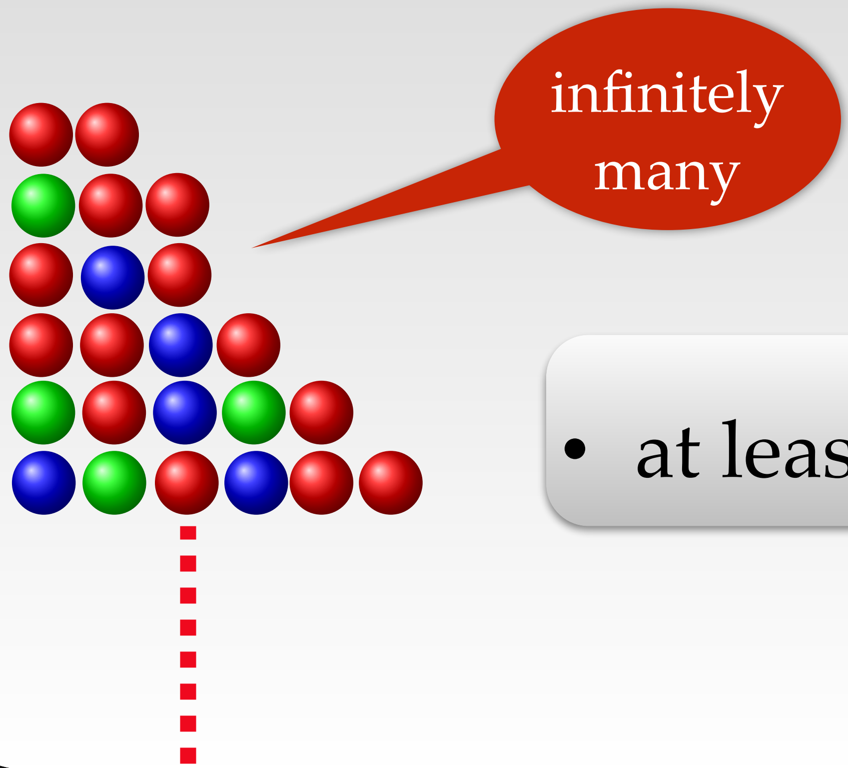


Initial Markings (*Init*)



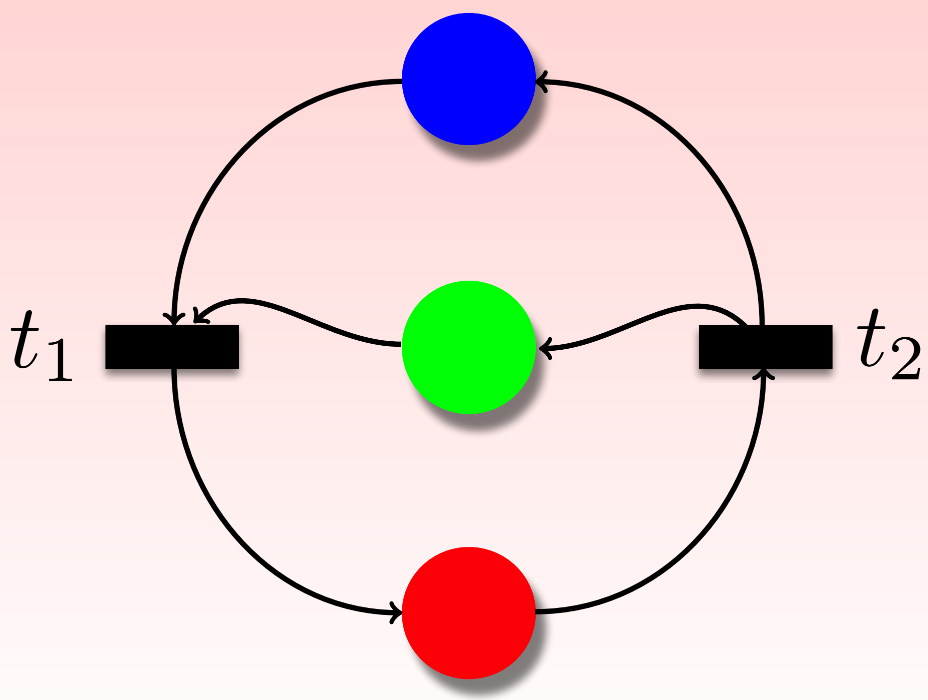
- one
- arbitrarily many

Bad Markings (*Bad*)

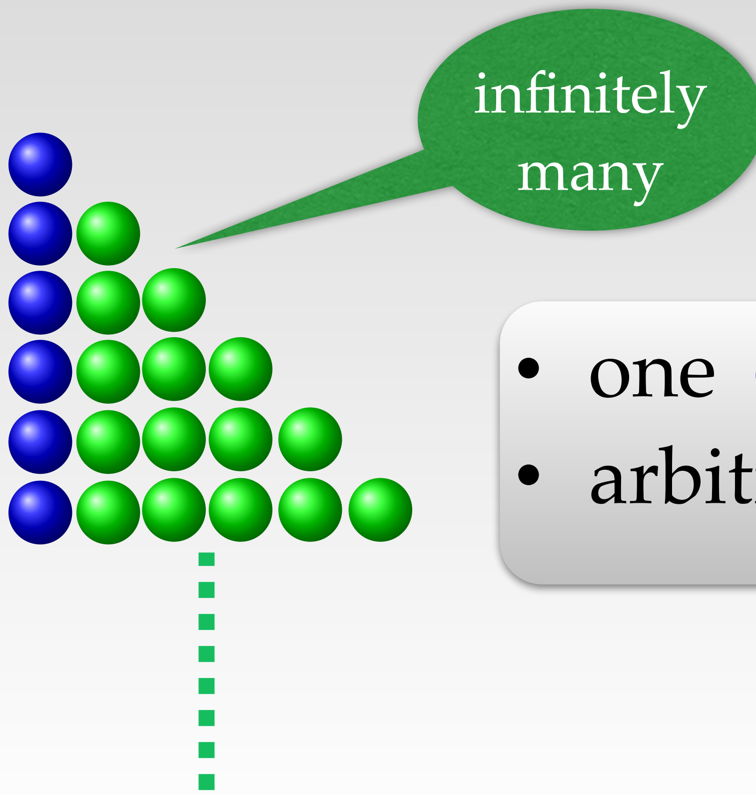


- at least two

Safety Properties

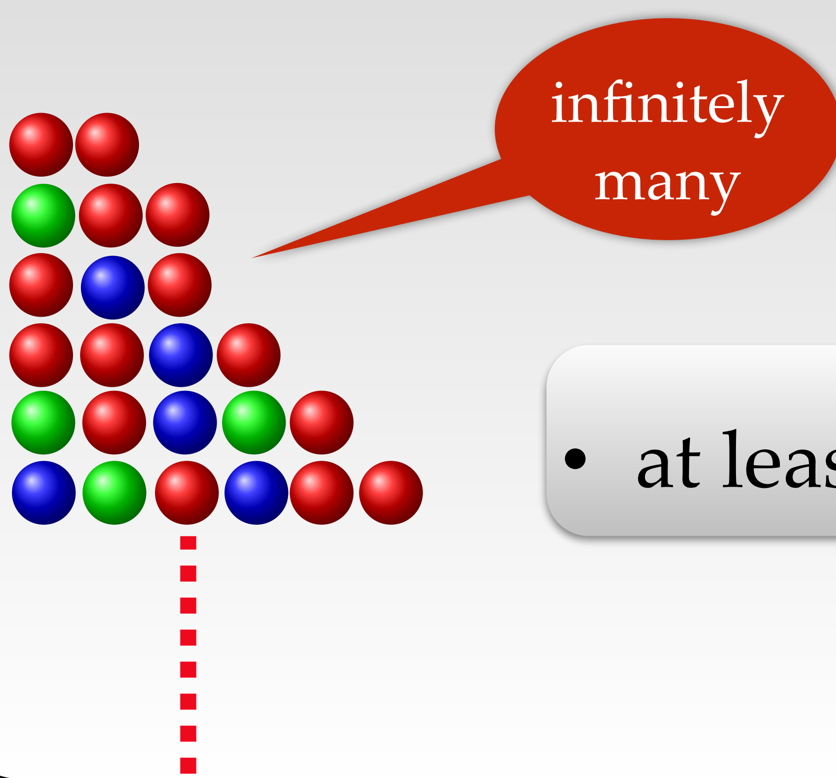


Initial Markings (*Init*)



- one
- arbitrarily many

Bad Markings (*Bad*)

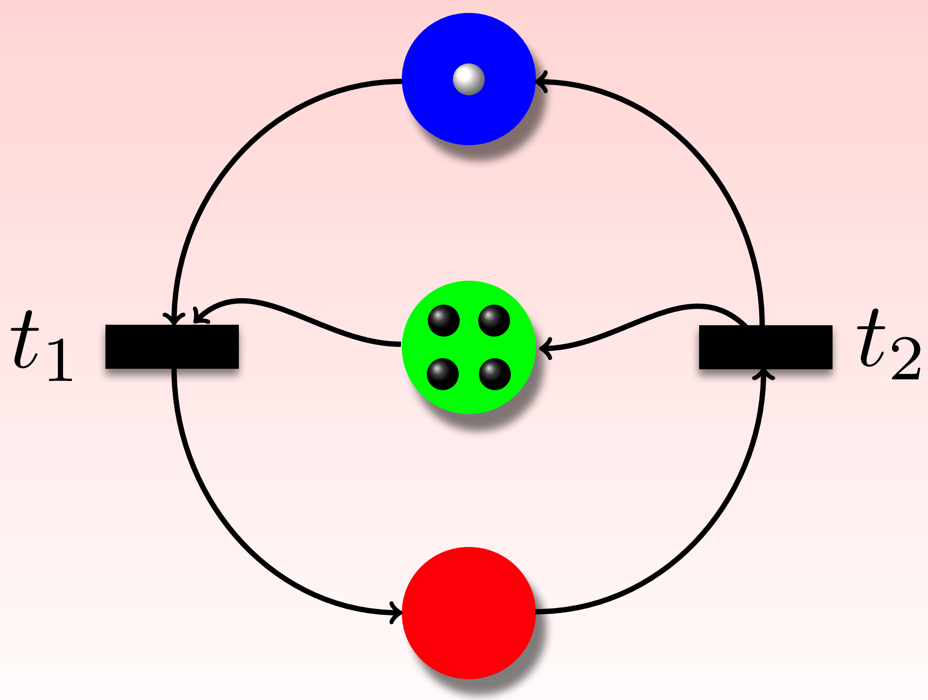


- at least two

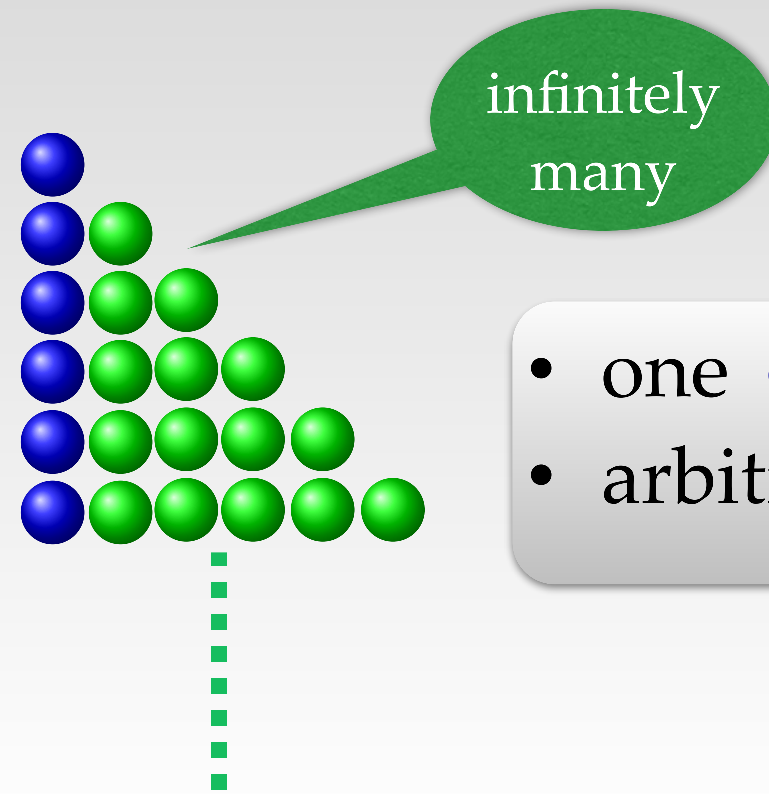
Safety Property

$$\mathit{Init} \xrightarrow{*} \mathit{Bad}\ ?$$

Safety Properties

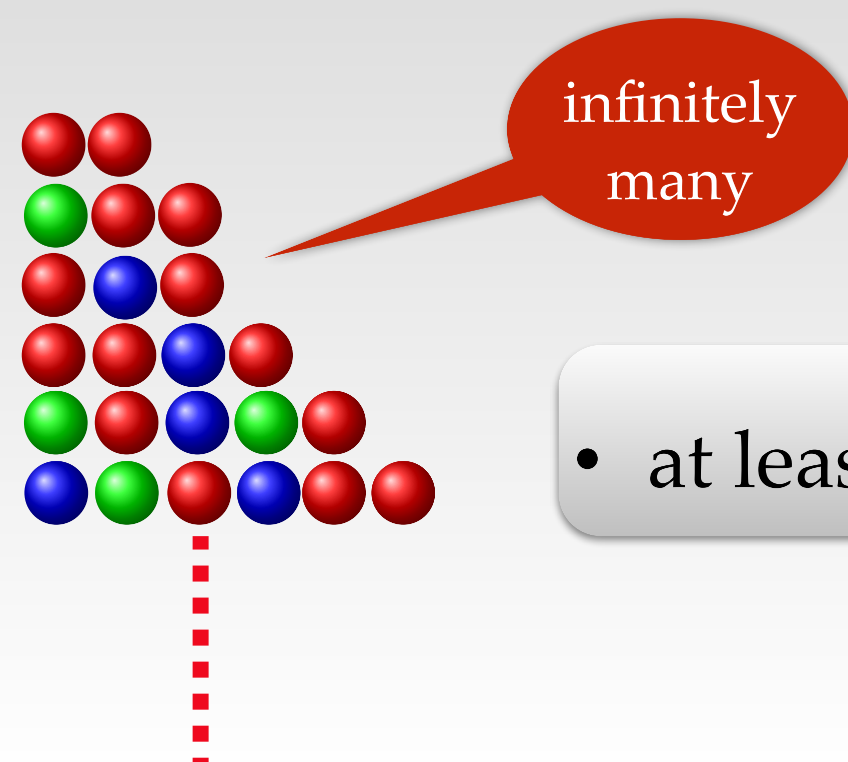


Initial Markings (*Init*)



- one
- arbitrarily many

Bad Markings (*Bad*)

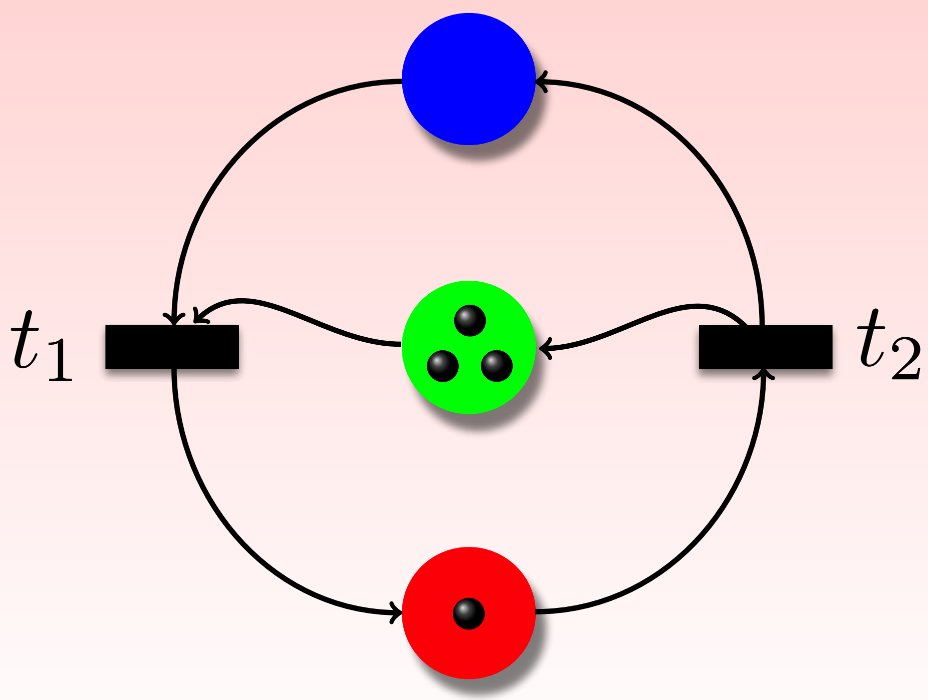


- at least two

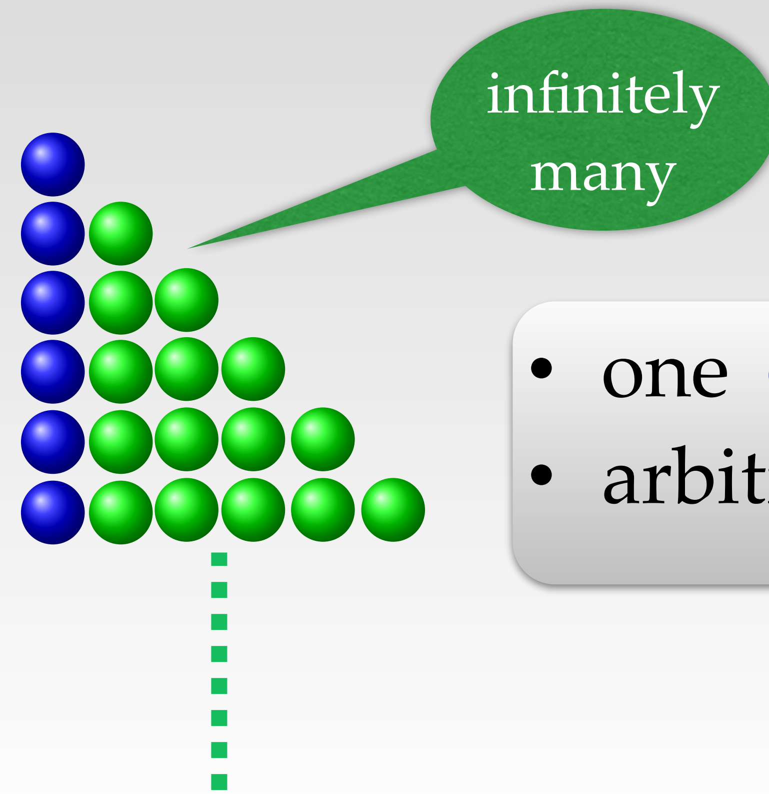
Safety Property

$$\mathit{Init} \xrightarrow{*} \mathit{Bad} \,?$$

Safety Properties

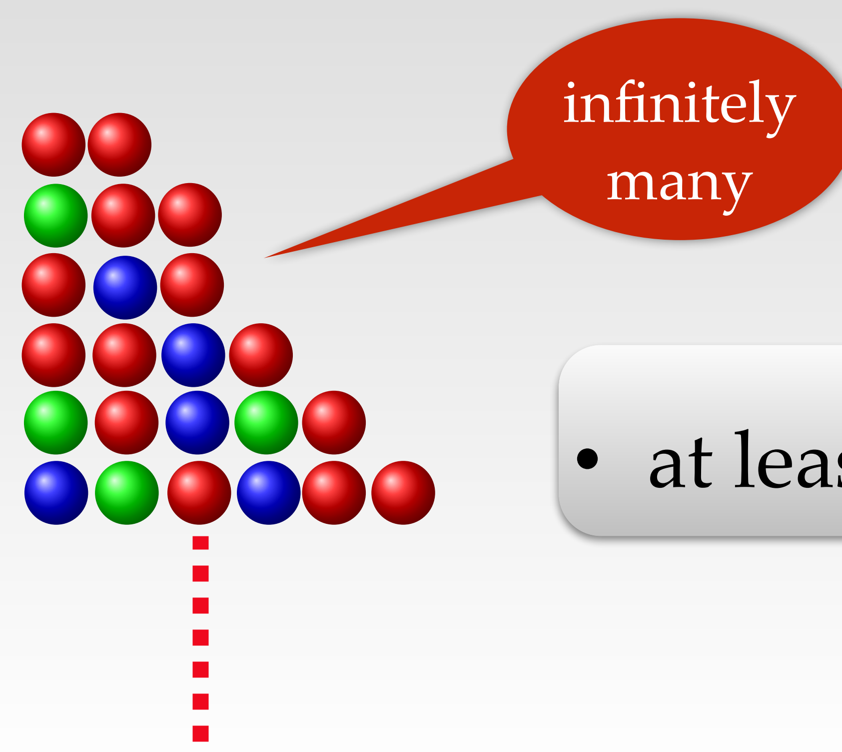


Initial Markings (*Init*)



- one (blue token)
- arbitrarily many (green tokens)

Bad Markings (*Bad*)

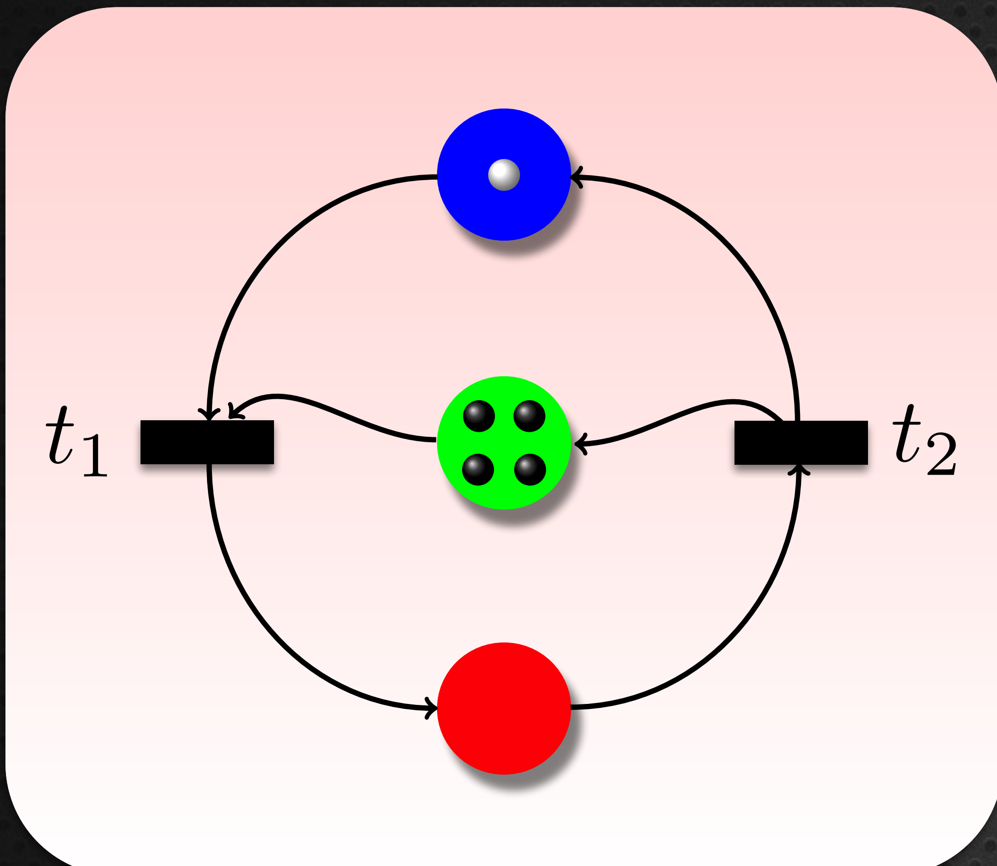


- at least two (red tokens)

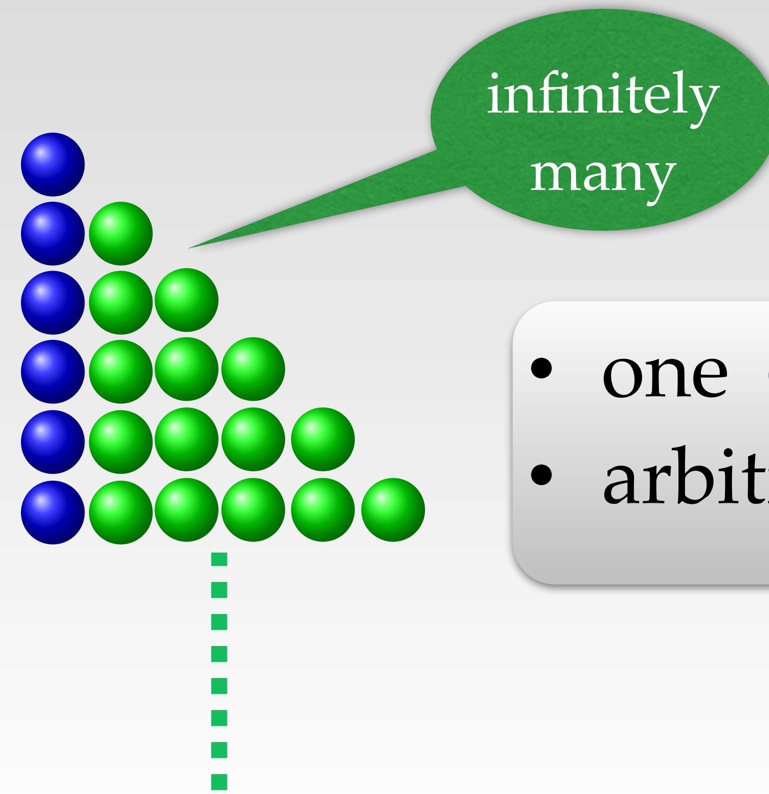
Safety Property

$$\mathit{Init} \xrightarrow{*} \mathit{Bad} \; ?$$

Safety Properties

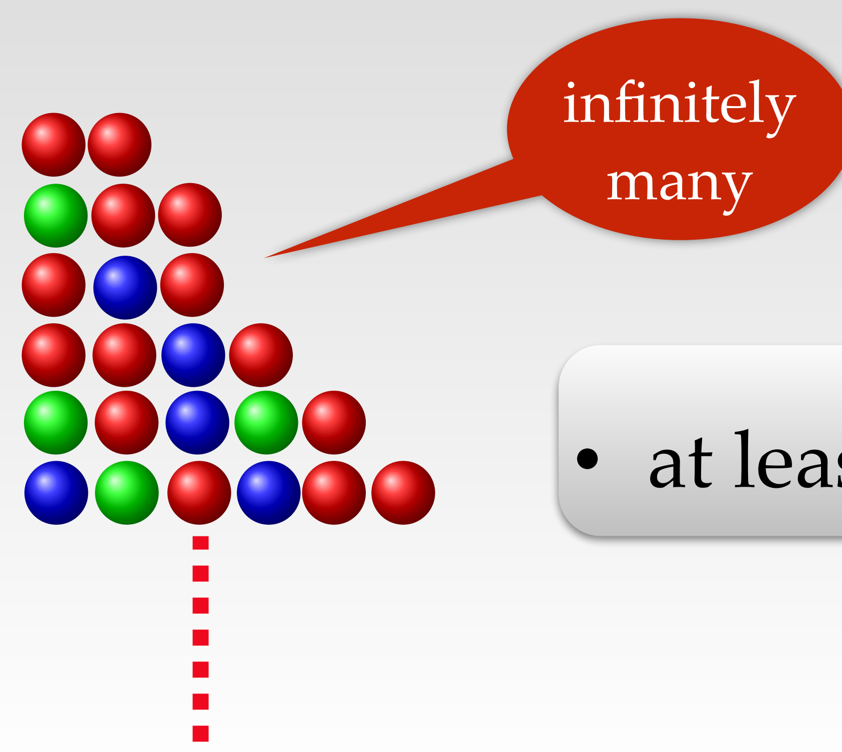


Initial Markings (*Init*)



- one
- arbitrarily many

Bad Markings (*Bad*)

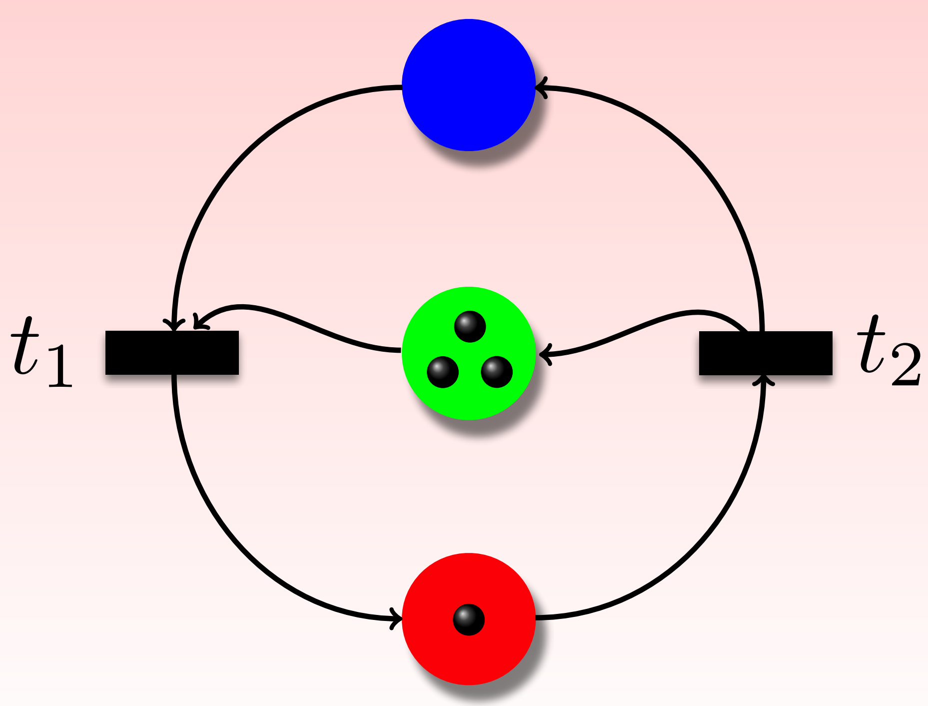


- at least two

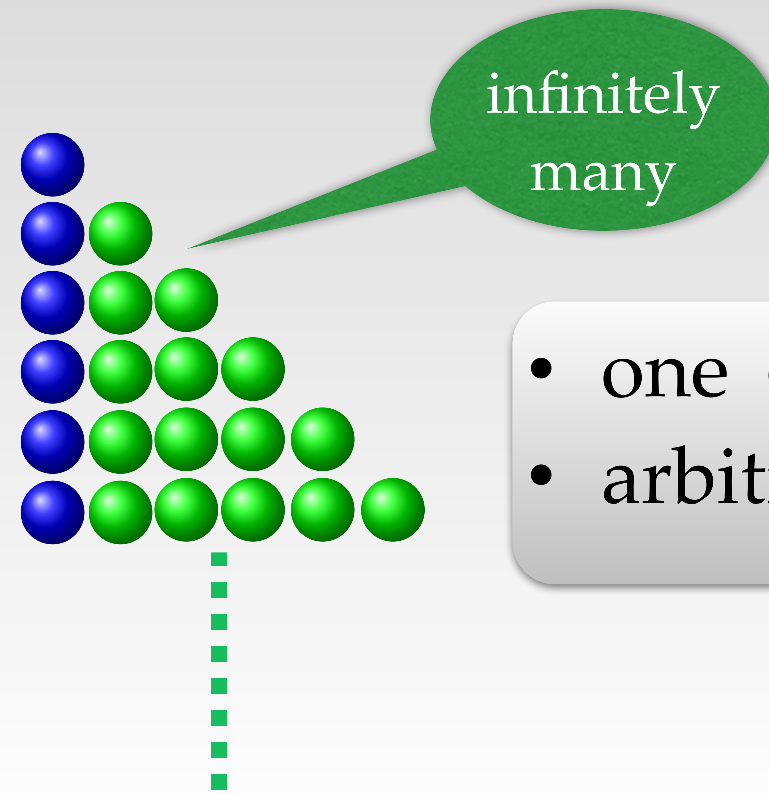
Safety Property

$$\mathit{Init} \xrightarrow{\;*\;} \mathit{Bad}\,?$$

Safety Properties

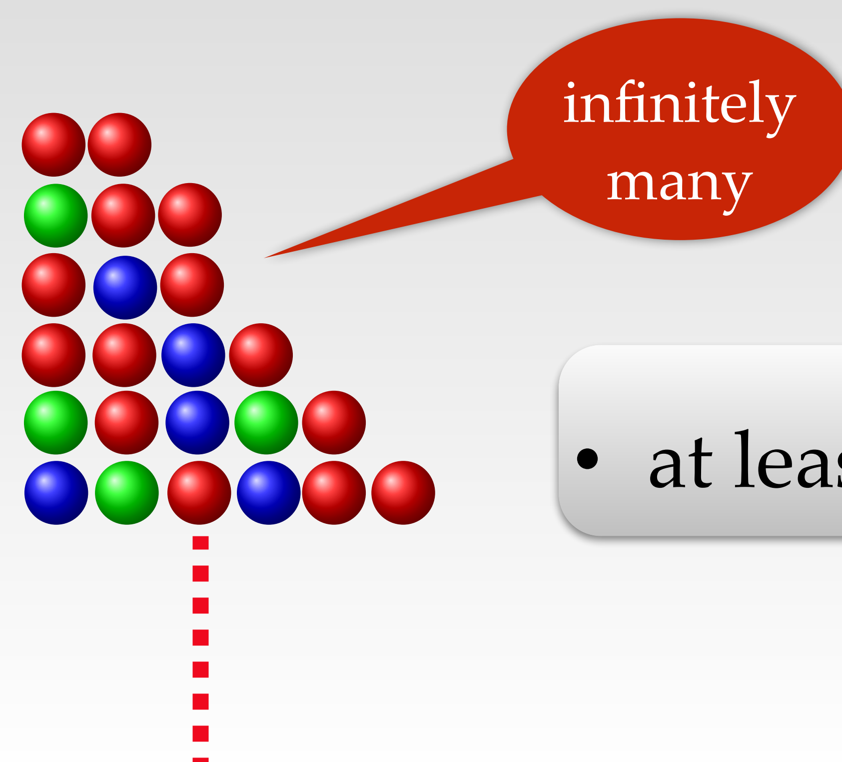


Initial Markings (*Init*)



- one
- arbitrarily many

Bad Markings (*Bad*)

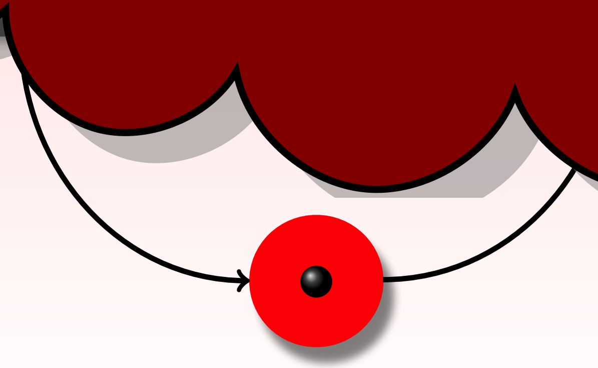


- at least two

Safety Property

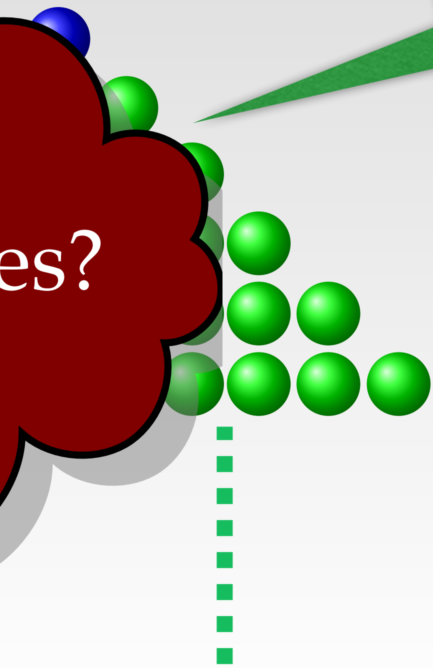
$$\mathit{Init} \xrightarrow{*} \mathit{Bad}\ ?$$

Safety Properties



How to check safety properties?

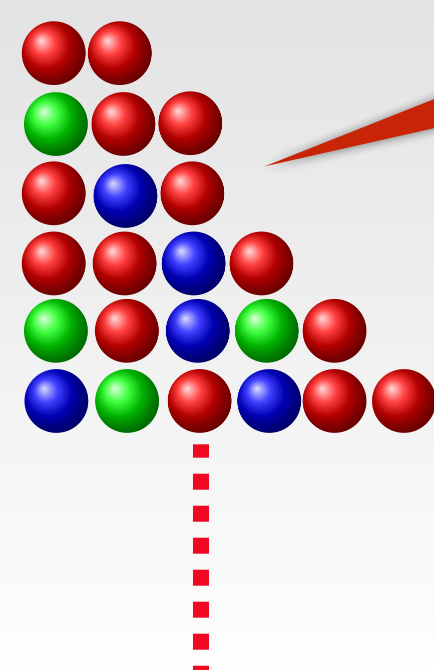
Initial Markings (*Init*)



infinitely many

- one
- arbitrarily many

Bad Markings (*Bad*)



infinitely many

- at least two

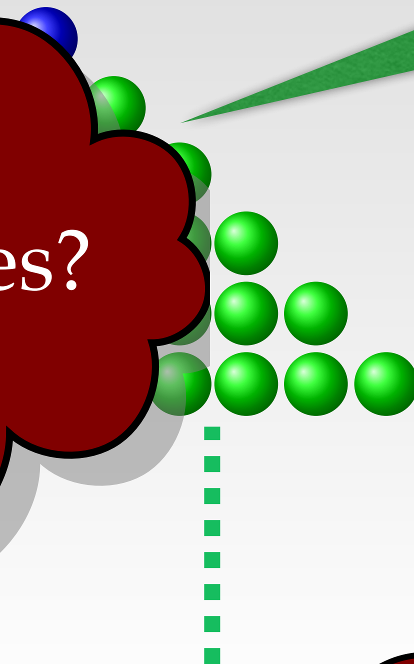
Safety Property

$$\mathit{Init} \xrightarrow{*} \mathit{Bad}\ ?$$

Safety Properties

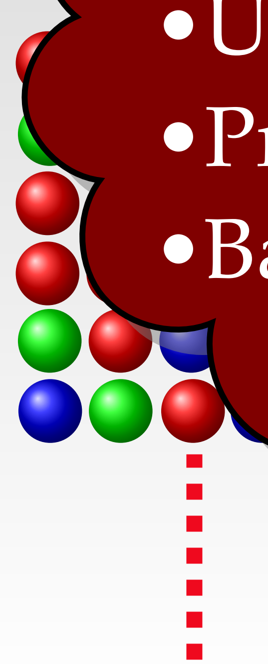
How to check safety properties?

Initial Markings (*Init*)



- one
- arbitrarily many

Bad



- Ordering
- Monotonicity
- Upward Closed sets
- Predecessors
- Backward Reachability

Safety Property

$$\mathit{Init} \xrightarrow{*} \mathit{Bad}\ ?$$

Petri Nets

Model

Configurations

Transitions

Ordering

Monotonicity

Upward Closed Sets

Computing Predecessors

Backward Reachability

Petri Nets

Model

Configurations

Transitions

Ordering

Monotonicity

Upward Closed Sets

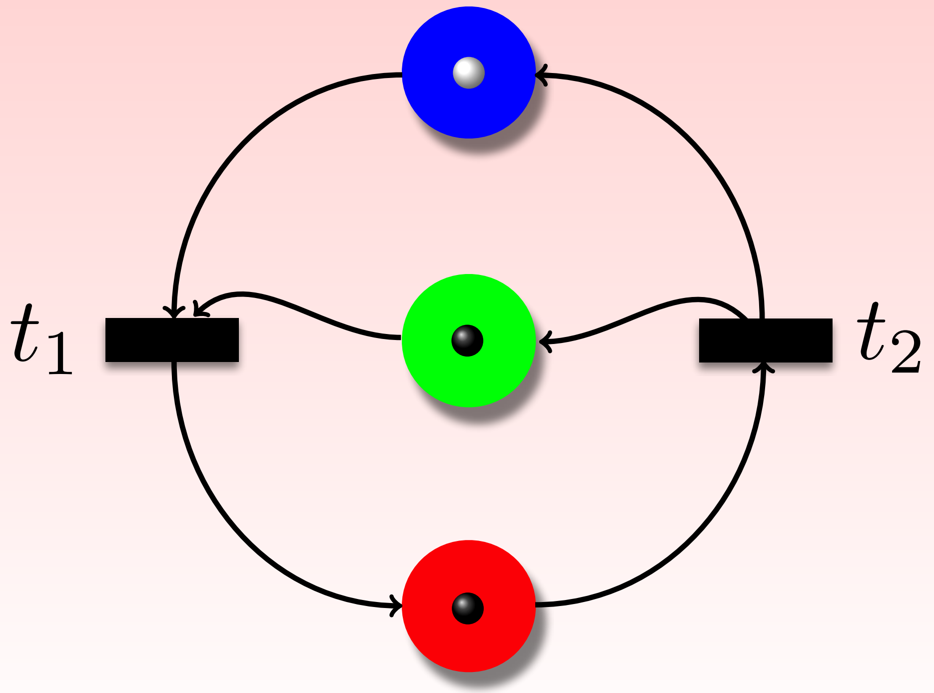
Computing Predecessors

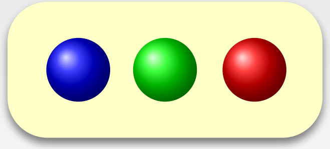
Backward Reachability

Ordering

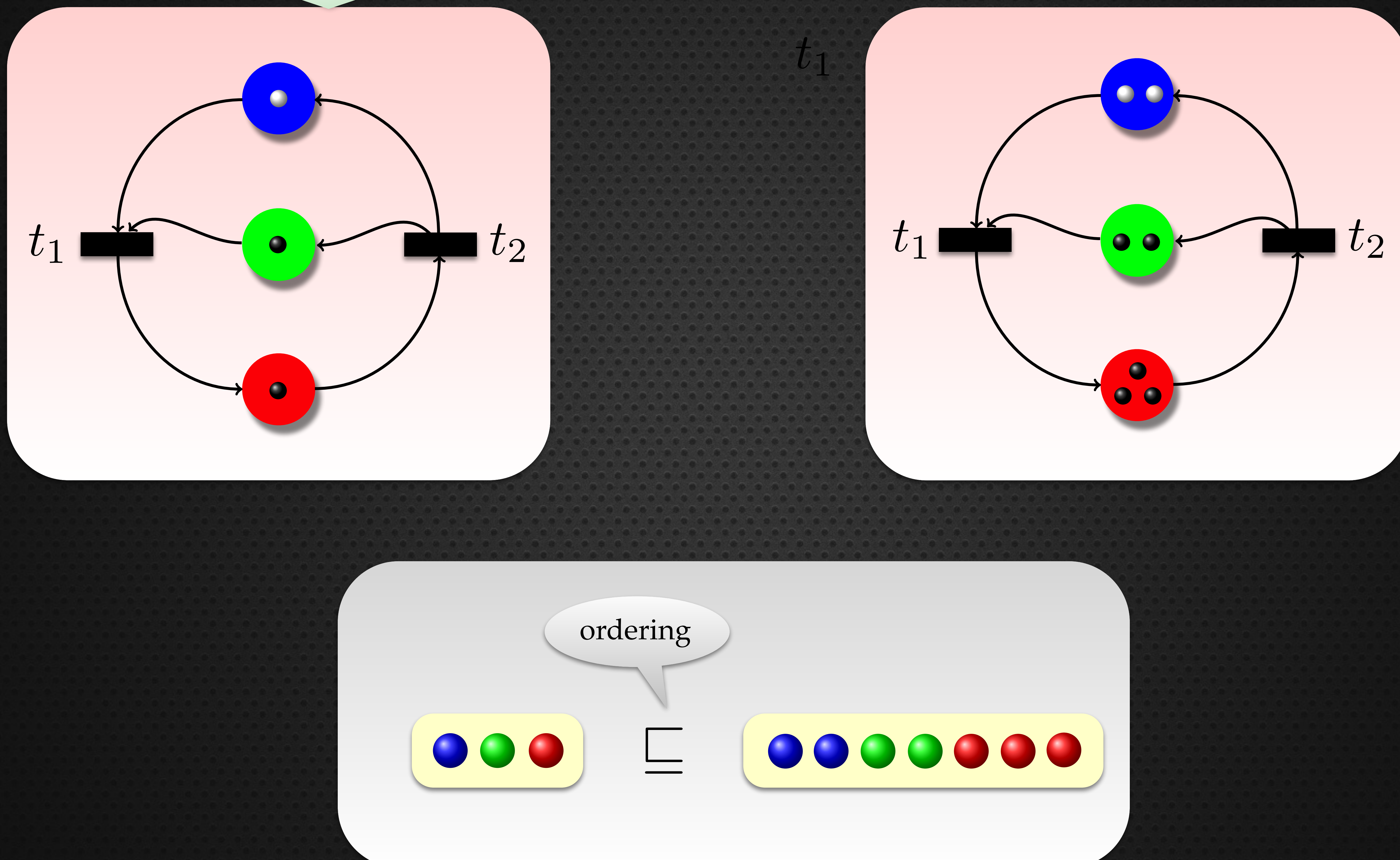
Pet

Ordering





Ordering



Petri Nets

Model ✓

Configurations ✓

Transitions ✓

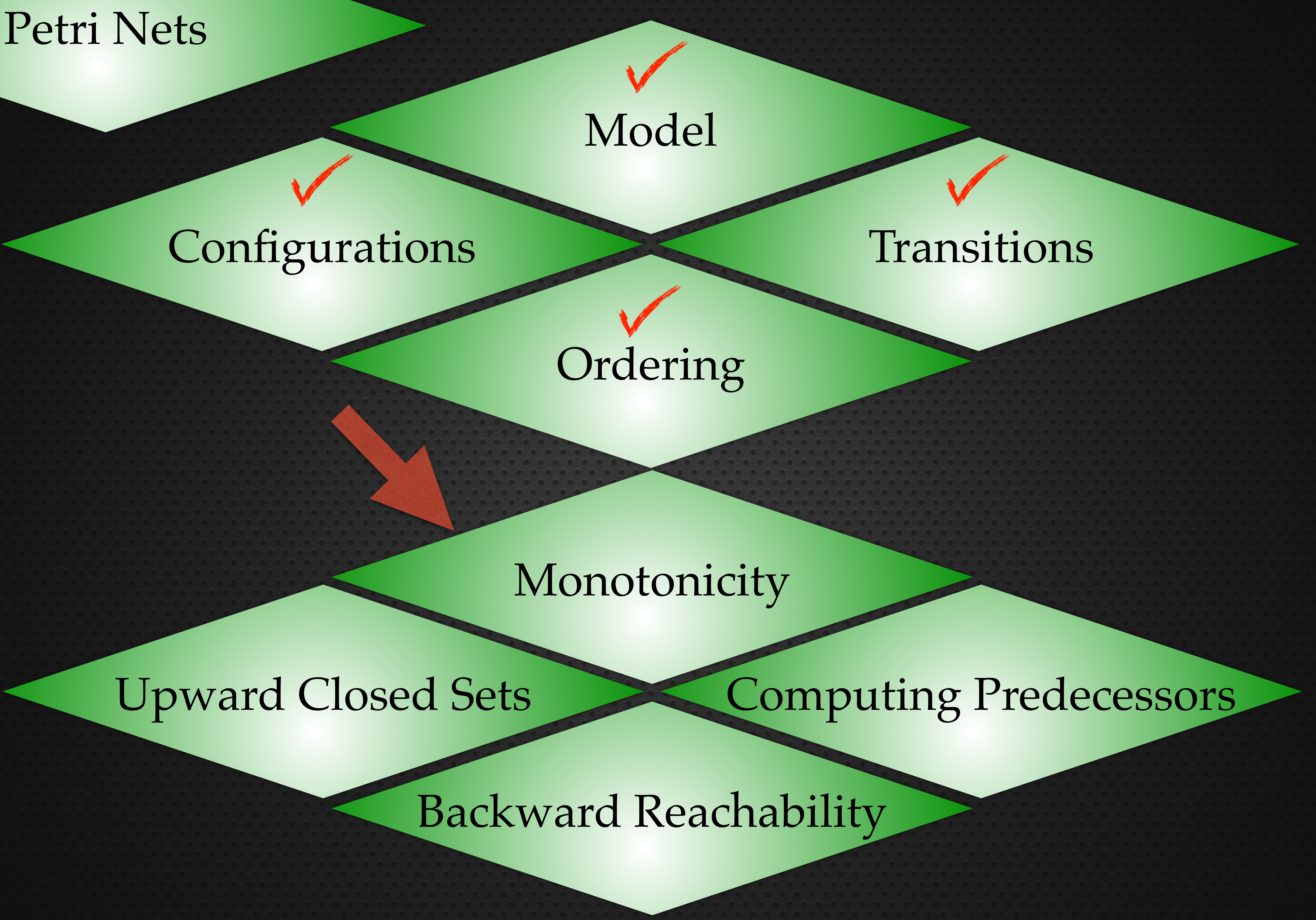
Ordering ✓

Monotonicity

Upward Closed Sets

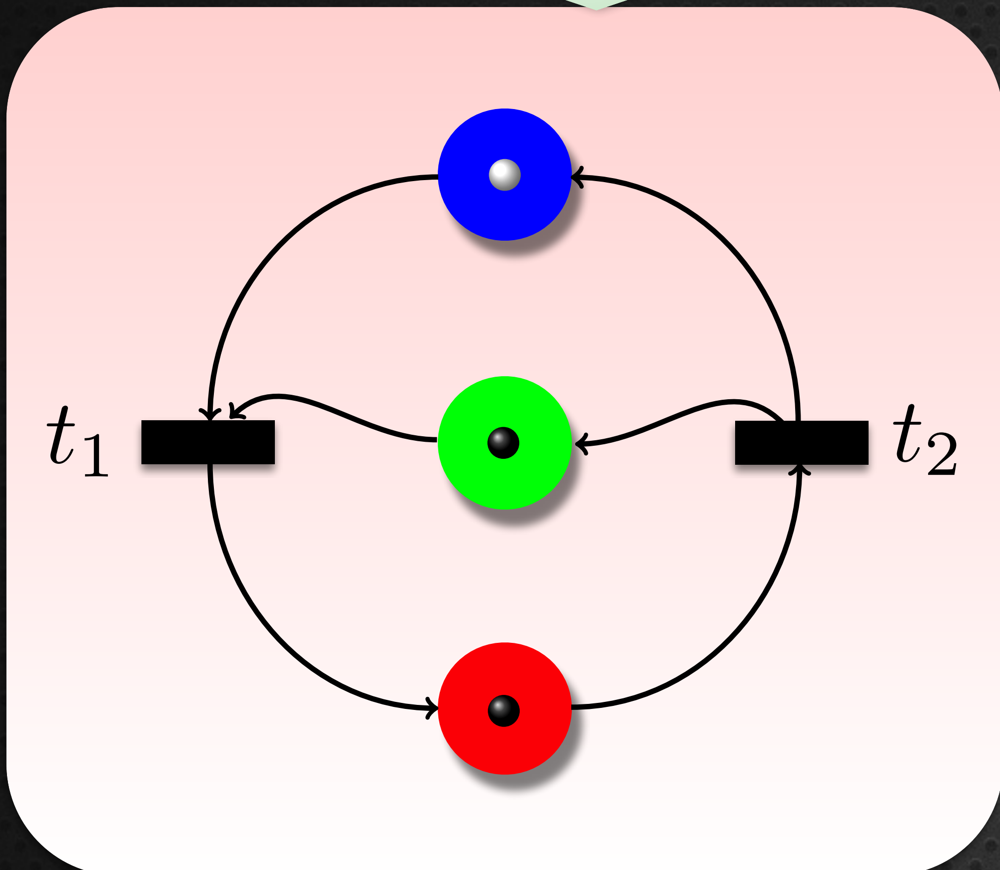
Computing Predecessors

Backward Reachability

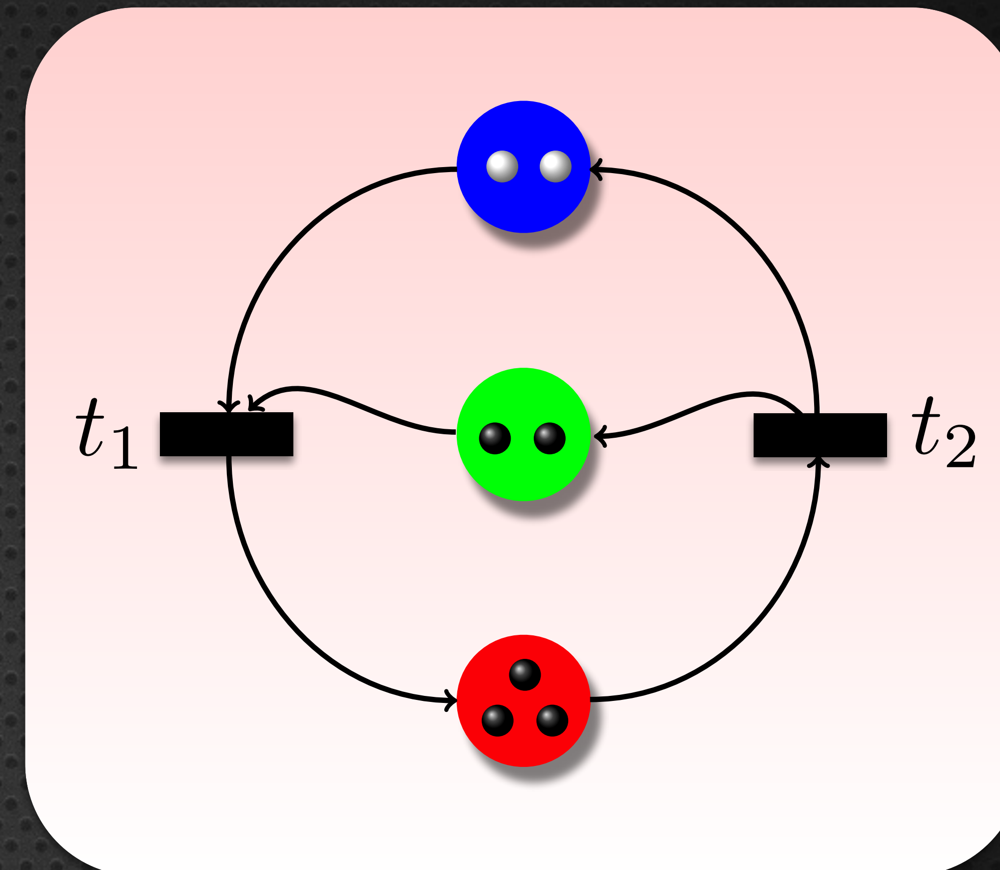

Petri Nets
Model
Configurations
Transitions
Ordering
Monotonicity
Upward Closed Sets
Computing Predecessors
Backward Reachability

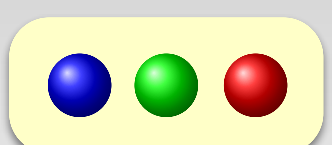
Monotonicity

Pet

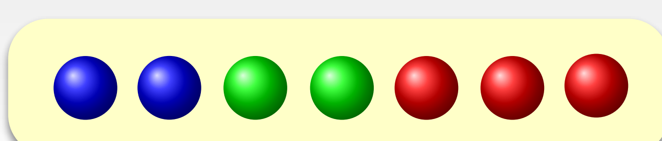


t_1



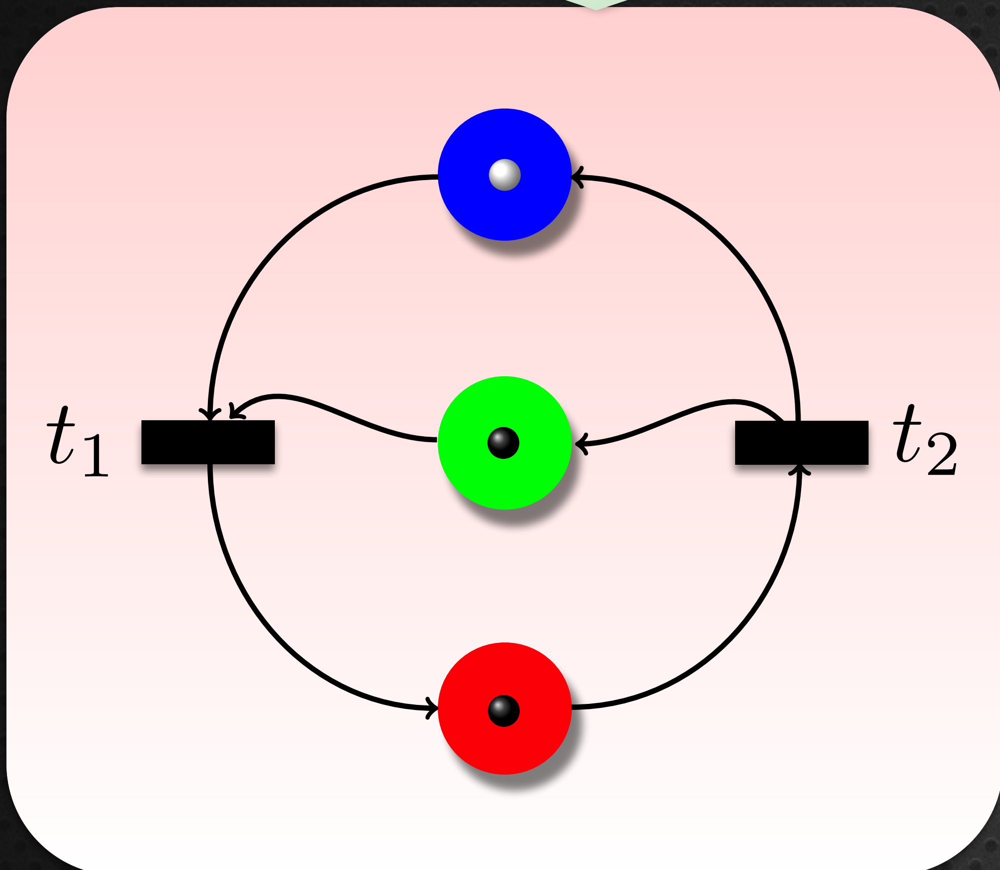


$\sqsubseteq$

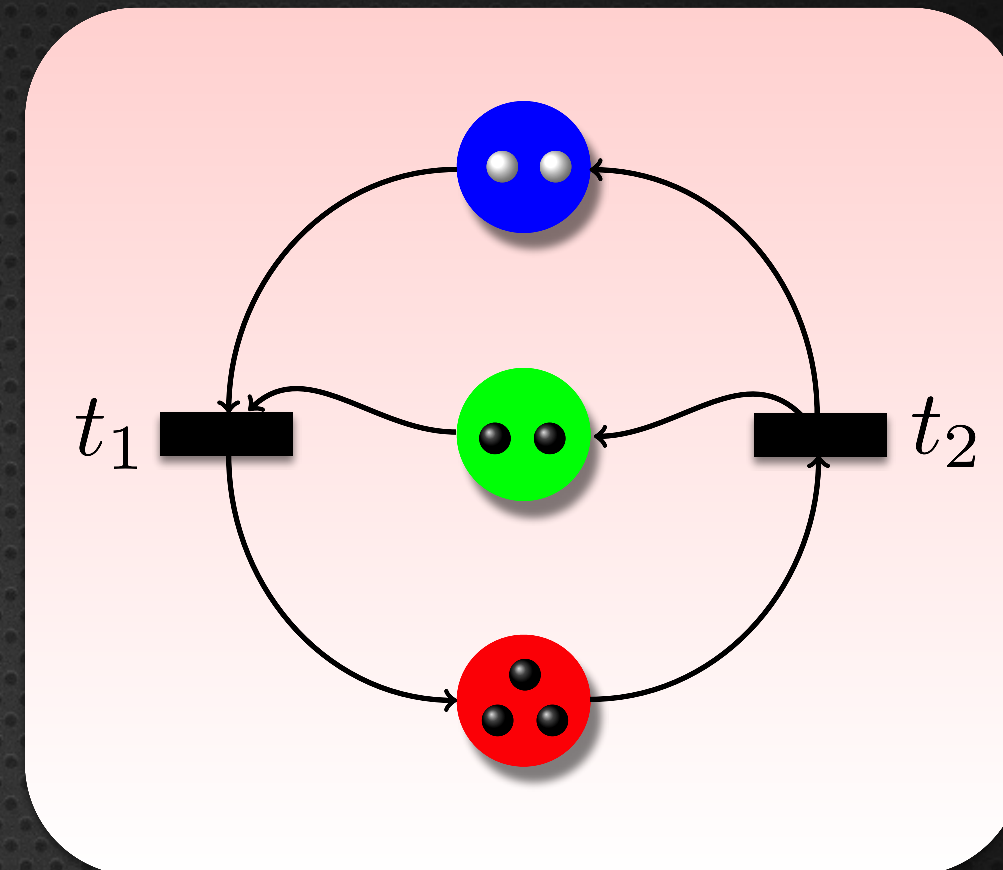


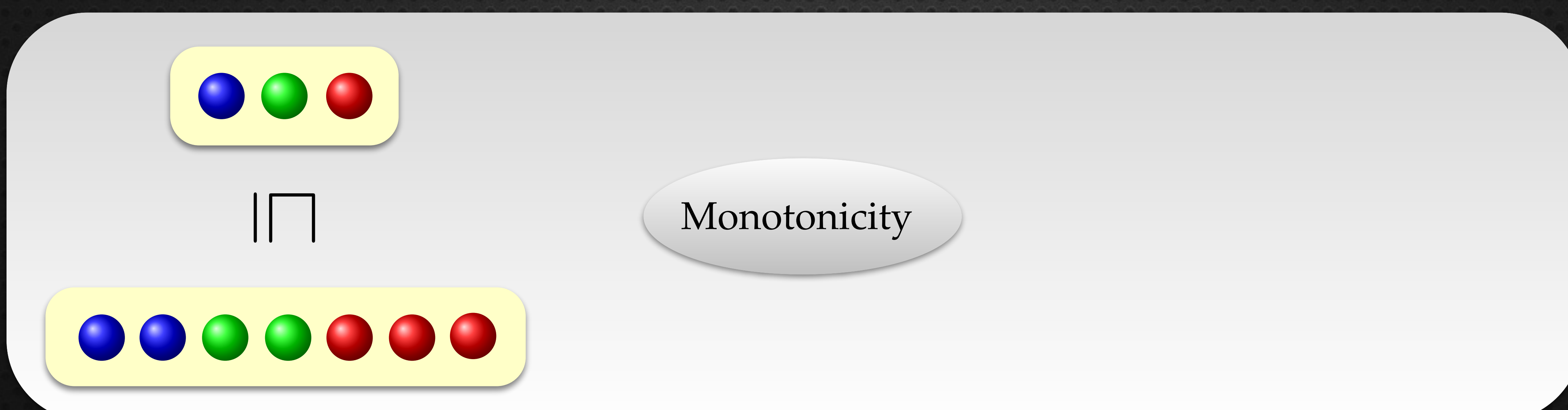
Monotonicity

Monotonicity

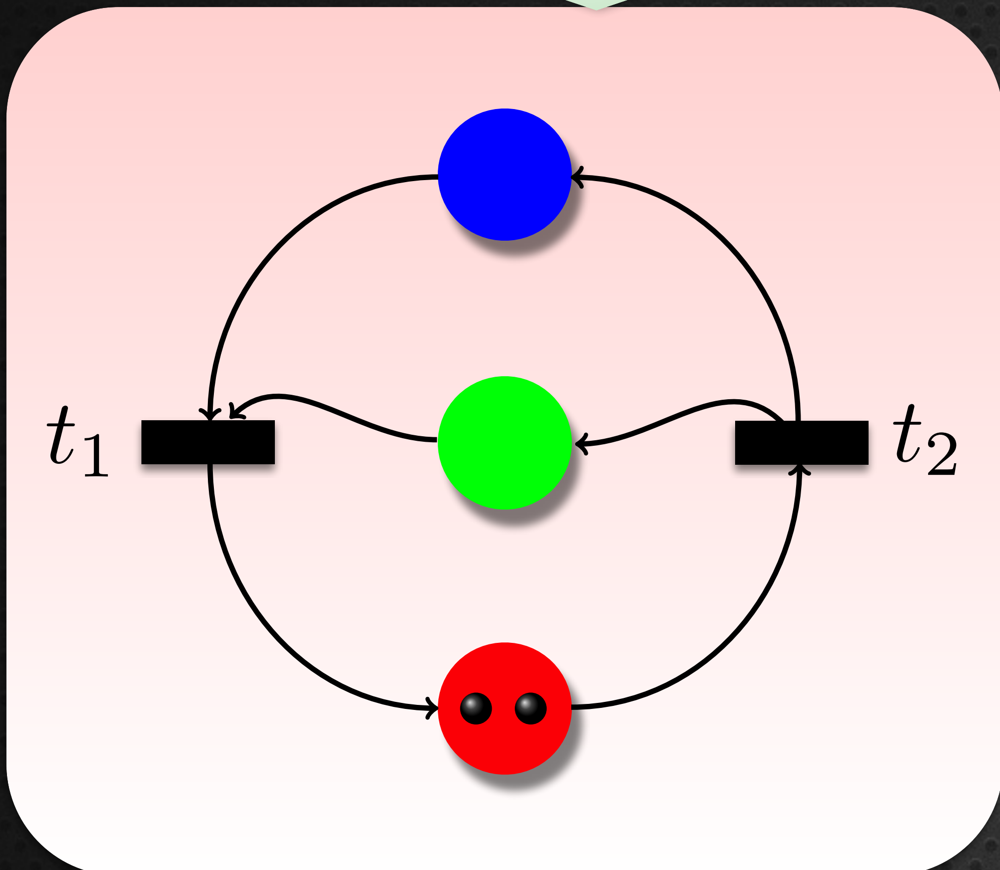


t_1

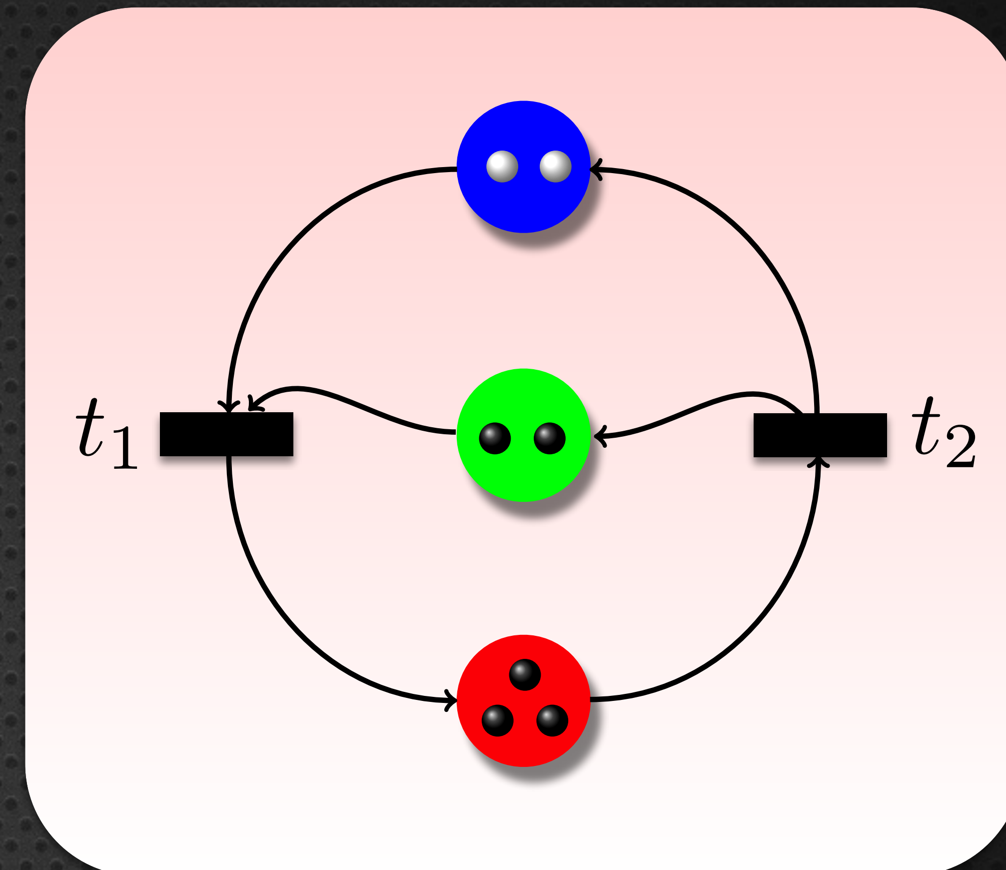


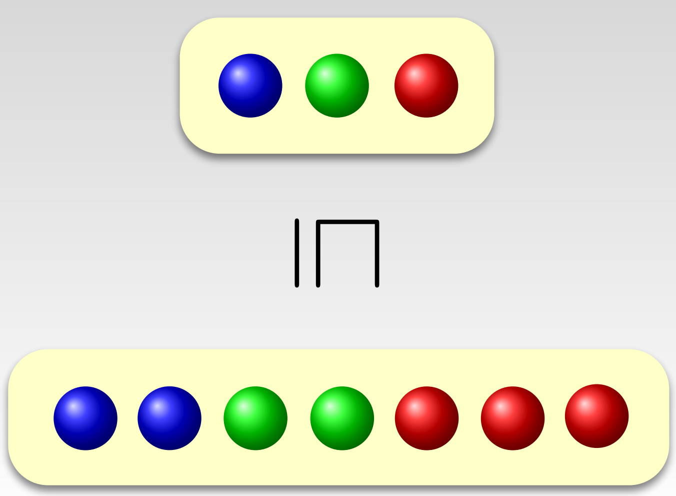


Monotonicity



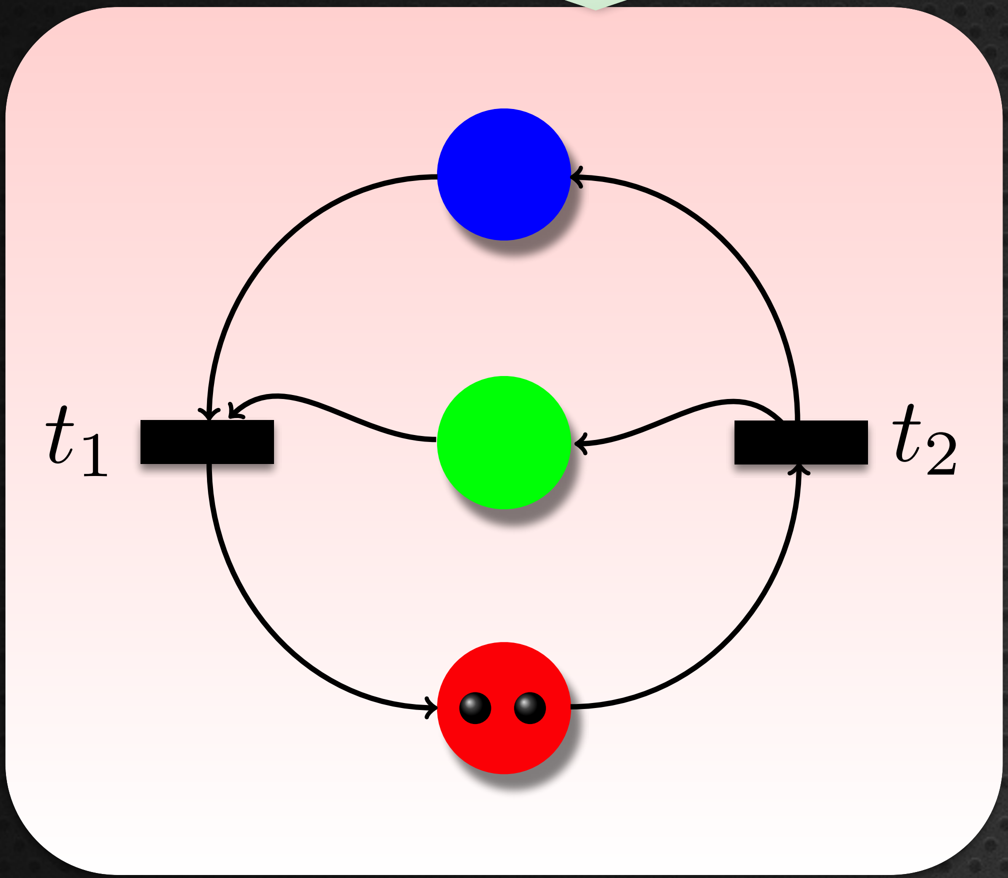
t_1



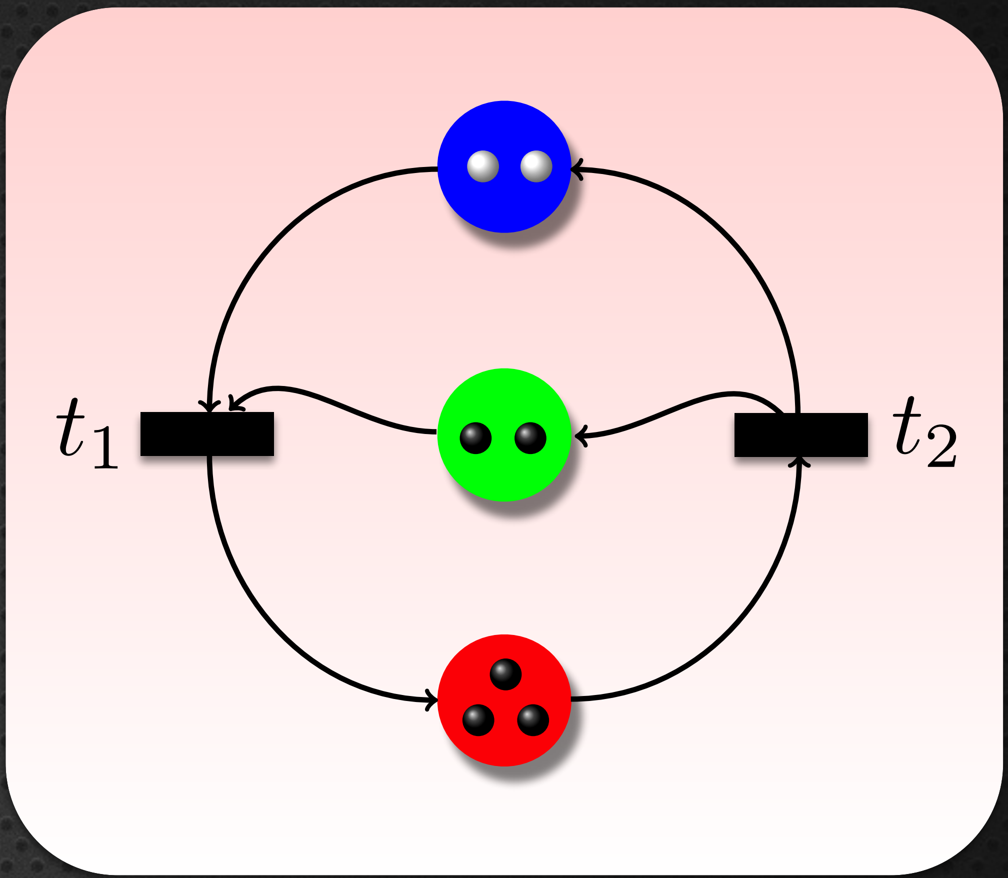


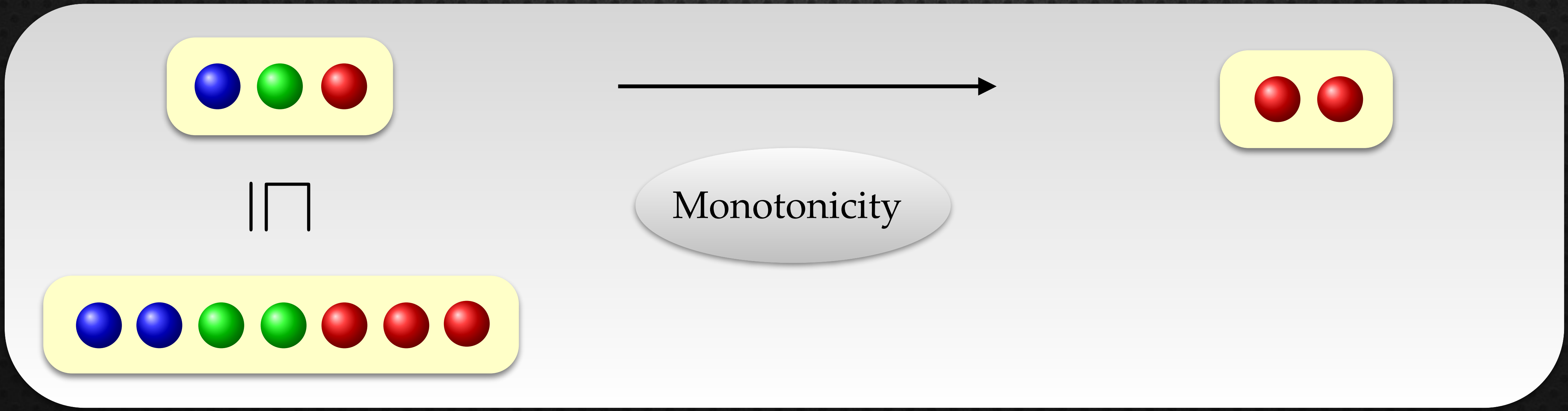
Monotonicity

Monotonicity

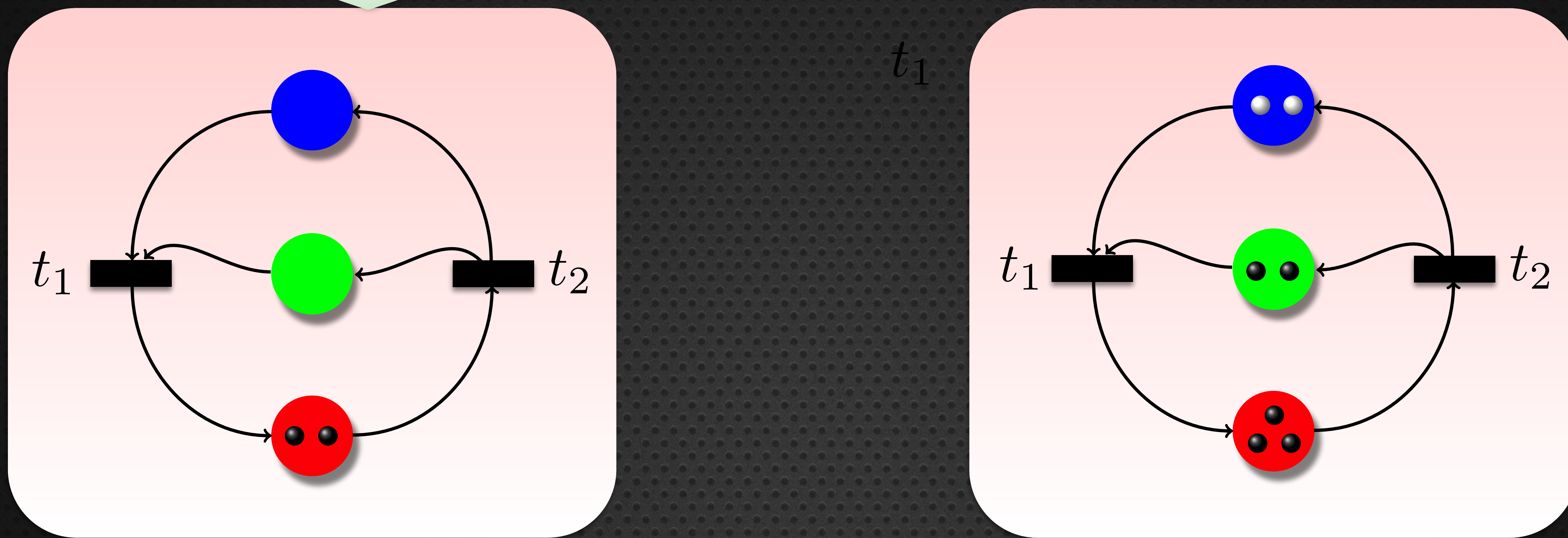


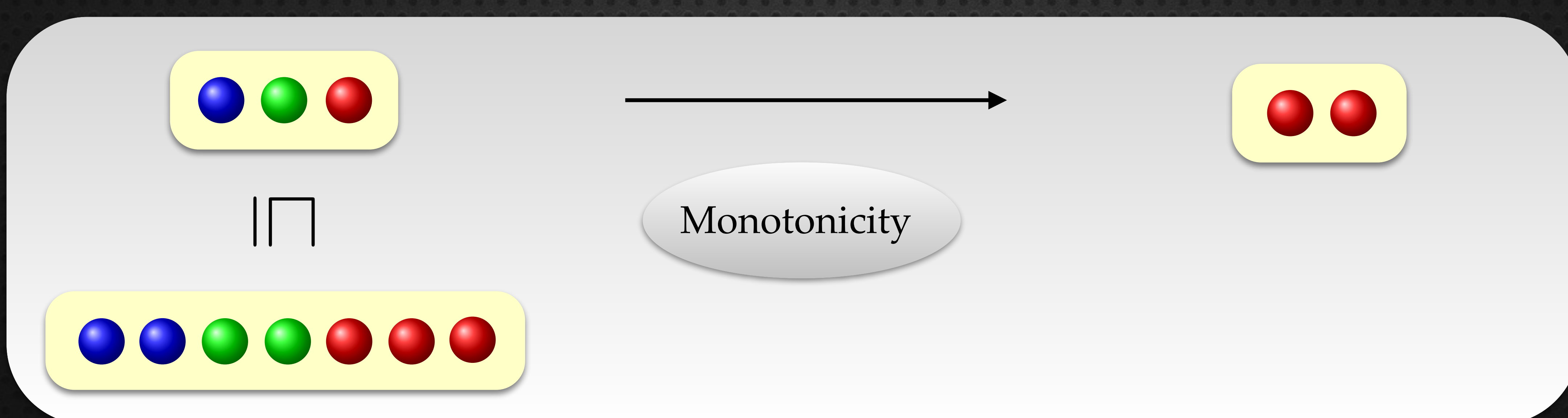
t_1



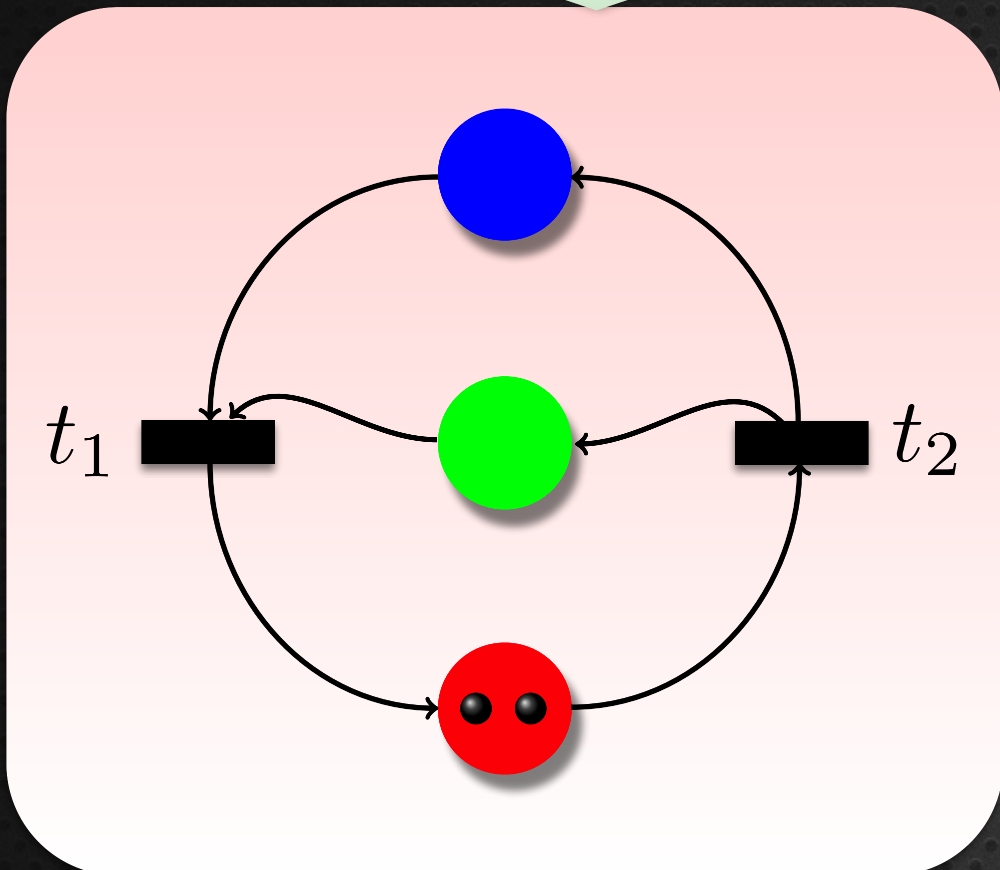


Monotonicity

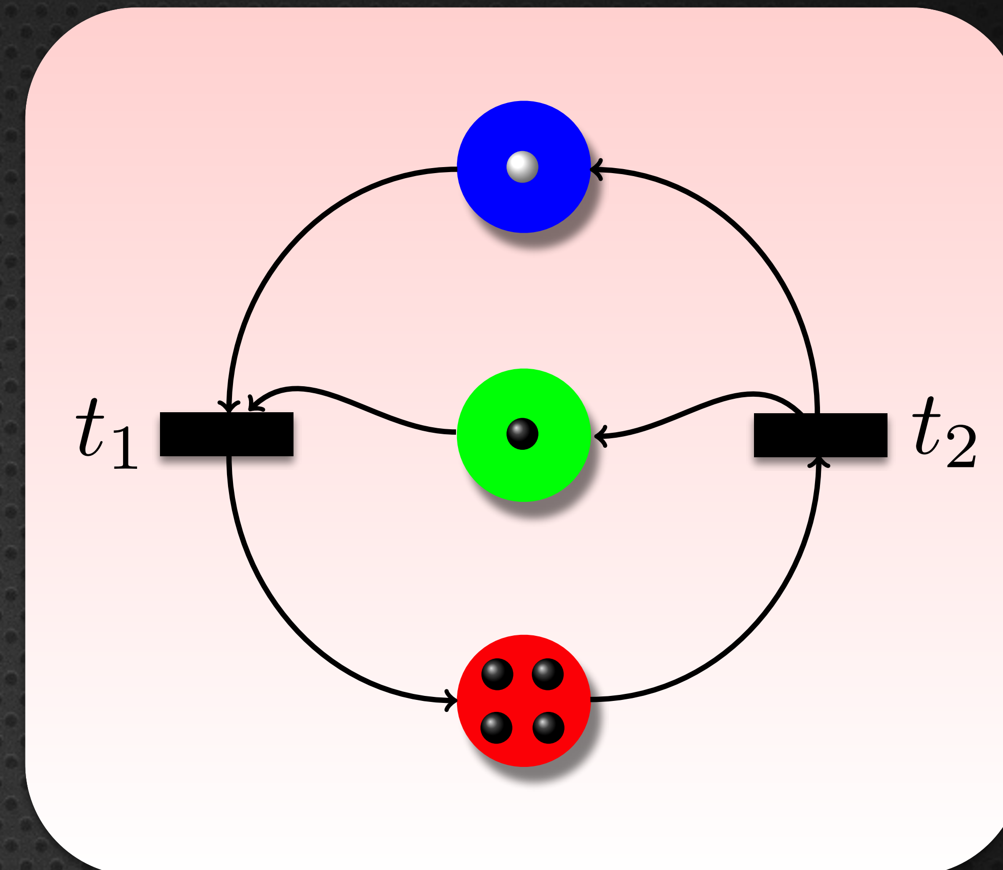


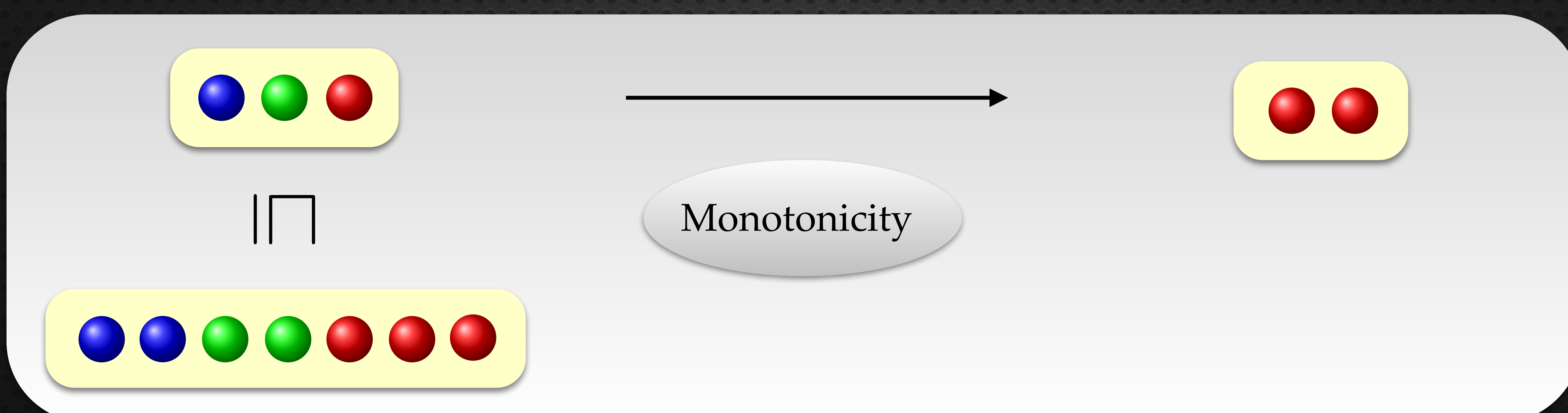


Monotonicity

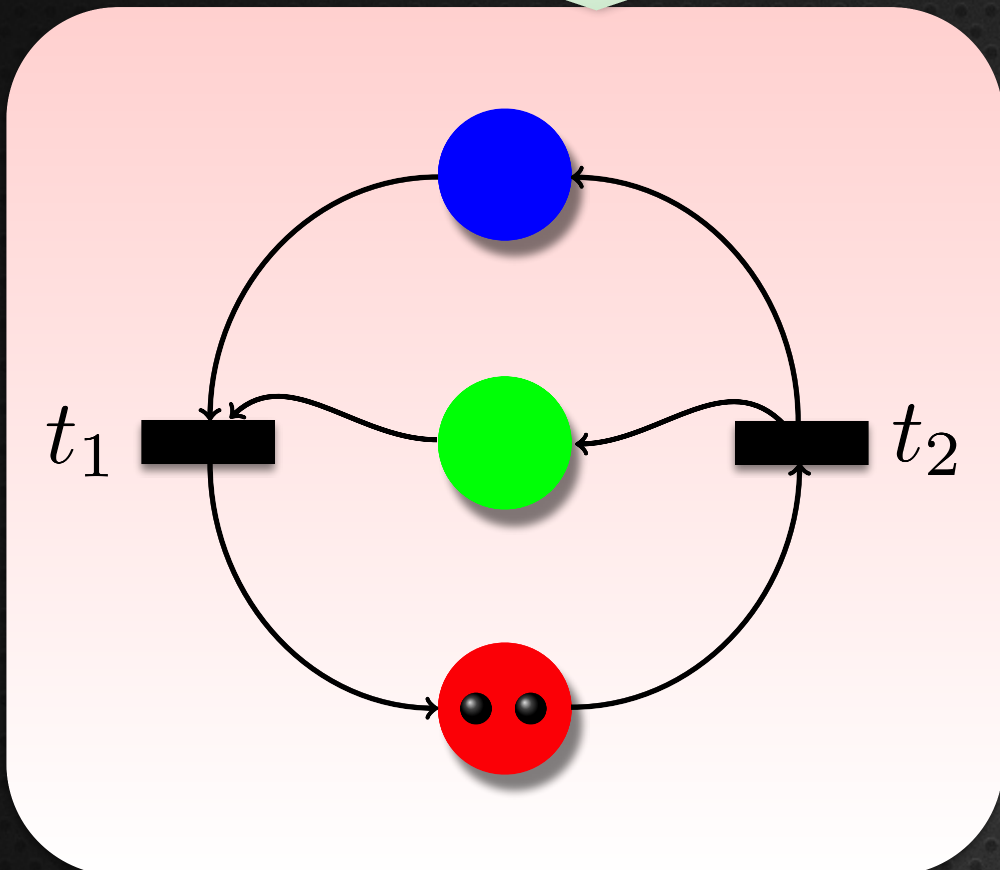


t_1

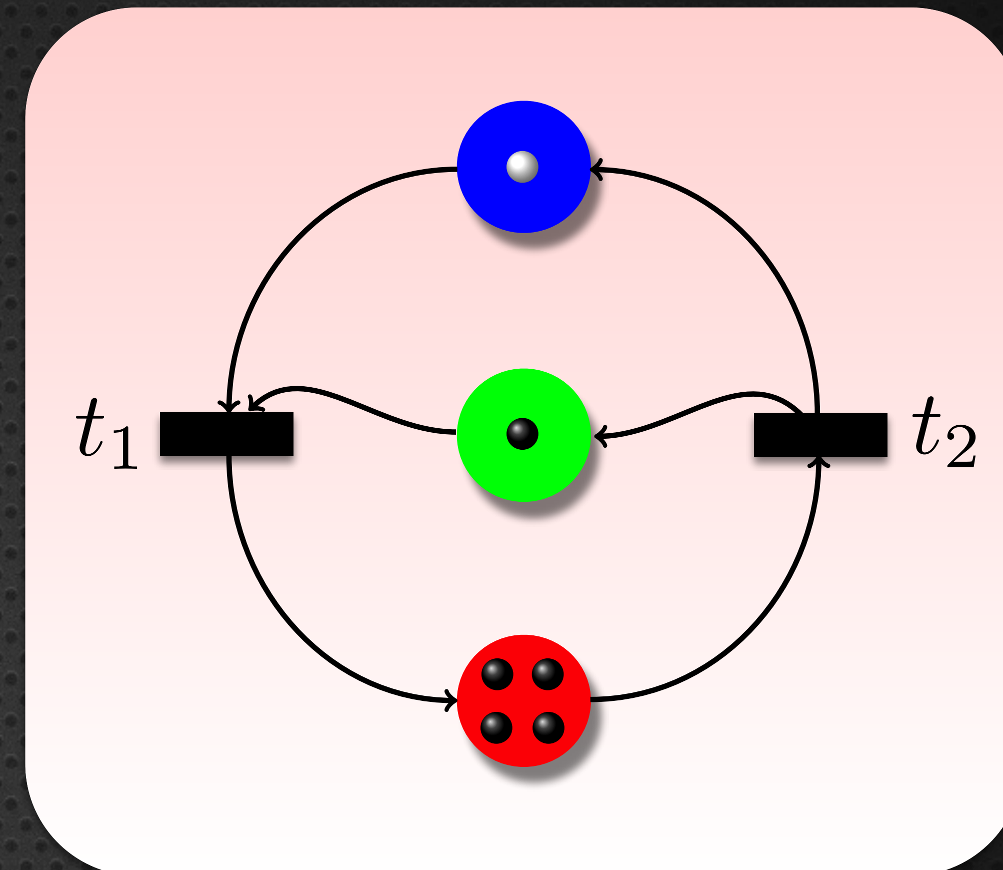


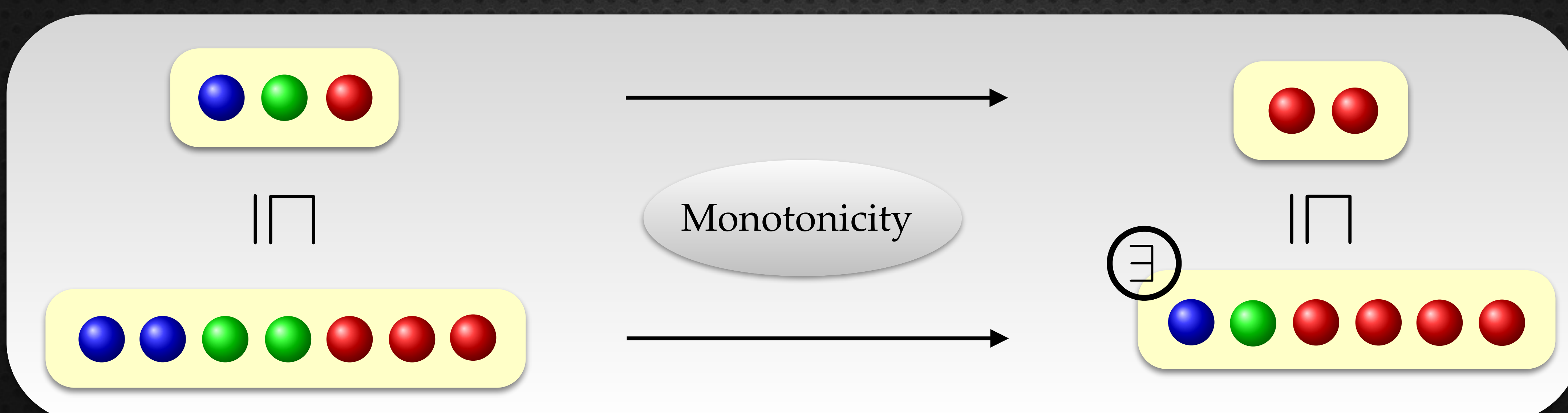


Monotonicity

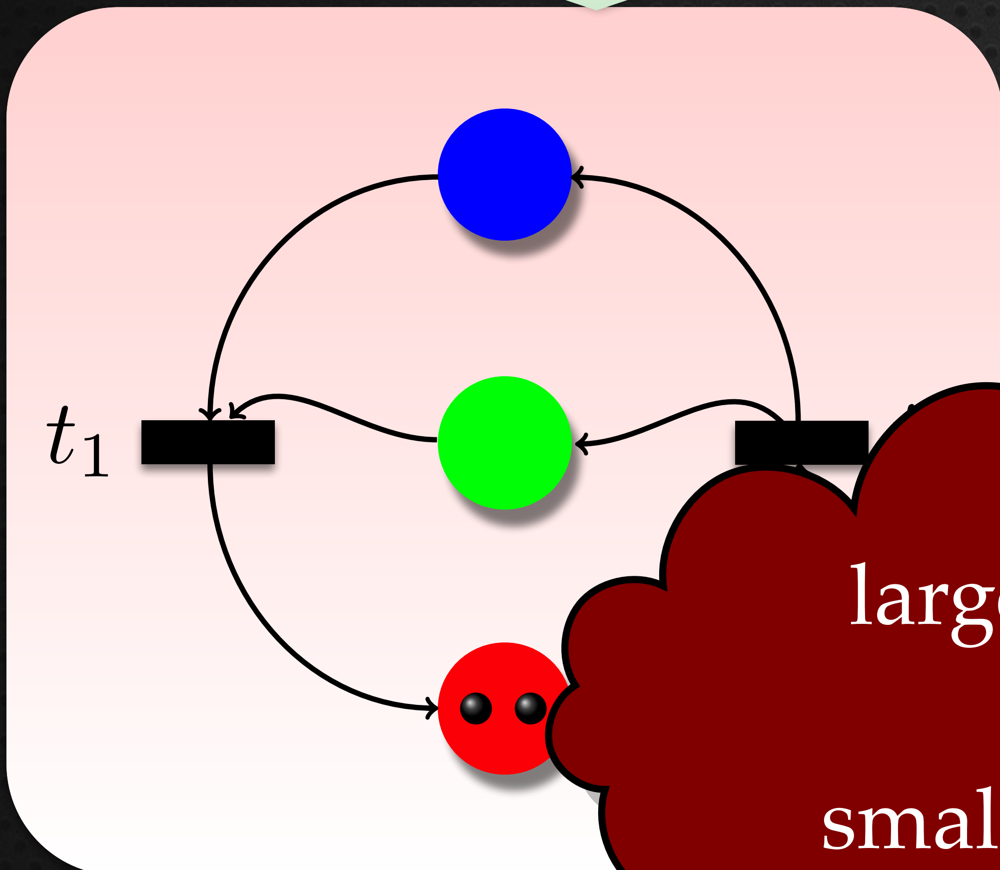


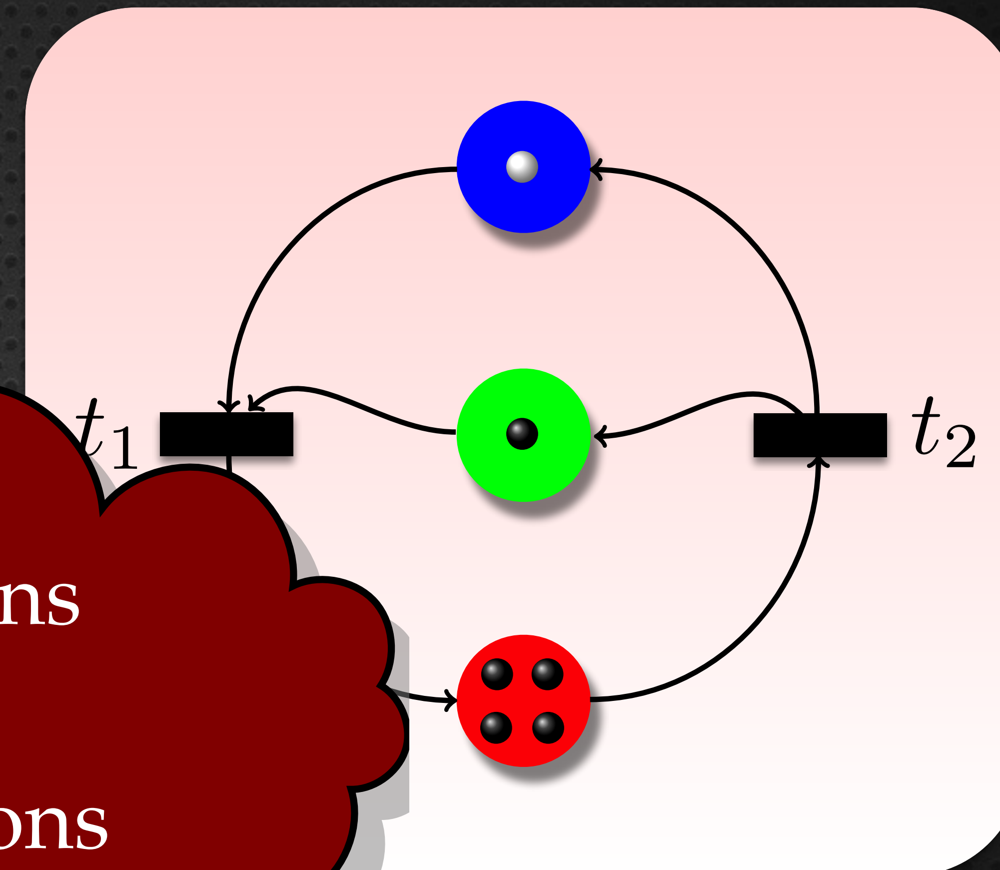
t_1





Monotonicity





larger configurations
"simulate"
smaller configurations

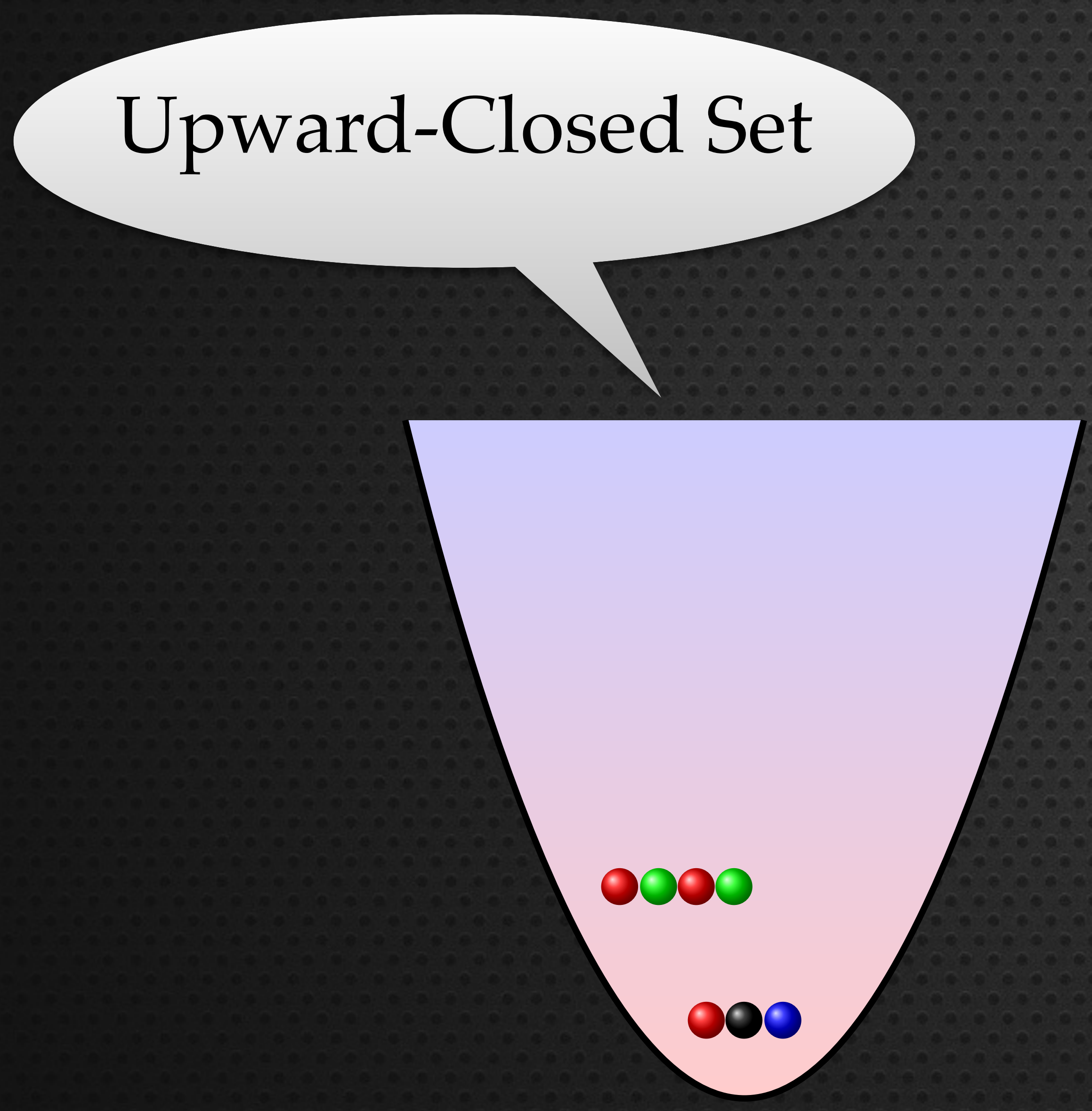
Petri Nets

- Model ✓
- Configurations ✓
- Transitions ✓
- Ordering ✓
- Monotonicity ✓
- Upward Closed Sets
- Computing Predecessors
- Backward Reachability

Petri Nets

- Model
- Configurations
- Transitions
- Ordering
- Monotonicity
- Upward Closed Sets
- Computing Predecessors
- Backward Reachability

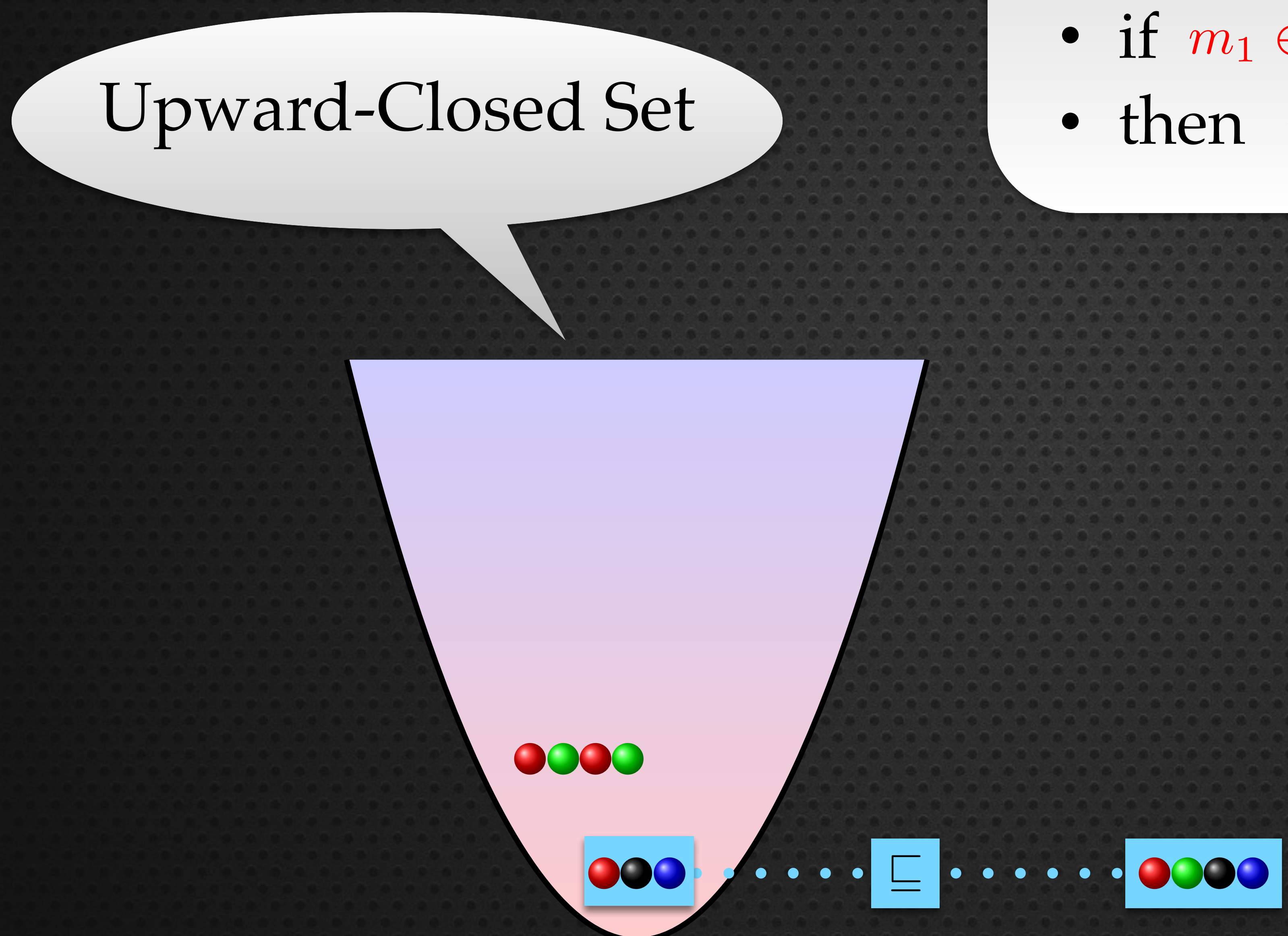
Upward Closed Sets



Upward Closed Set (UC)

- if $m_1 \in U$ and $m_1 \sqsubseteq m_2$
- then $m_2 \in U$

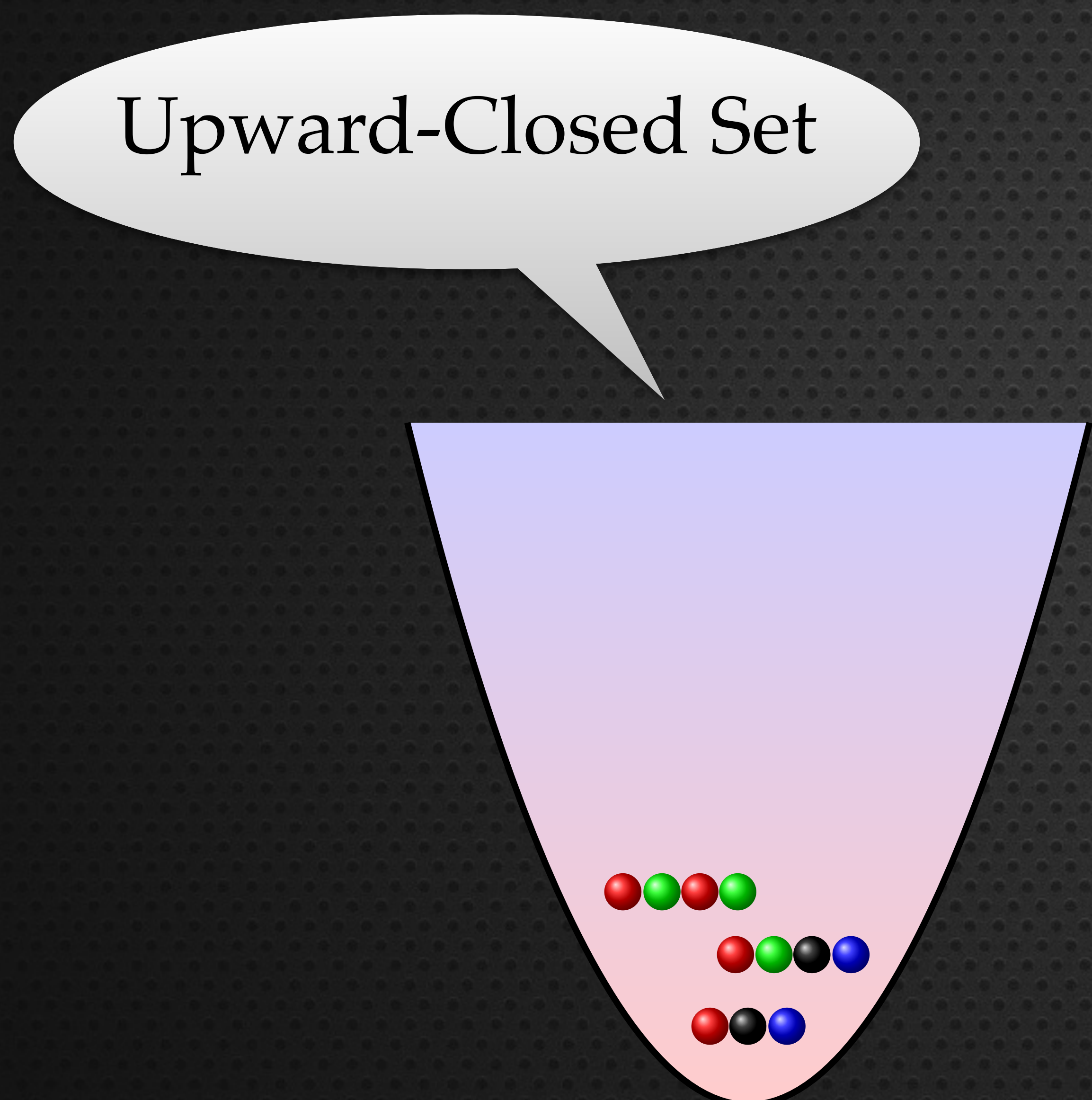
Upward Closed Sets



Upward Closed Set (UC)

- if $m_1 \in U$ and $m_1 \sqsubseteq m_2$
- then $m_2 \in U$

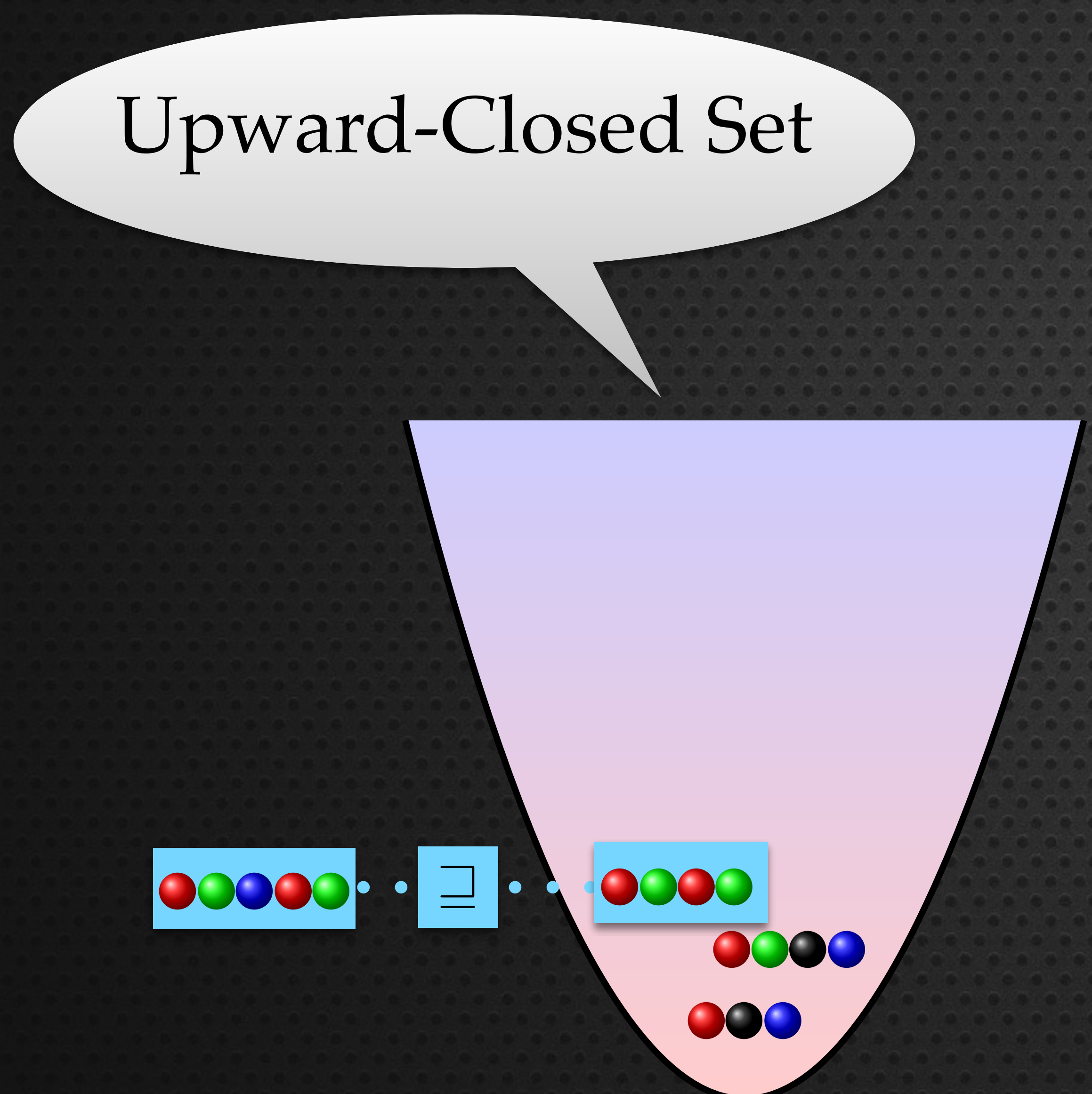
Upward Closed Sets



Upward Closed Set (UC)

- if $m_1 \in U$ and $m_1 \sqsubseteq m_2$
- then $m_2 \in U$

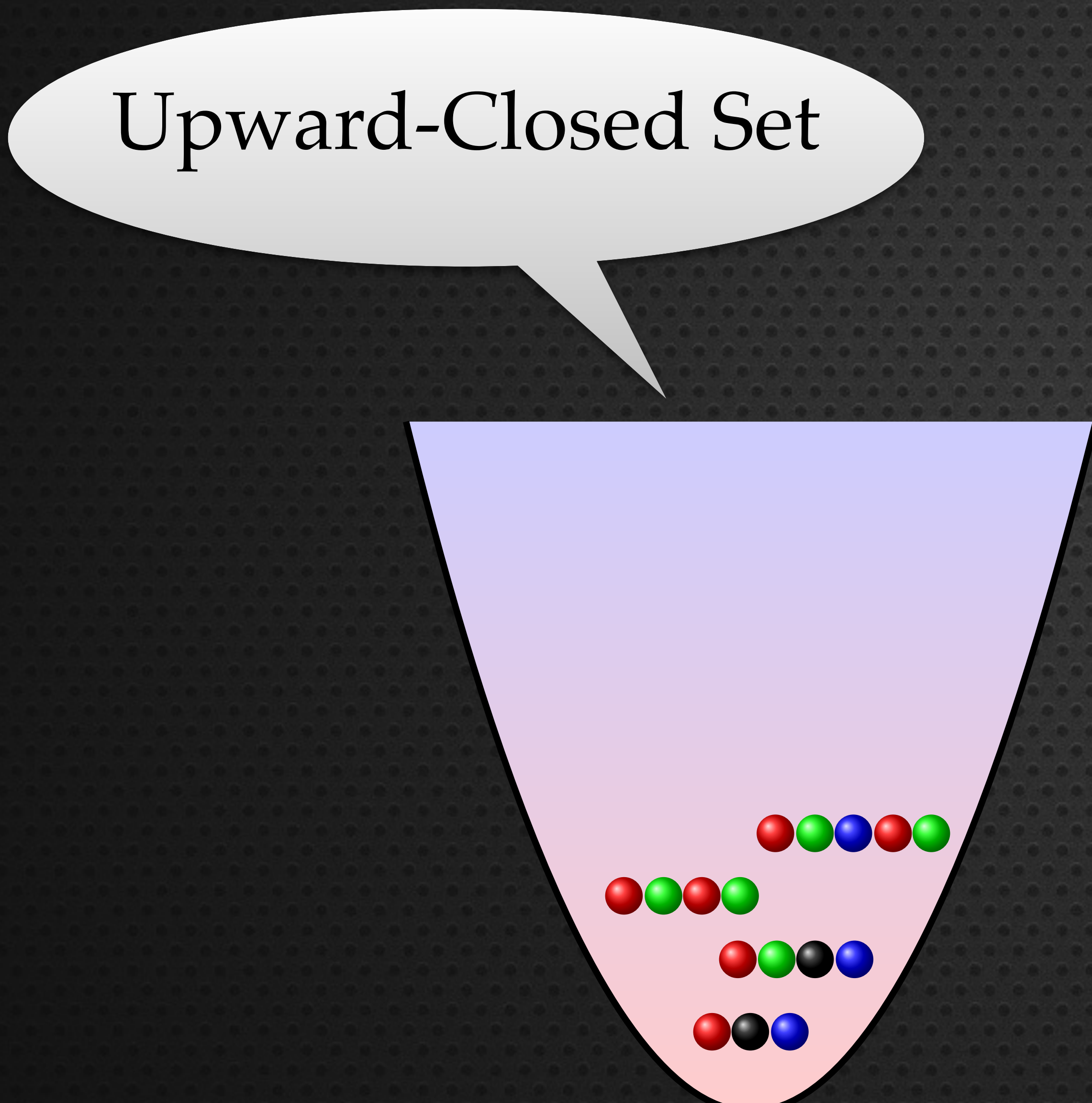
Upward Closed Sets



Upward Closed Set (UC)

- if $m_1 \in U$ and $m_1 \sqsubseteq m_2$
- then $m_2 \in U$

Upward Closed Sets



Upward Closed Set (UC)

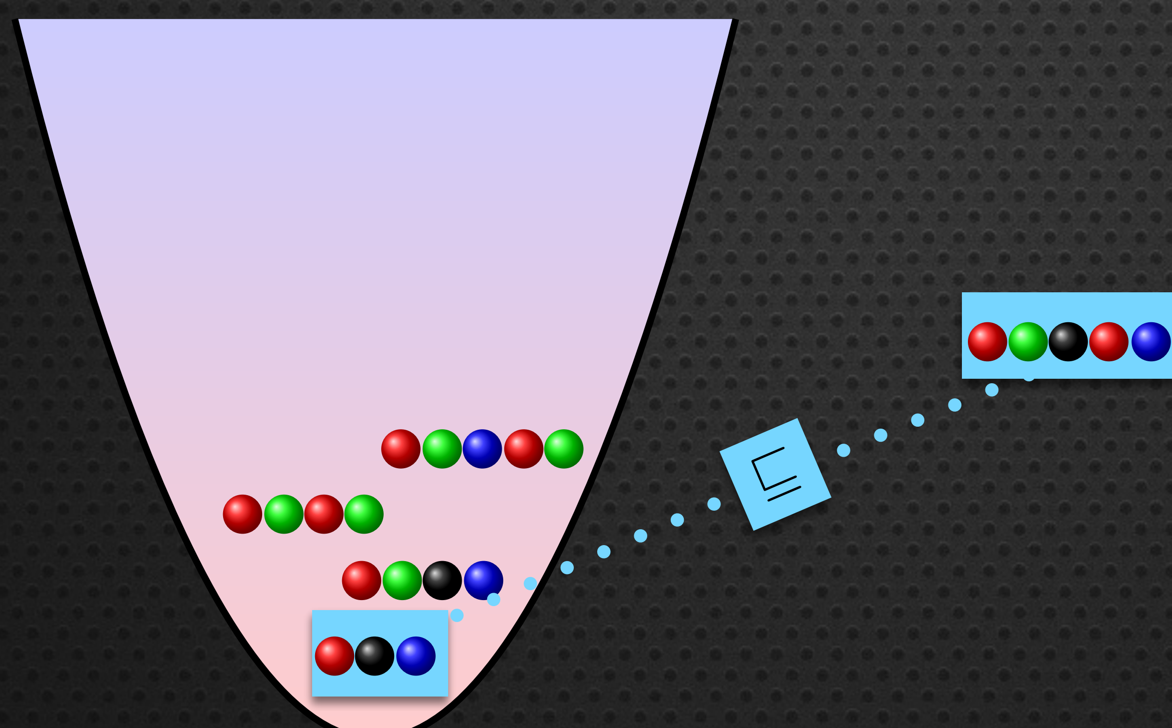
- if $m_1 \in U$ and $m_1 \sqsubseteq m_2$
- then $m_2 \in U$

Upward Closed Sets

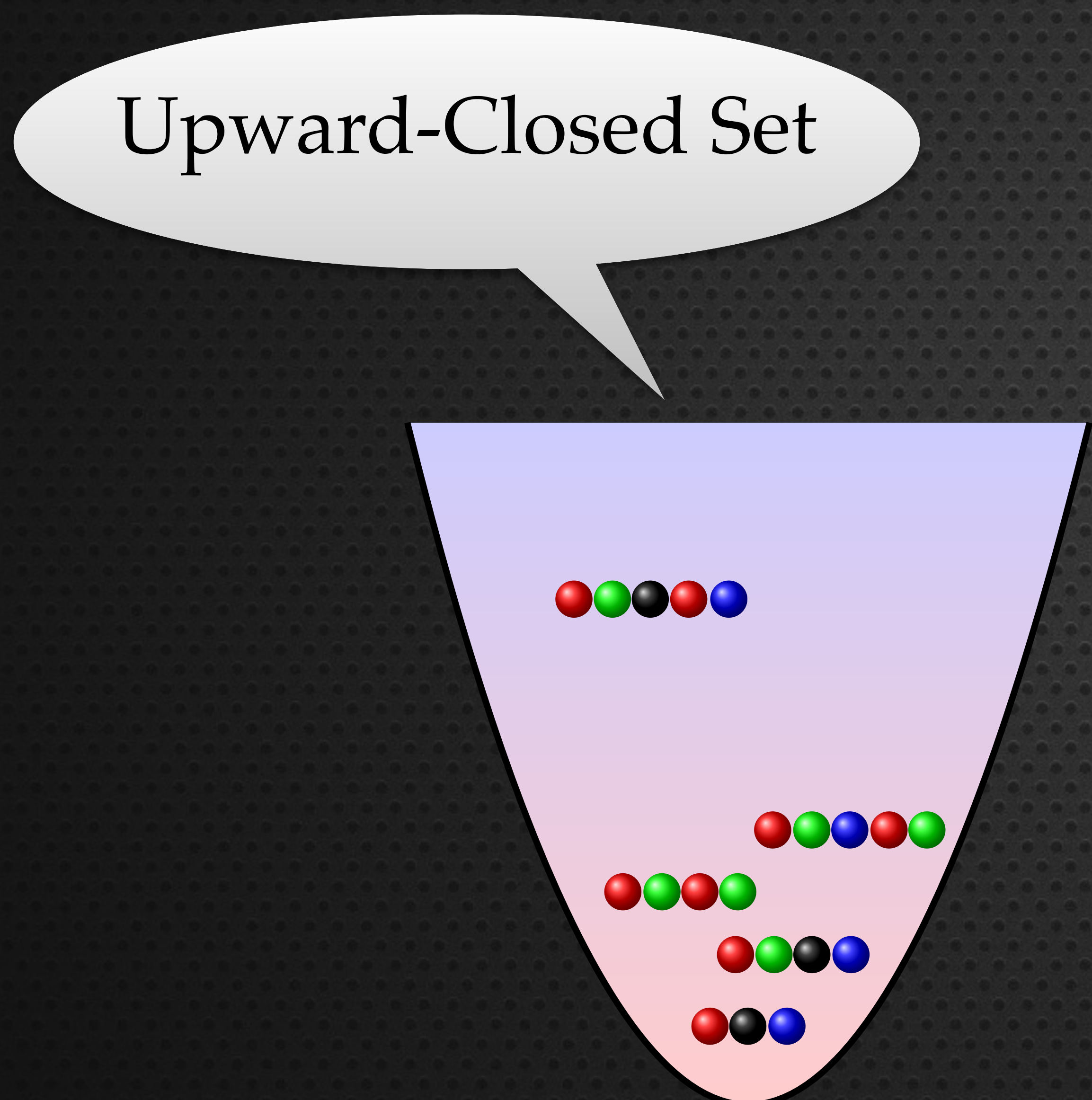
Upward Closed Set (UC)

- if $m_1 \in U$ and $m_1 \sqsubseteq m_2$
- then $m_2 \in U$

Upward-Closed Set



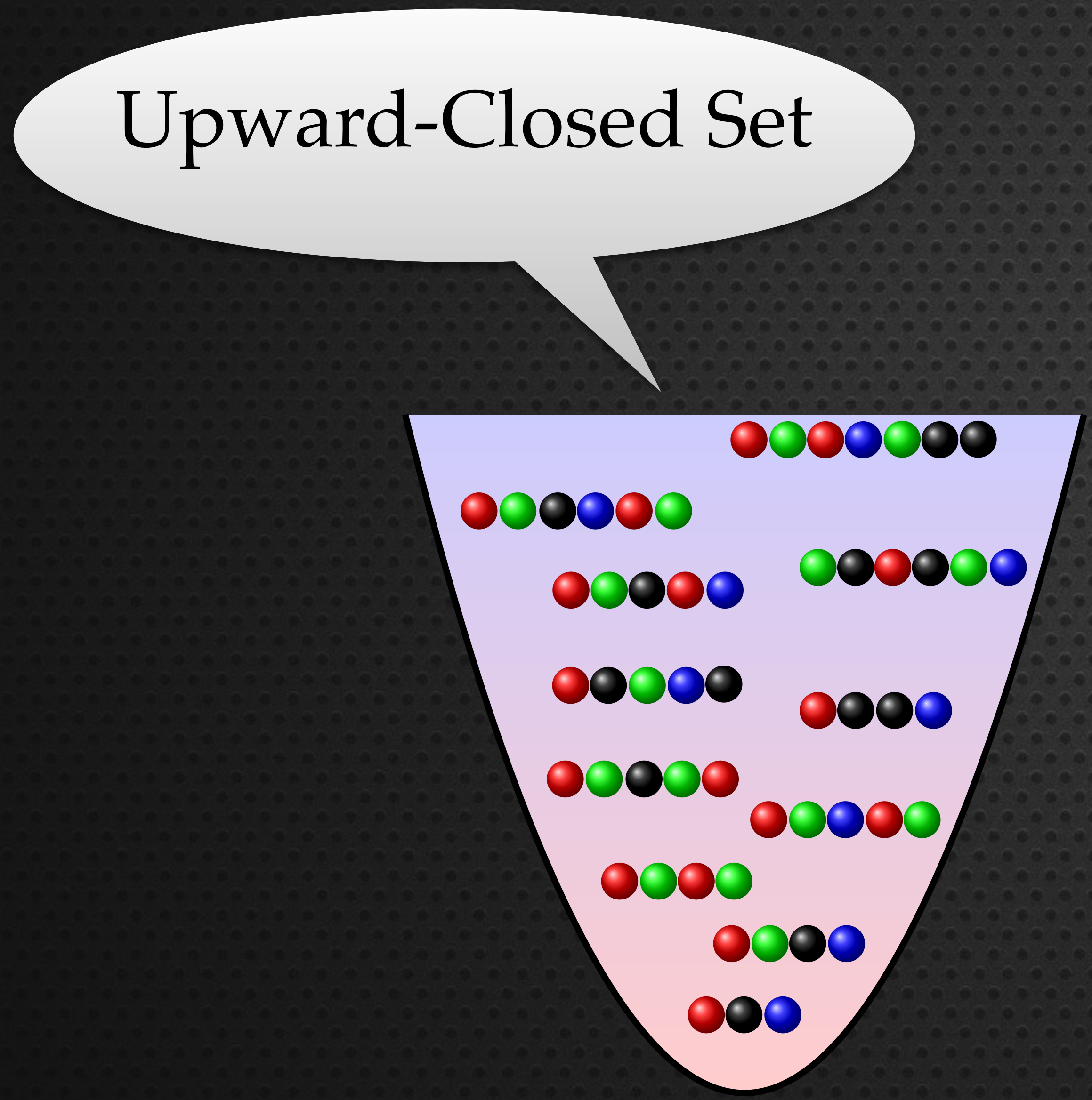
Upward Closed Sets



Upward Closed Set (UC)

- if $m_1 \in U$ and $m_1 \sqsubseteq m_2$
- then $m_2 \in U$

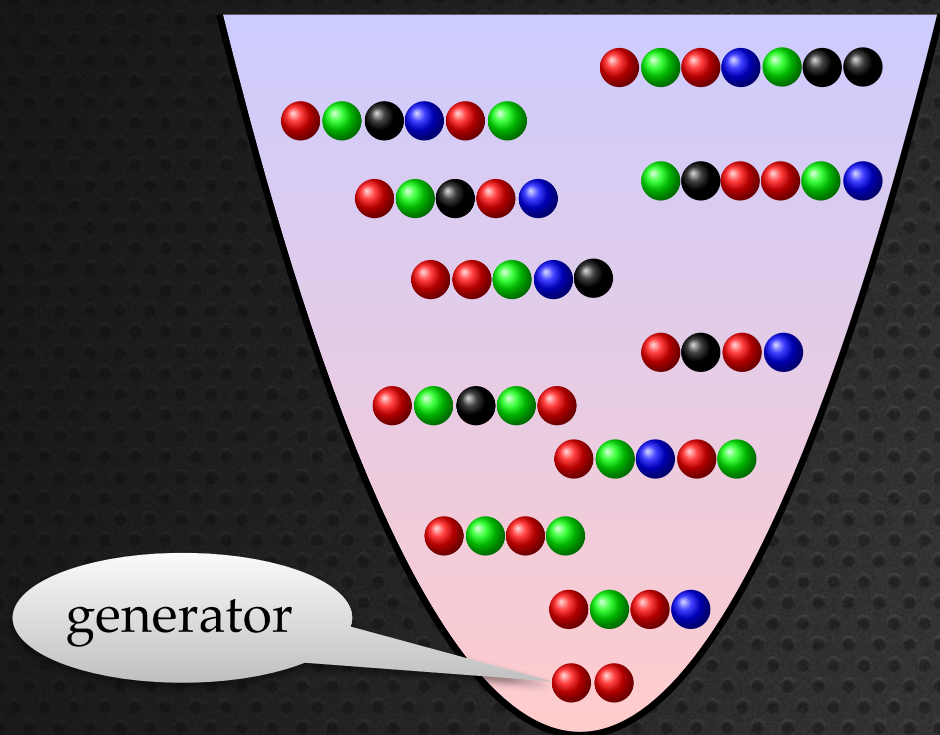
Upward Closed Sets



Upward Closed Set (UC)

- if $m_1 \in U$ and $m_1 \sqsubseteq m_2$
- then $m_2 \in U$

Upward Closed Sets



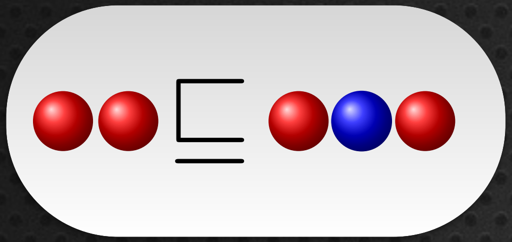
Upward Closed Set (UC)

- if $m_1 \in U$ and $m_1 \sqsubseteq m_2$
- then $m_2 \in U$

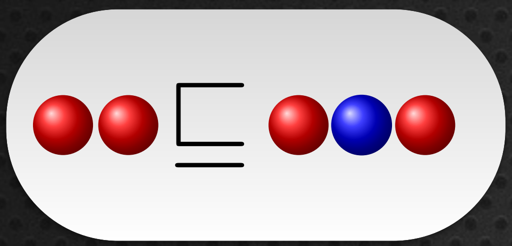
Why UC?

- Bad sets of markings are UC
  - checking safety properties = reachability of bad markings
- Uniquely characterized by generator
  - simple representation= finite multiset

Upward Closed Sets

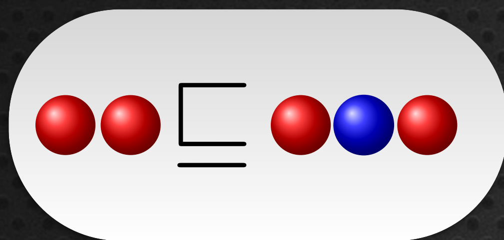


Upward Closed Sets

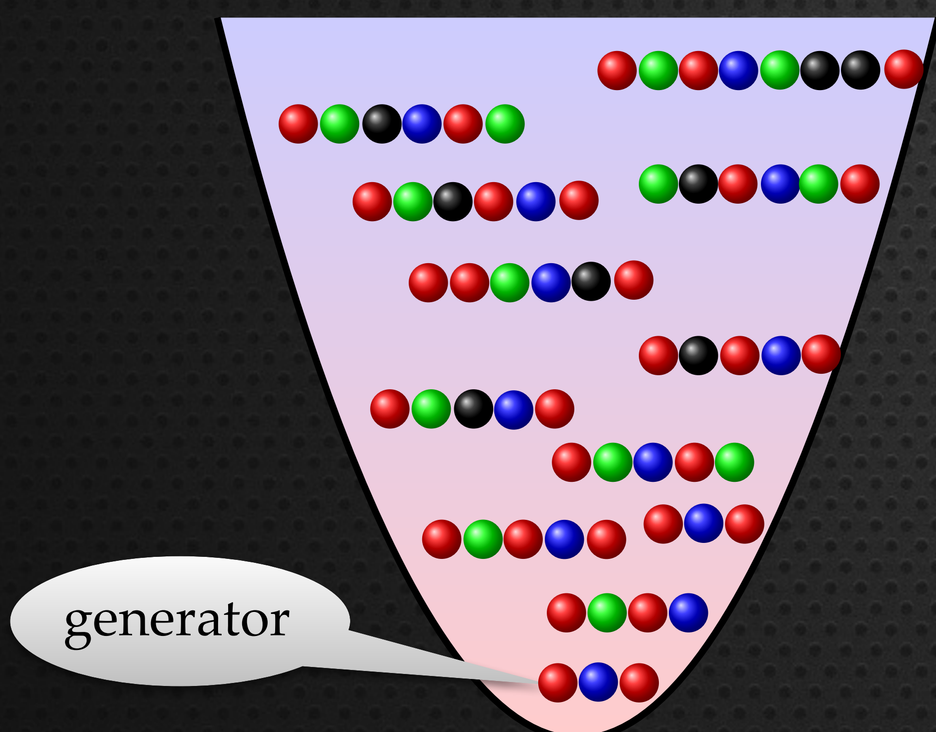


implies

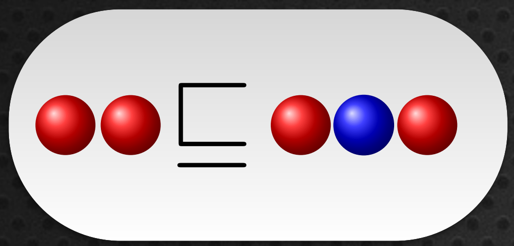
Upward Closed Sets



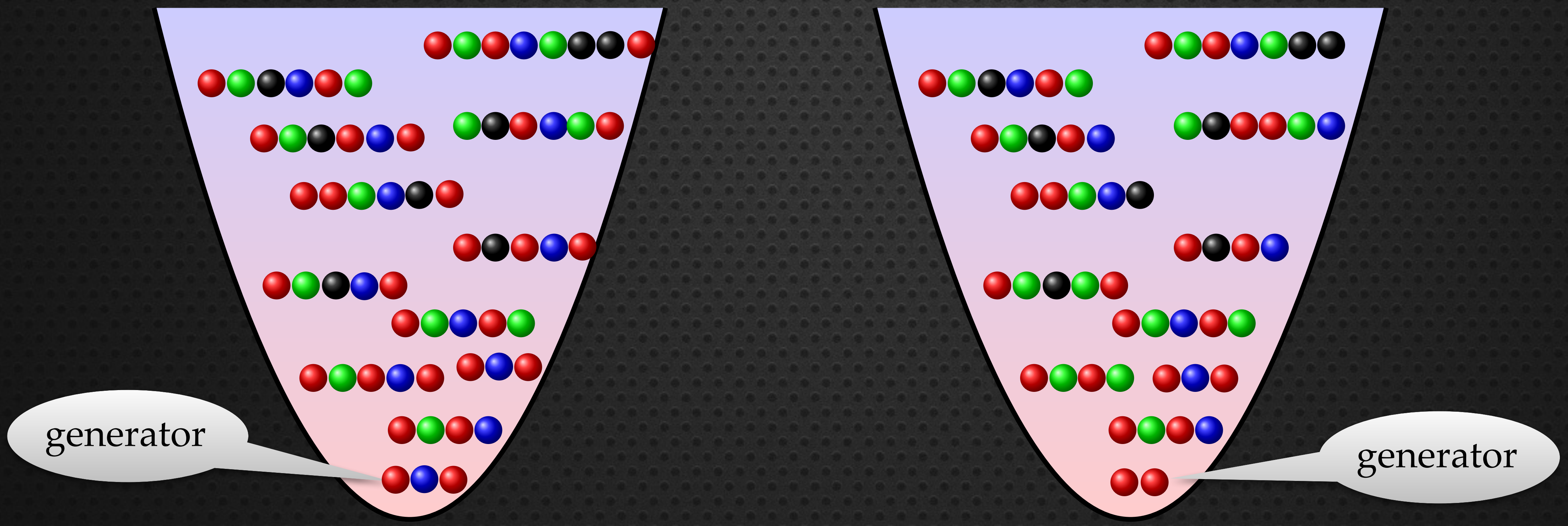
implies



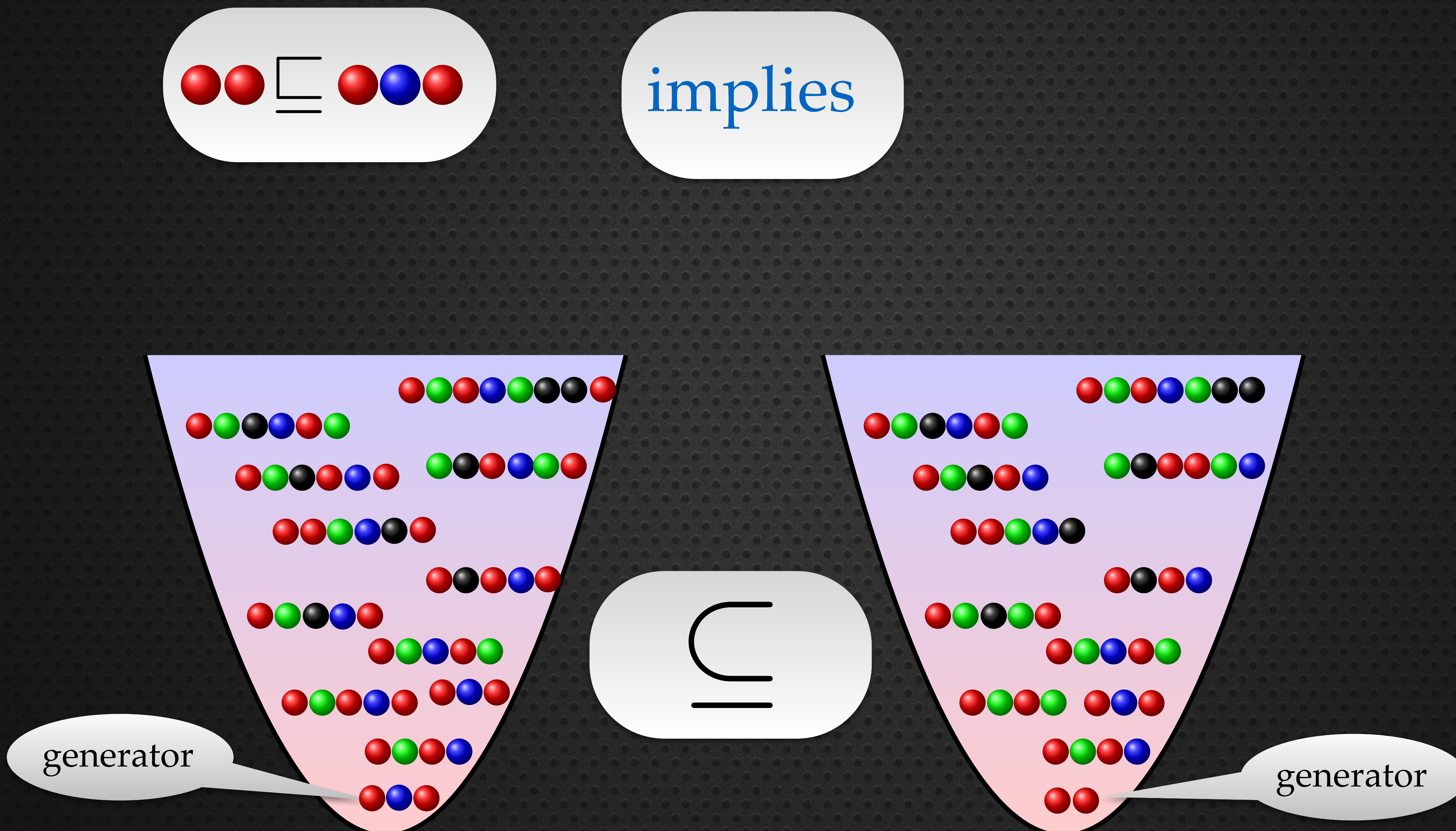
Upward Closed Sets



implies

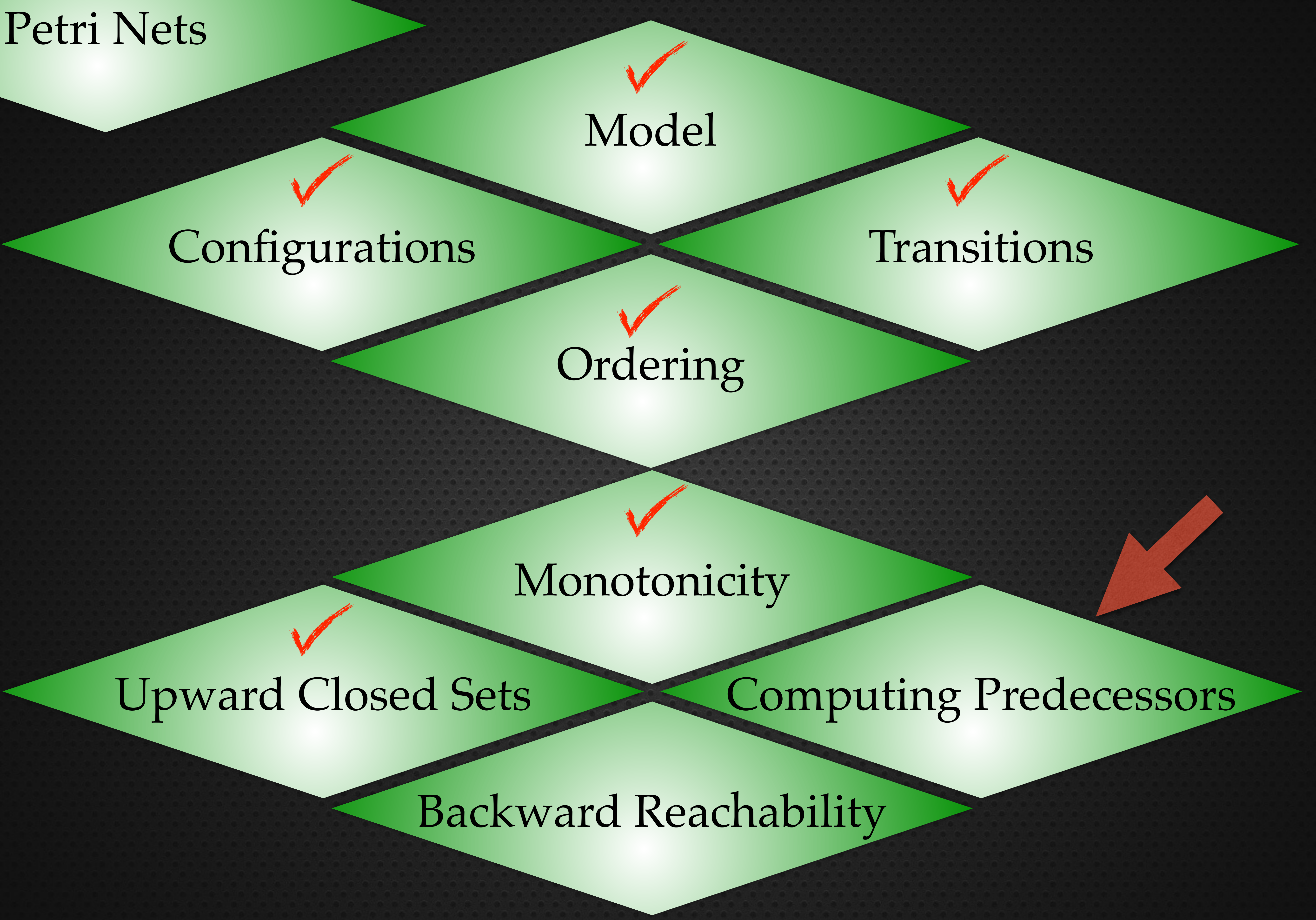


Upward Closed Sets

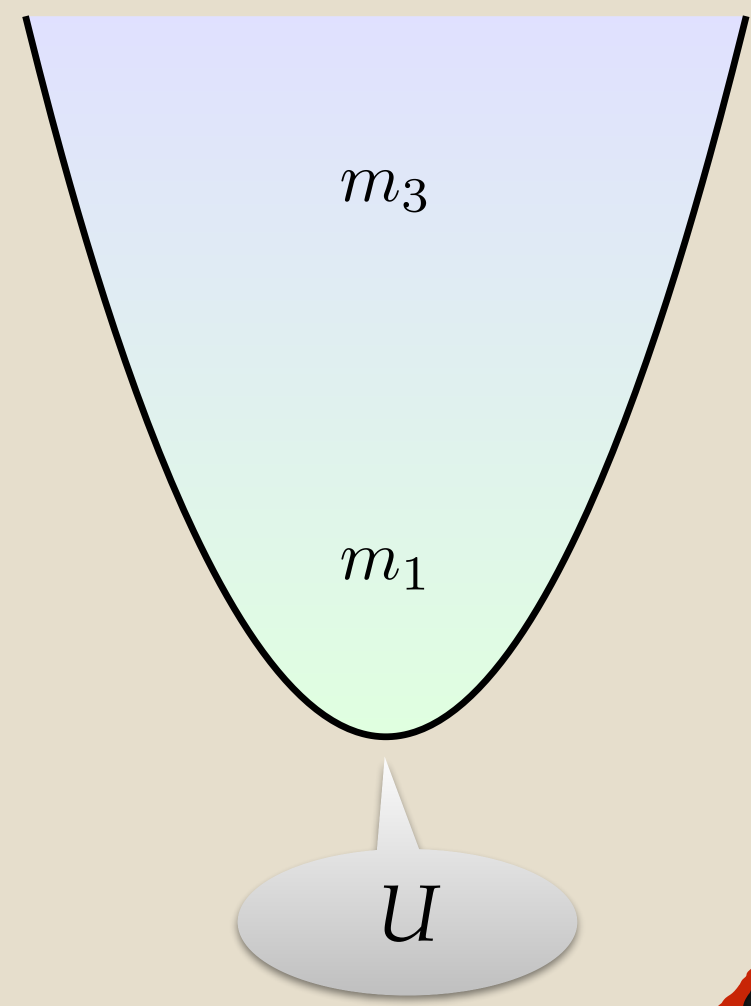


Petri Nets

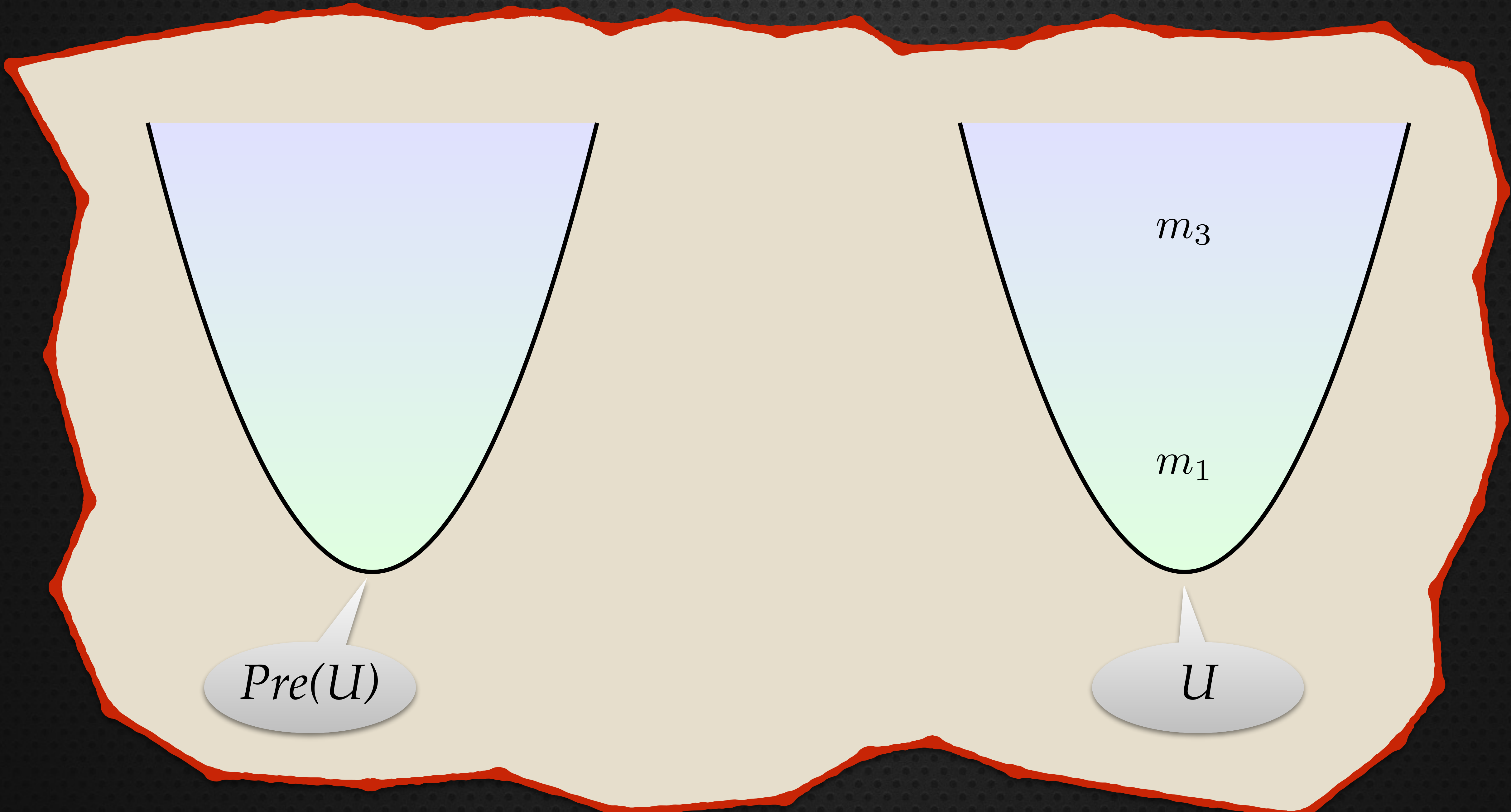
- Model
- Configurations
- Transitions
- Ordering
- Monotonicity
- Upward Closed Sets
- Computing Predecessors
- Backward Reachability


Petri Nets
Model
Configurations
Transitions
Ordering
Monotonicity
Upward Closed Sets
Computing Predecessors
Backward Reachability

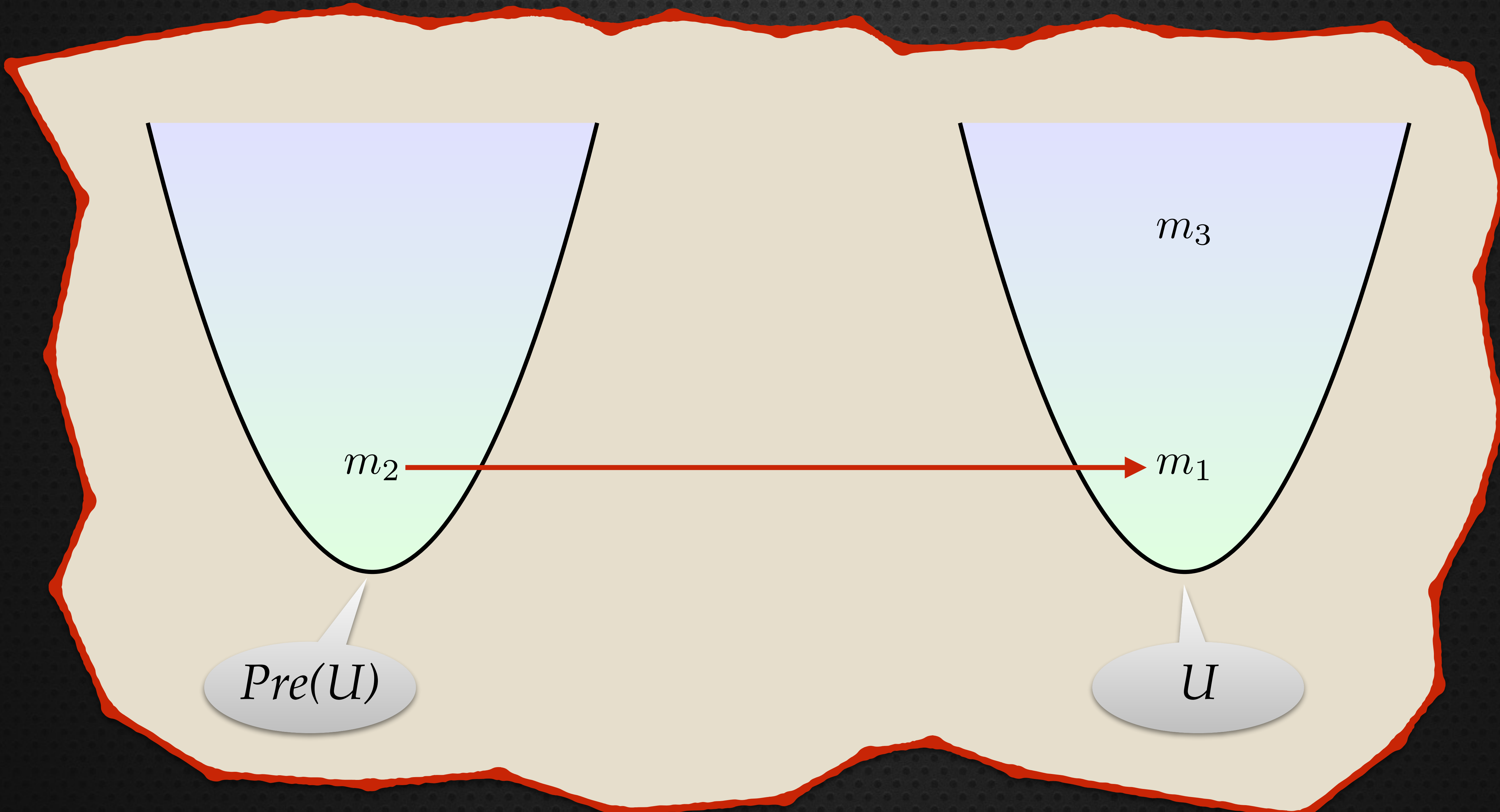
Predecessors



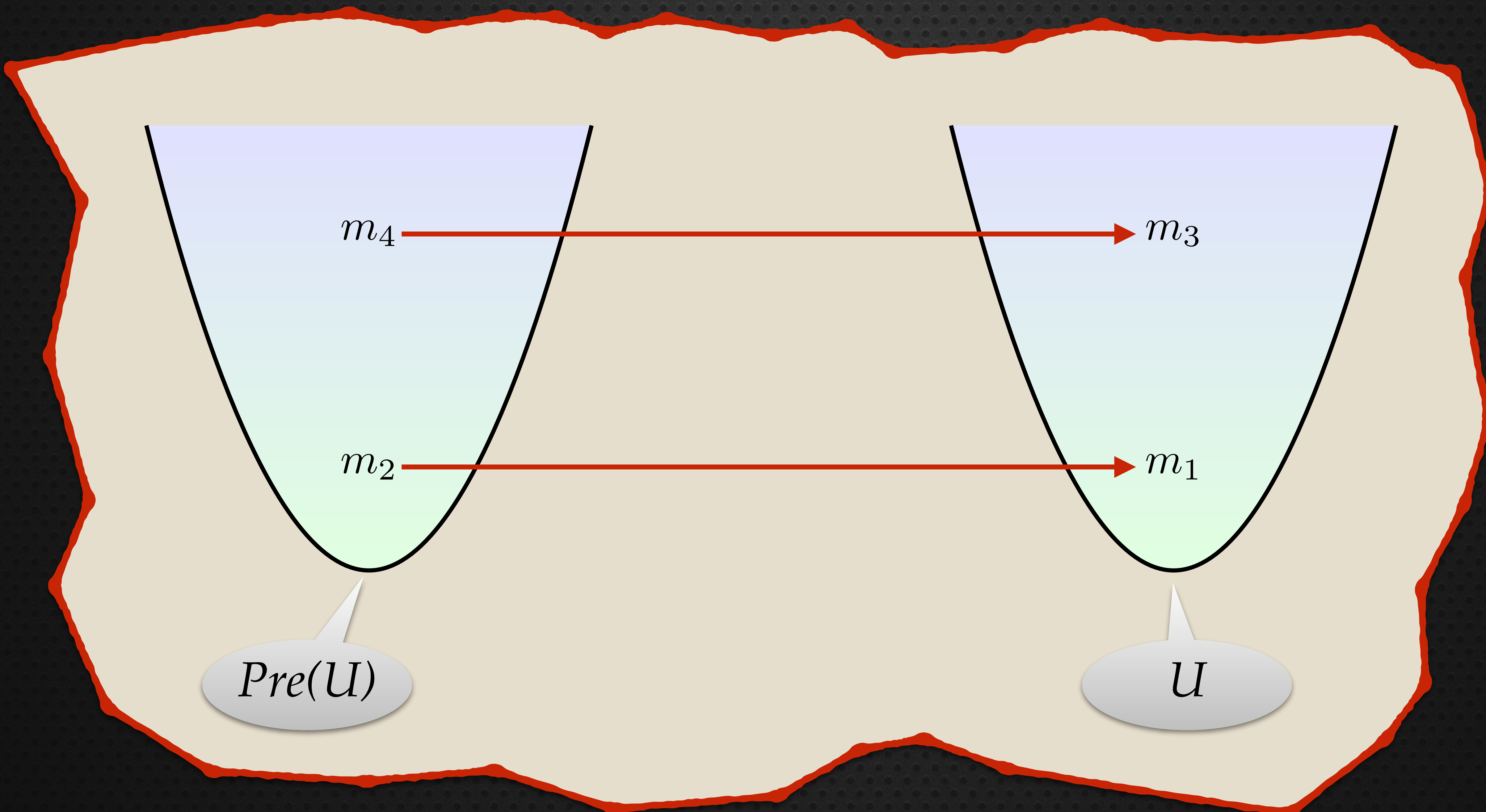
Predecessors



Predecessors



Predecessors



Predecessors

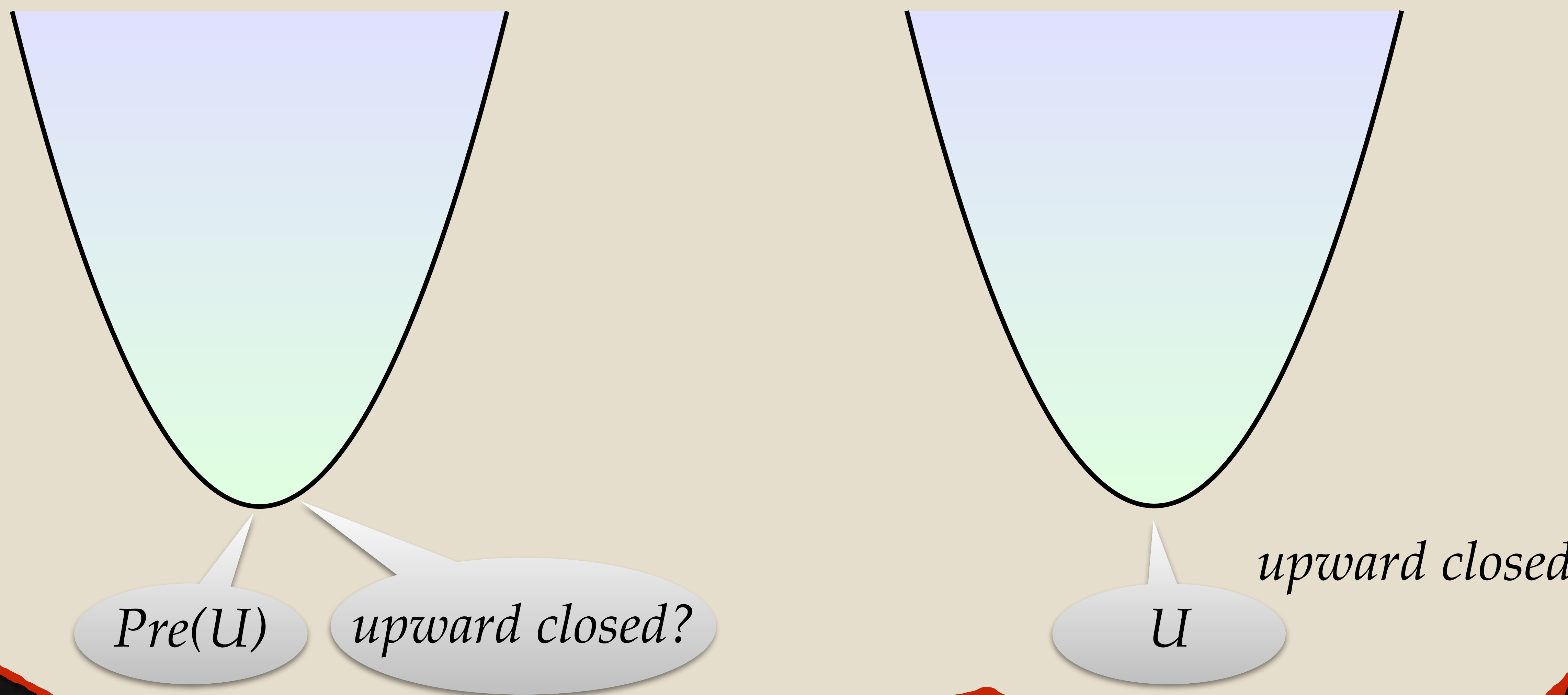
$$\begin{tikzcd} m_1 \arrow[r] \arrow[d, phantom, "\rotatebox{90}{$\sqsubseteq$}"] & m_2 \arrow[d, phantom, "\rotatebox{90}{$\sqsubseteq$}"] \\ m_3 \arrow[r, "\exists"] & m_4 \end{tikzcd}$$

Monotonicity: UC persevered by *Pre*

Predecessors

Monotonicity: UC persevered by *Pre*

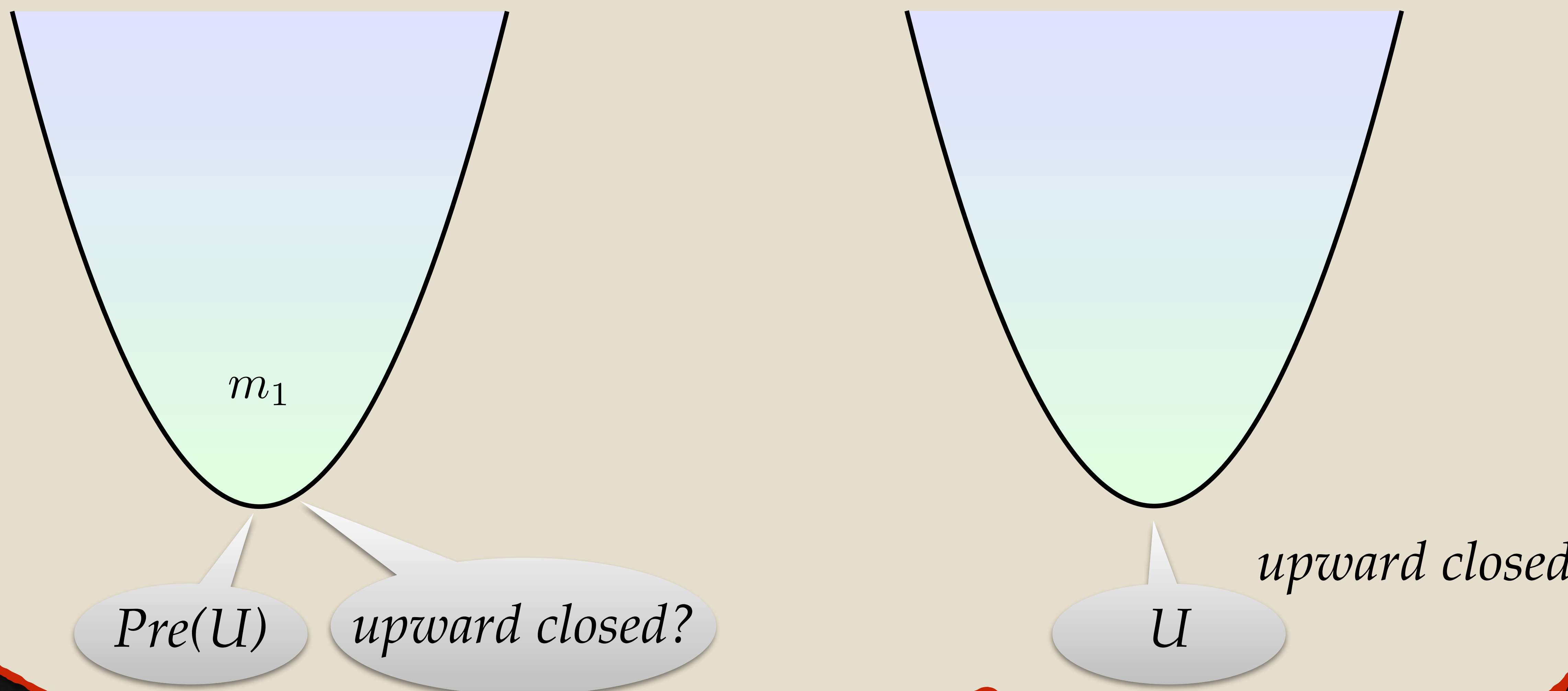
$$
\begin{tikzcd}
m_1 \arrow[r] \arrow[d, phantom, sloped, "\sqsubseteq"] & m_2 \arrow[d, phantom, sloped, "\sqsubseteq"] \\
m_3 \arrow[r] & \exists\, m_4
\end{tikzcd}
$$



Predecessors

Monotonicity: UC persevered by *Pre*

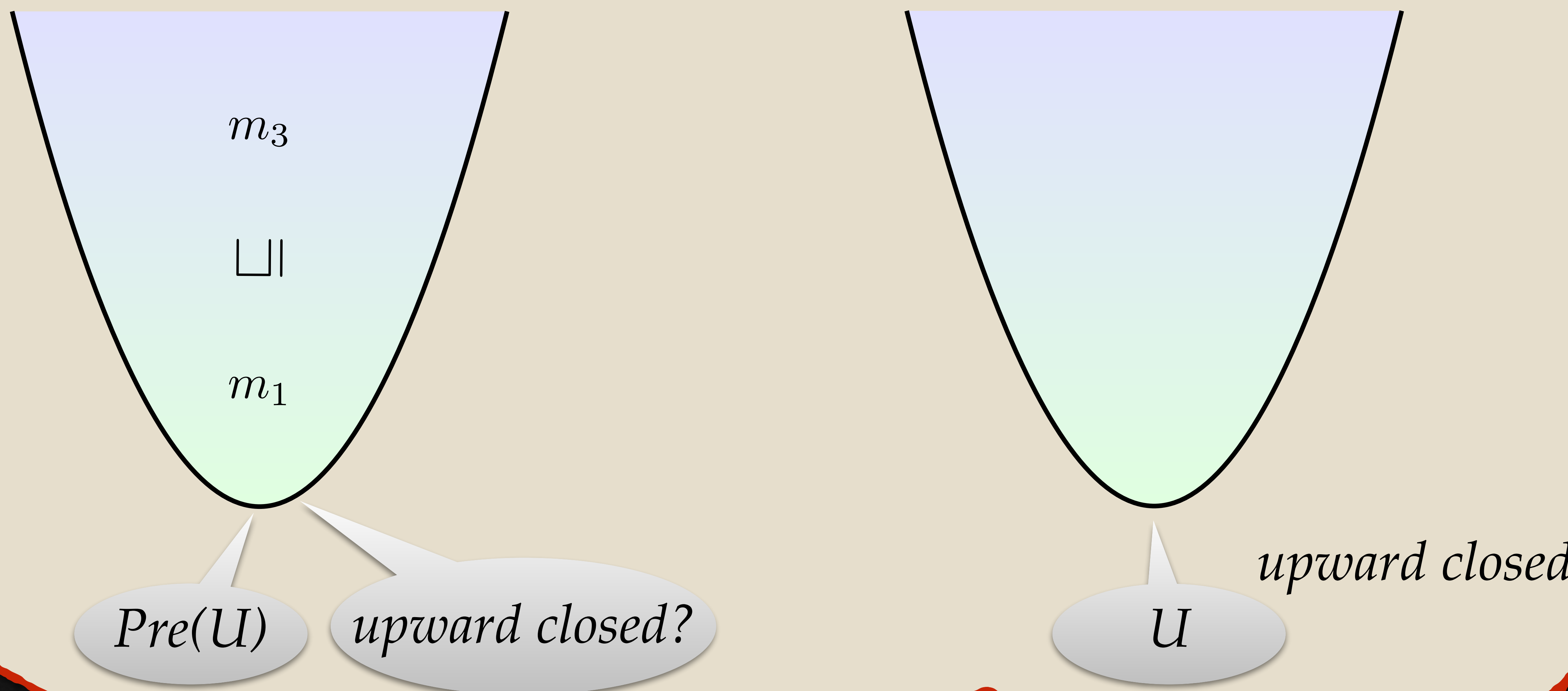
$$\begin{array}{ccc} m_1 & \longrightarrow & m_2 \\ \rotatebox{-90}{$\sqsubseteq$} & & \rotatebox{-90}{$\sqsubseteq$} \\ m_3 & \longrightarrow & \exists\; m_4 \end{array}$$



Predecessors

Monotonicity: UC persevered by *Pre*

$$
\begin{array}{ccc}
m_1 & \longrightarrow & m_2 \\
\sqsubseteq & & \sqsubseteq \\
m_3 & \xrightarrow{\exists} & m_4
\end{array}
$$

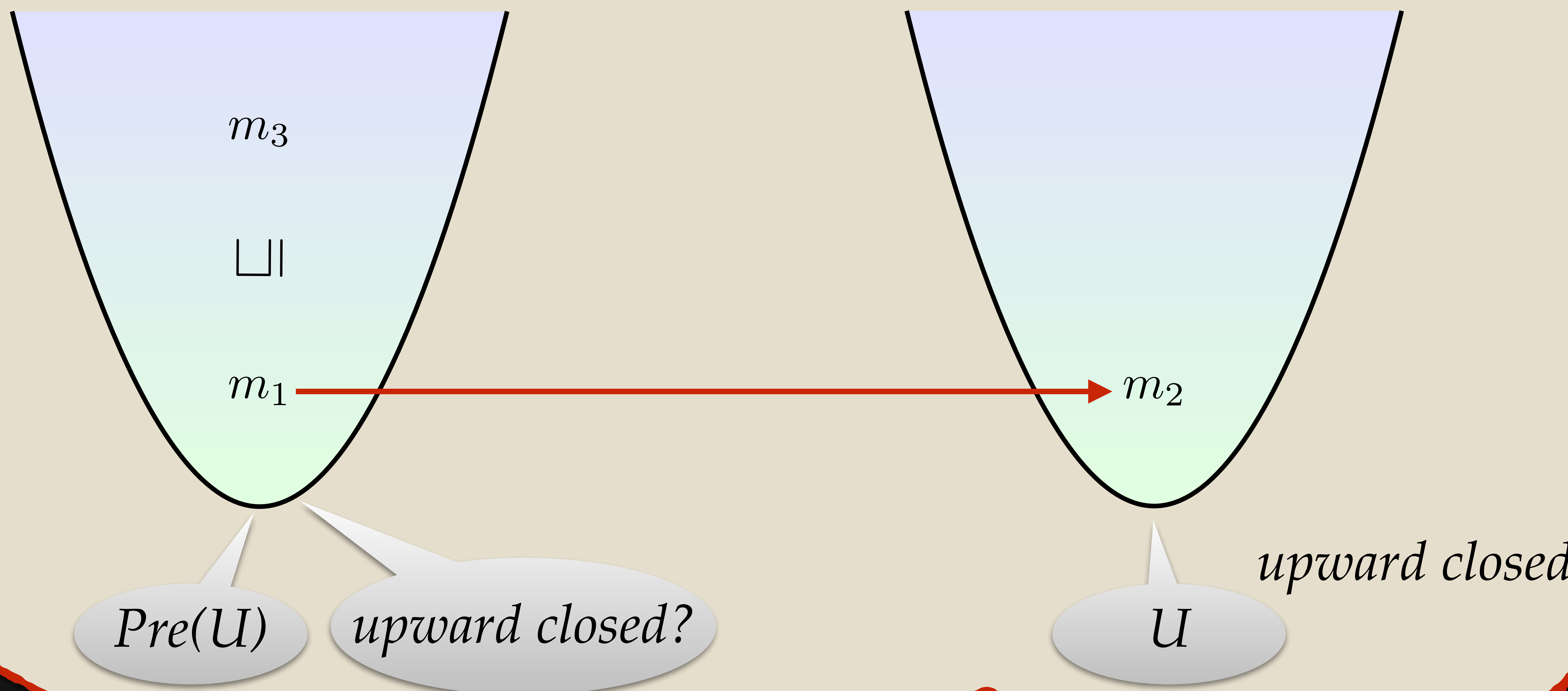


Petri...

Predecessors

Monotonicity: UC persevered by *Pre*

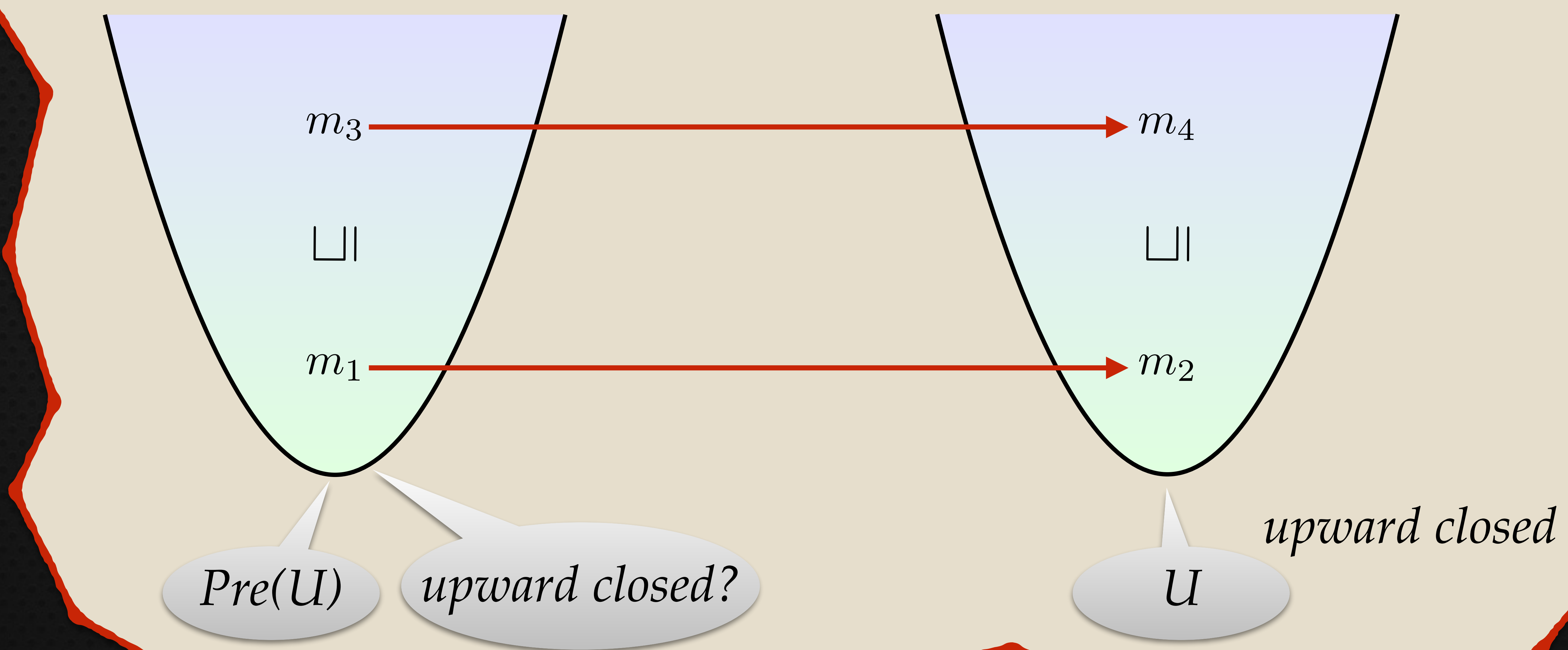
$$\begin{array}{ccc} m_1 & \longrightarrow & m_2 \\ \sqsubseteq & & \sqsubseteq \\ m_3 & \longrightarrow & \exists\, m_4 \end{array}$$



Predecessors

$$\begin{tikzcd} m_1 \arrow[r] \arrow[d, phantom, sloped, "\sqsubseteq"] & m_2 \arrow[d, phantom, sloped, "\sqsubseteq"] \\ m_3 \arrow[r, "\exists"] & m_4 \end{tikzcd}$$

Monotonicity: UC persevered by *Pre*

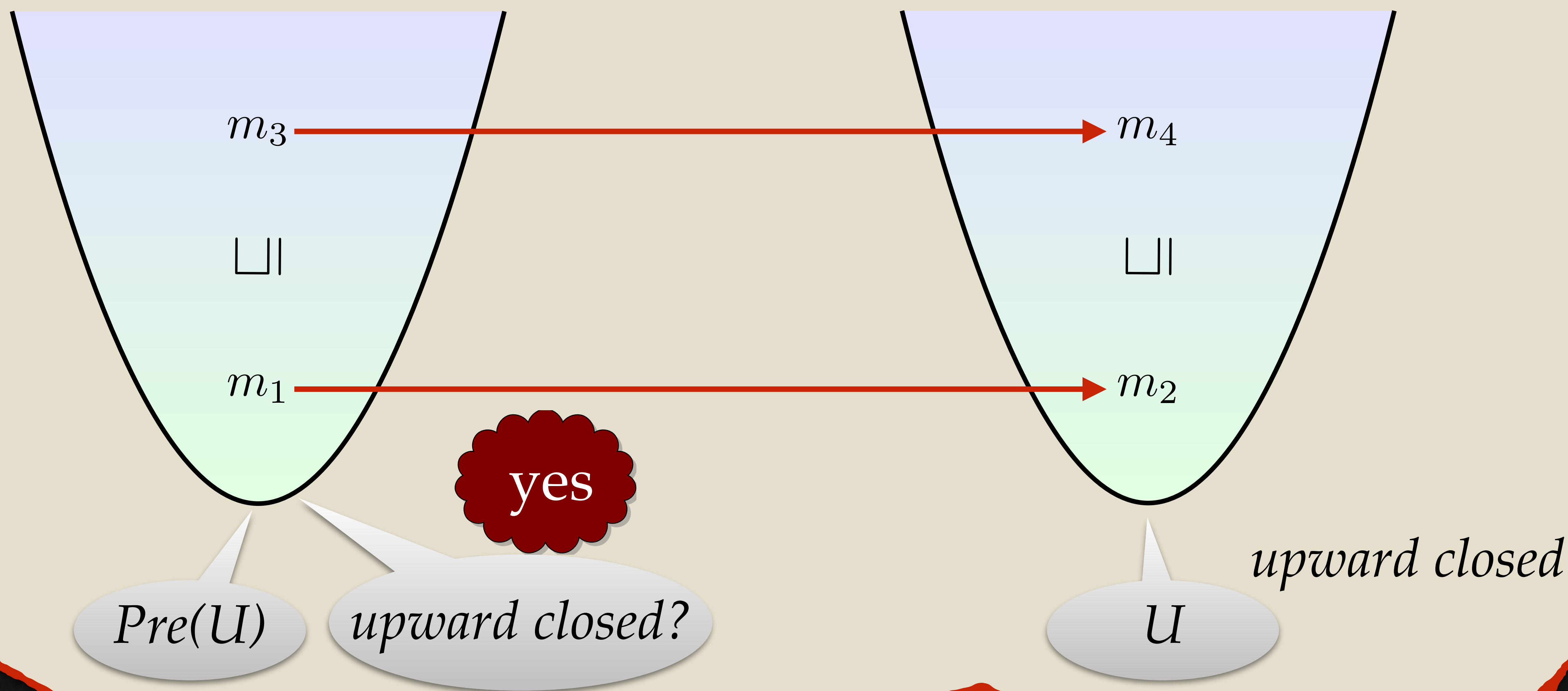


Petri

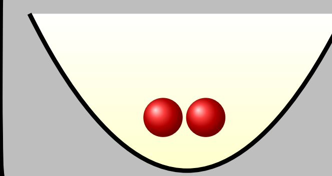
Predecessors

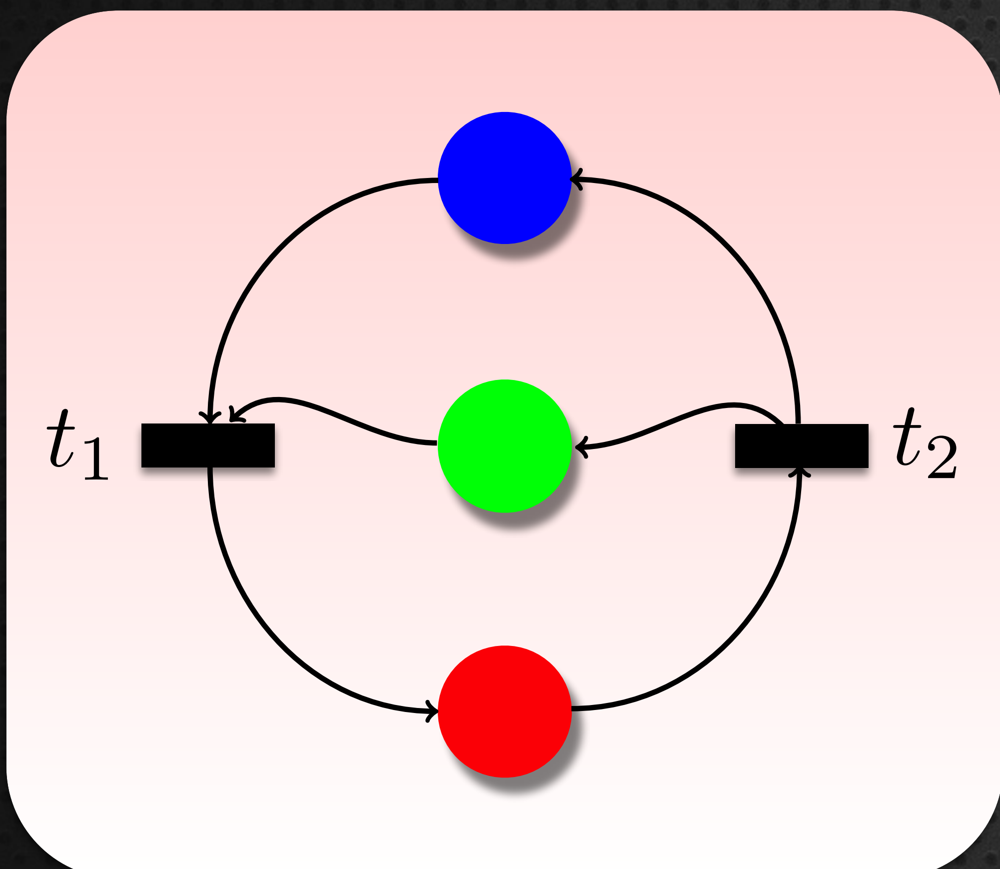
Monotonicity: UC persevered by *Pre*

$$\begin{tikzcd}
m_1 \arrow[r] \arrow[d, phantom, "\rotatebox{-90}{$\sqsubseteq$}"] & m_2 \arrow[d, phantom, "\rotatebox{-90}{$\sqsubseteq$}"] \\
m_3 \arrow[r, "\exists"] & m_4
\end{tikzcd}$$

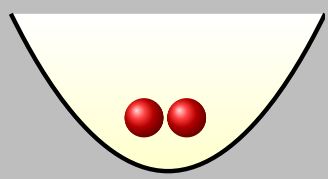
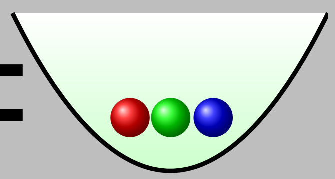


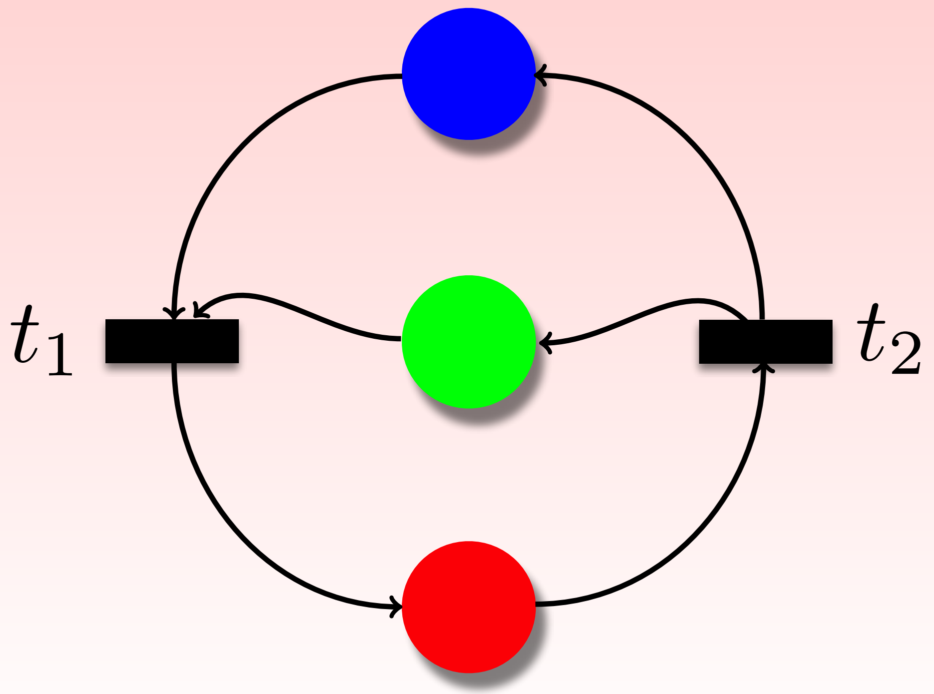
Computing Predecessors

$Pre_{t_1}\Big[$  $\Big] =$

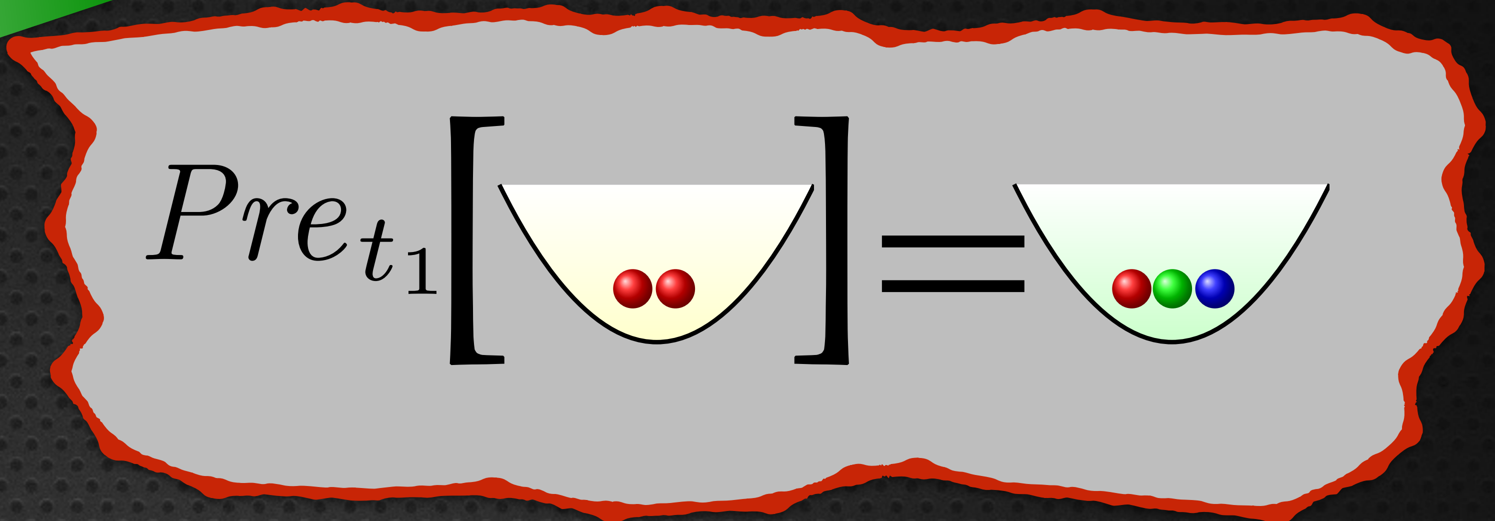


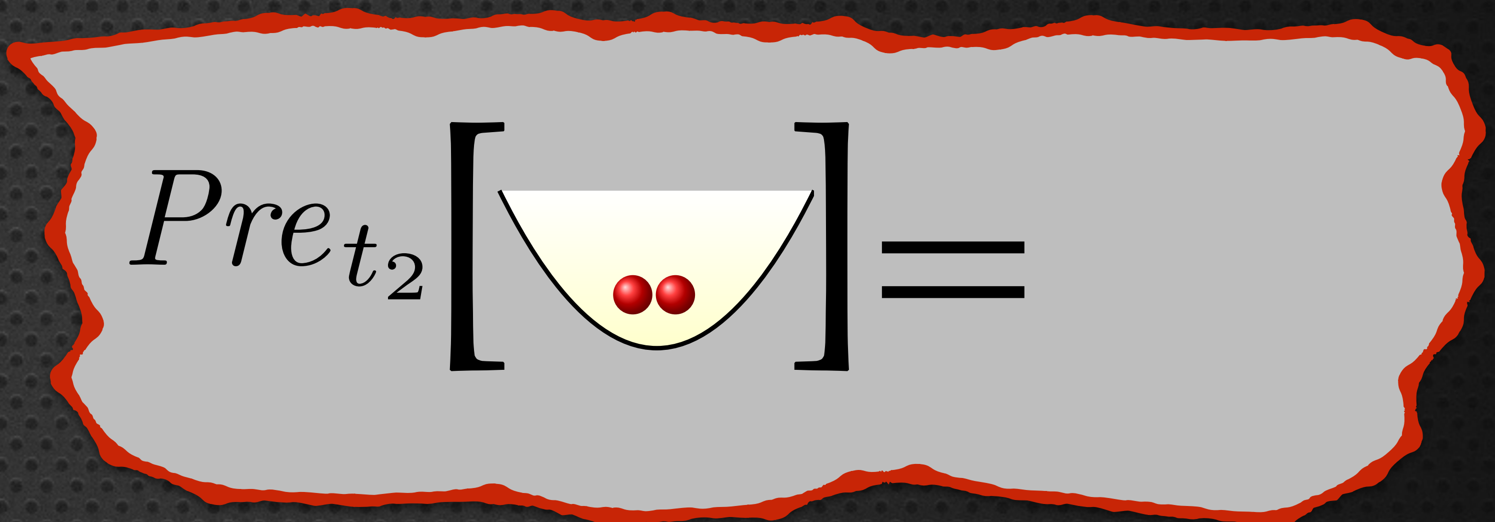
Computing Predecessors

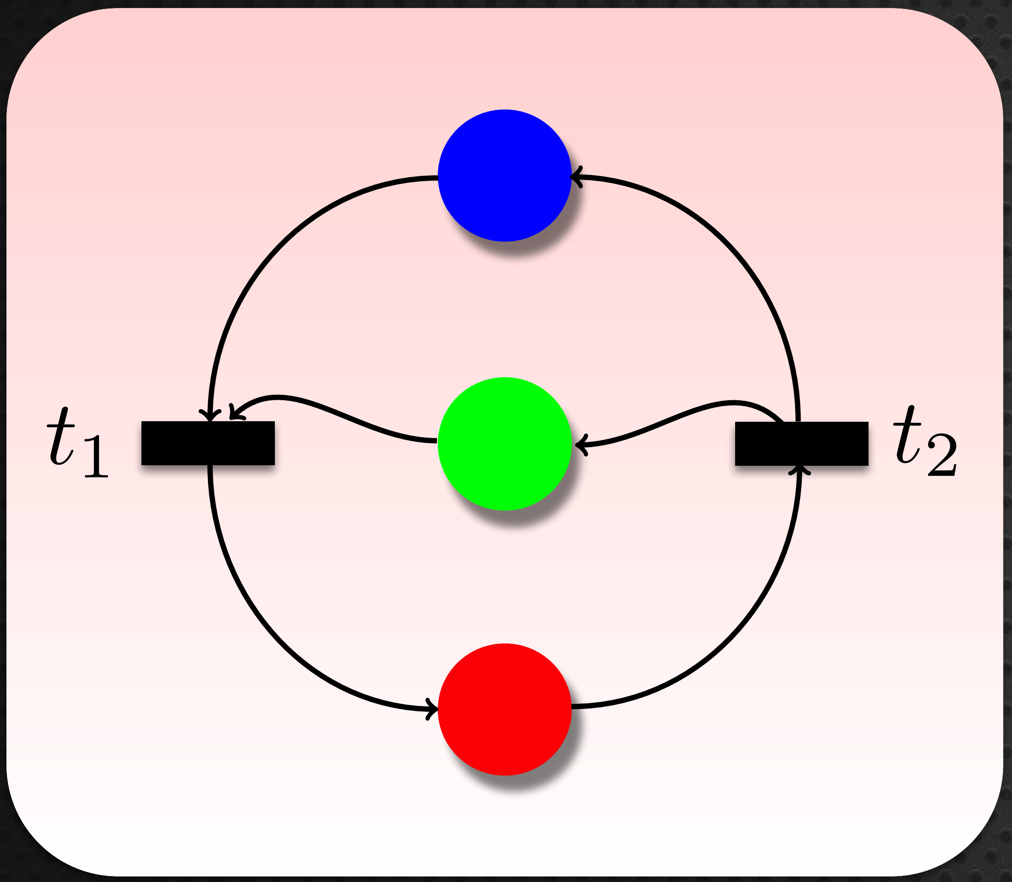
Pre_{t_1} [] = 



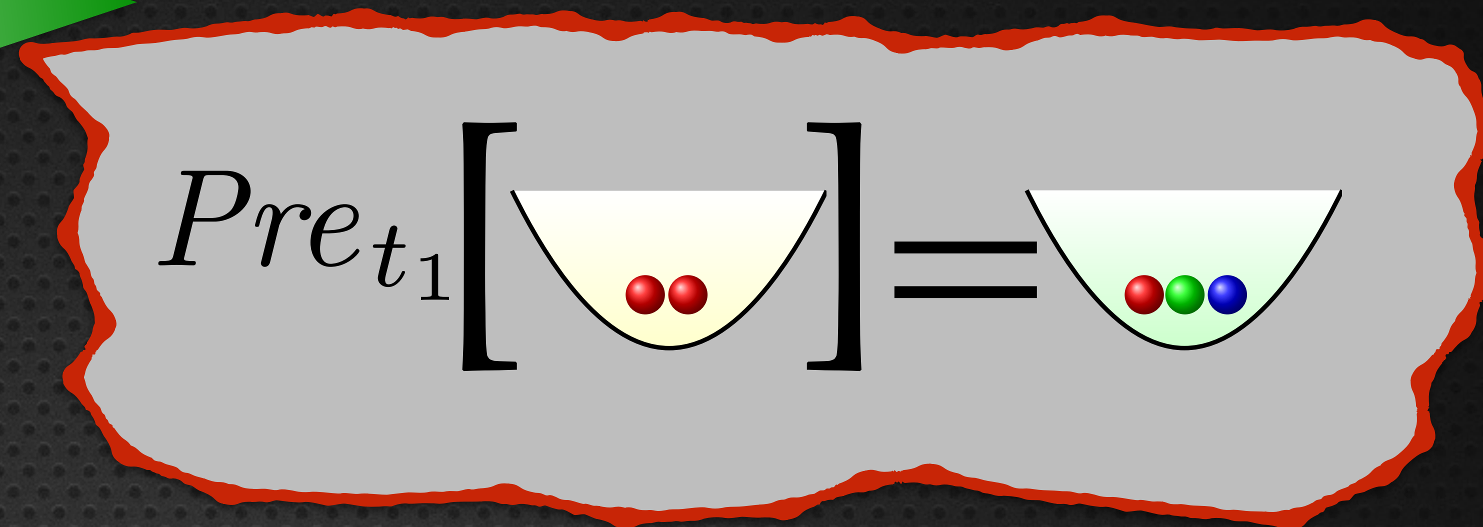
Computing Predecessors

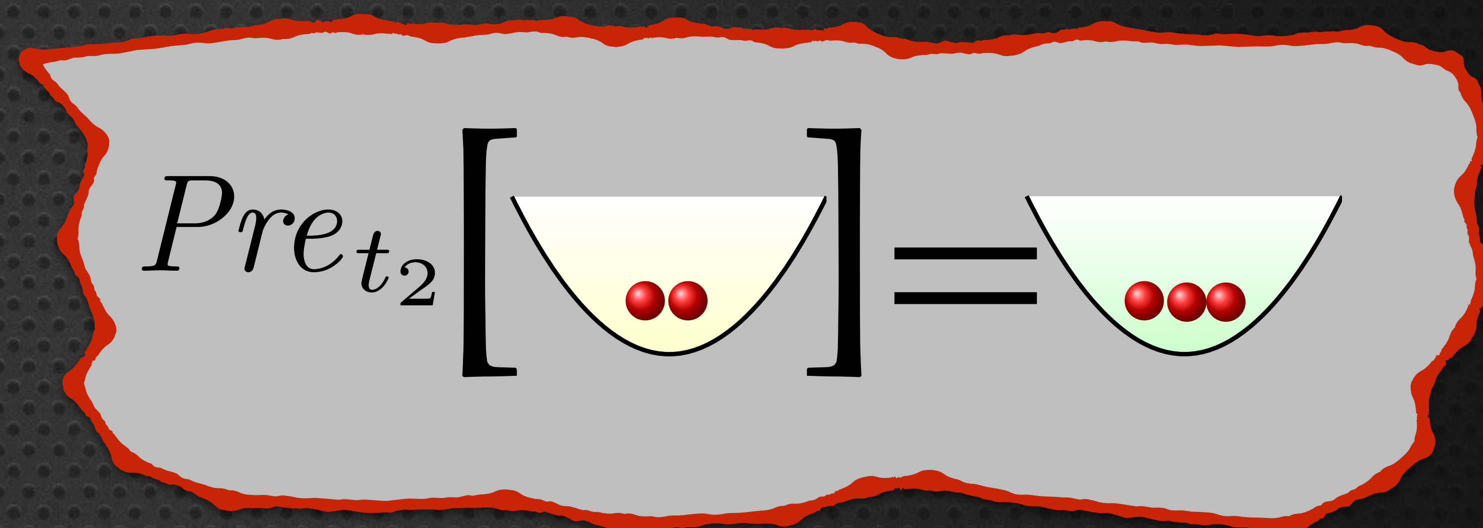


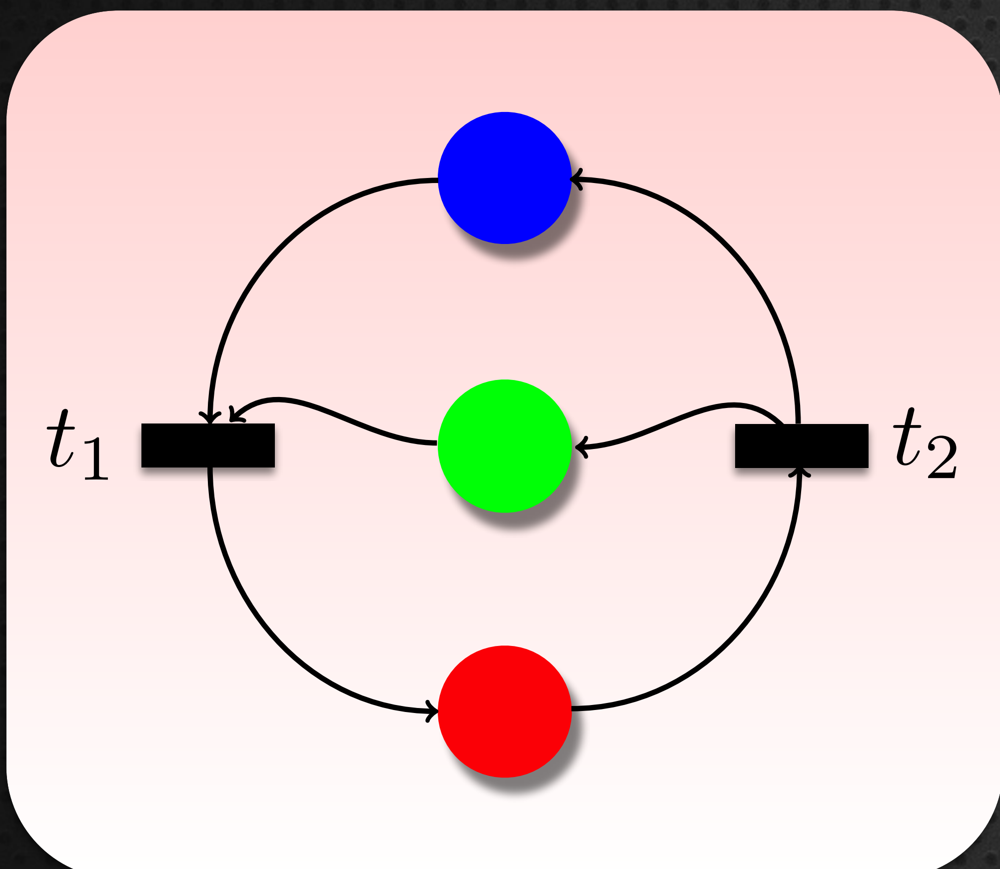




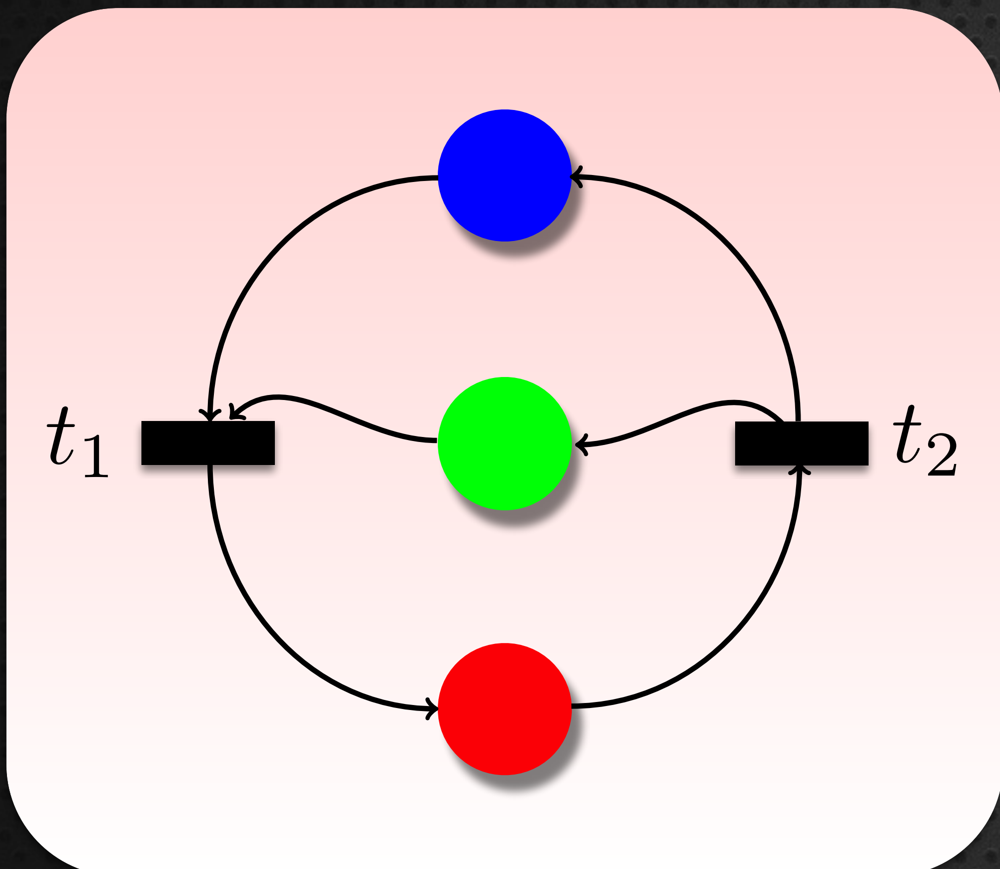
Computing Predecessors

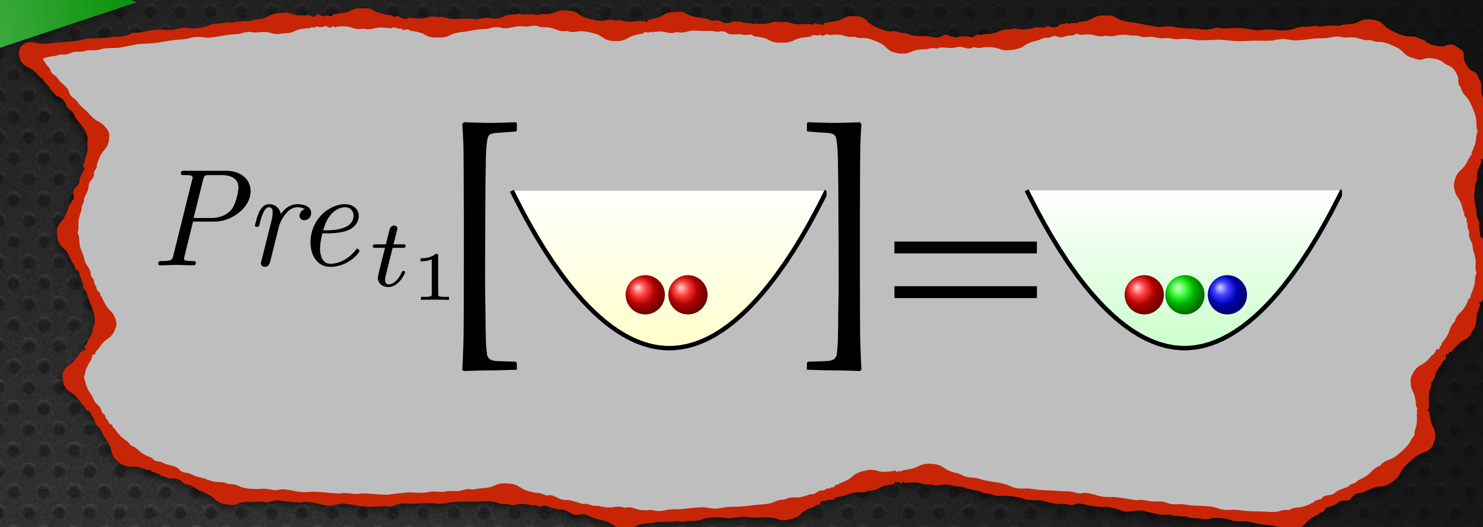


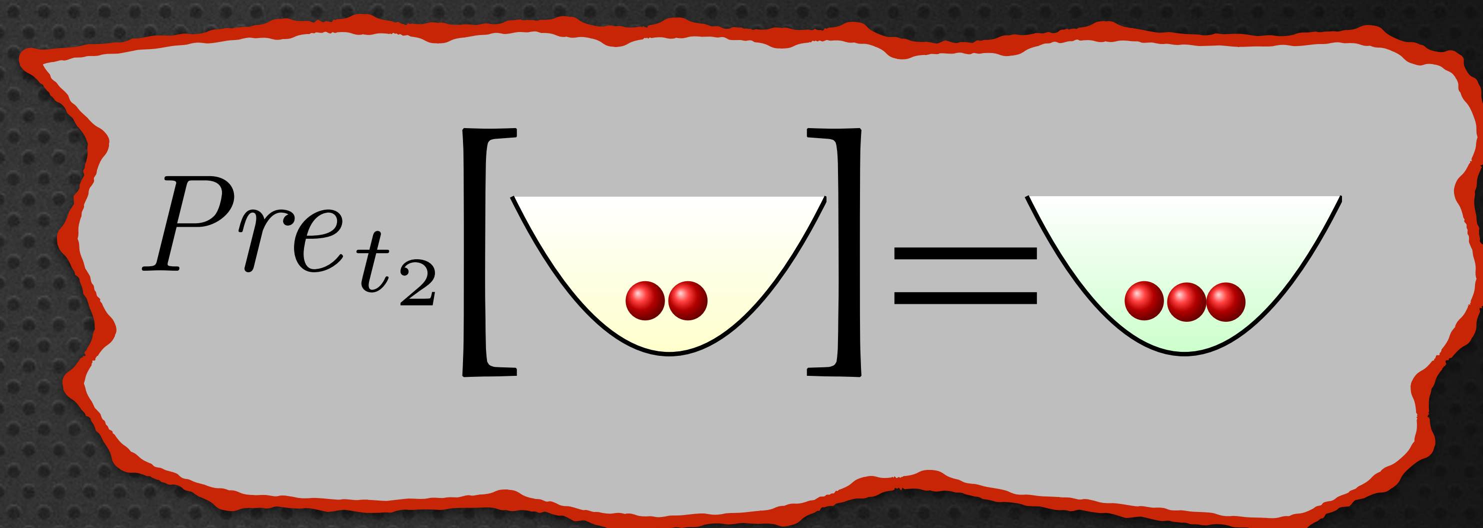


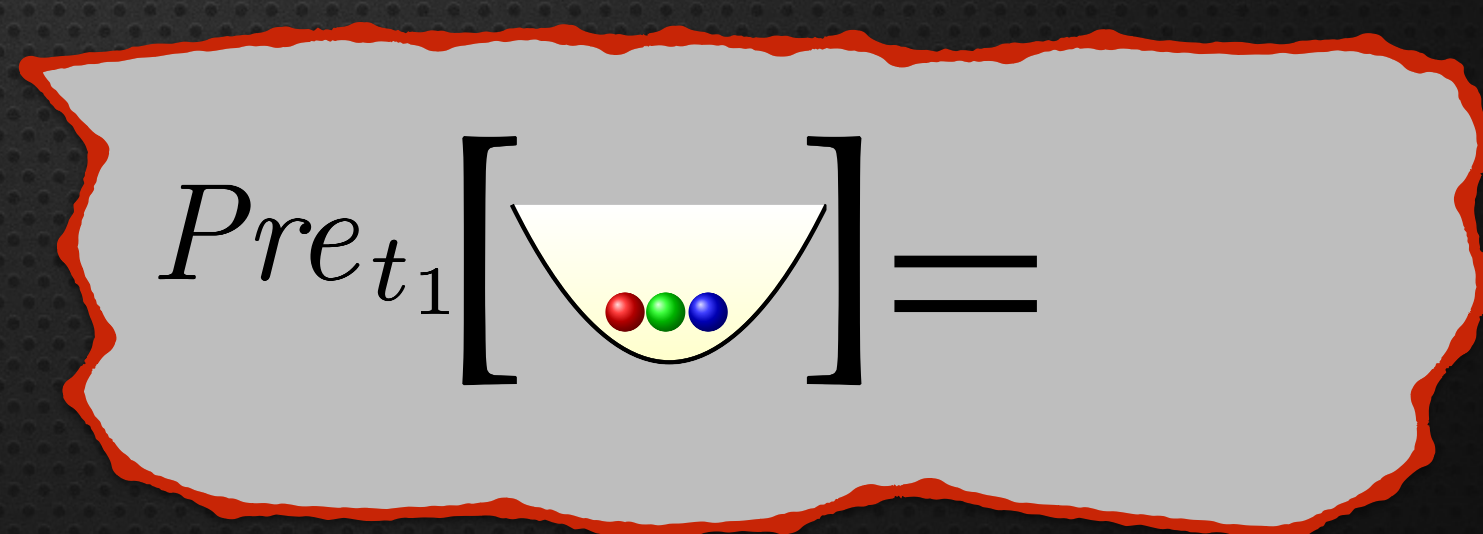


Computing Predecessors

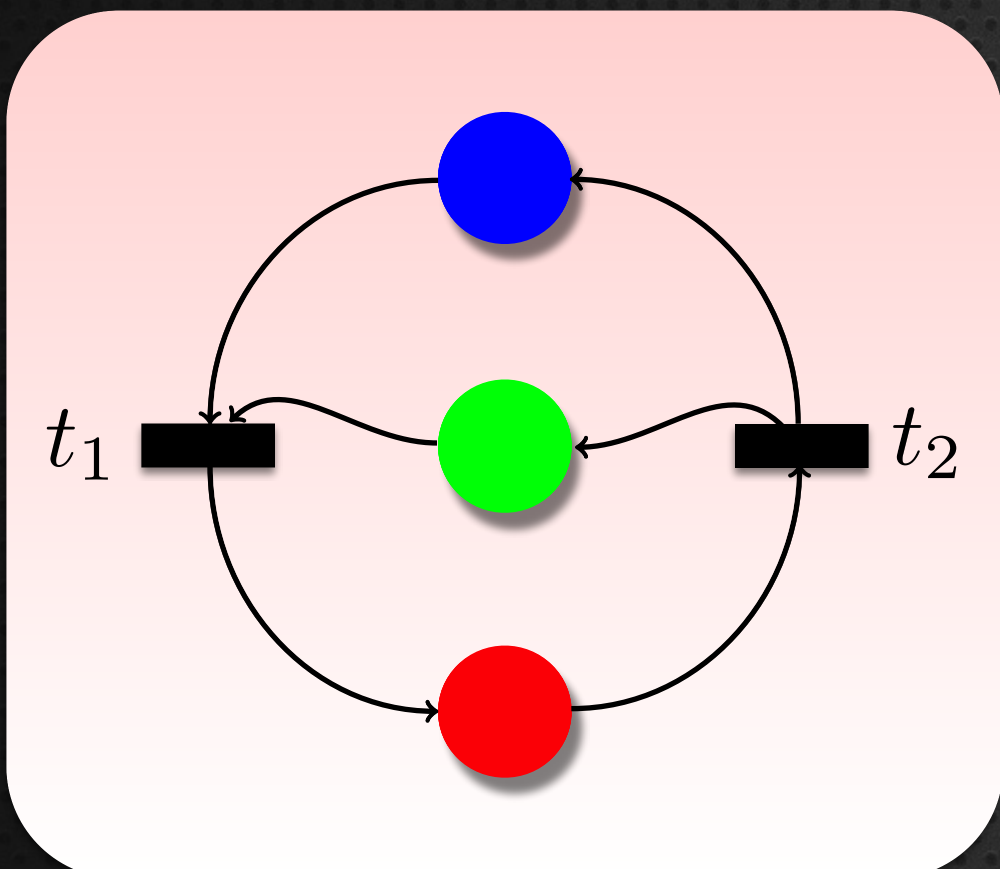


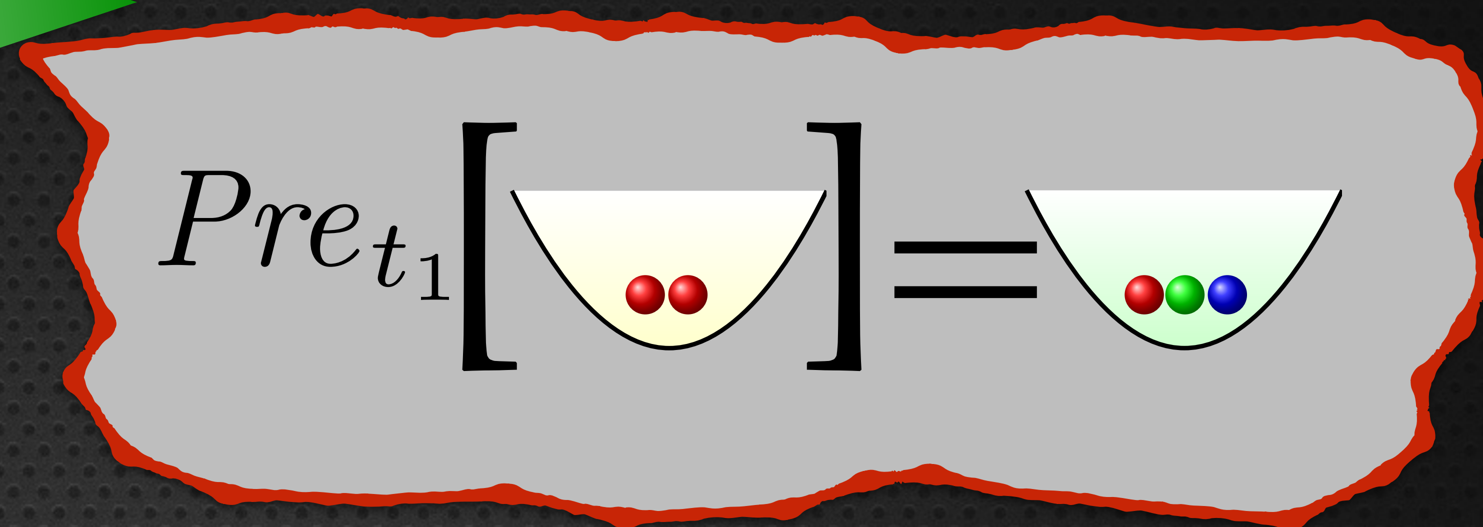


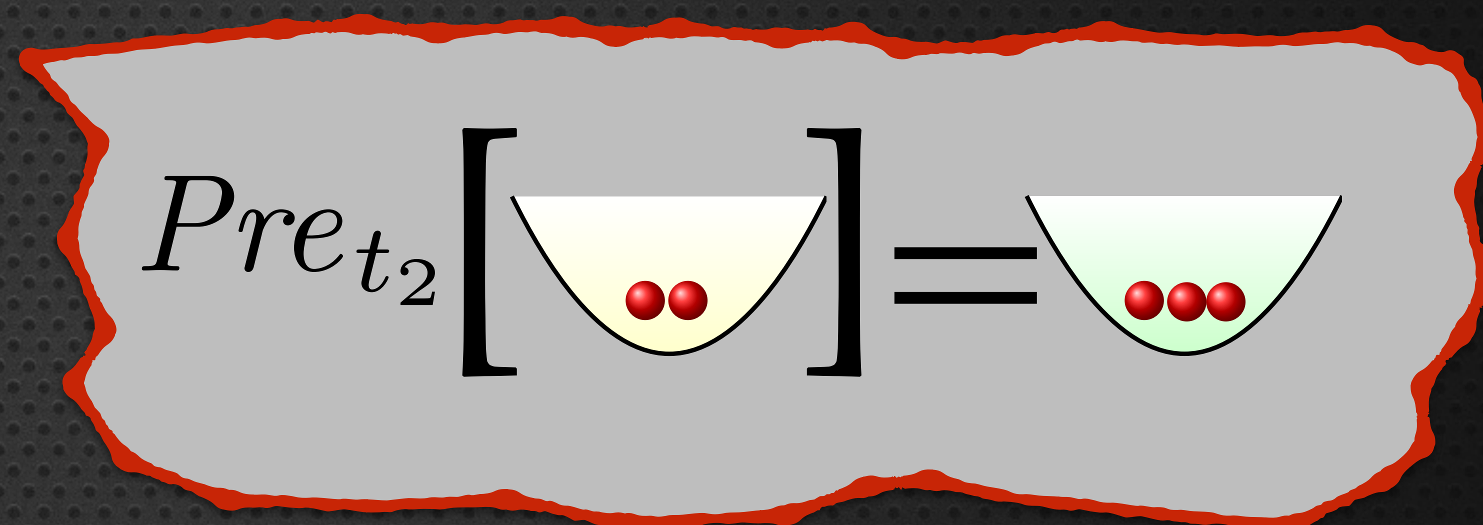


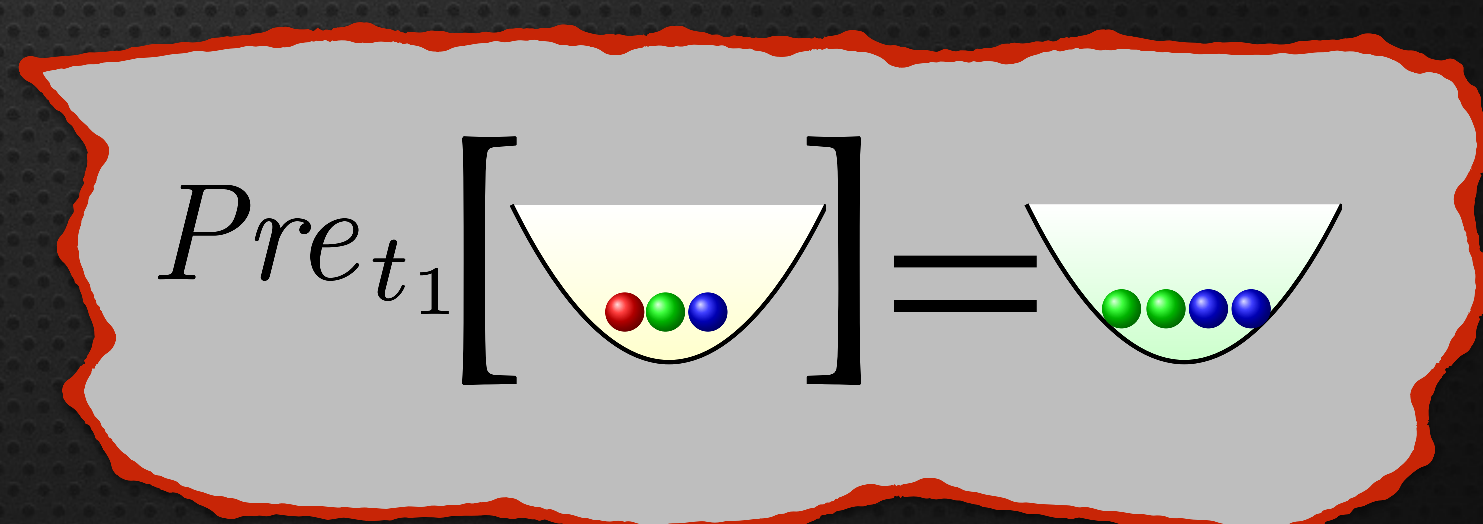


Computing Predecessors



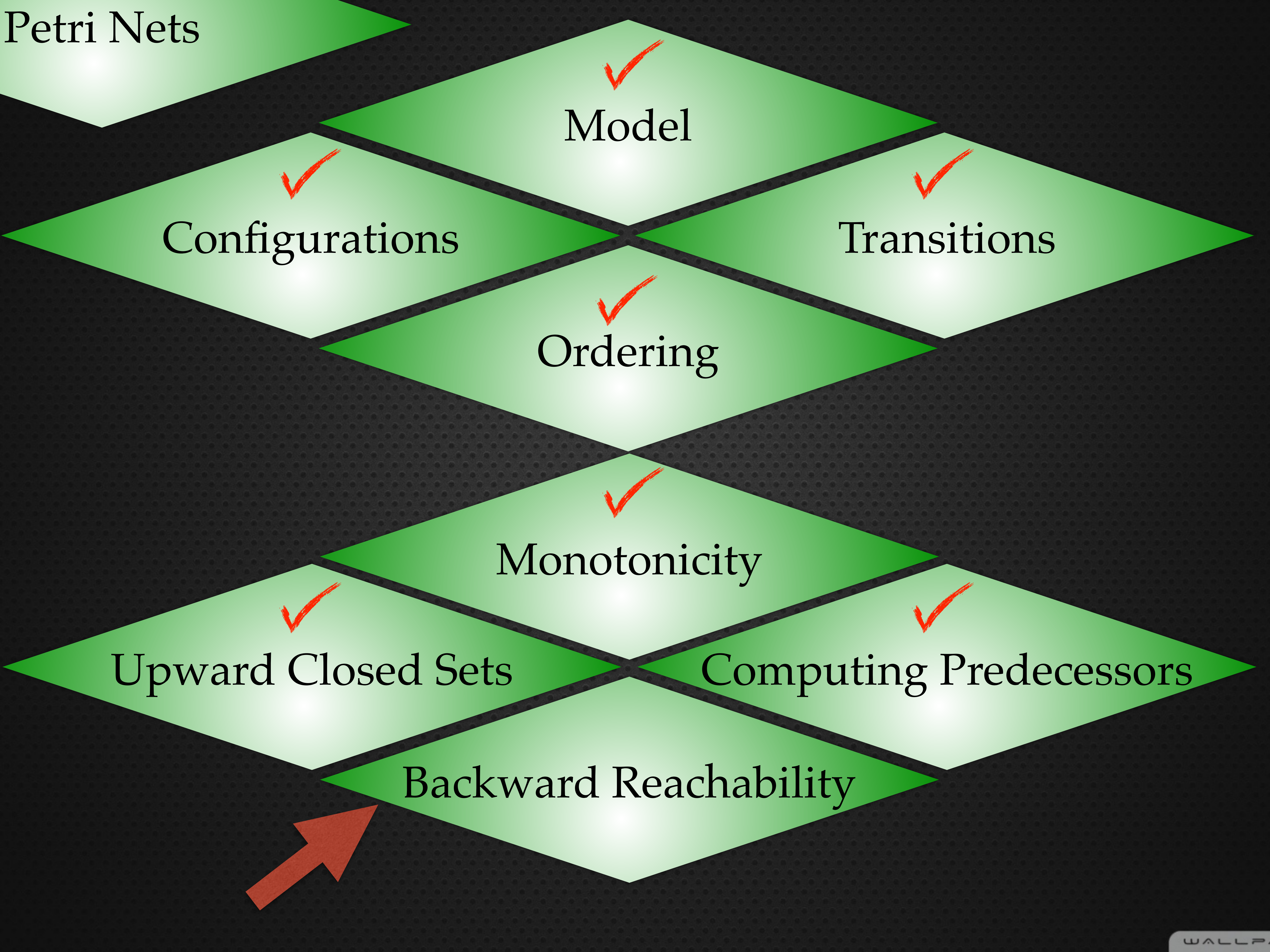




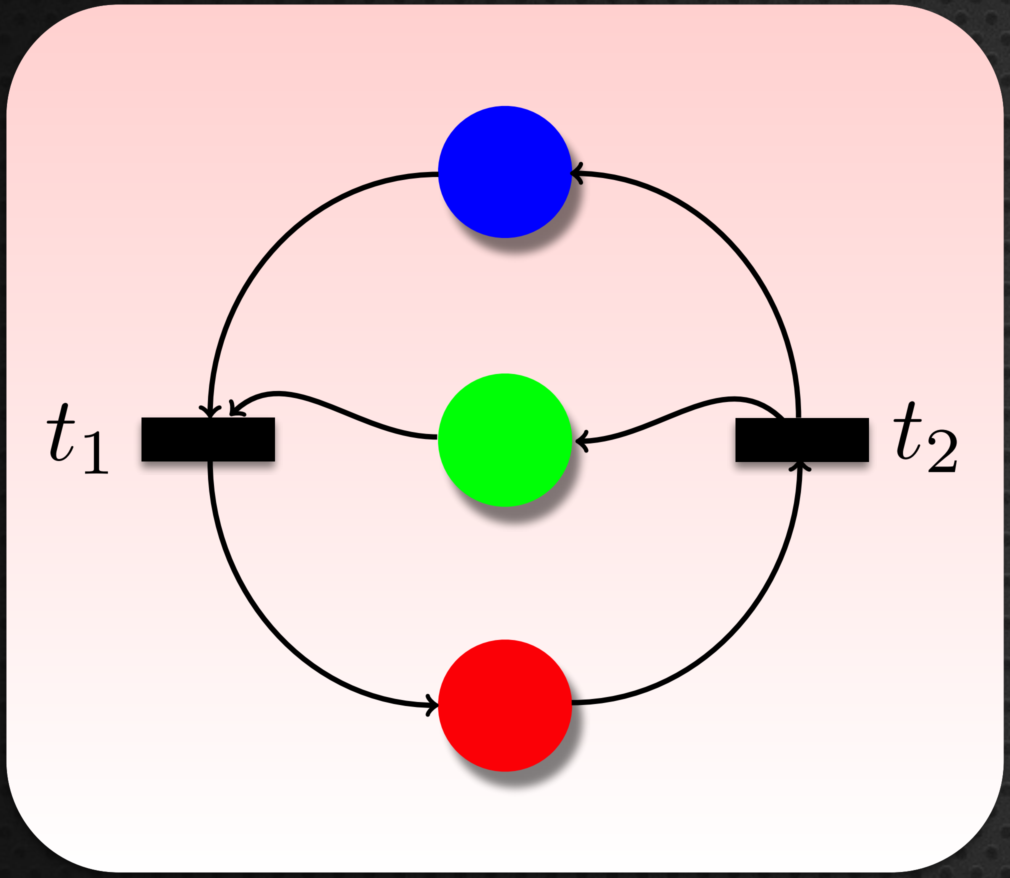


Petri Nets

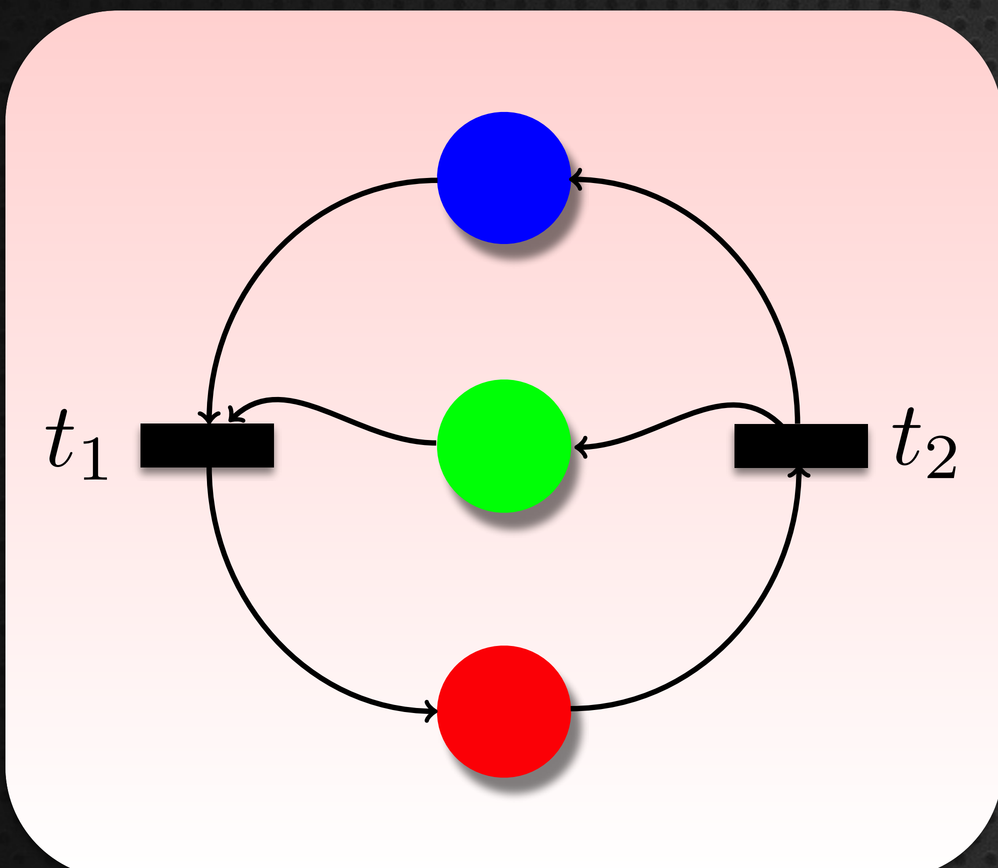
- Model
- Configurations
- Transitions
- Ordering
- Monotonicity
- Upward Closed Sets
- Computing Predecessors
- Backward Reachability


Petri Nets
Model
Configurations
Transitions
Ordering
Monotonicity
Upward Closed Sets
Computing Predecessors
Backward Reachability

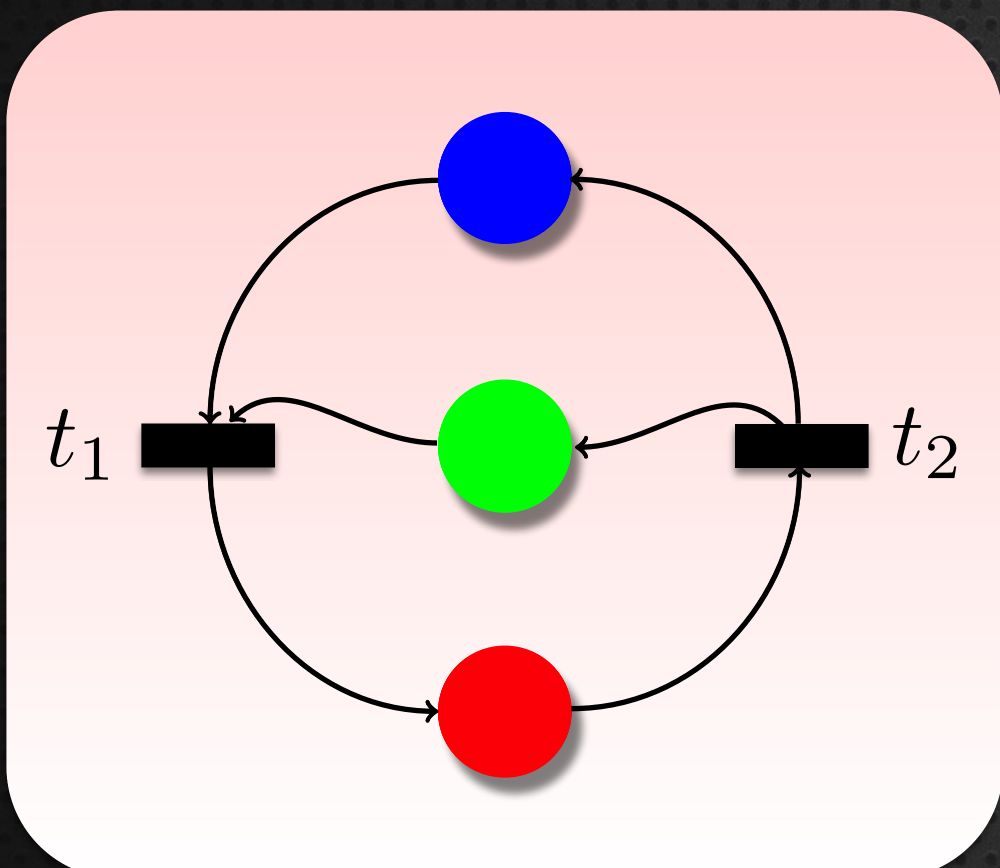
Backward Reachability

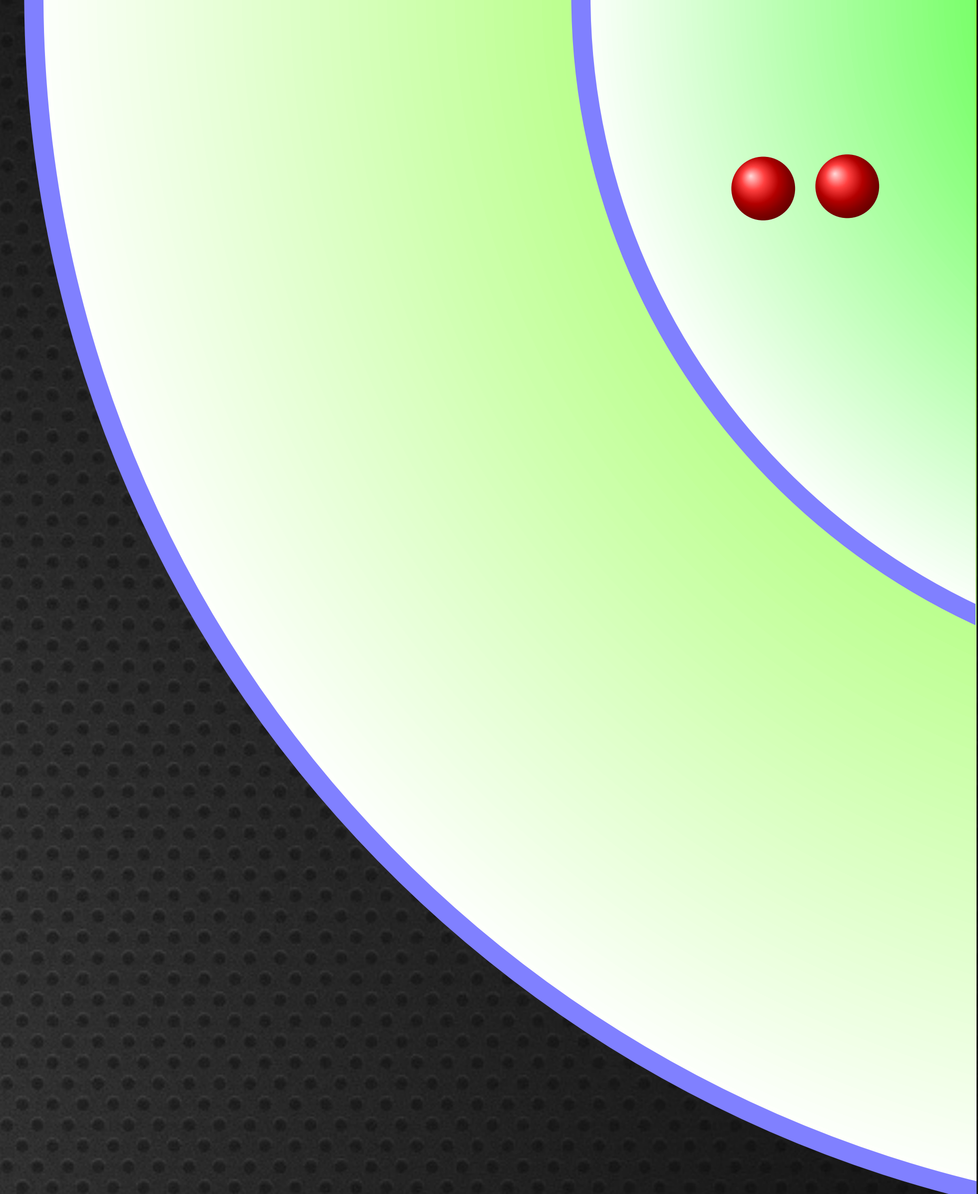


Backward Reachability

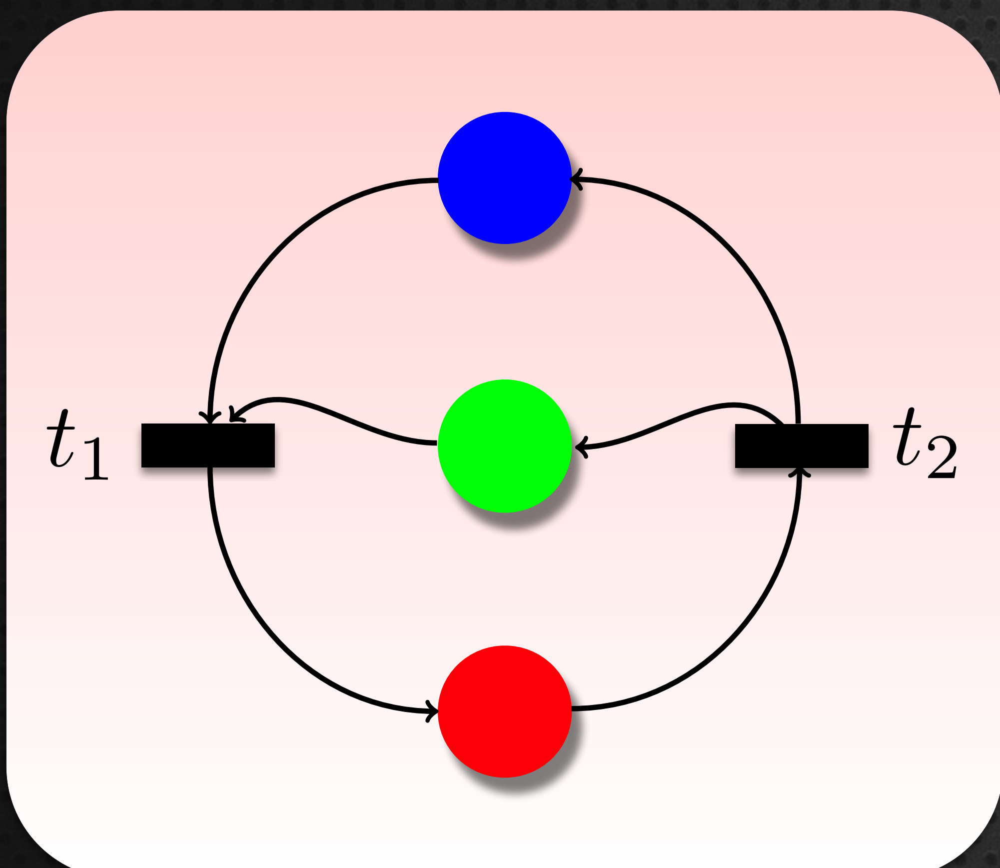


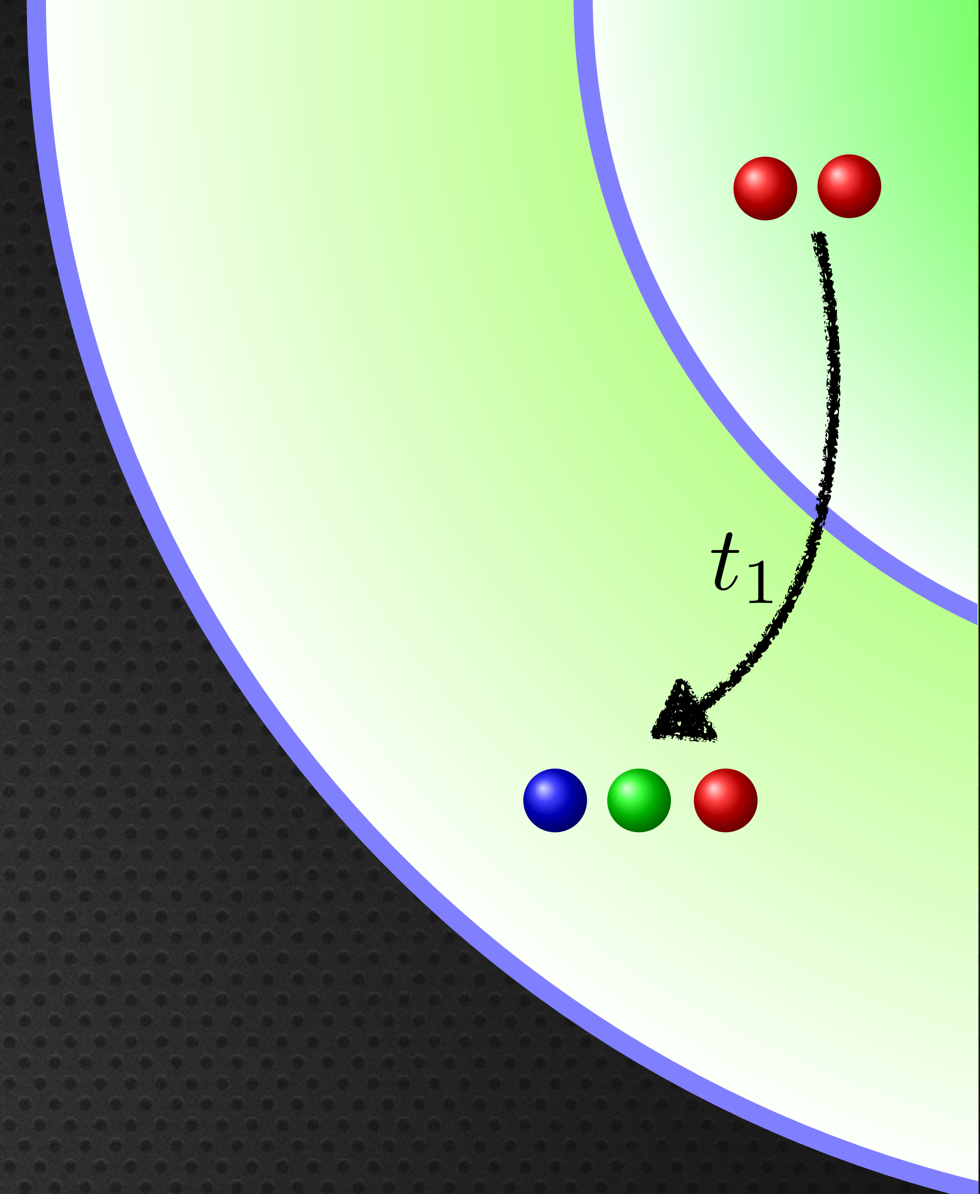
Backward Reachability



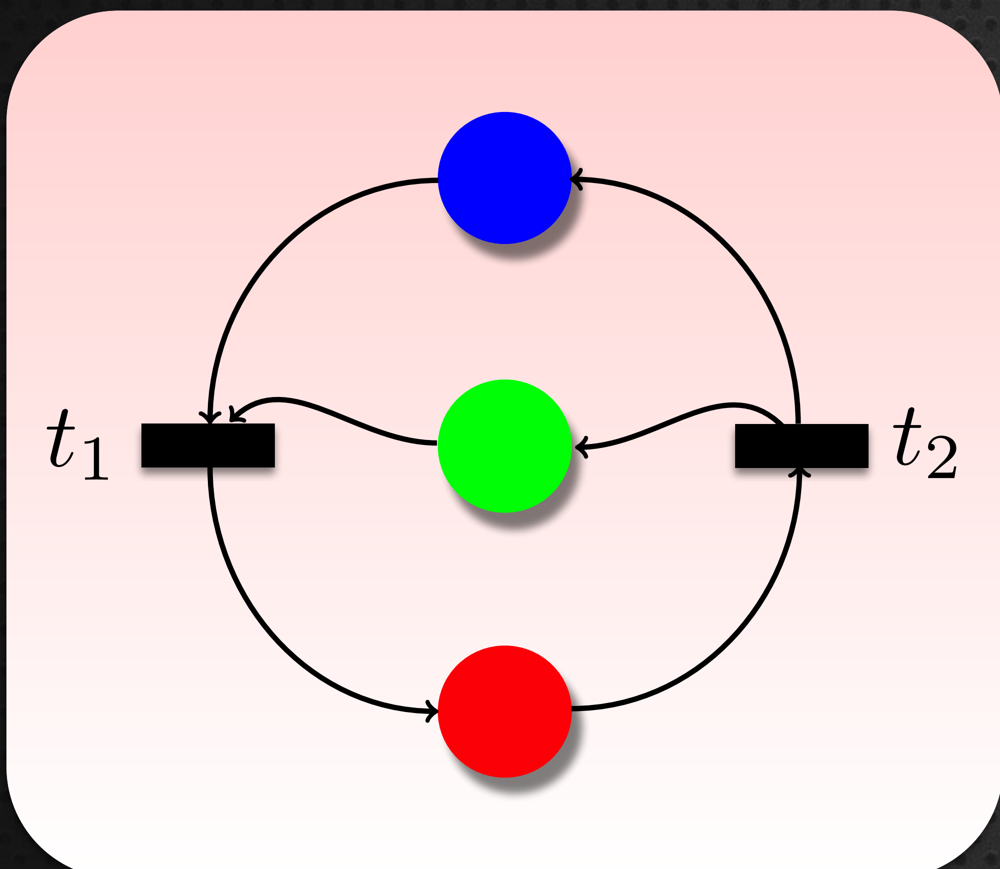


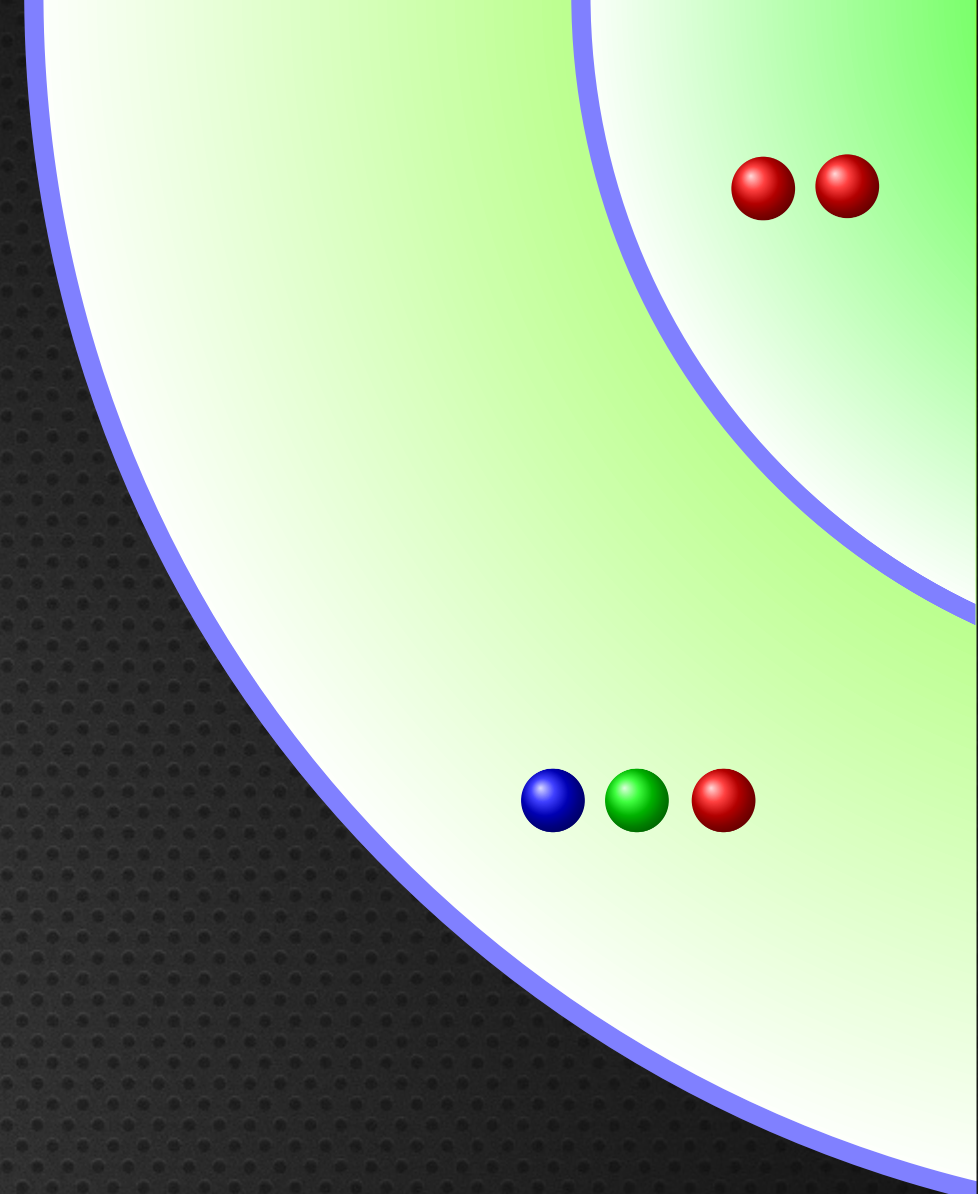
Backward Reachability



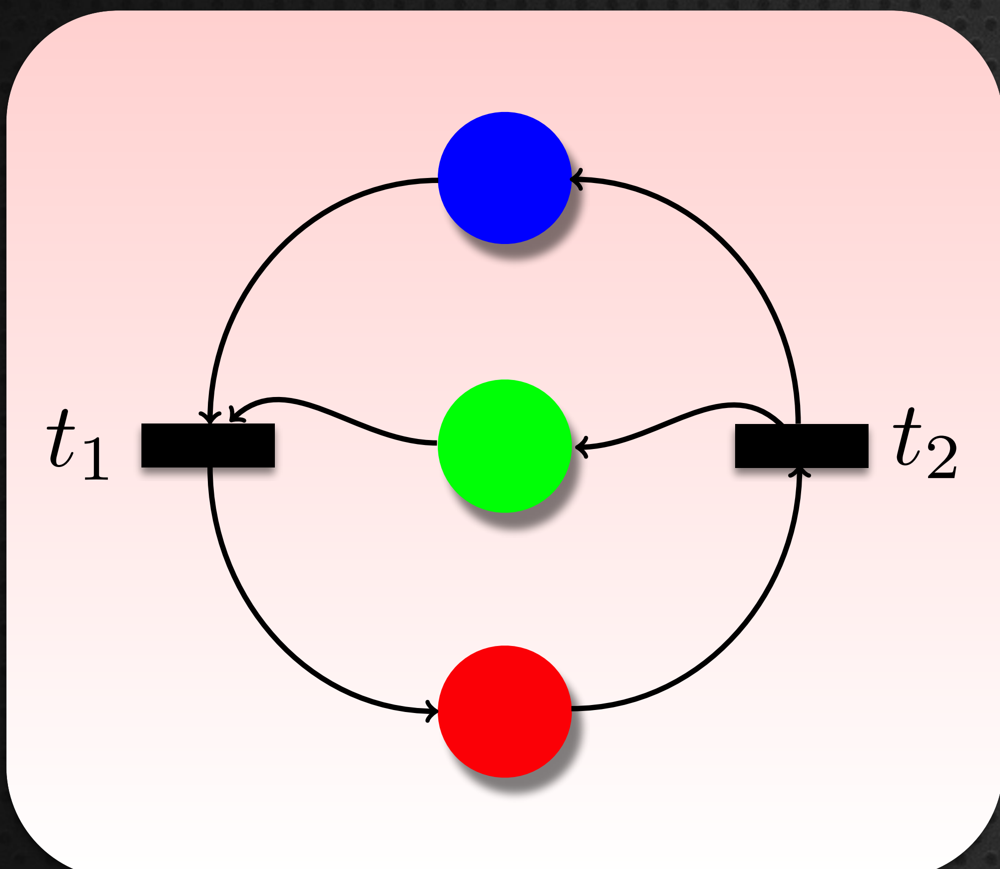


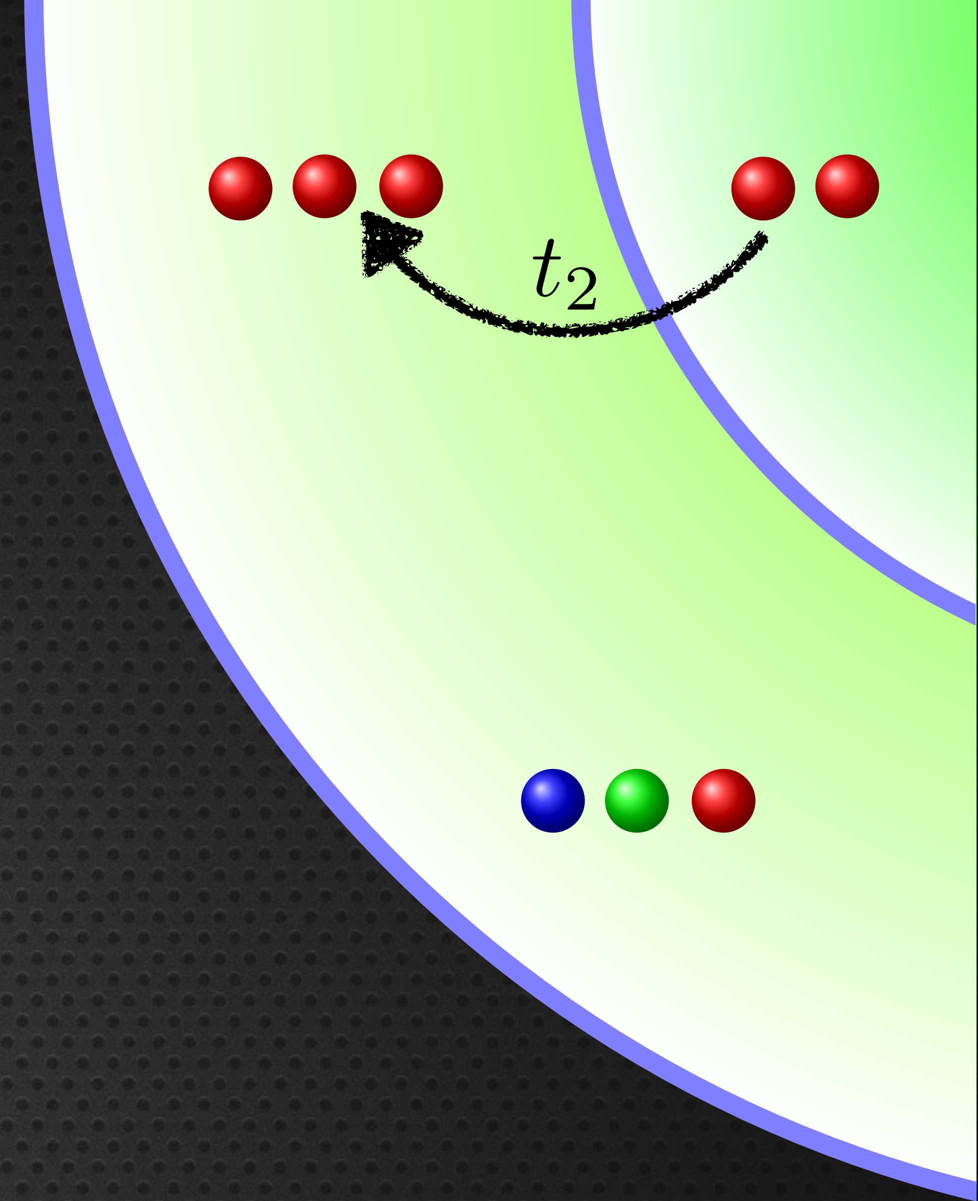
Backward Reachability



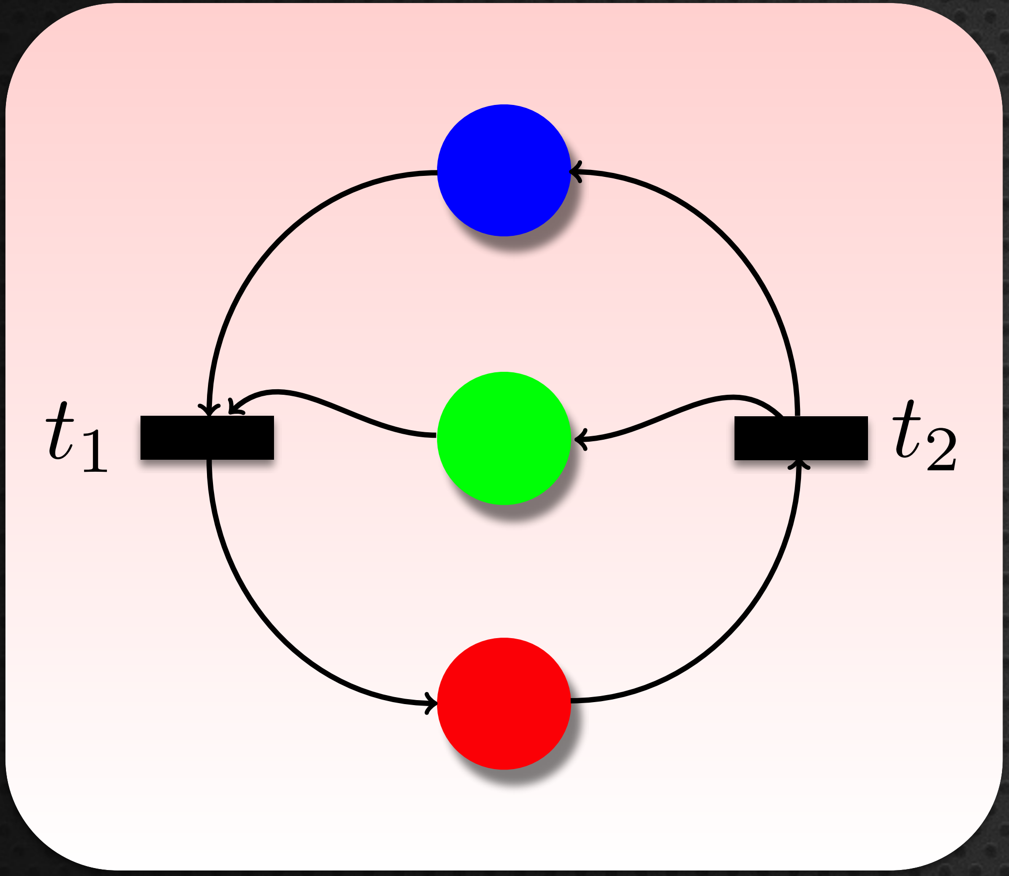


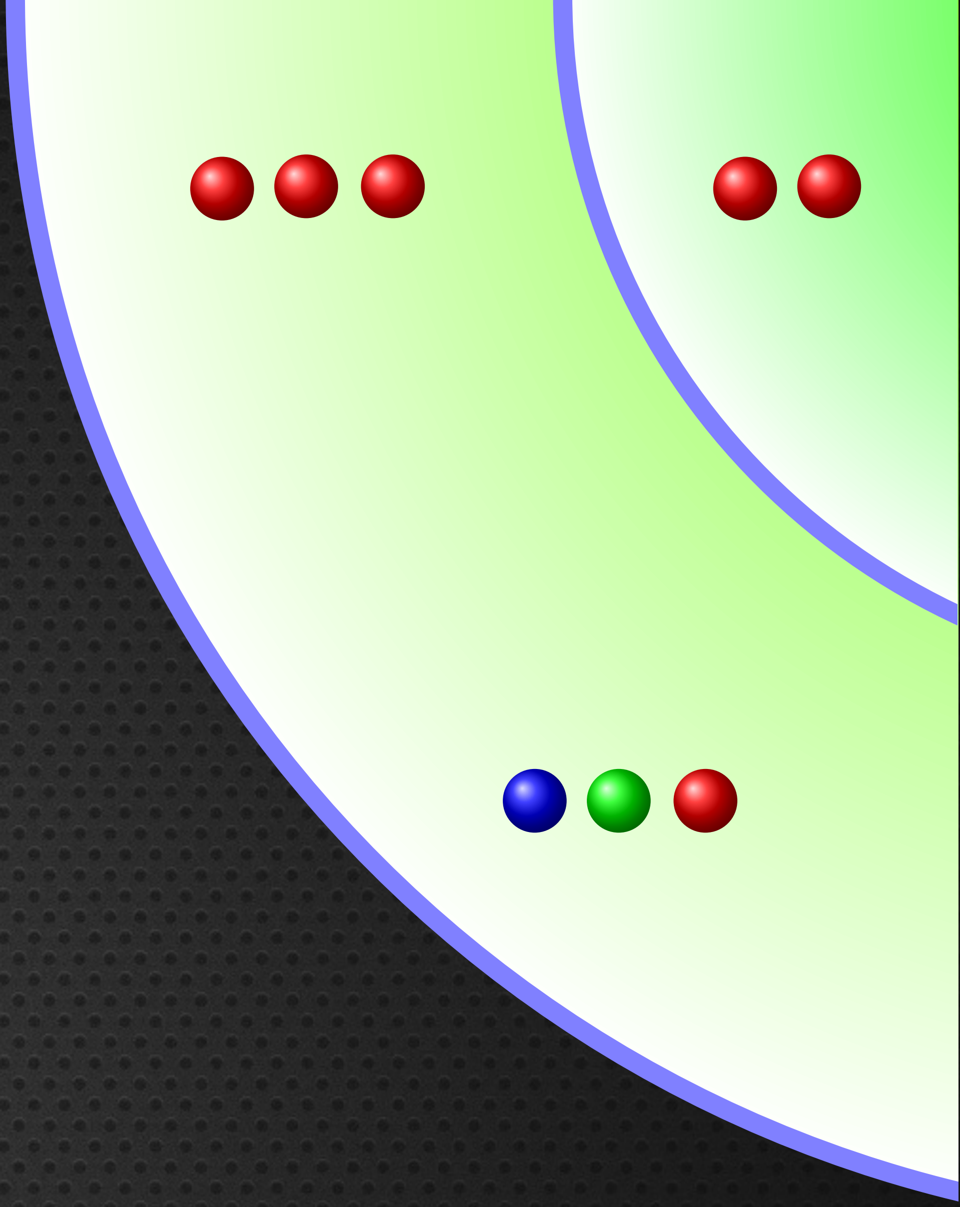
Backward Reachability



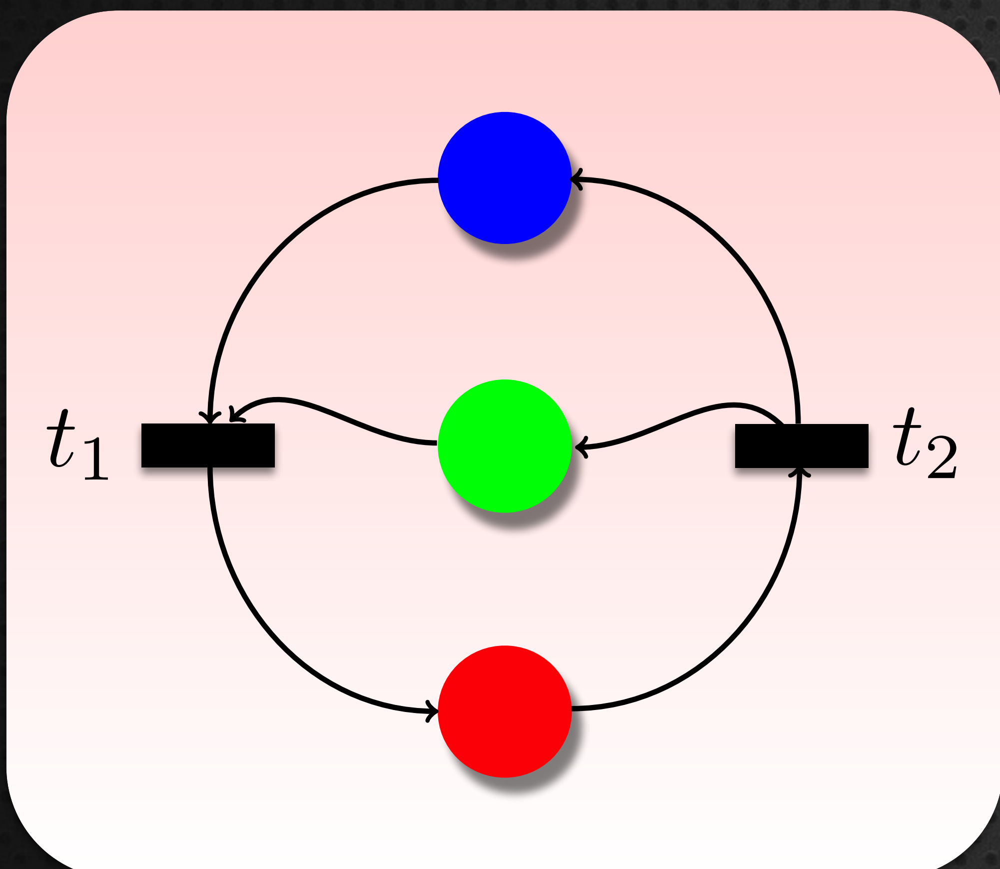


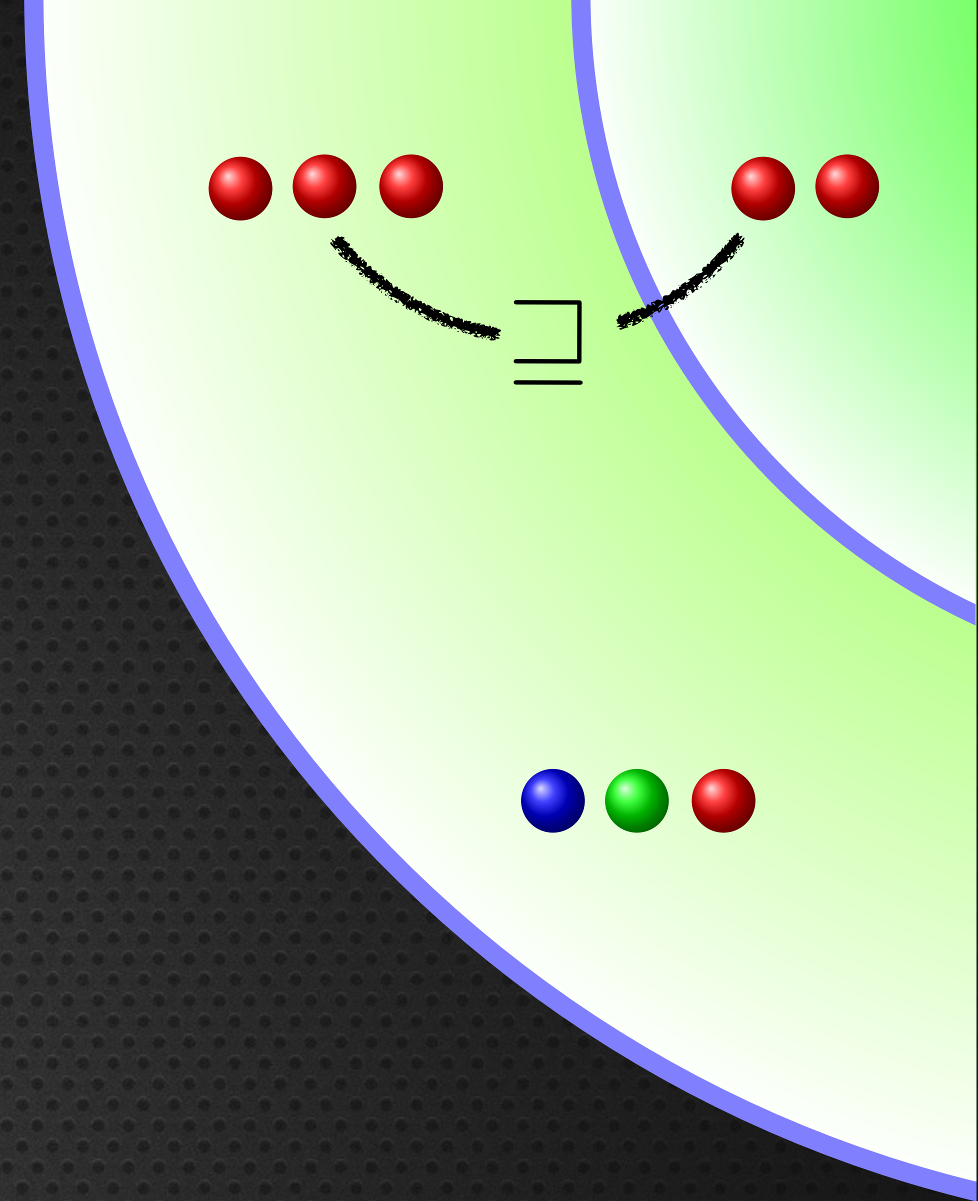
Backward Reachability



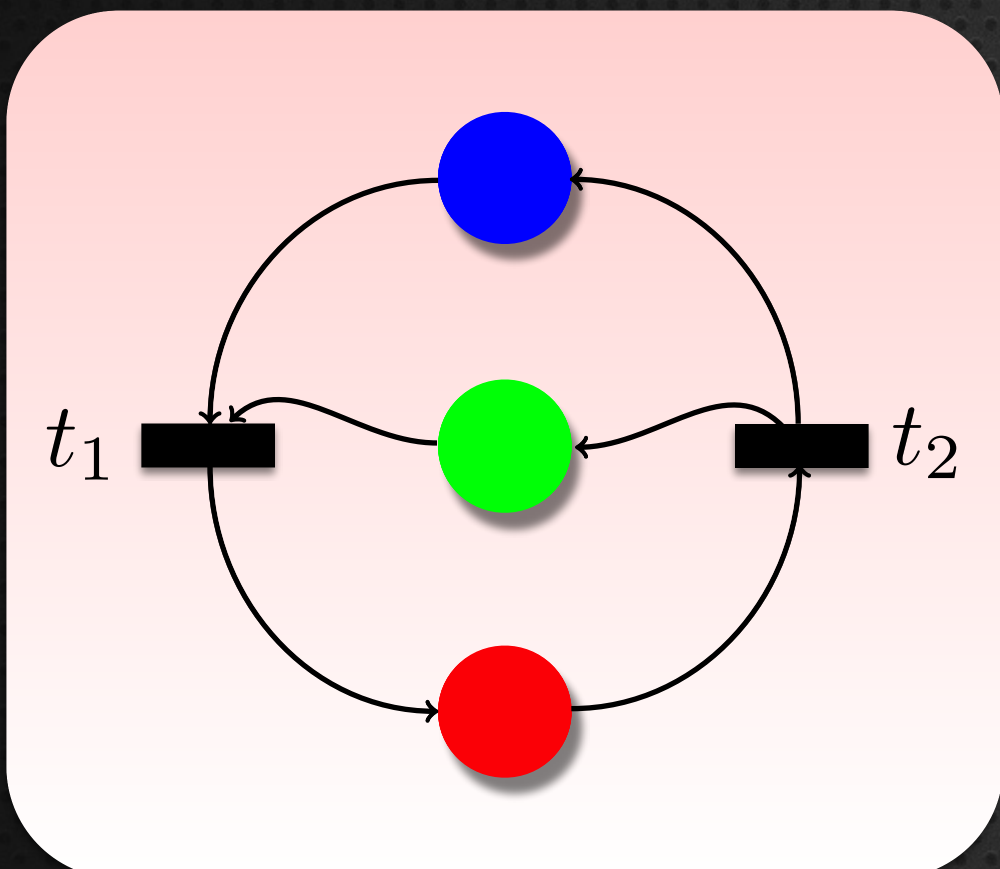


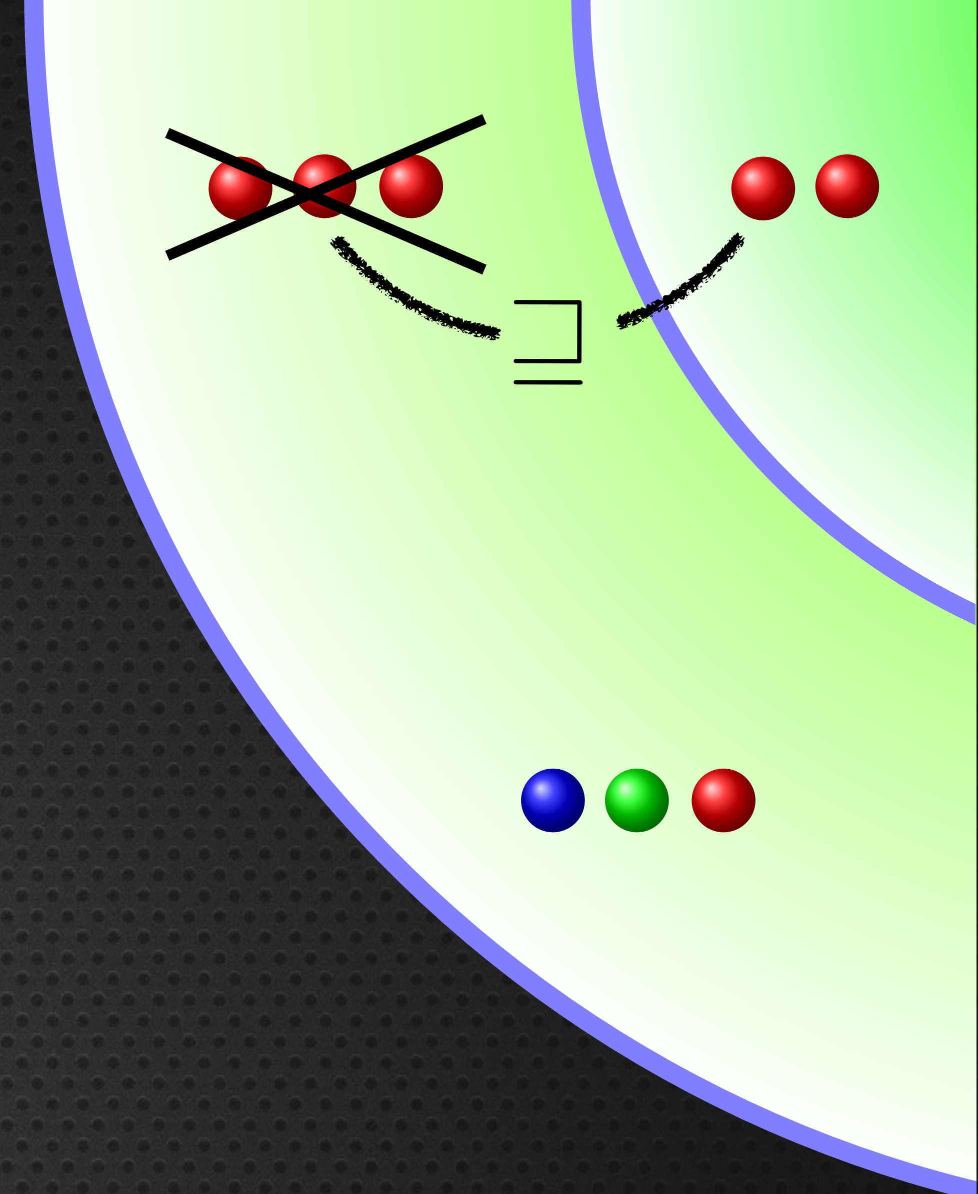
Backward Reachability



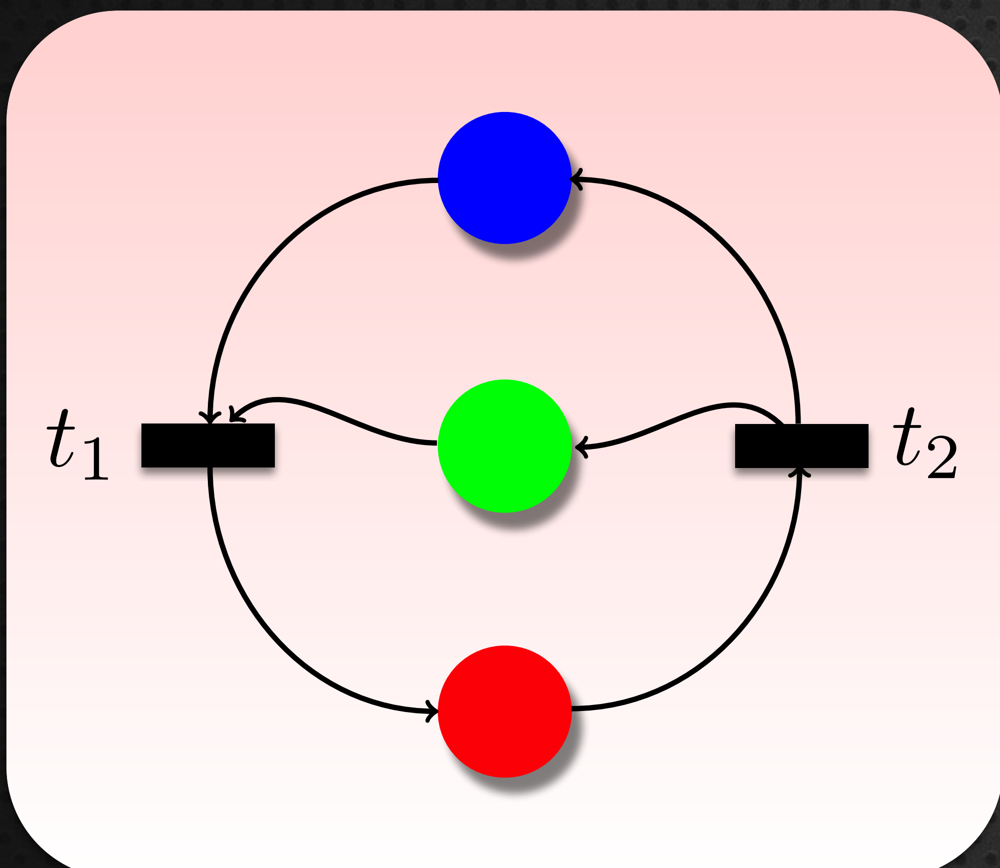


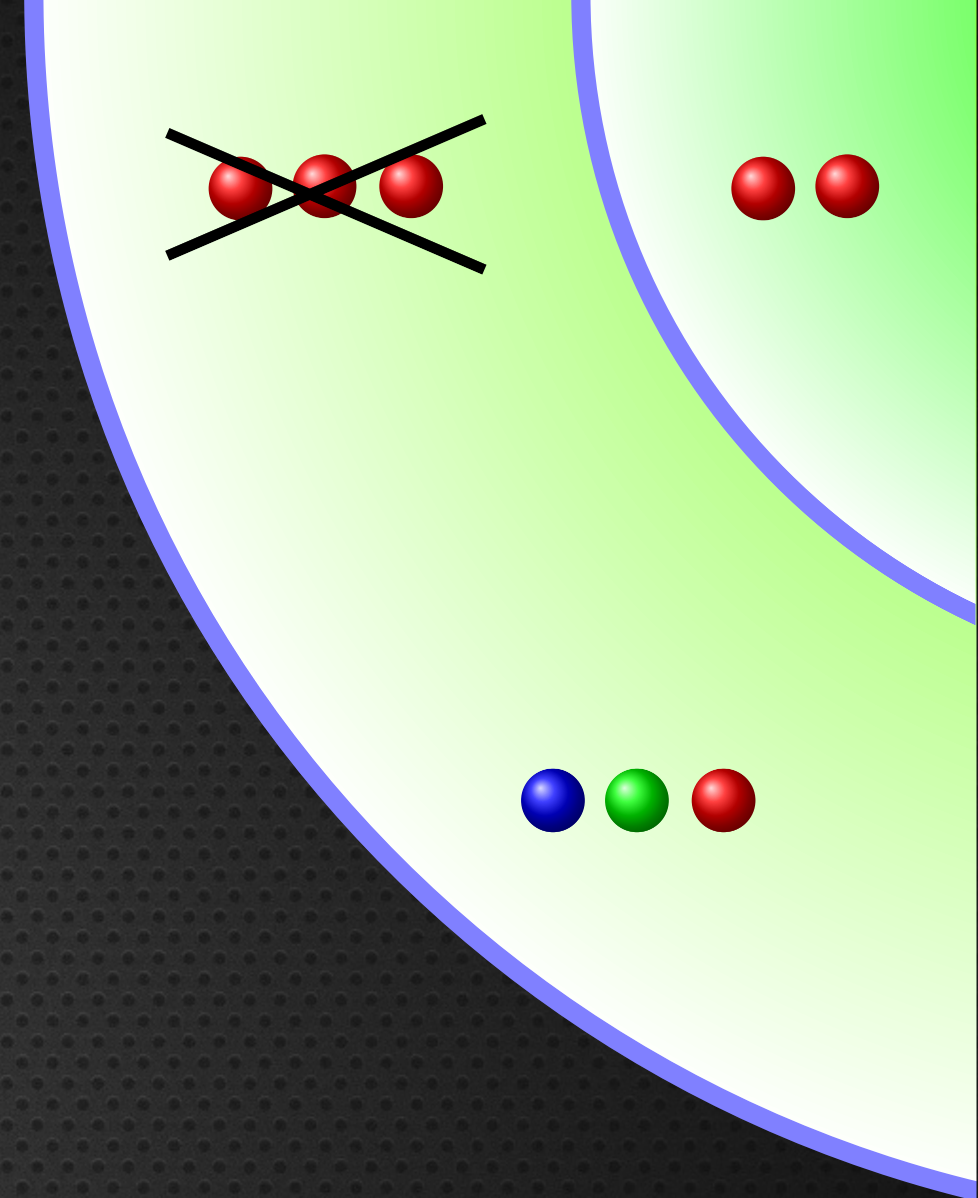
Backward Reachability



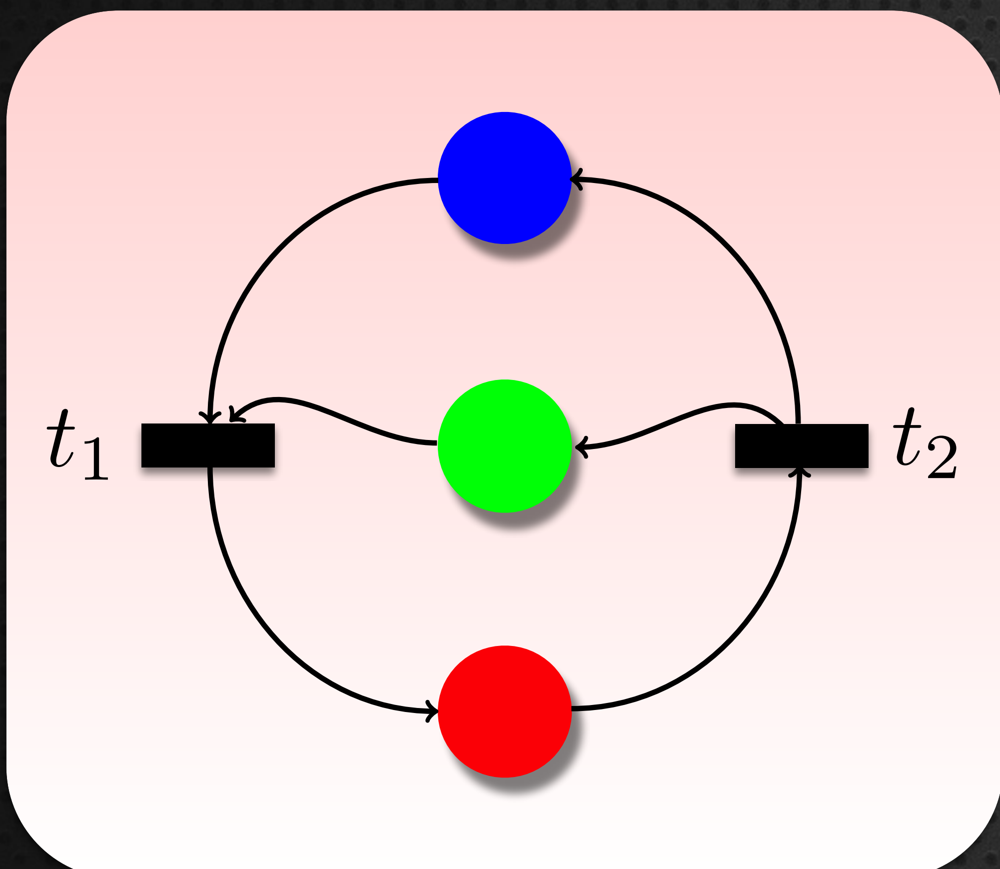


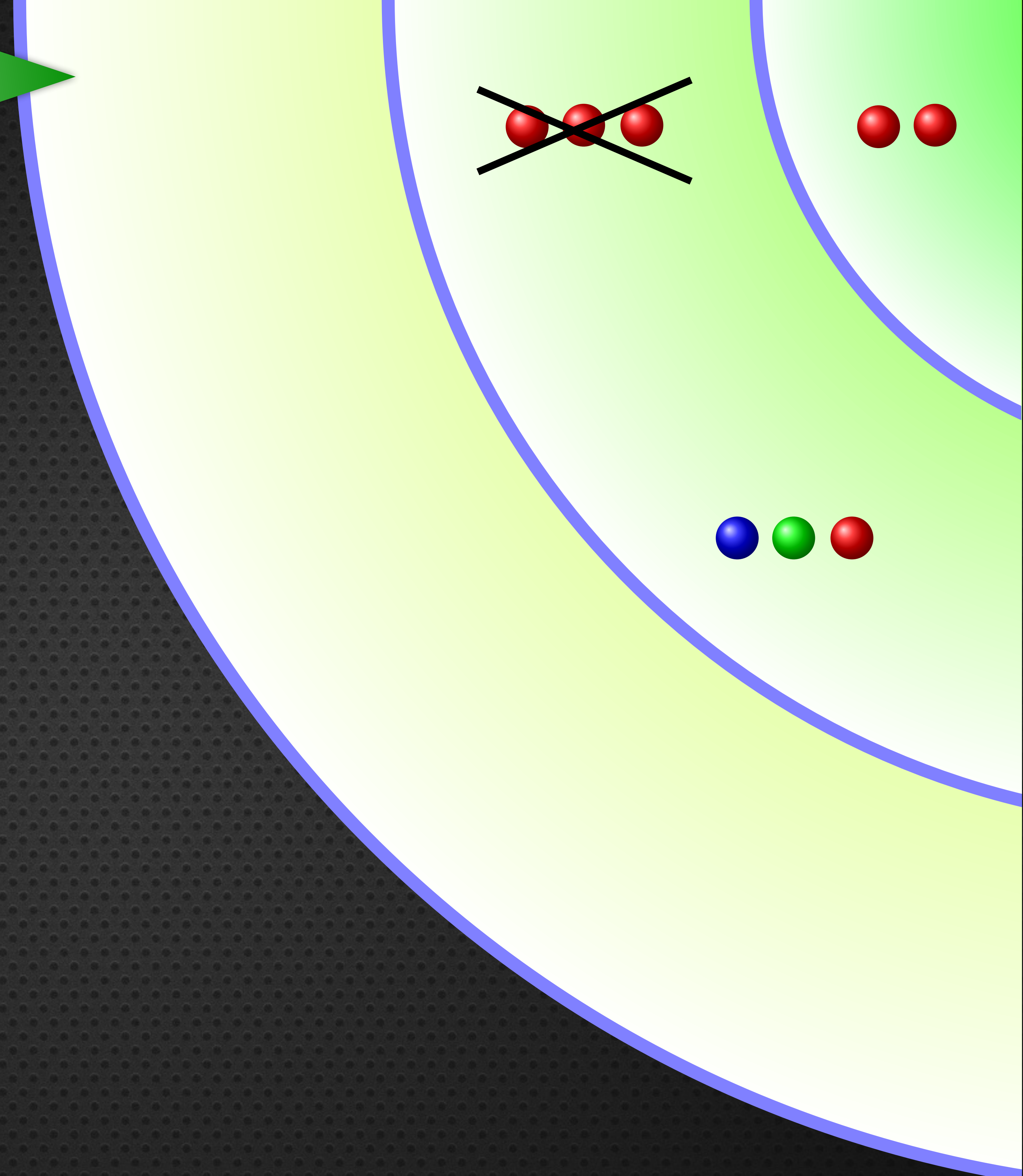
Backward Reachability



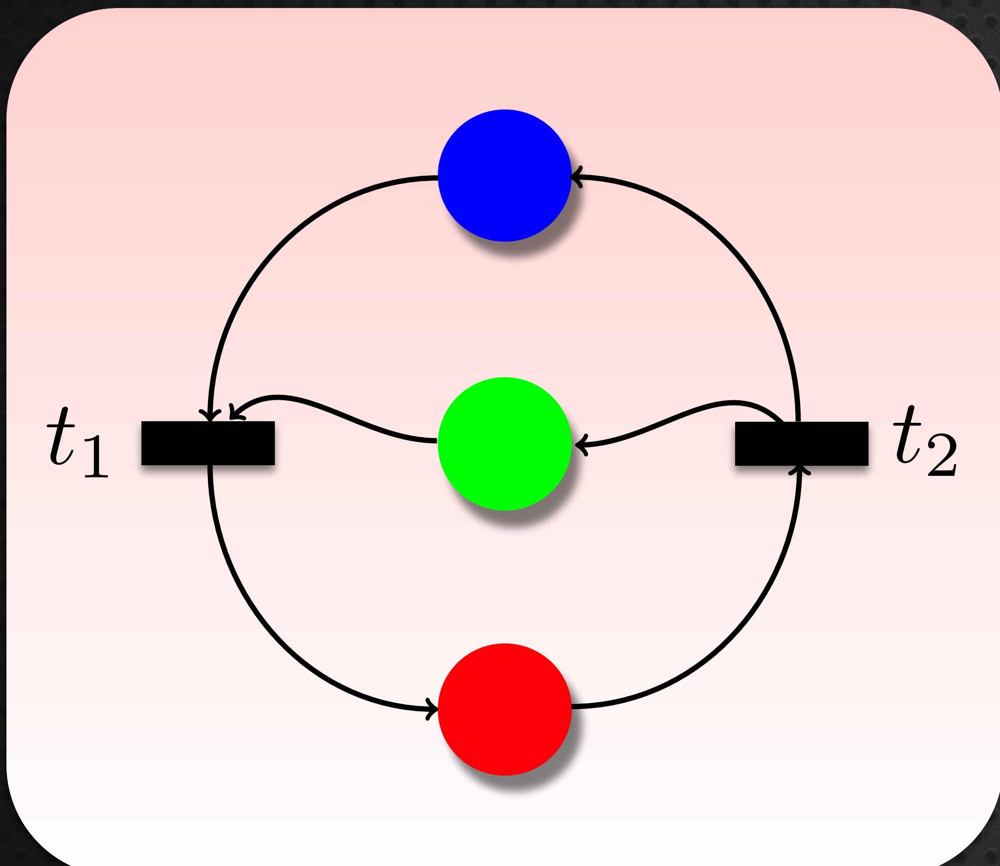


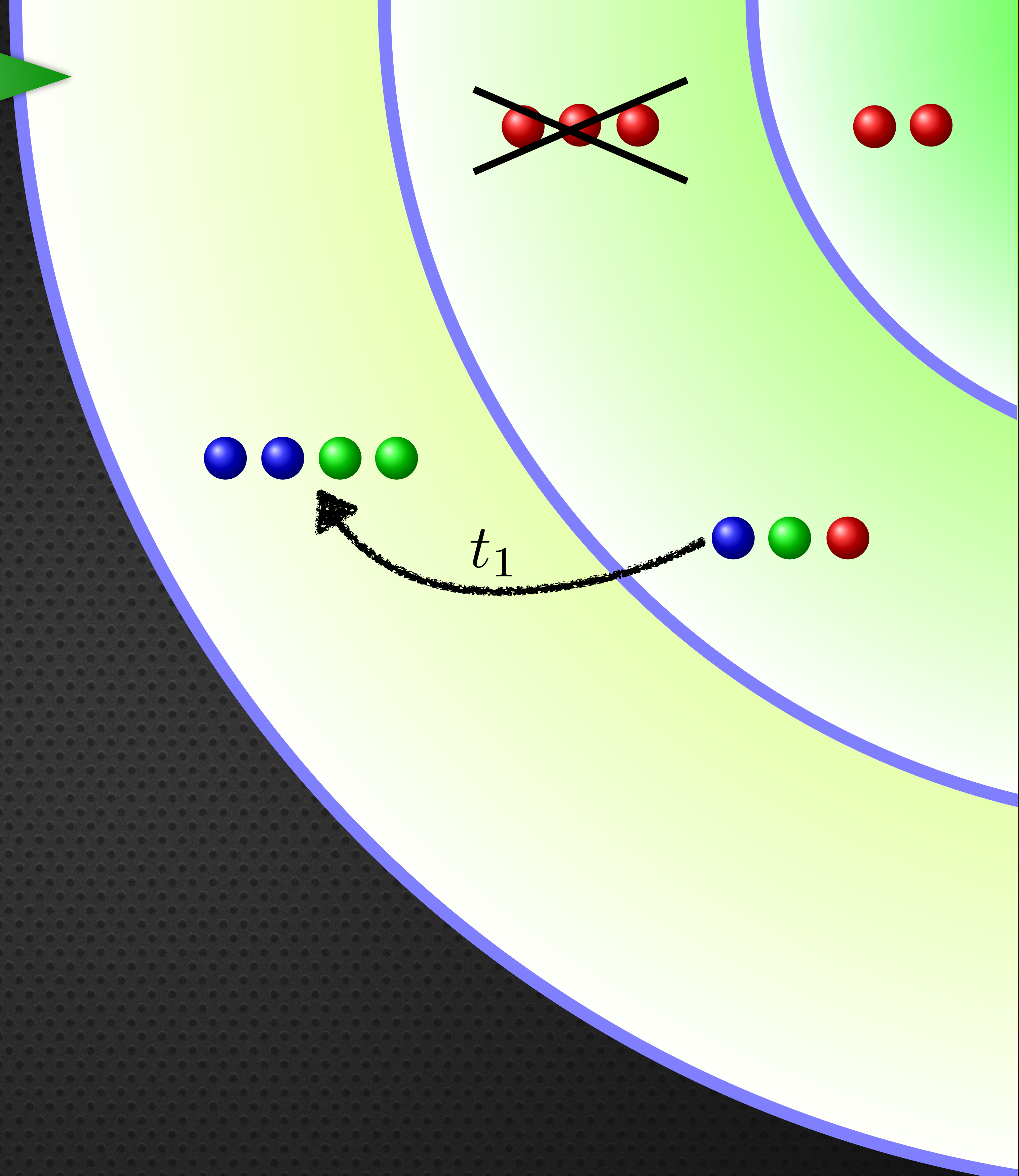
Backward Reachability



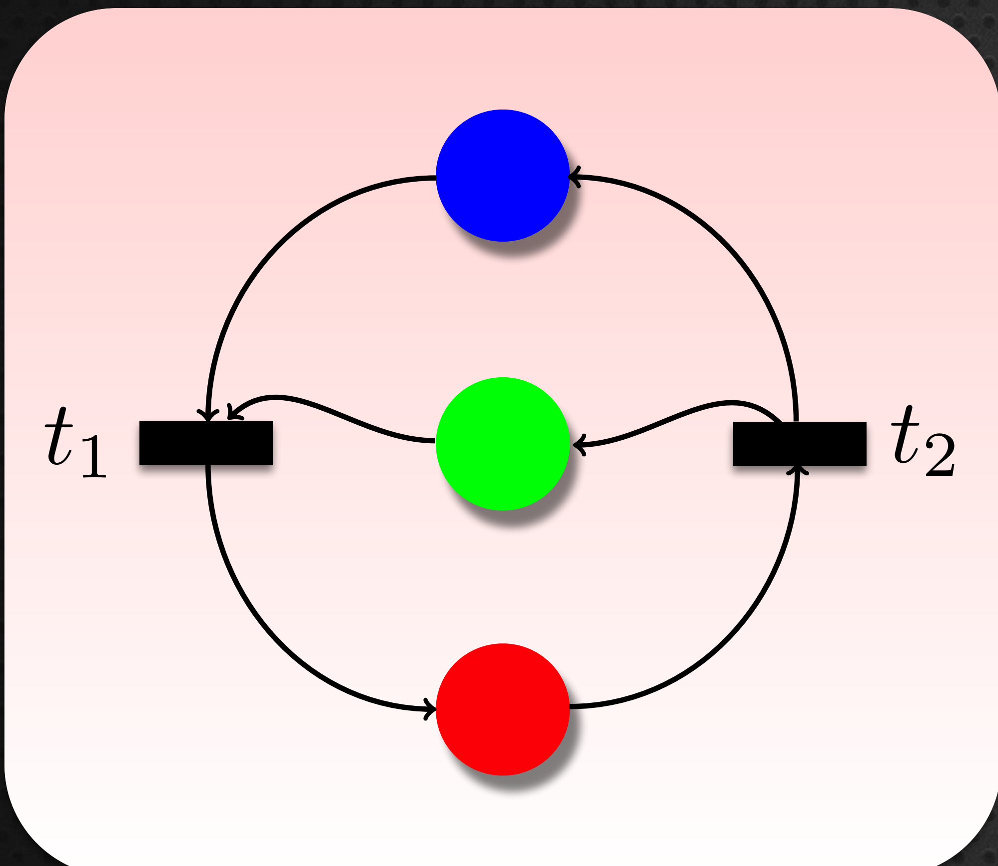


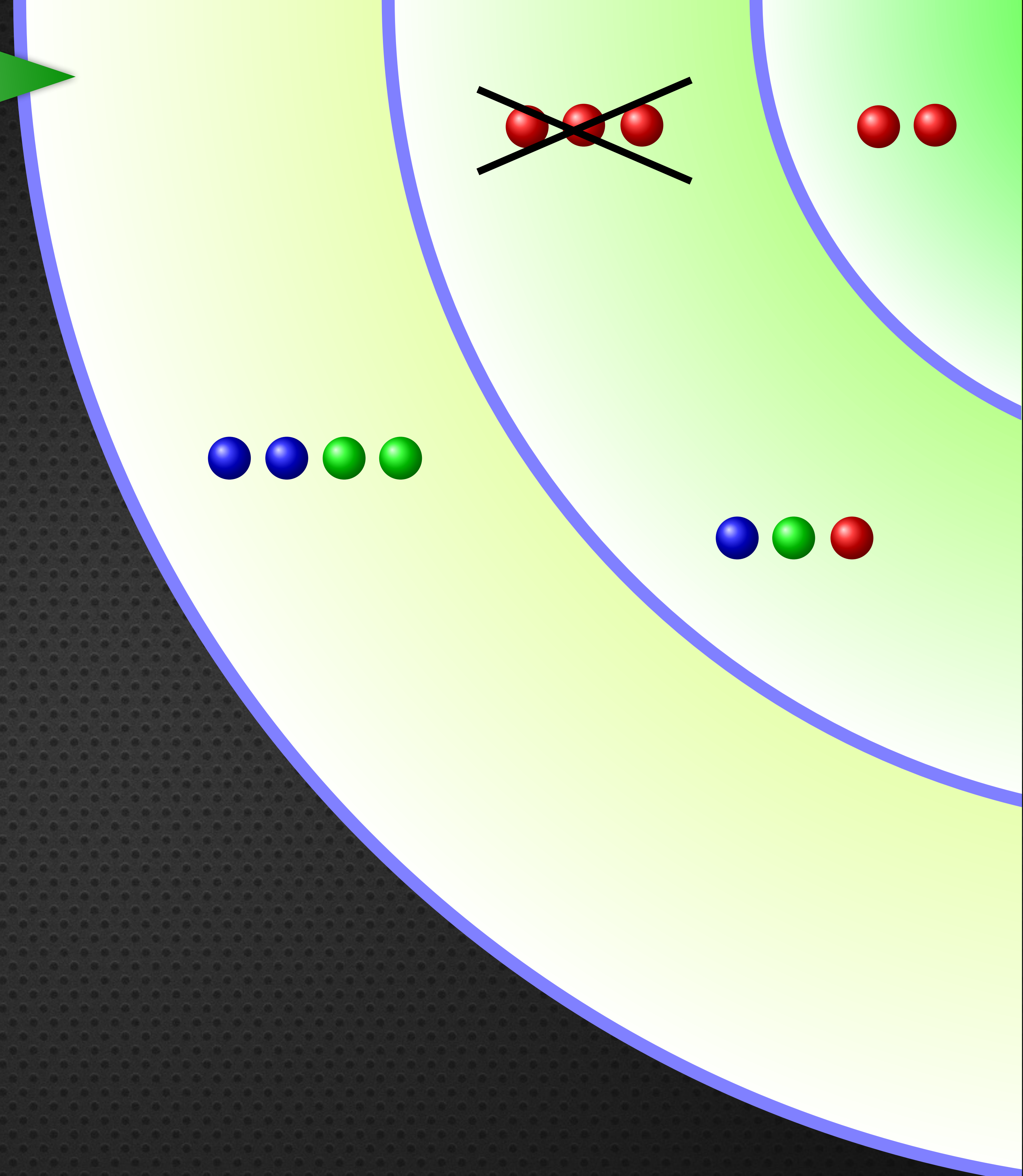
Backward Reachability



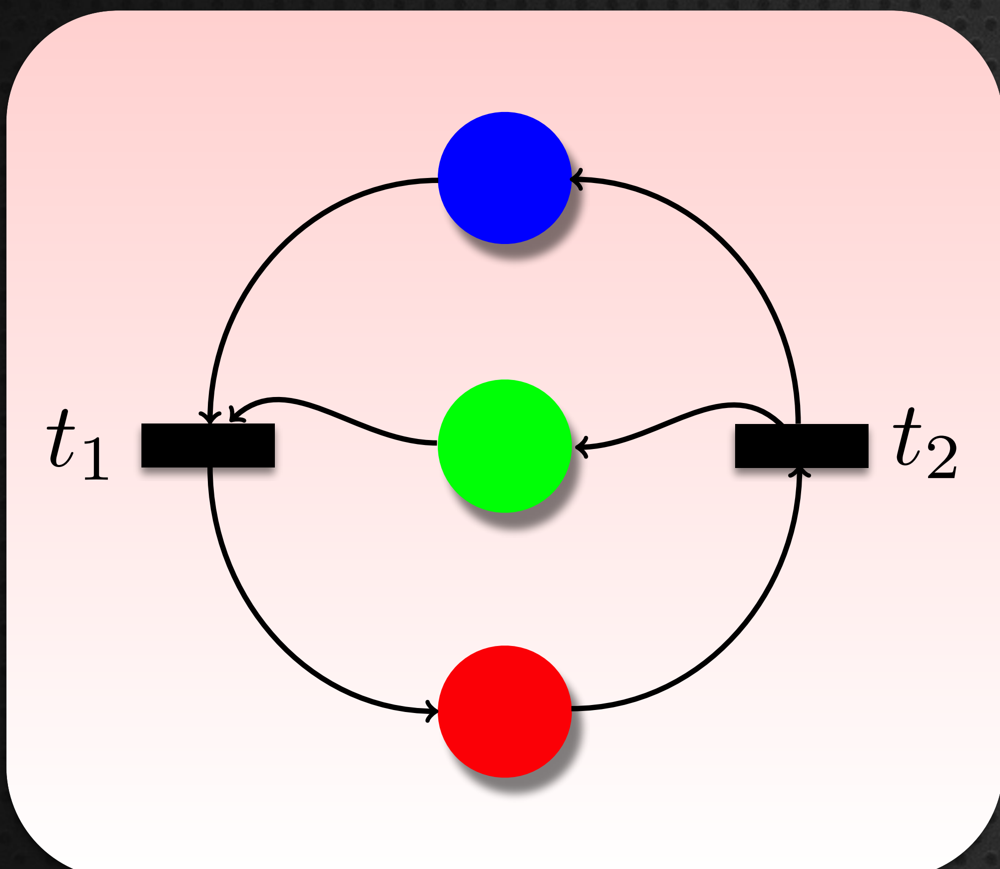


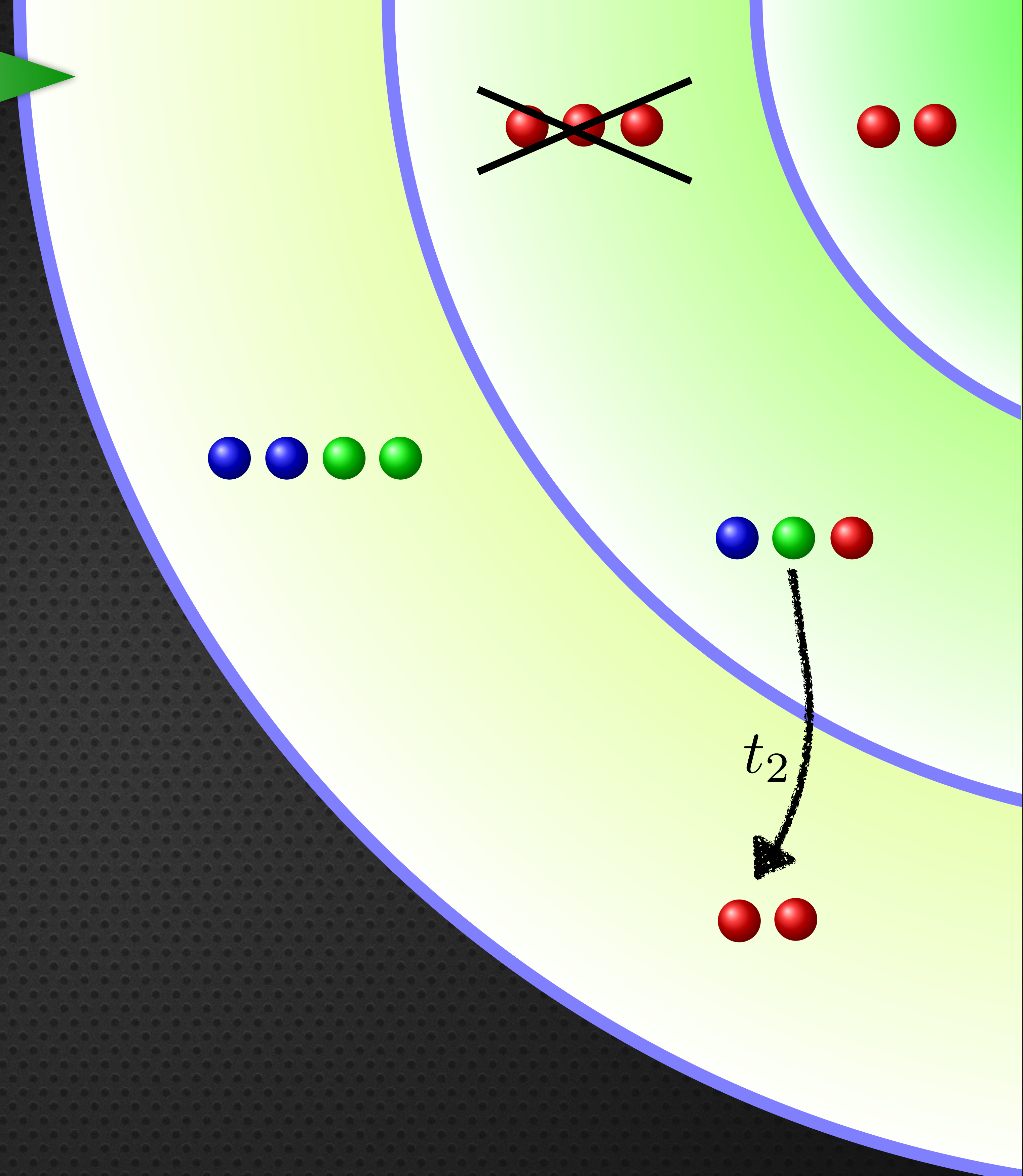
Backward Reachability



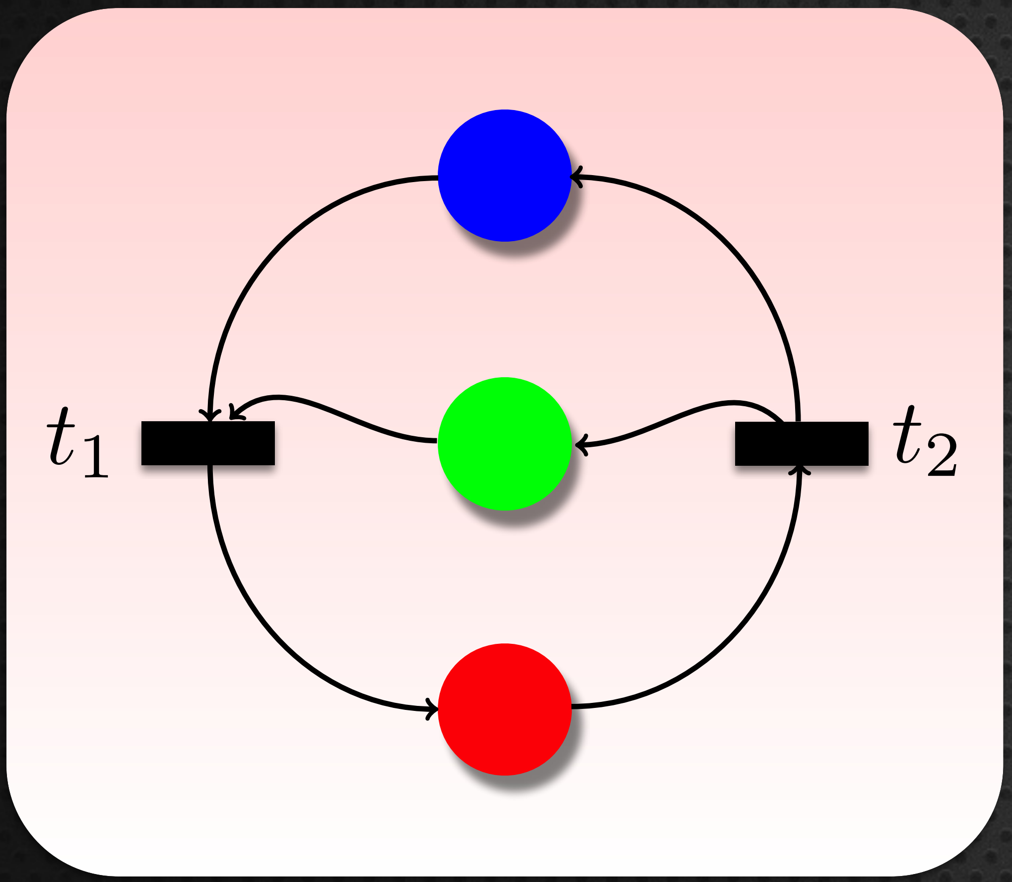


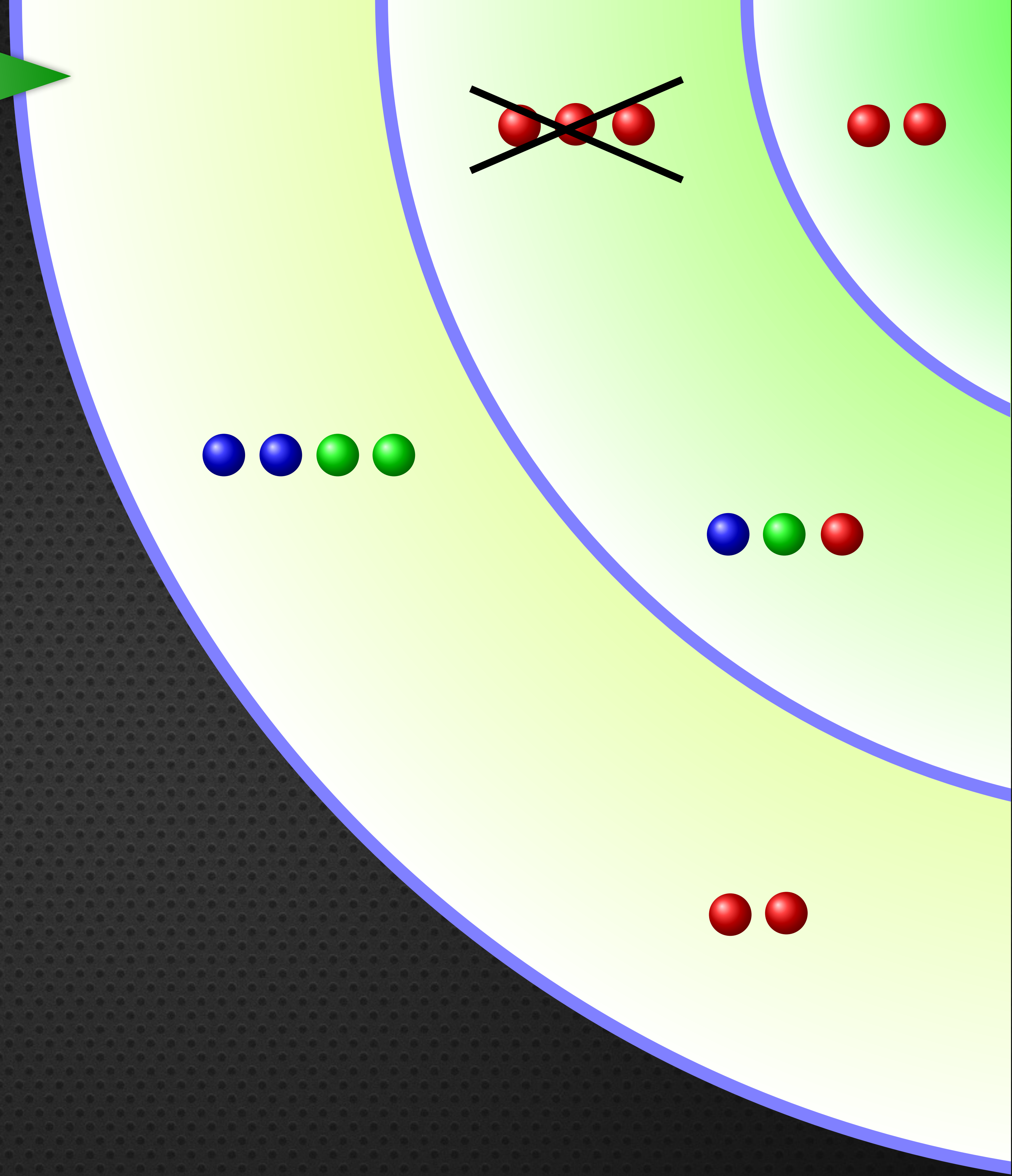
Backward Reachability



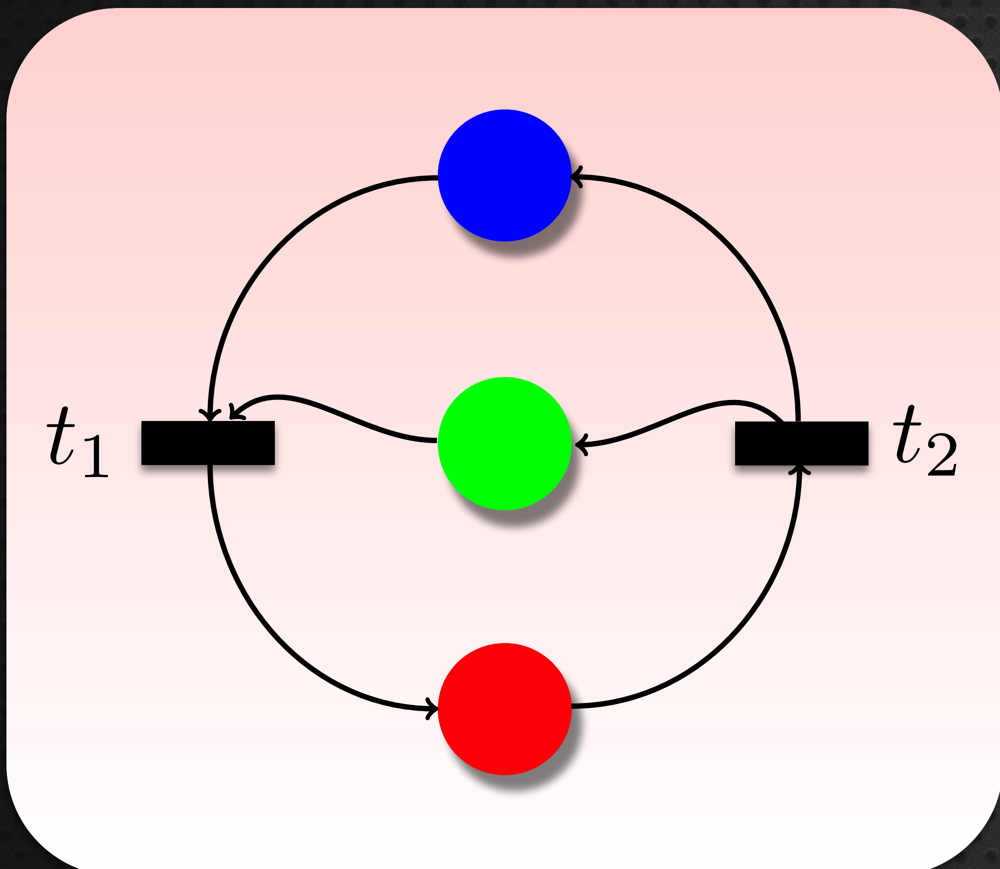


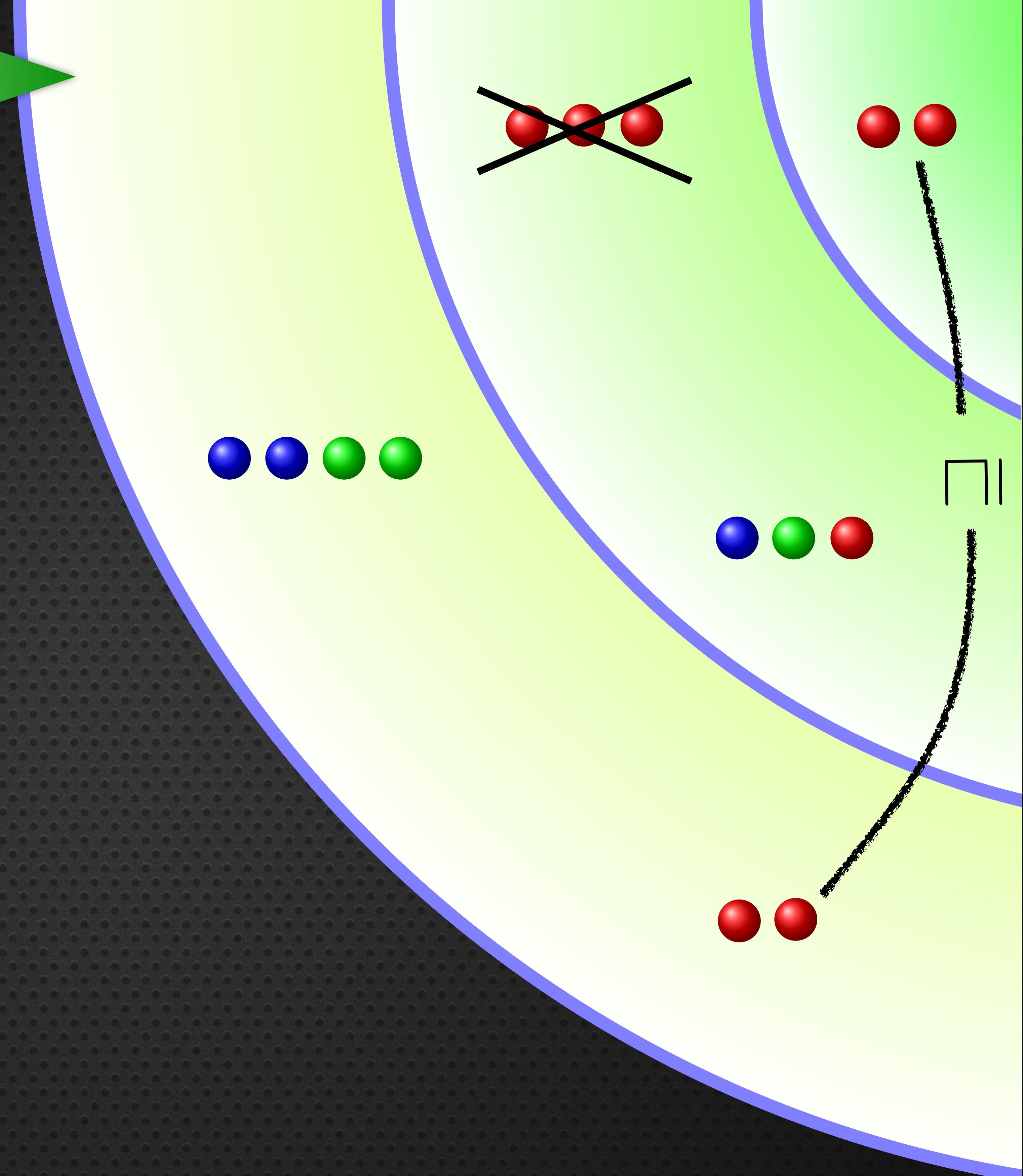
Backward Reachability



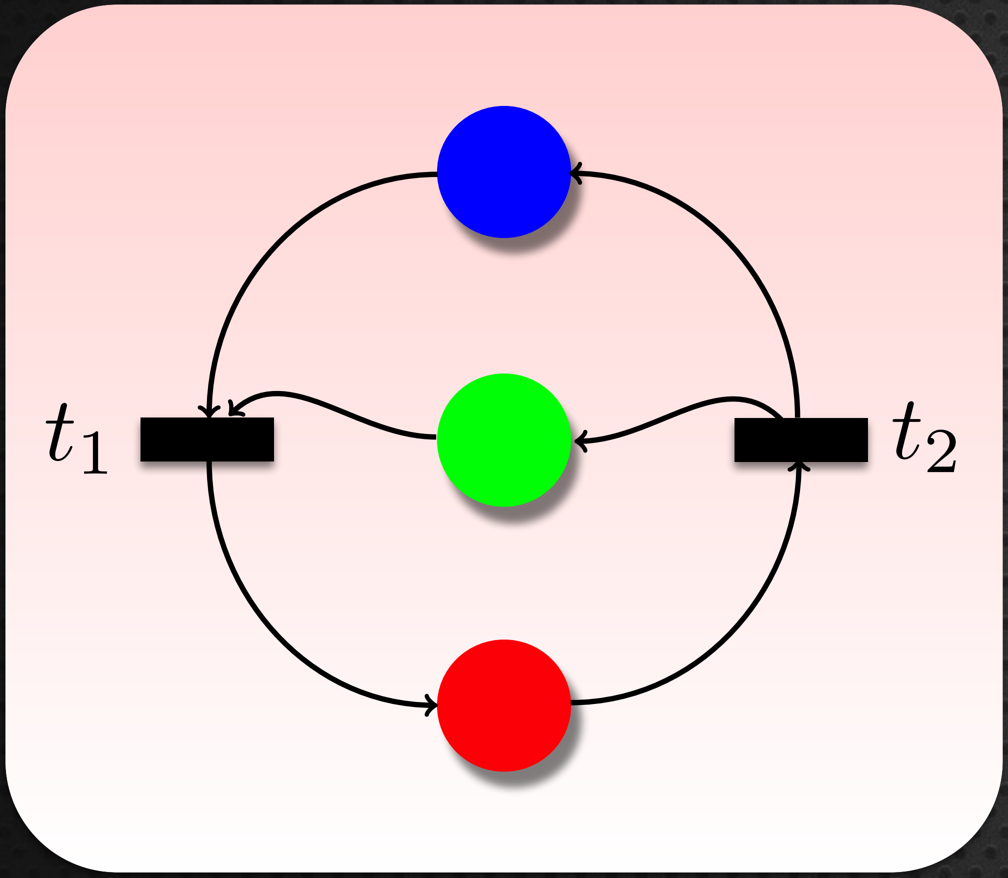


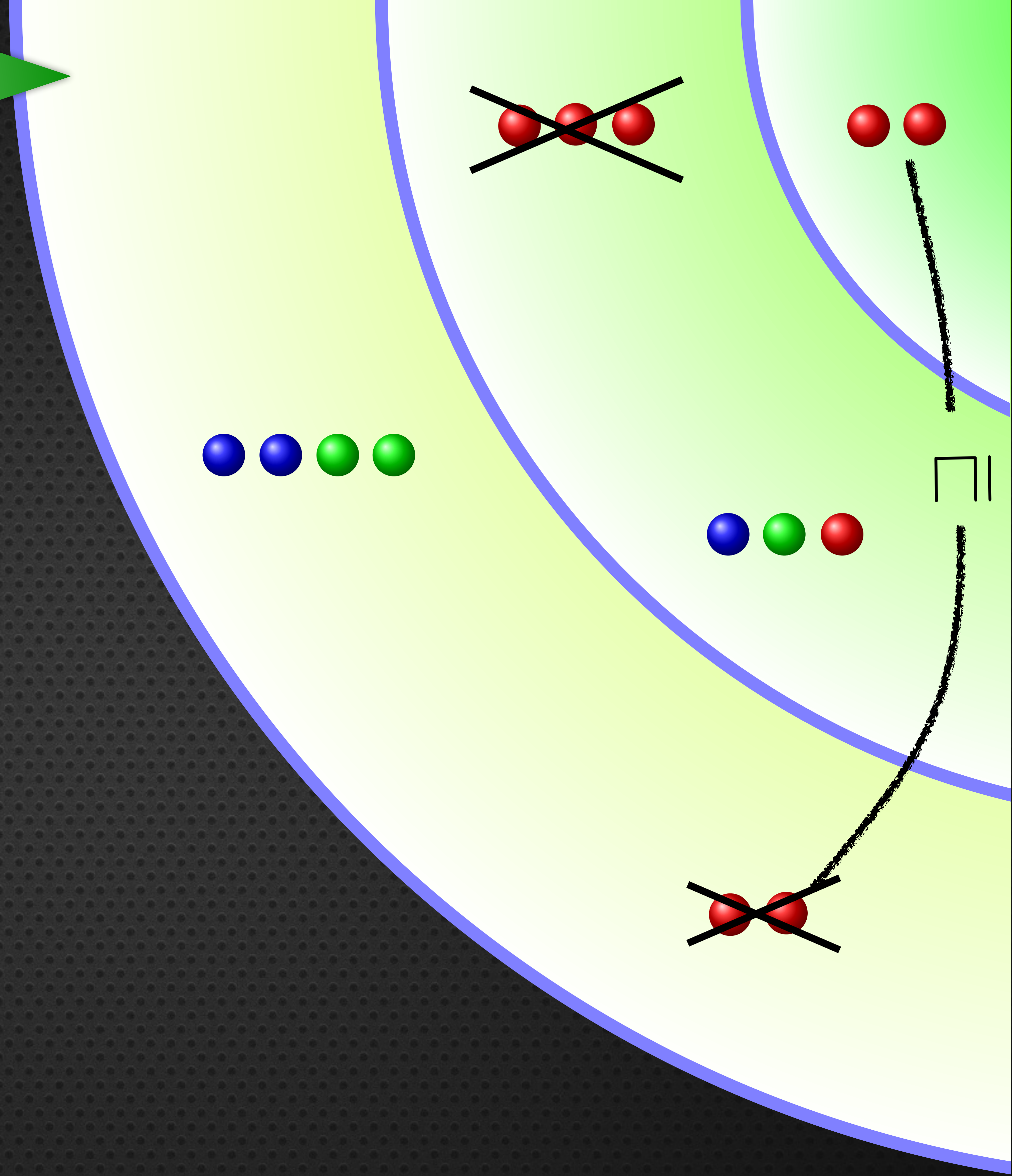
Backward Reachability



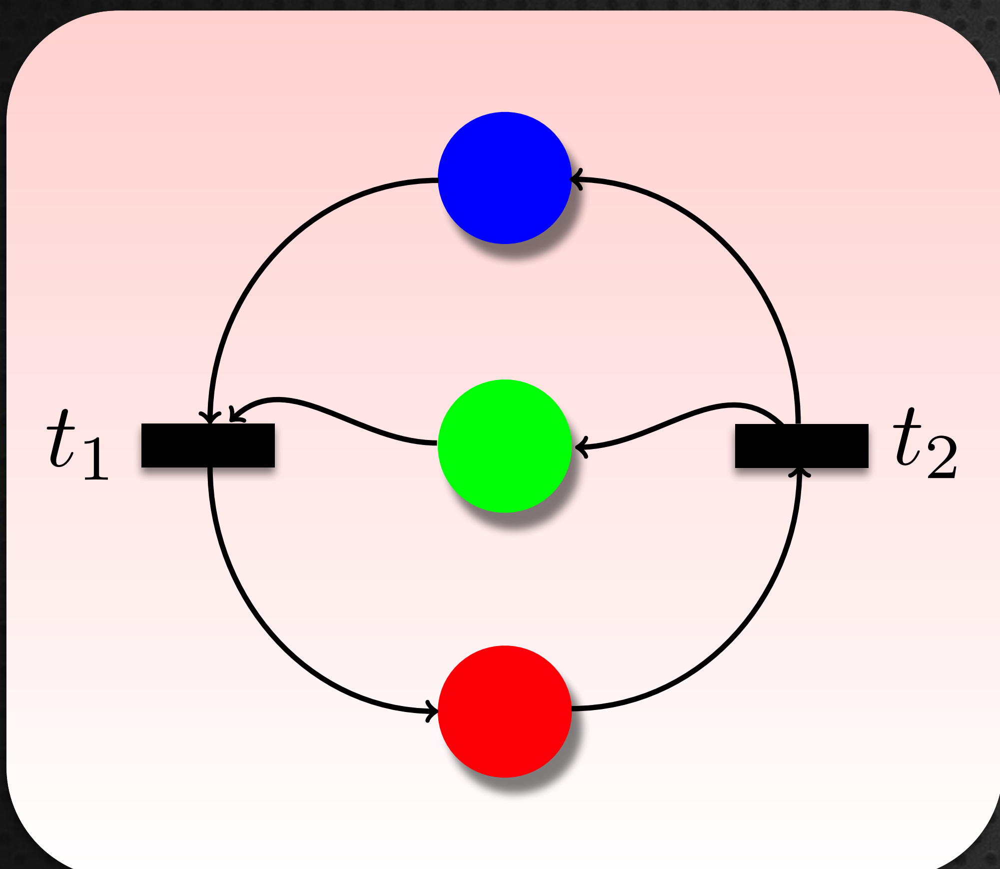


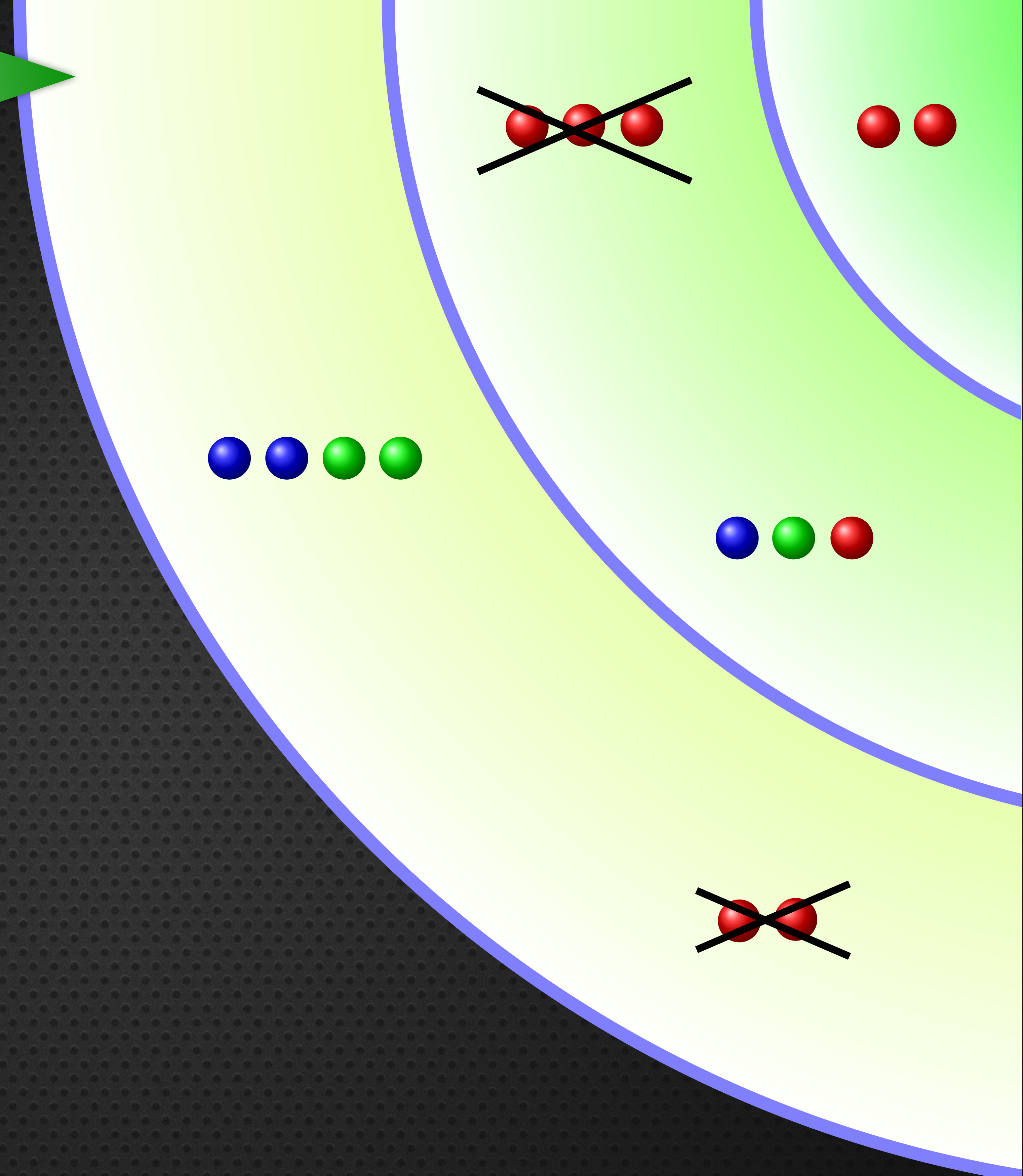
Backward Reachability



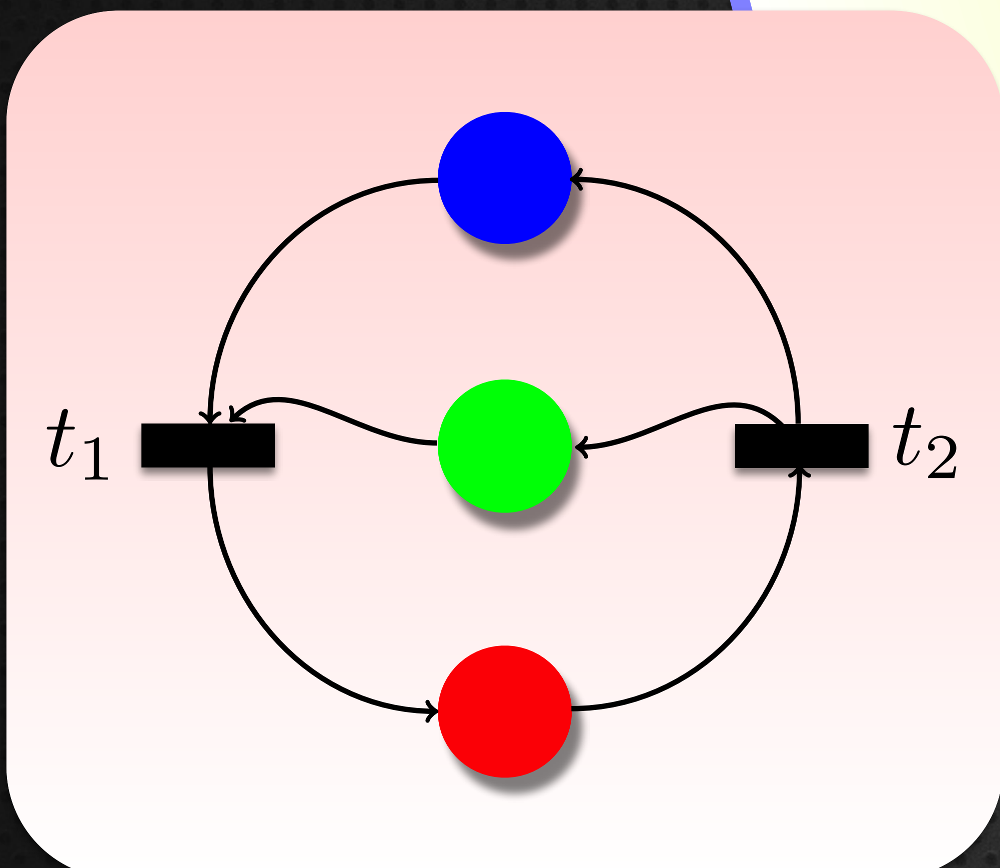


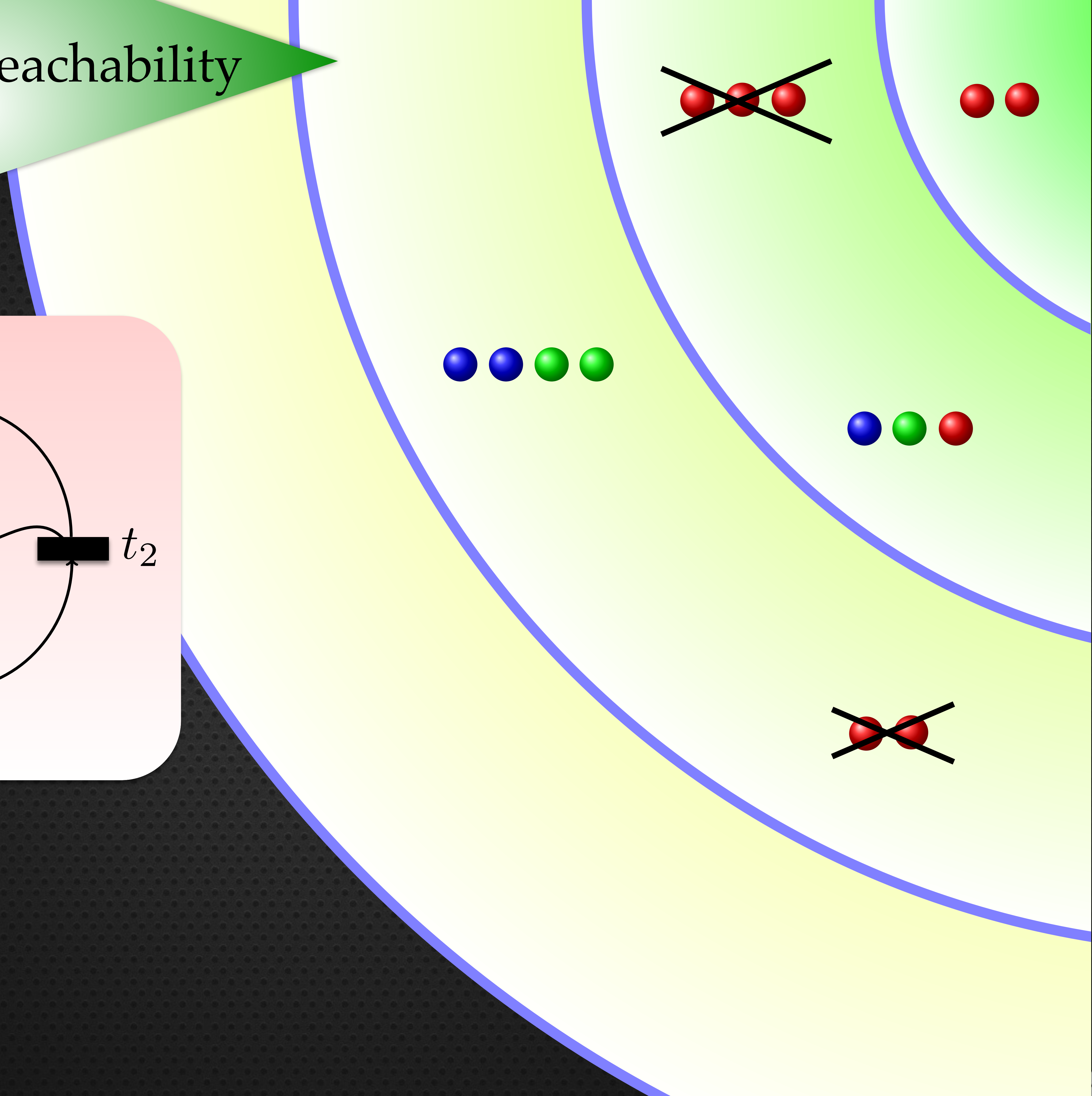
Backward Reachability



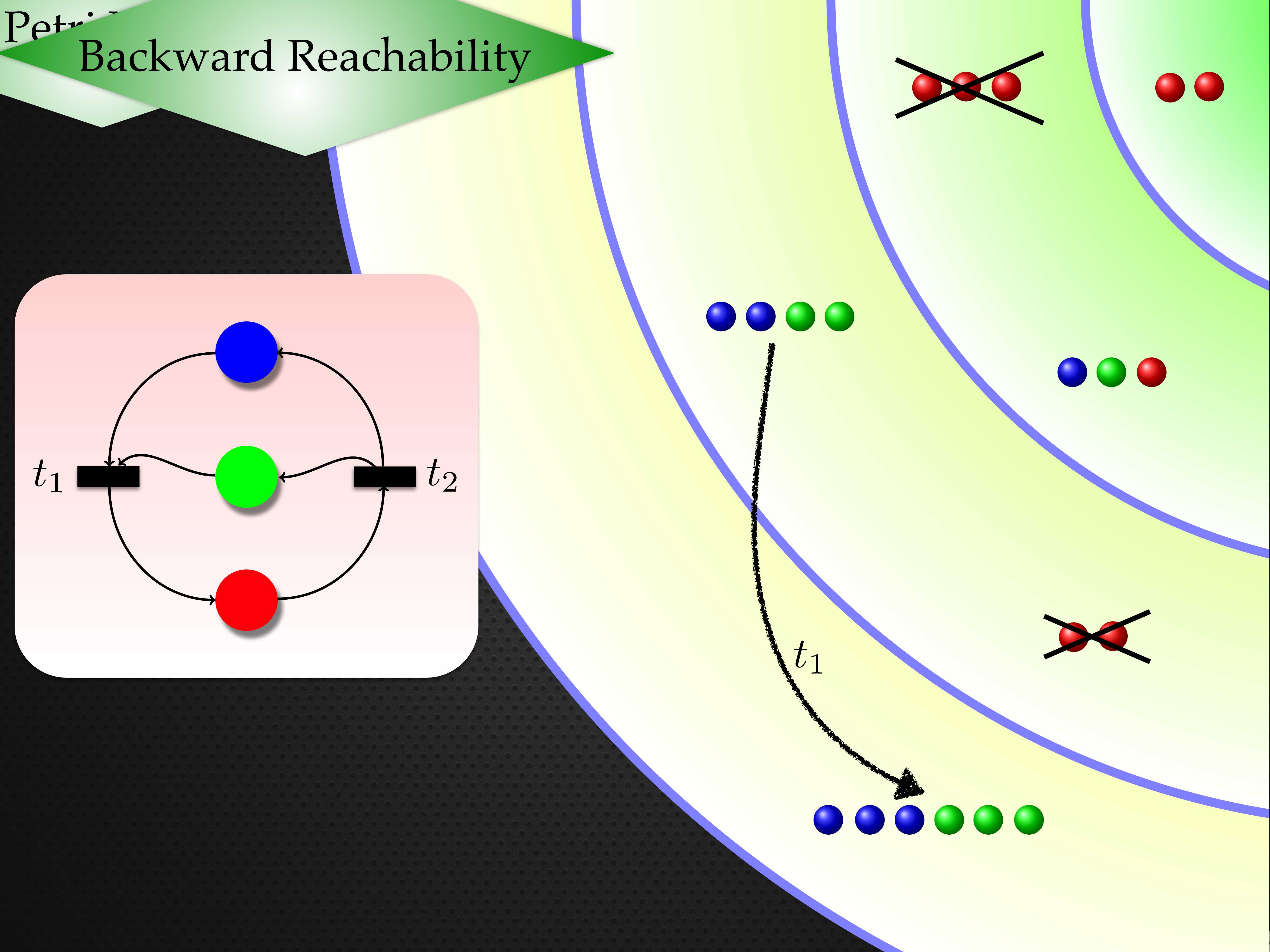


Backward Reachability

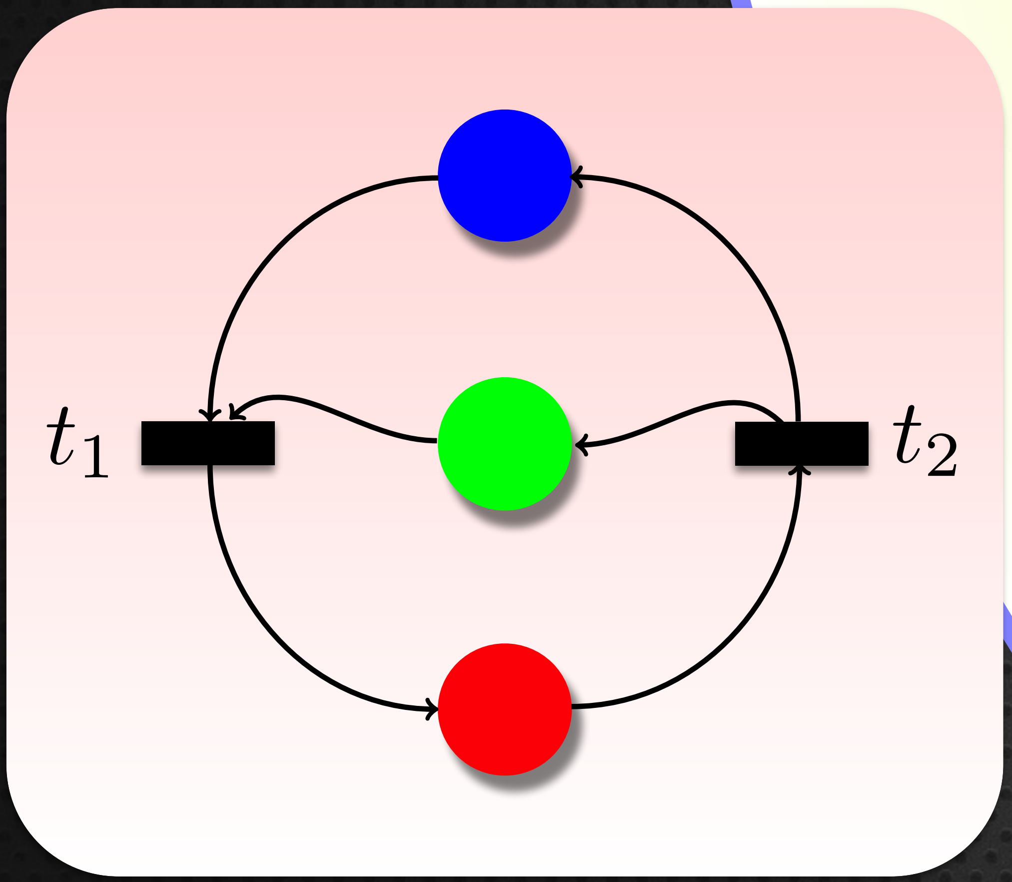


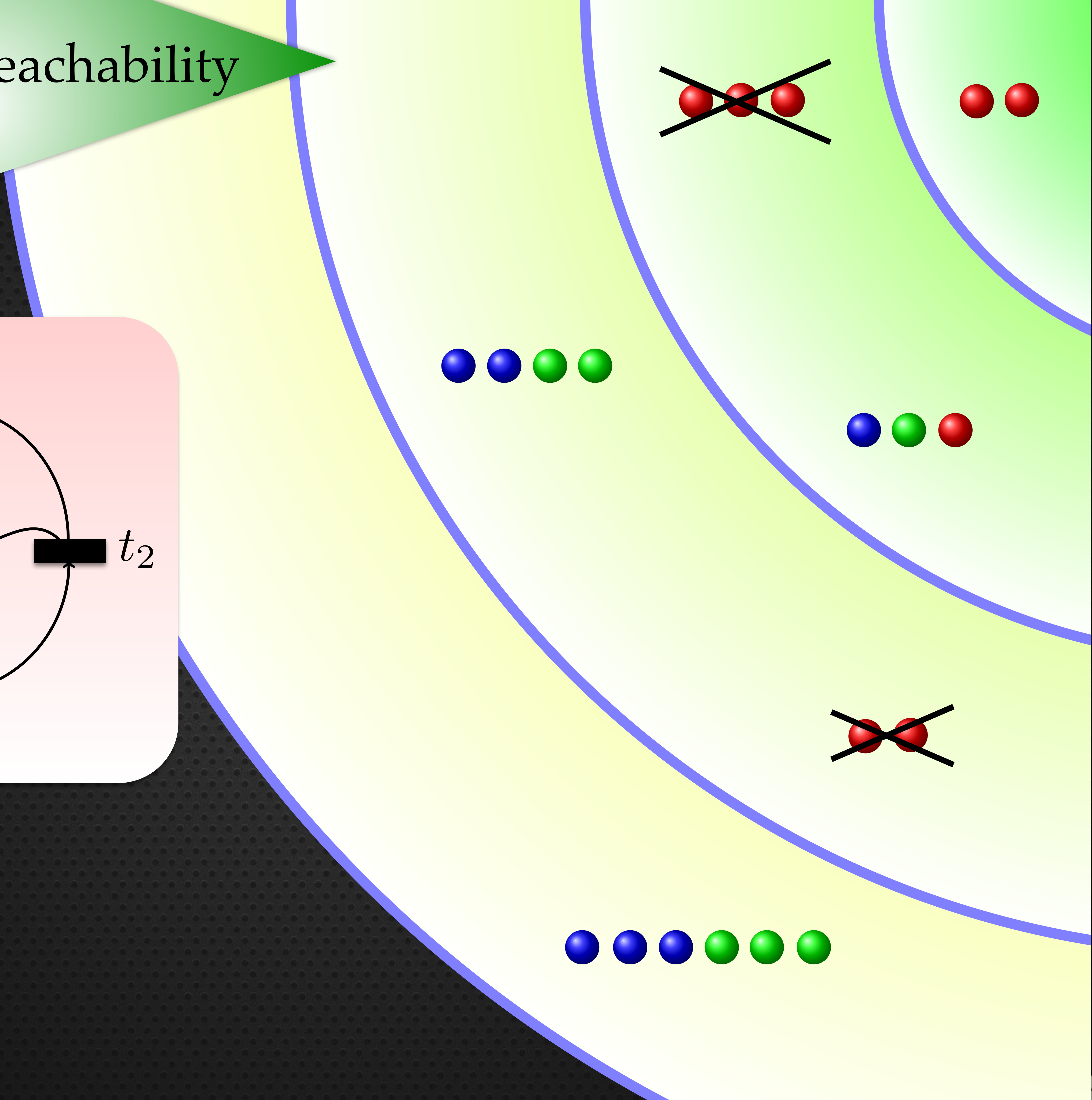


Backward Reachability

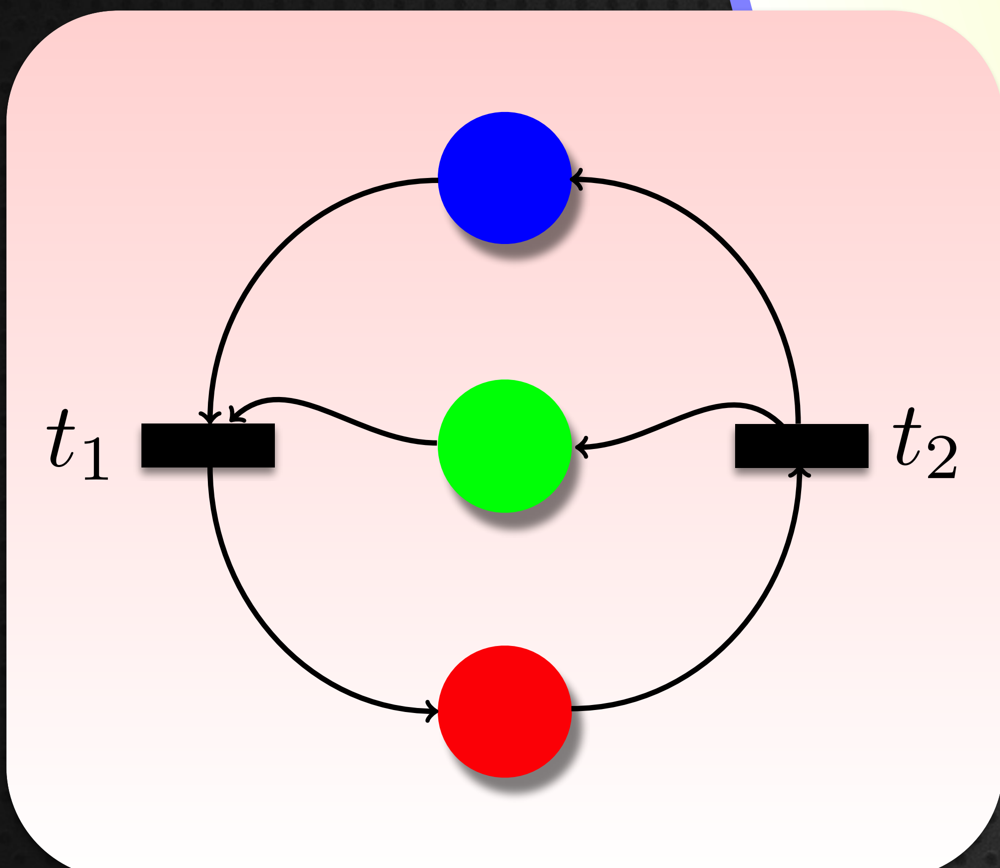


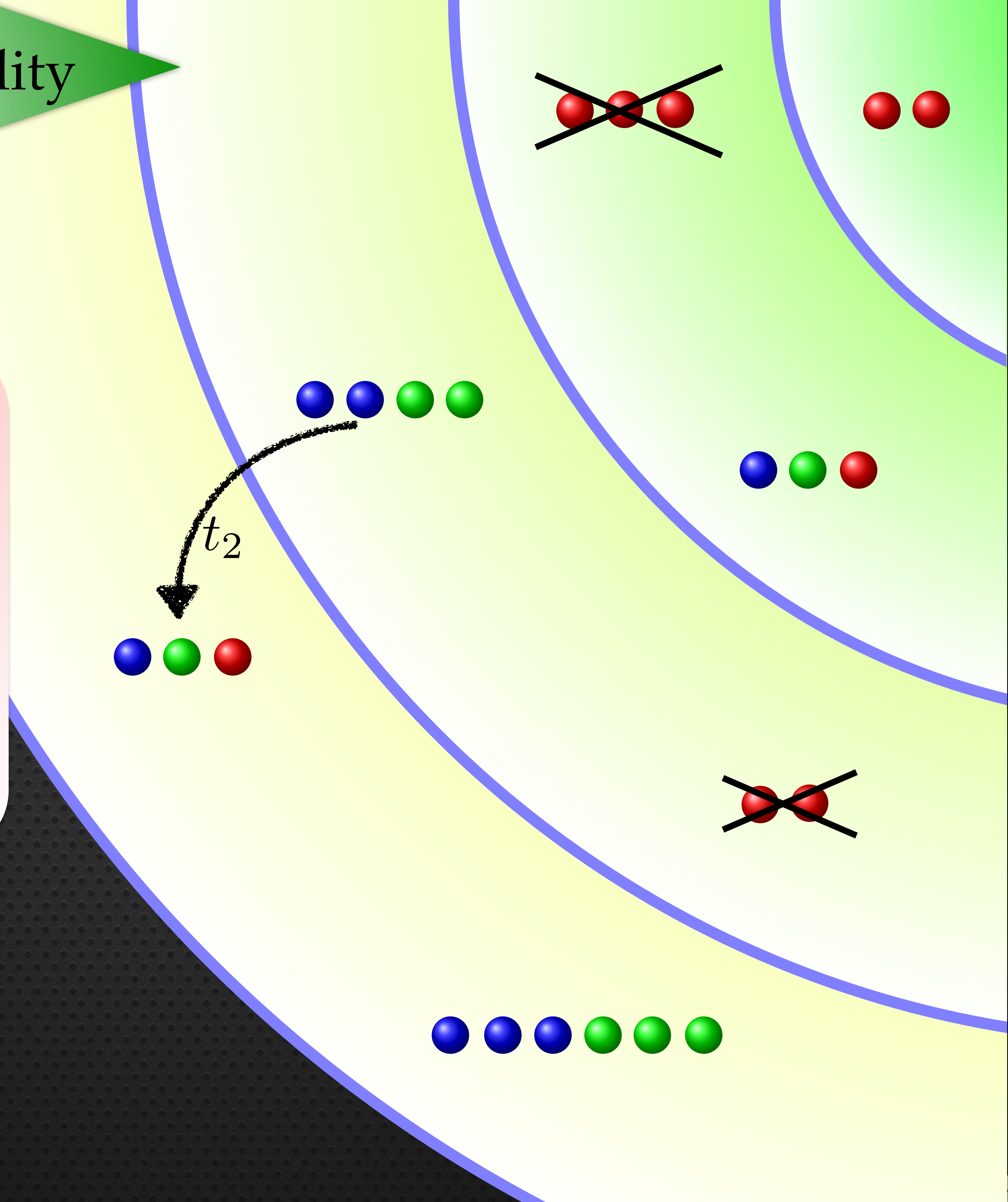
Backward Reachability



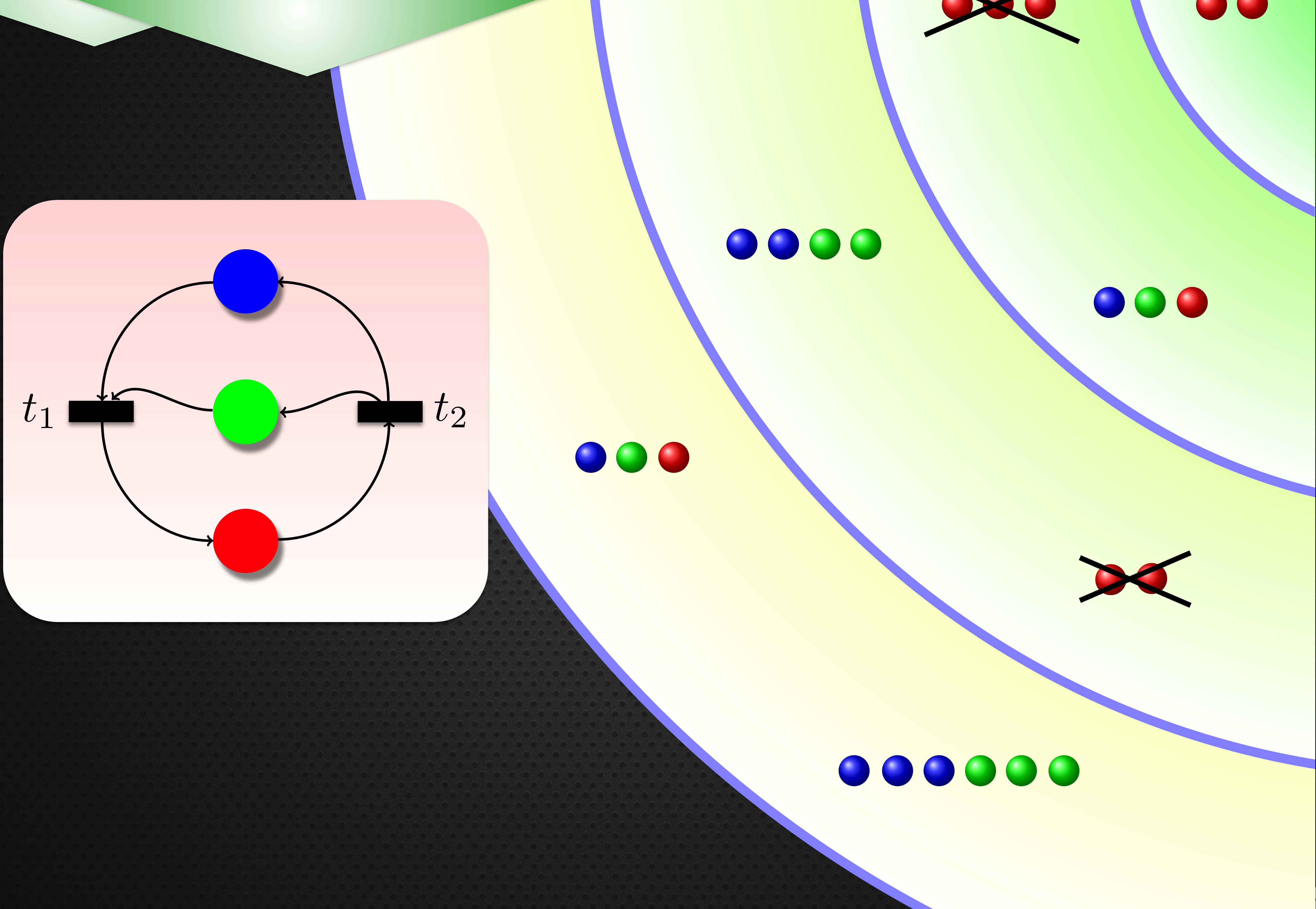


Backward Reachability

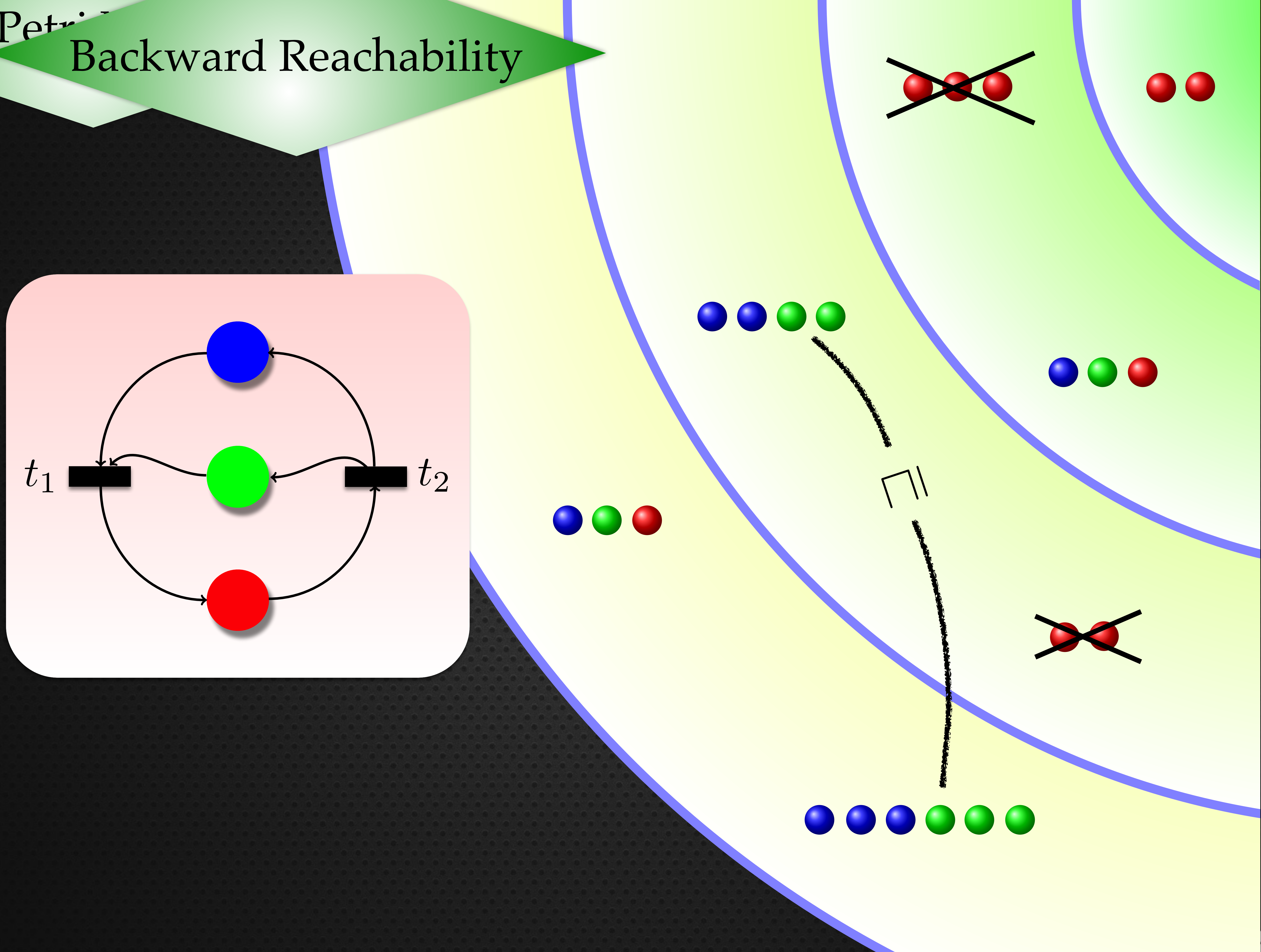




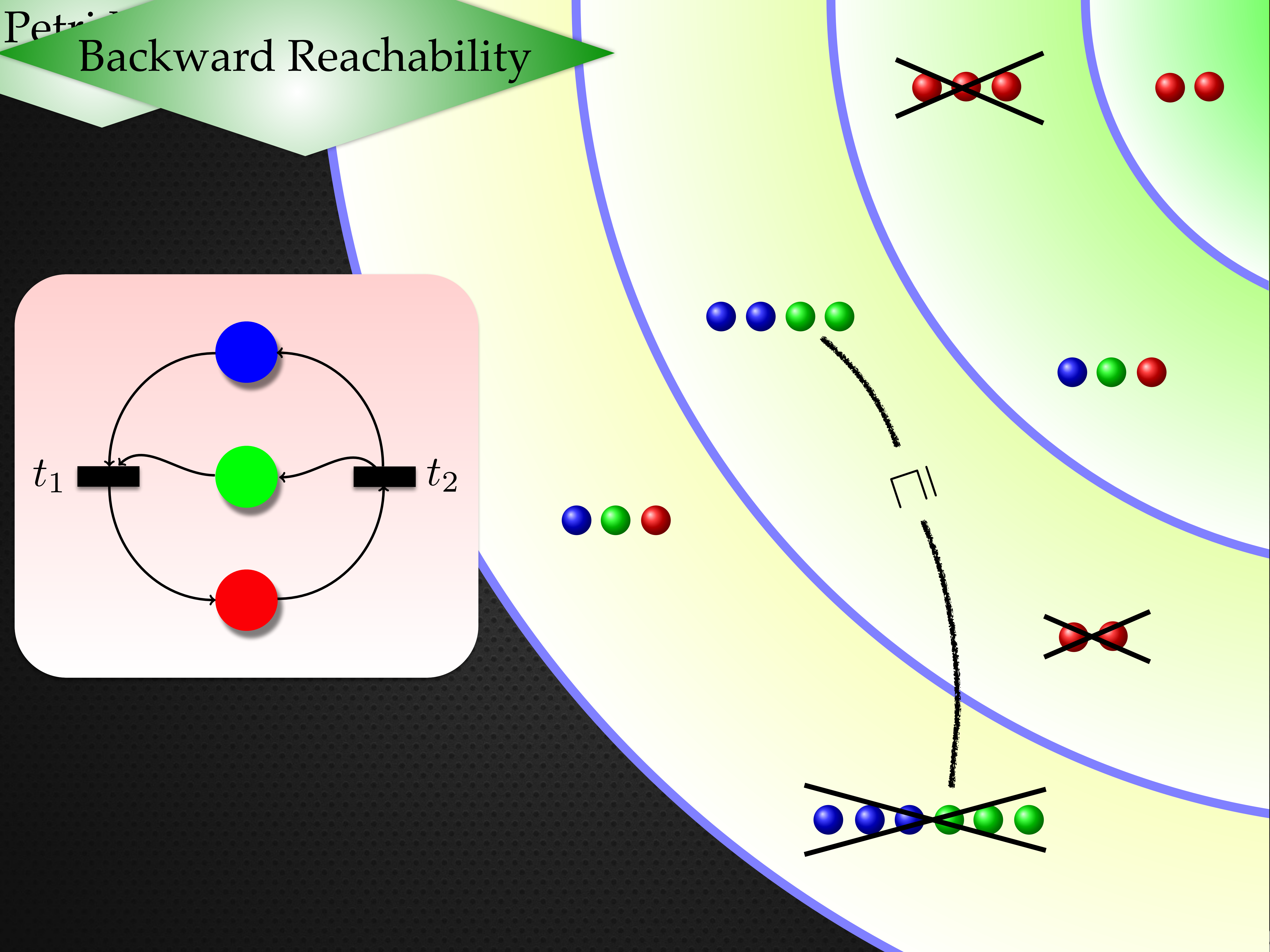
Backward Reachability



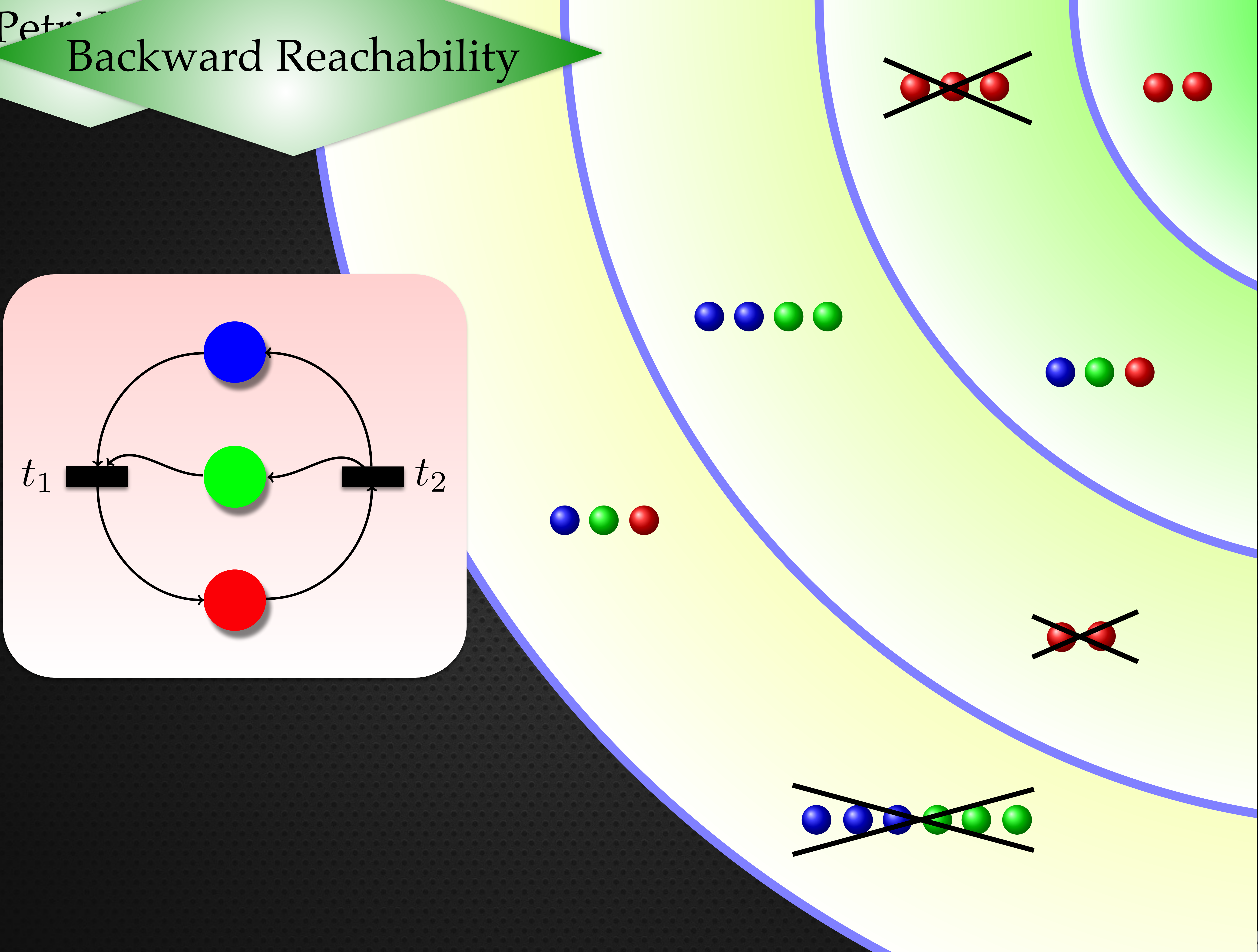
Backward Reachability



Backward Reachability

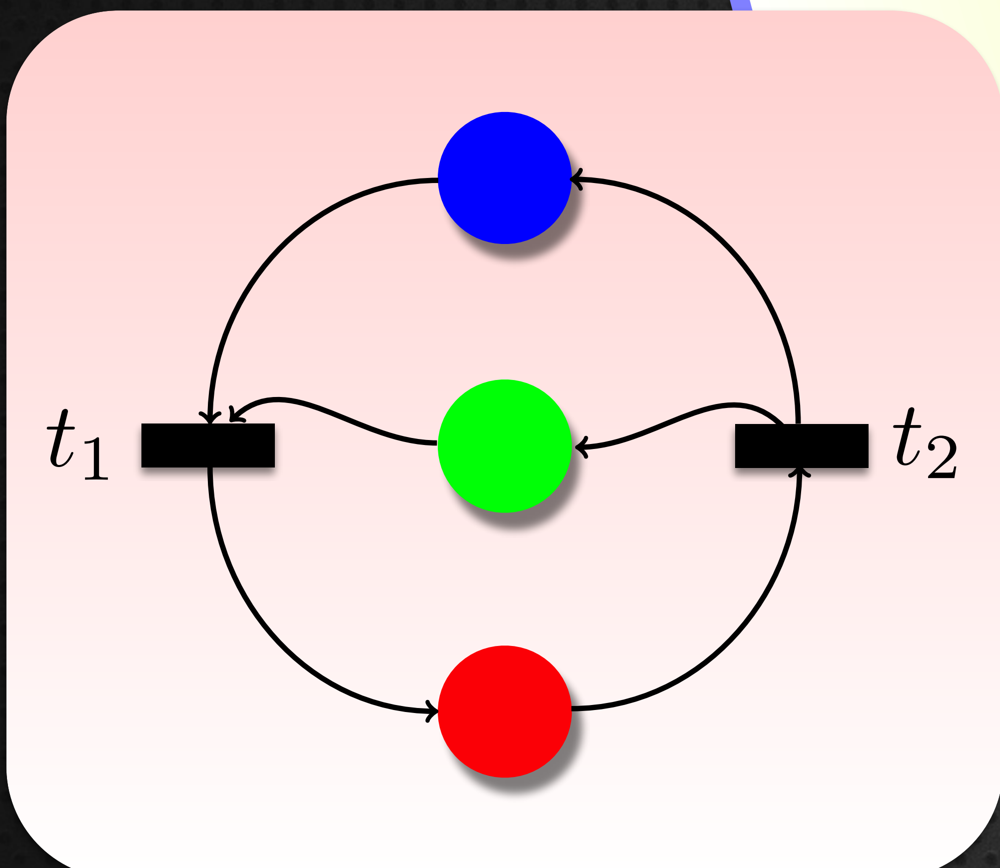


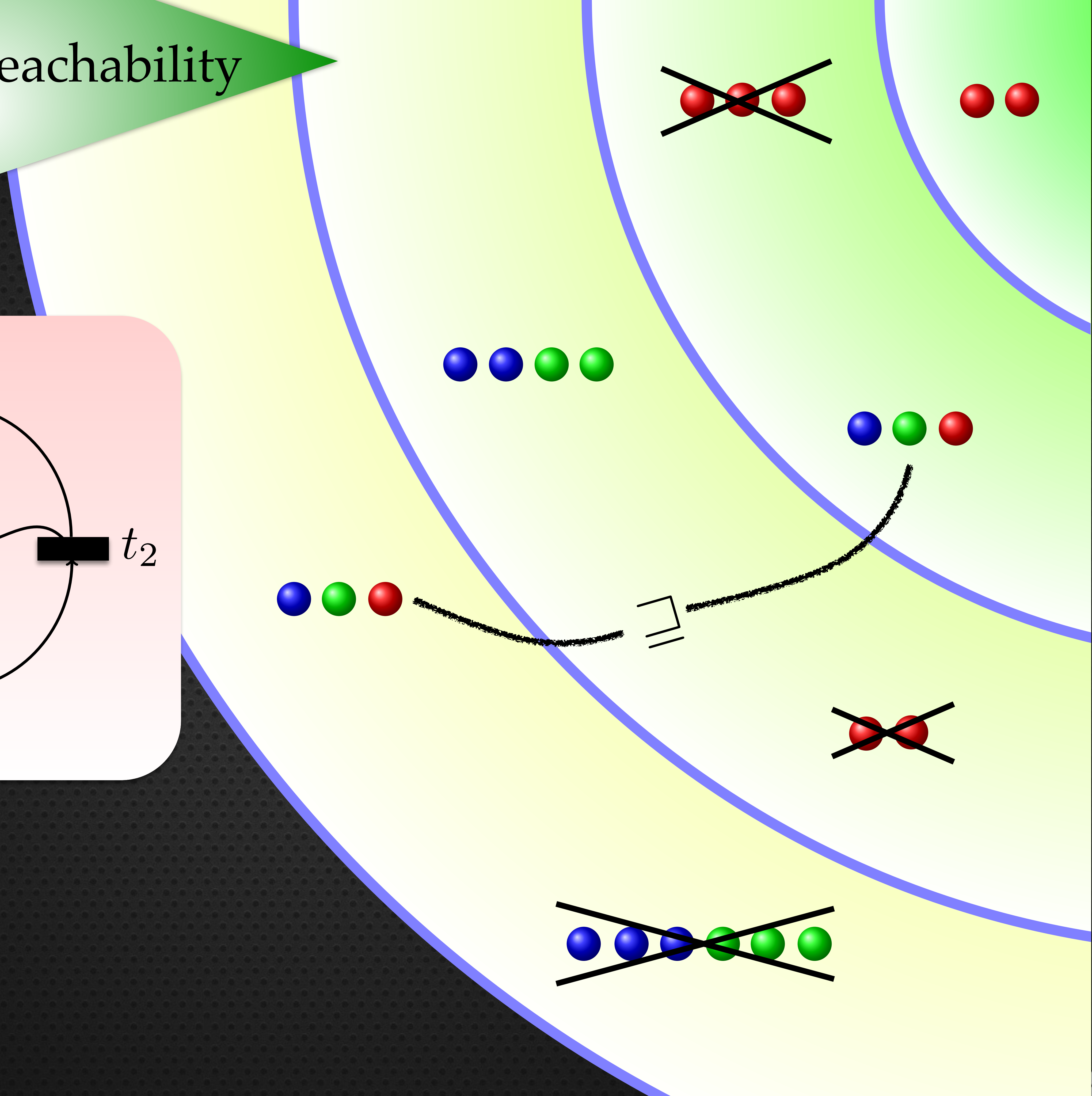
Backward Reachability



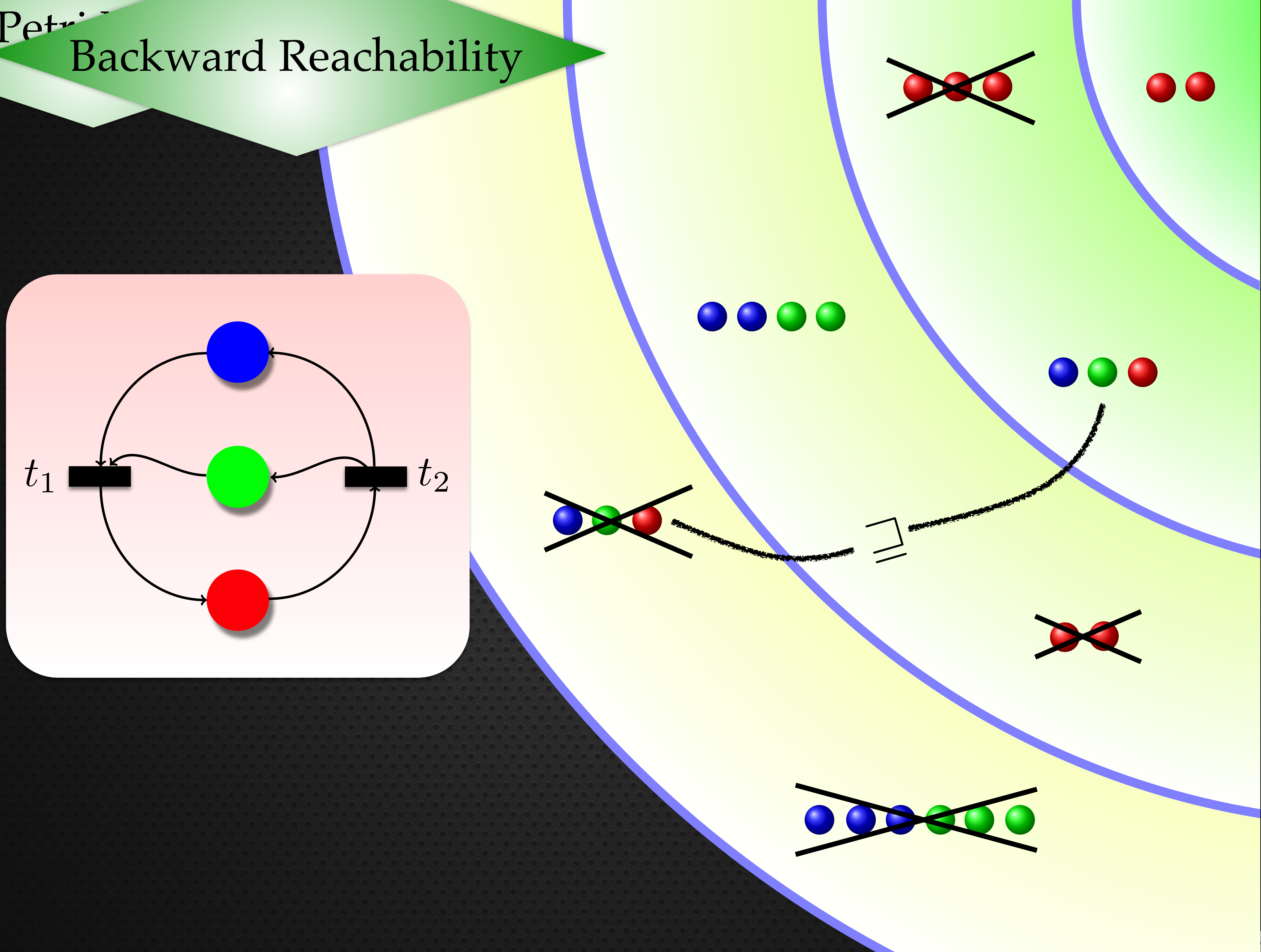
Petri ...

Backward Reachability

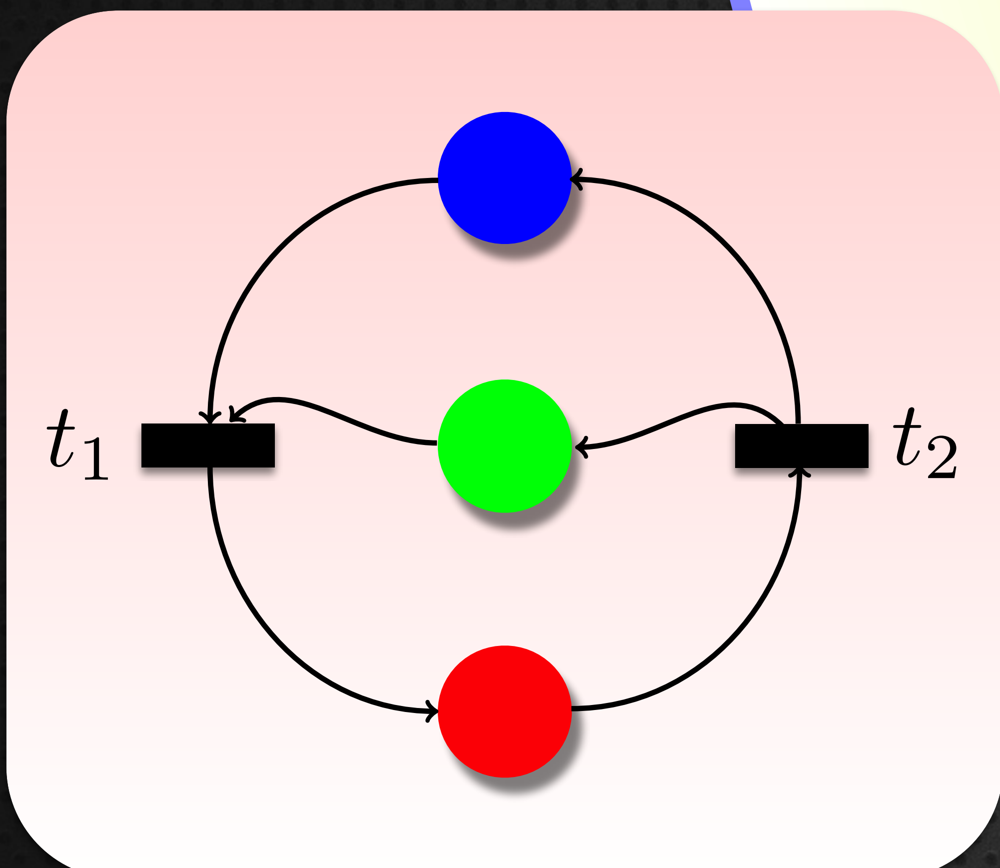


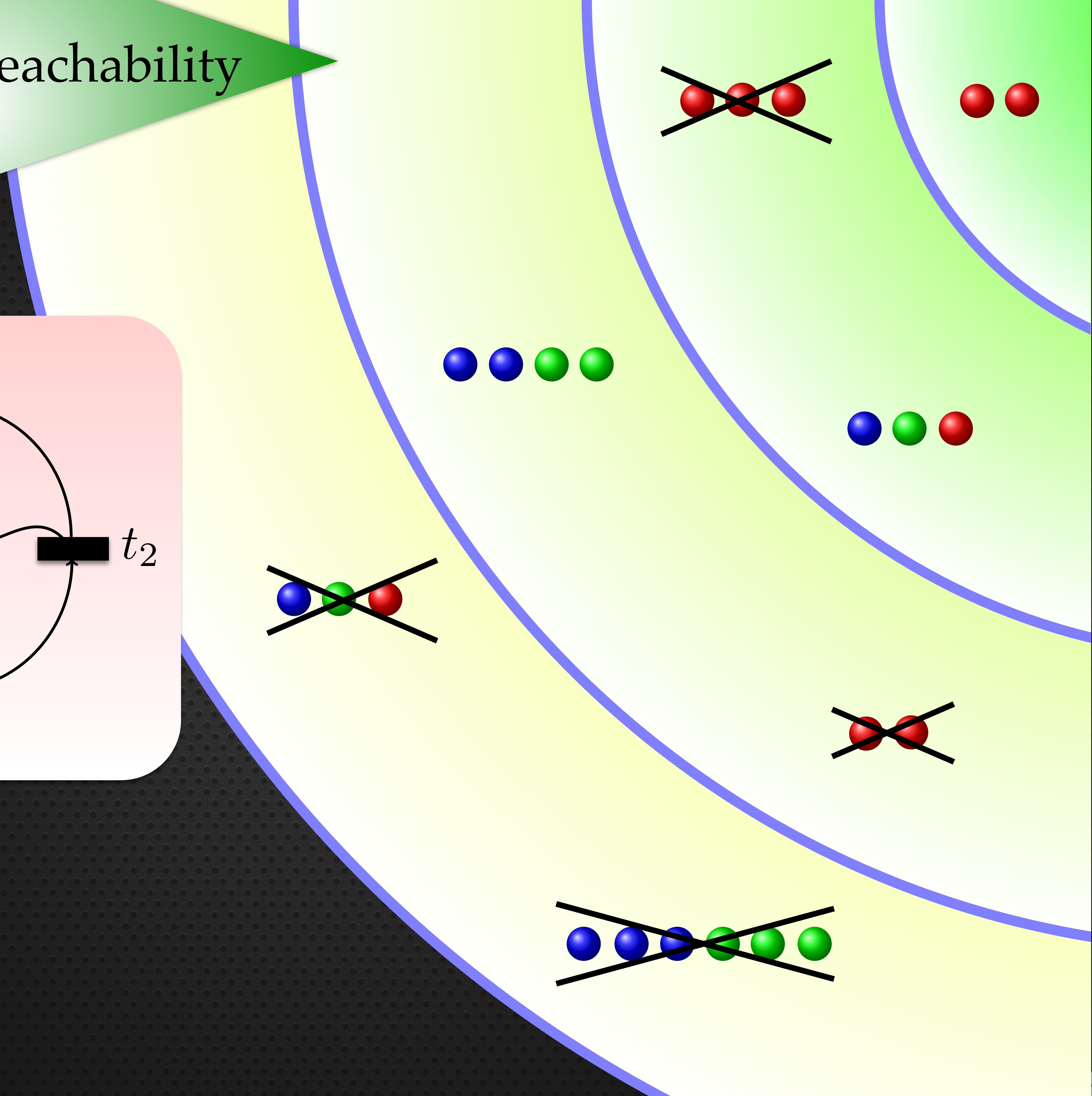


Backward Reachability

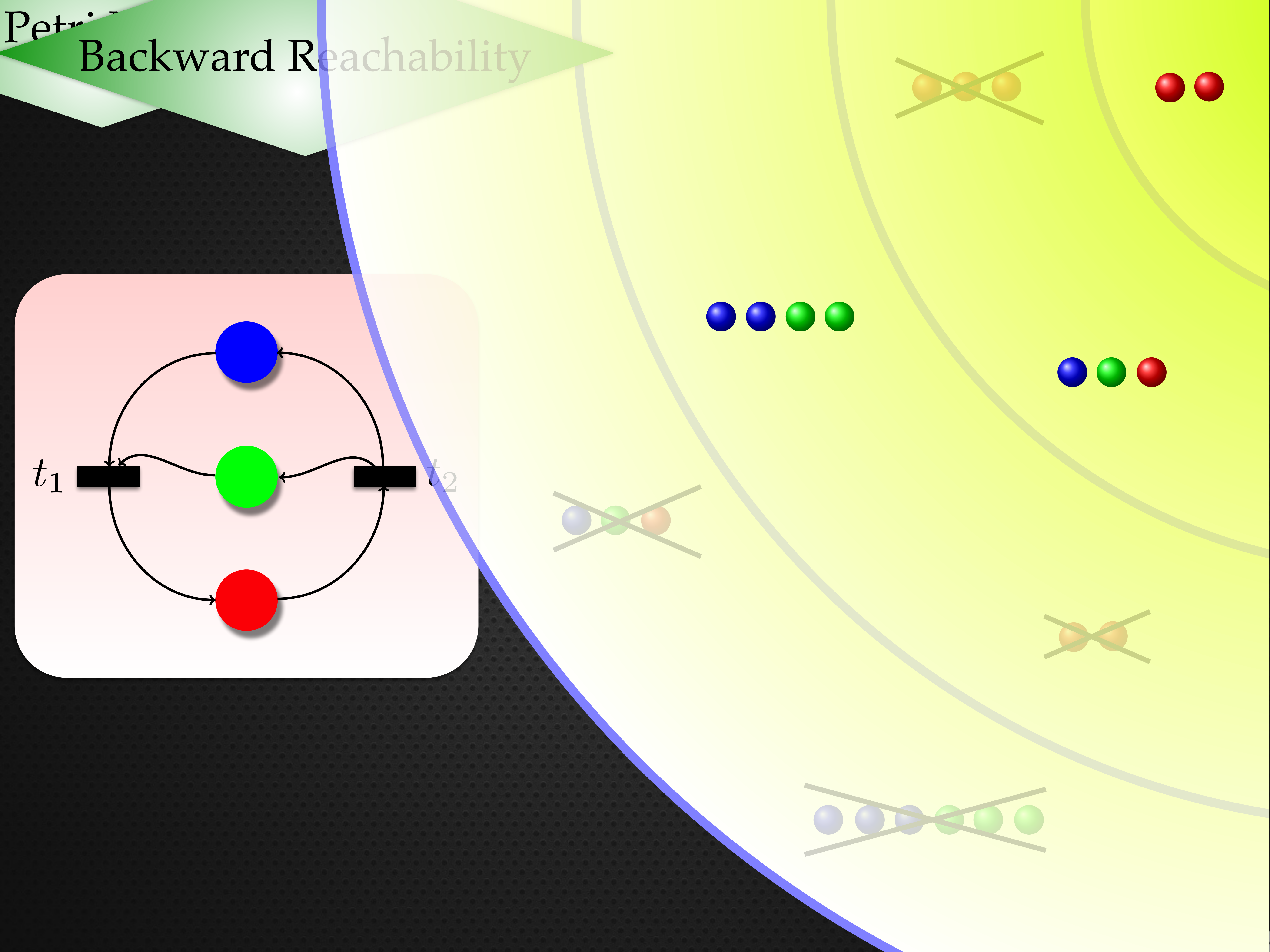


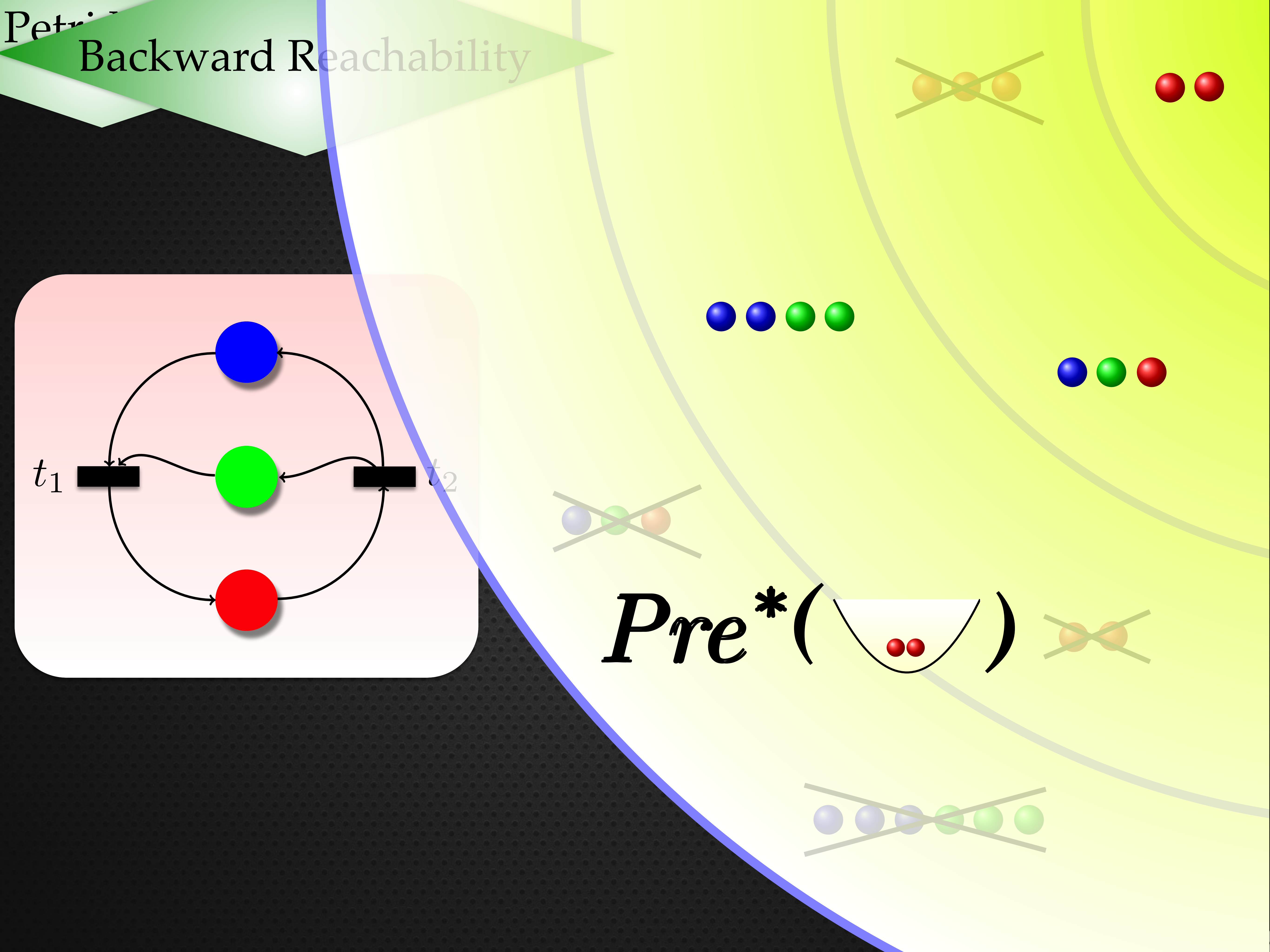
Backward Reachability



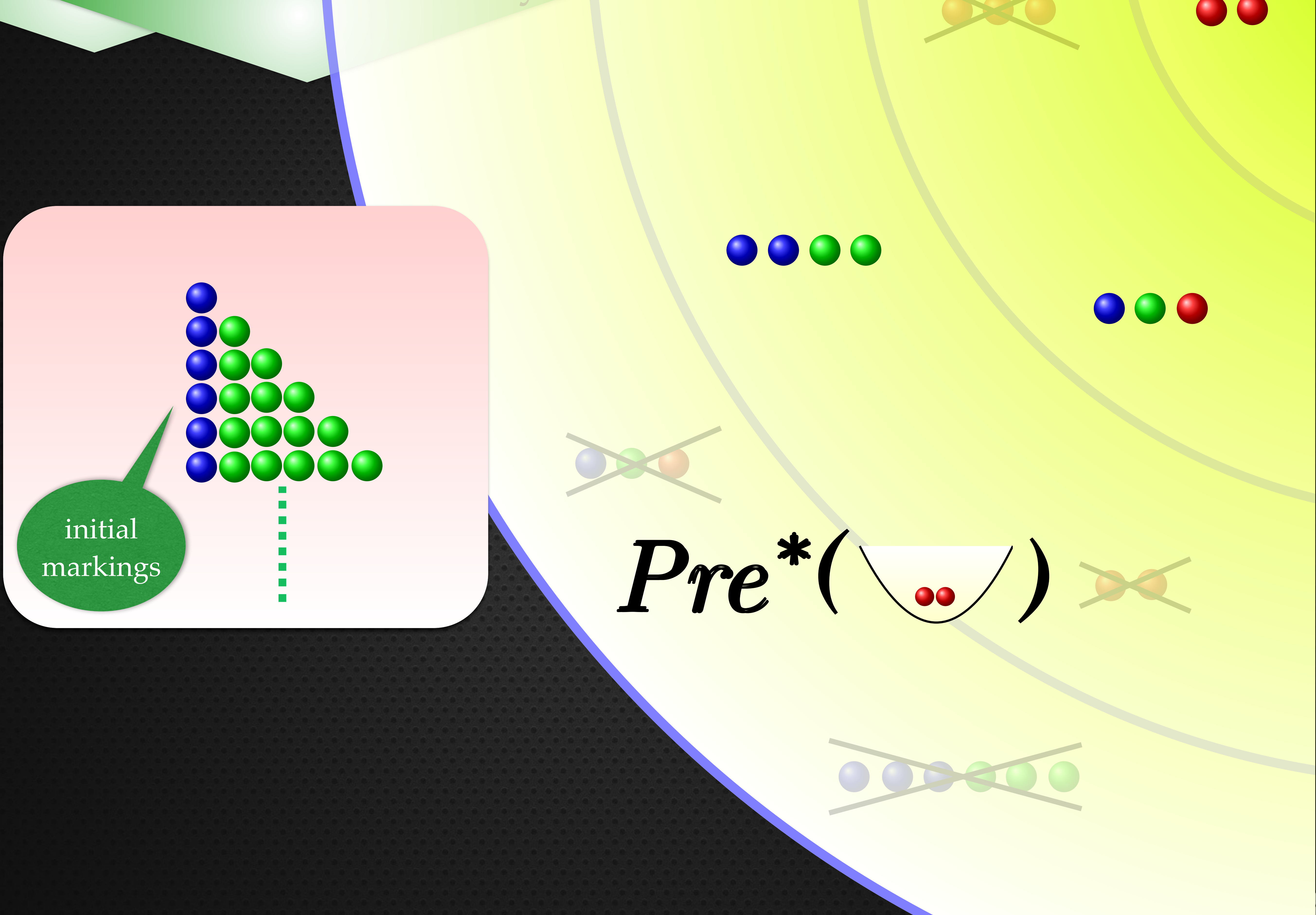


Backward Reachability

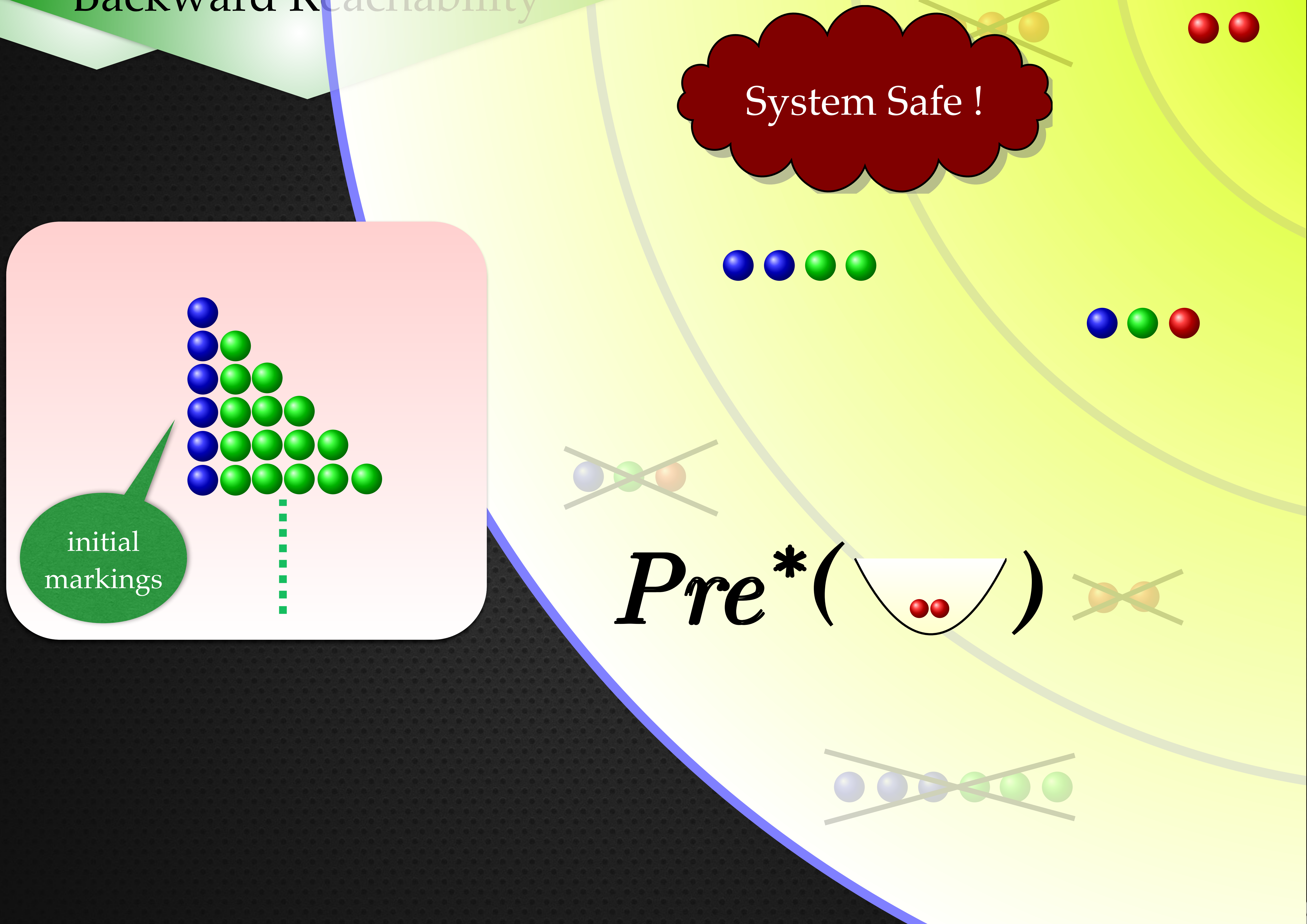



Backward Reachability
t_1
t_2
$Pre^*(\;)$

Backward Reachability



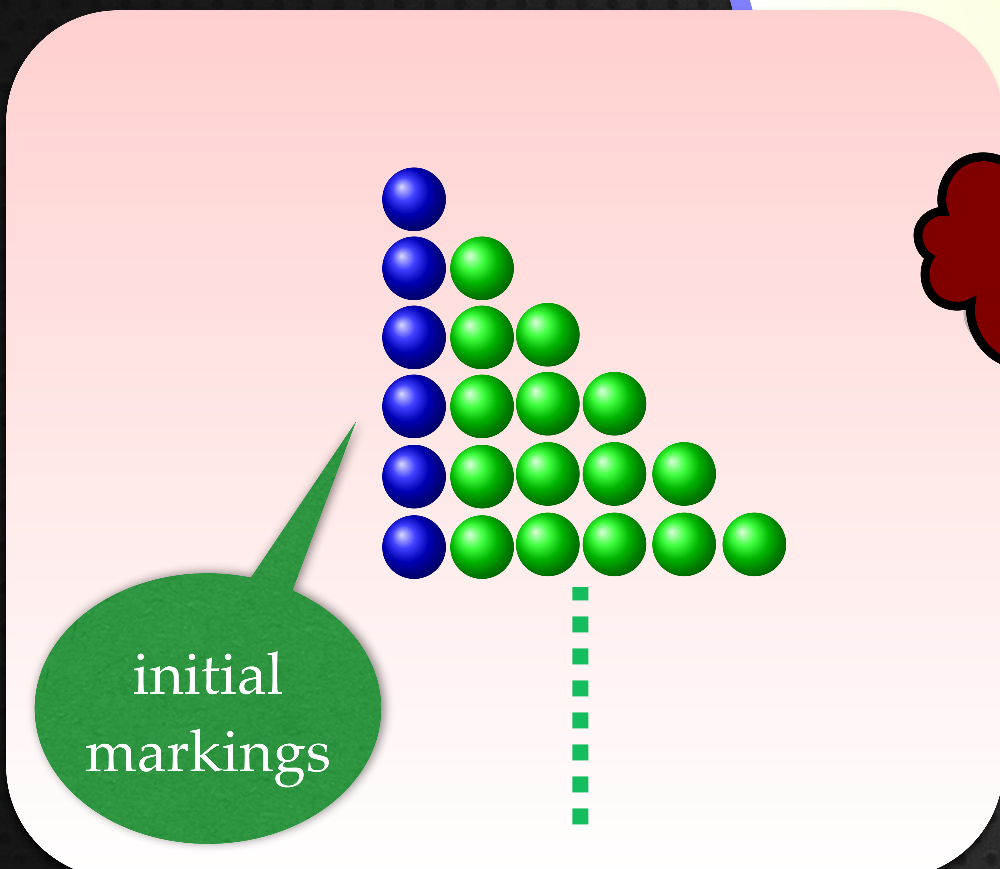
Backward Reachability



Backward Reachability

System Safe !

symbolic representation = finite multisets



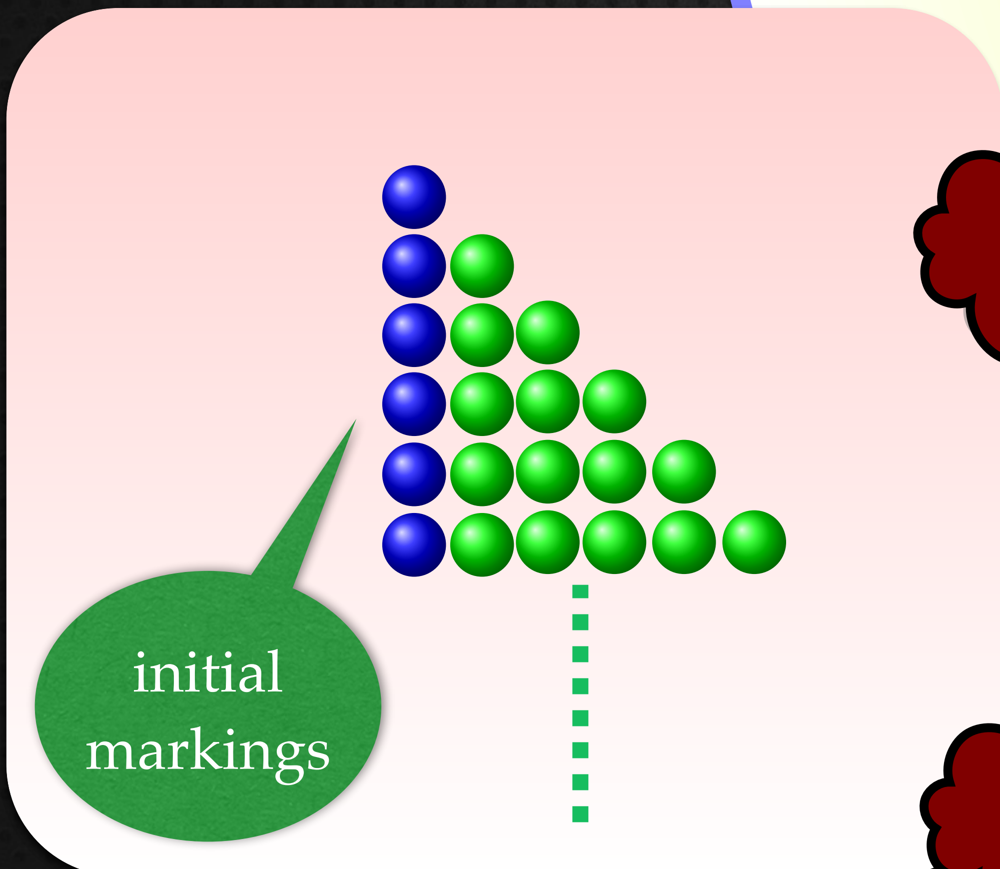
$Pre^*(\quad)$

Backward Reachability

System Safe !

symbolic representation = finite multisets

Termination: multisets well quasi-ordered



Well Quasi-Ordering

Well Quasi-Ordering

infinite sequence of markings

$$m_0, m_1, m_2, \ldots, m_i, \ldots, m_j, \ldots$$

Well Quasi-Ordering

Well Quasi-Ordering

infinite sequence of markings

$$m_0, m_1, m_2, \ldots, m_i, \ldots, m_j, \ldots$$

$$\exists i < j : m_i \sqsubseteq m_j$$

Well Quasi-Ordering

Well Quasi-Ordering

infinite sequence of markings

$$m_0, m_1, m_2, \ldots, m_i, \ldots, m_j, \ldots$$

$$\sqsubseteq$$

$$\exists i < j : m_i \sqsubseteq m_j$$

Assume: non-termination

Petri

Backwa

Termination

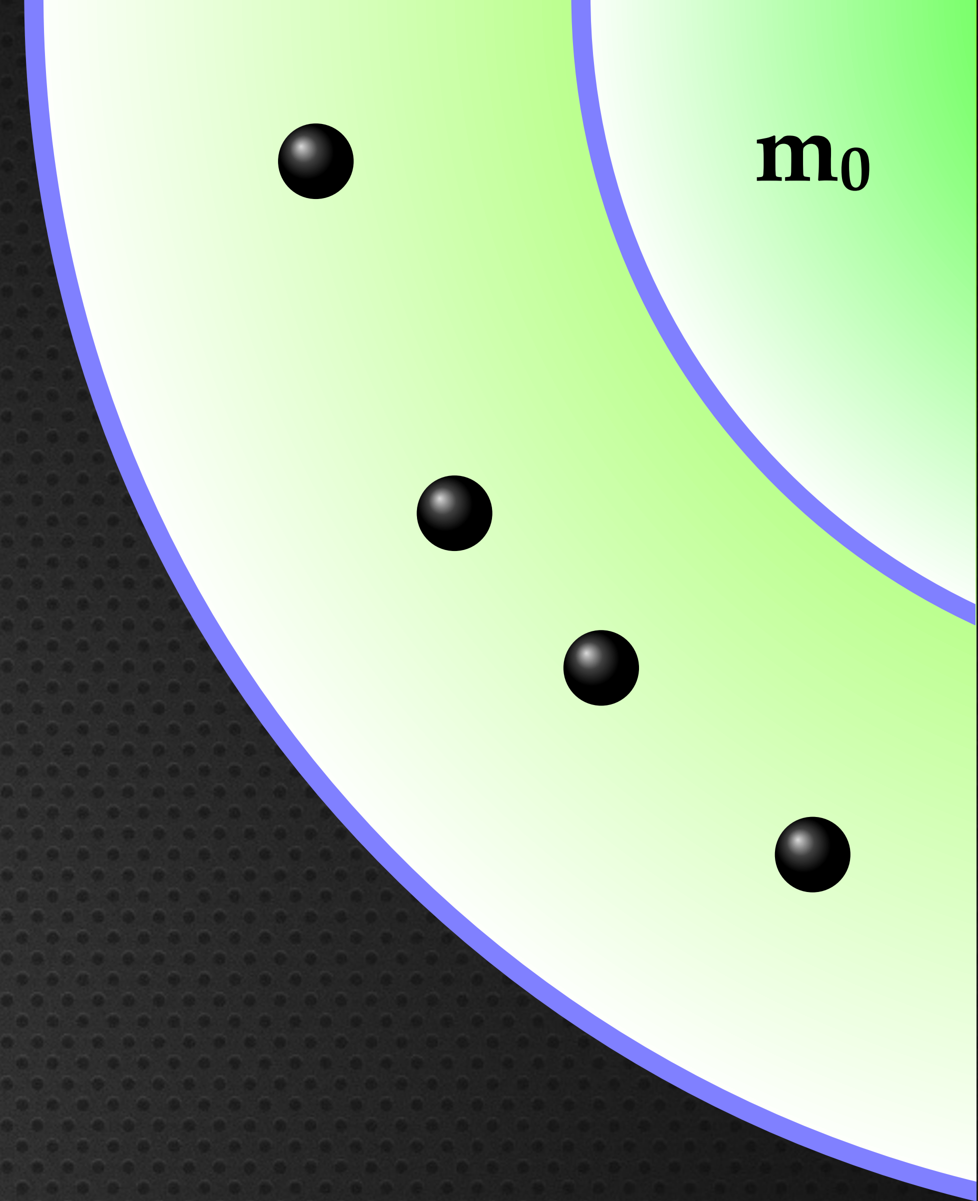
$\mathbf{m_0}$

Assume: non-termination

Pet

Backwa

Termination



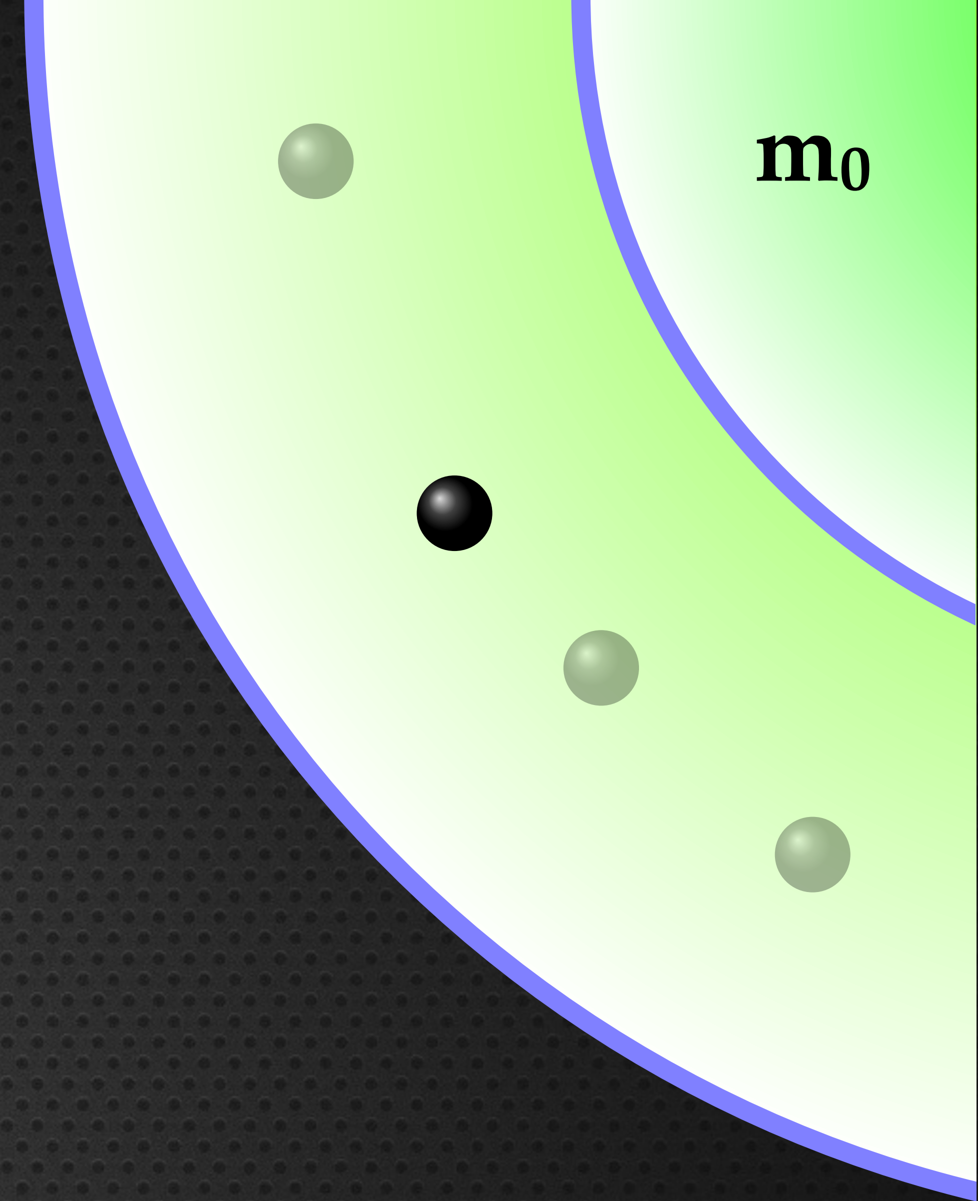
Assume: non-termination

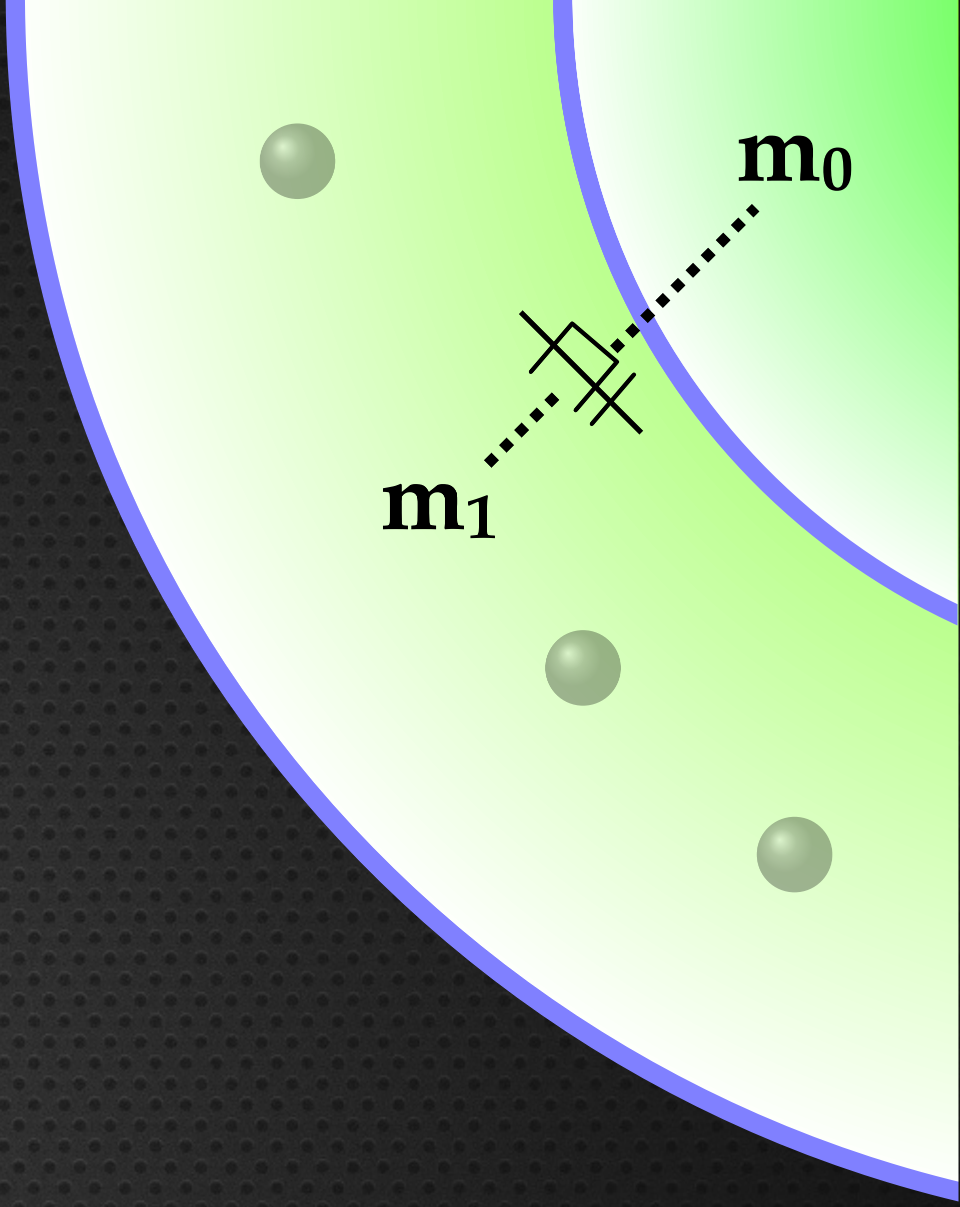
Petri

Backwa

Termination

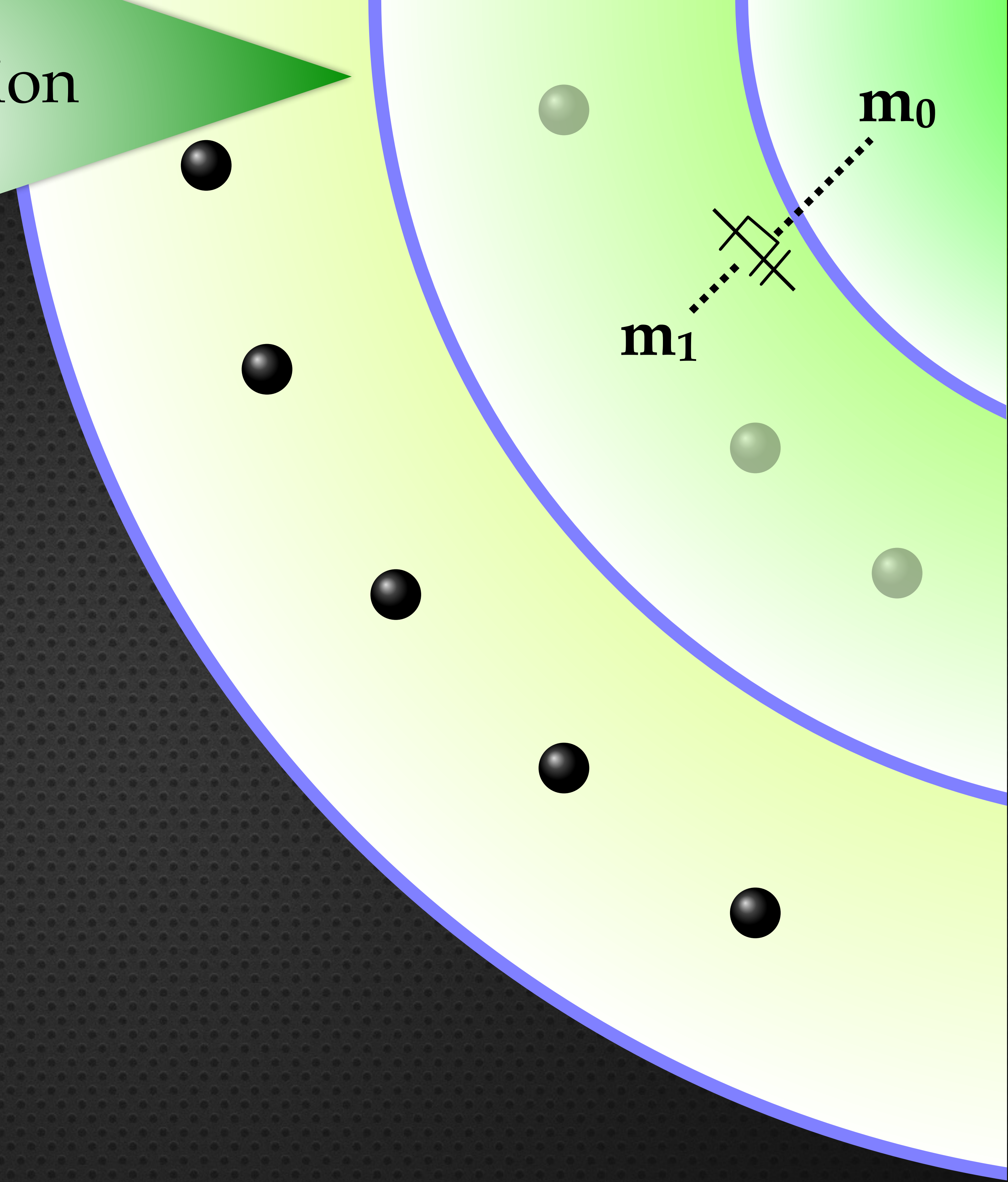
Assume: non-termination



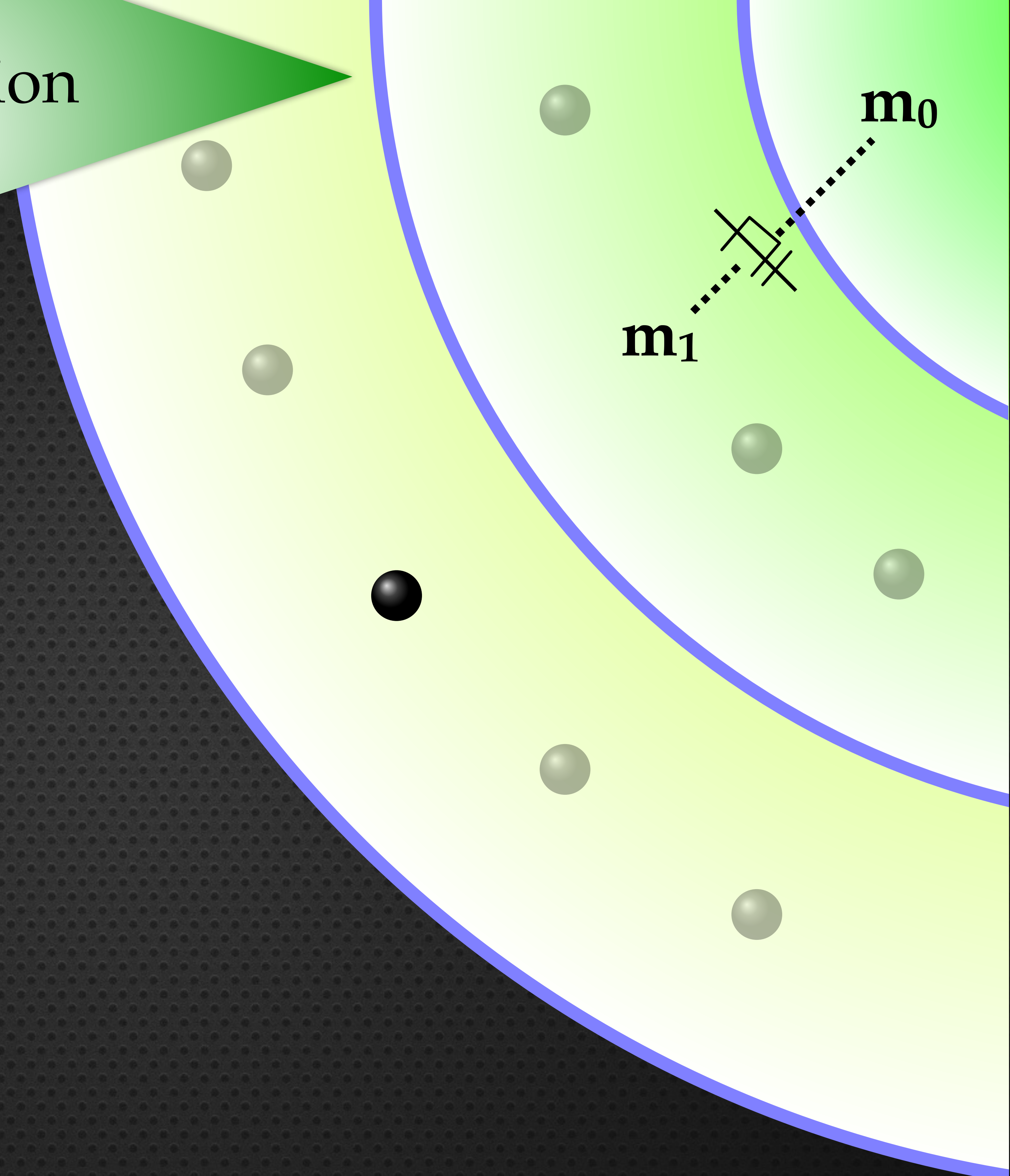


Assume: non-termination

Assume: non-termination



Assume: non-termination

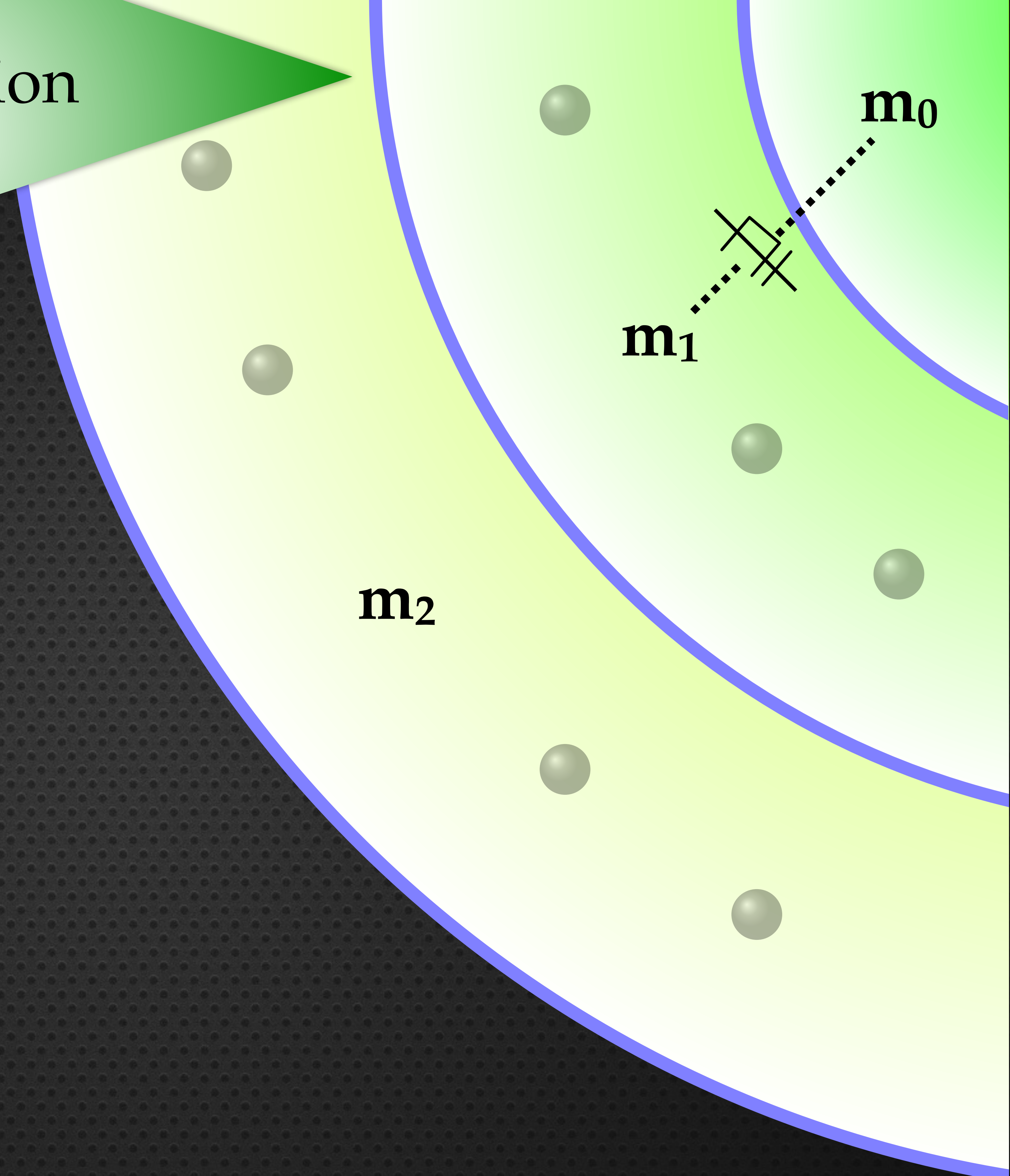


Petri

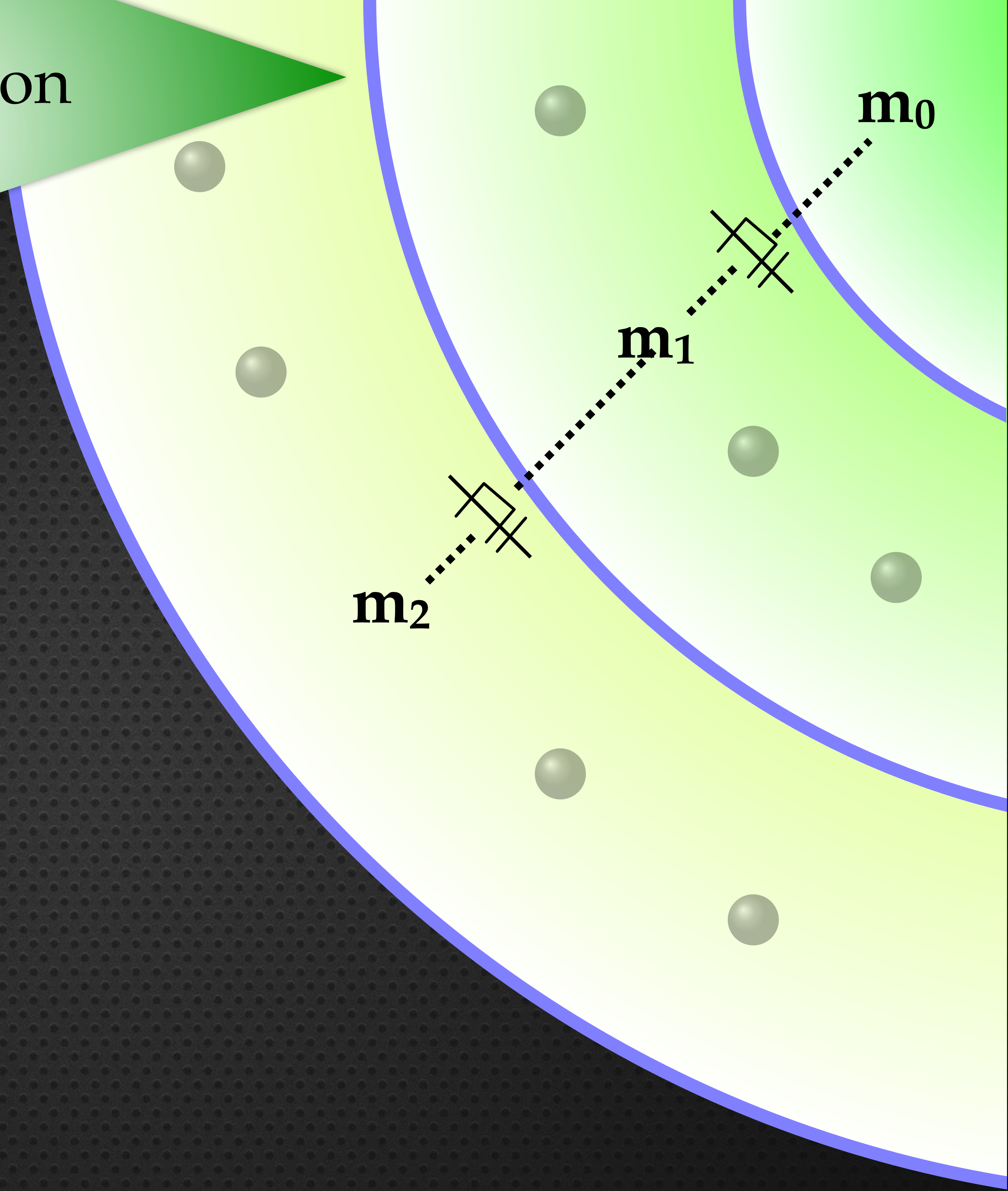
Backwa

Termination

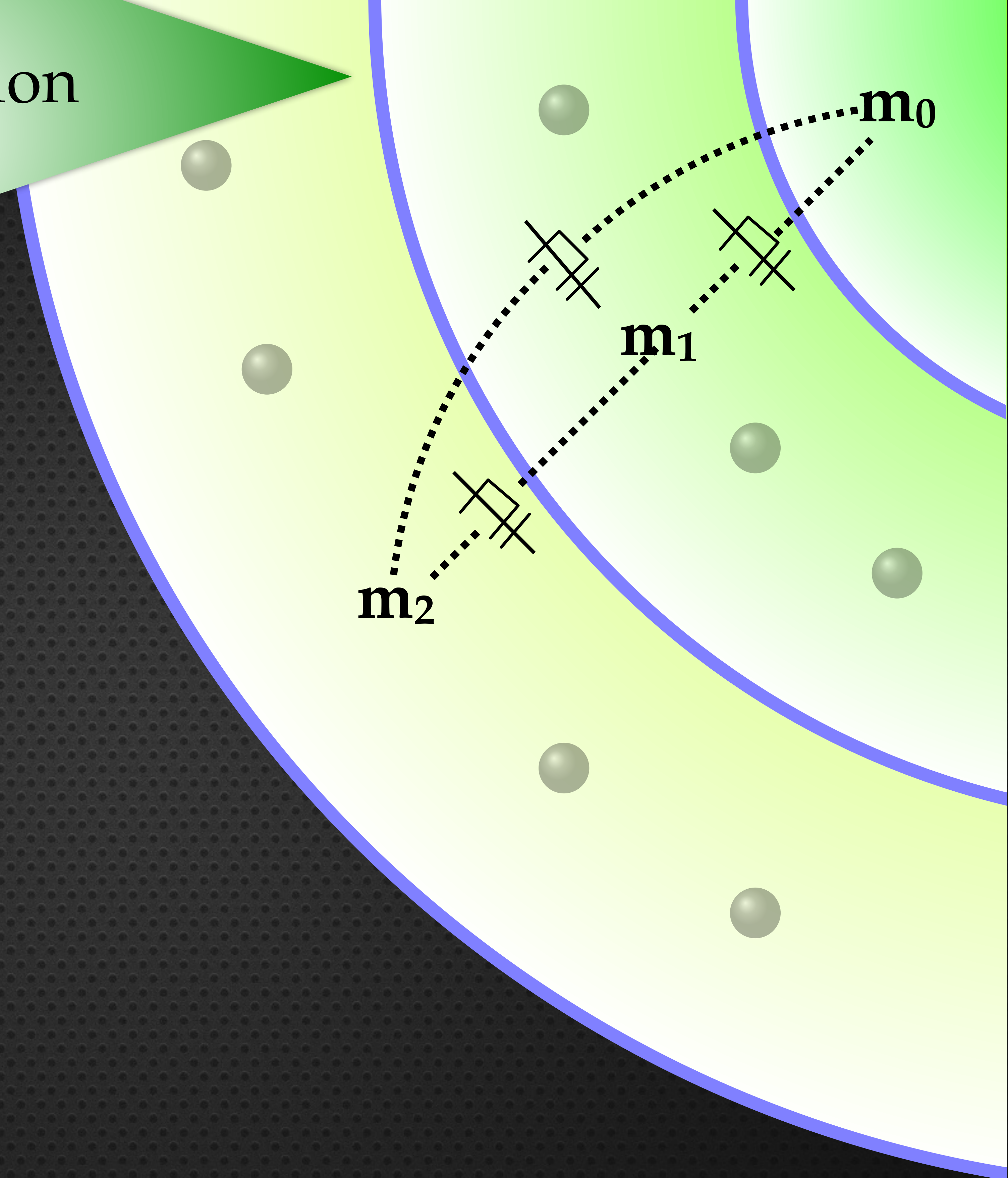
Assume: non-termination

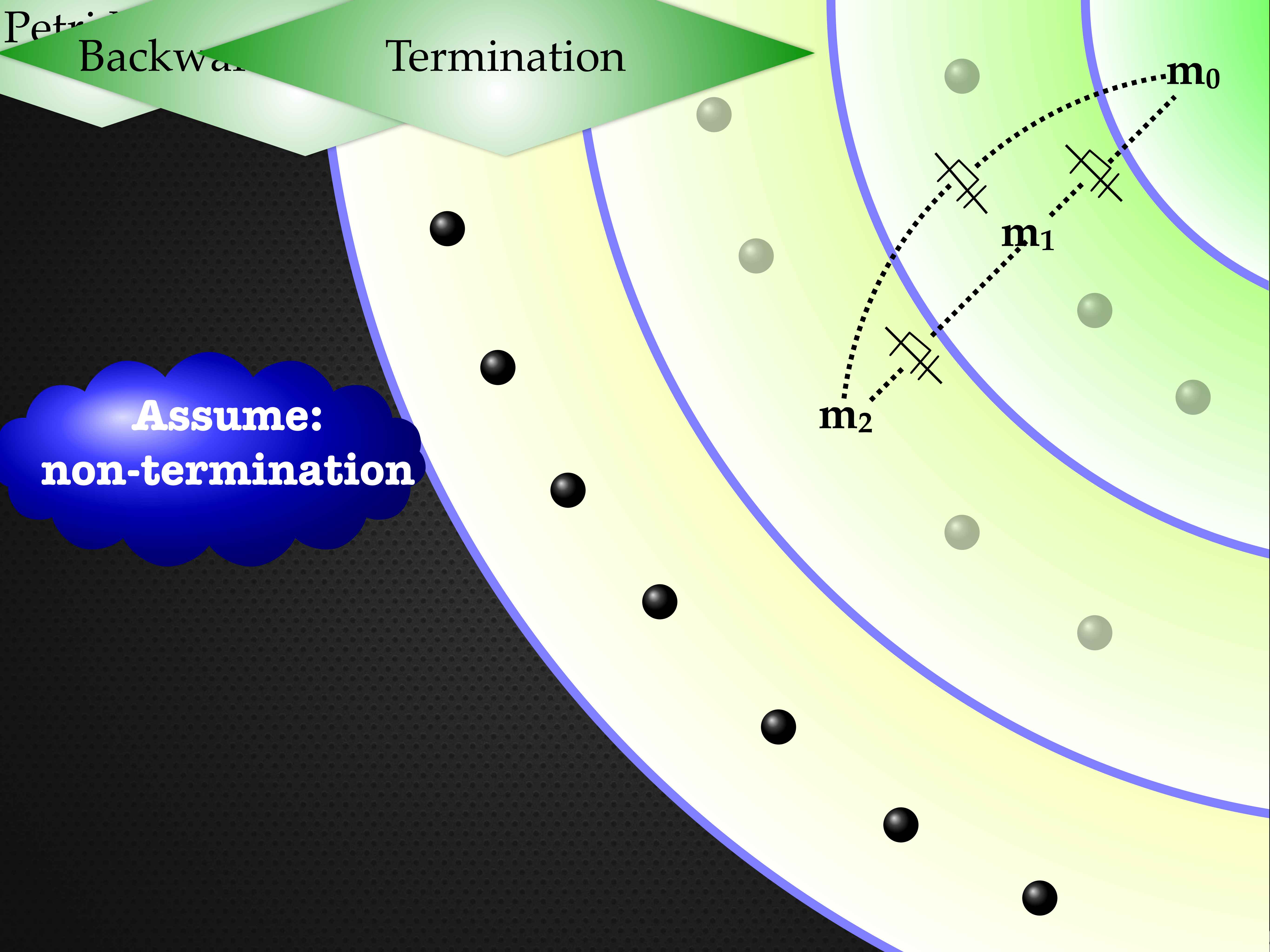


Assume: non-termination

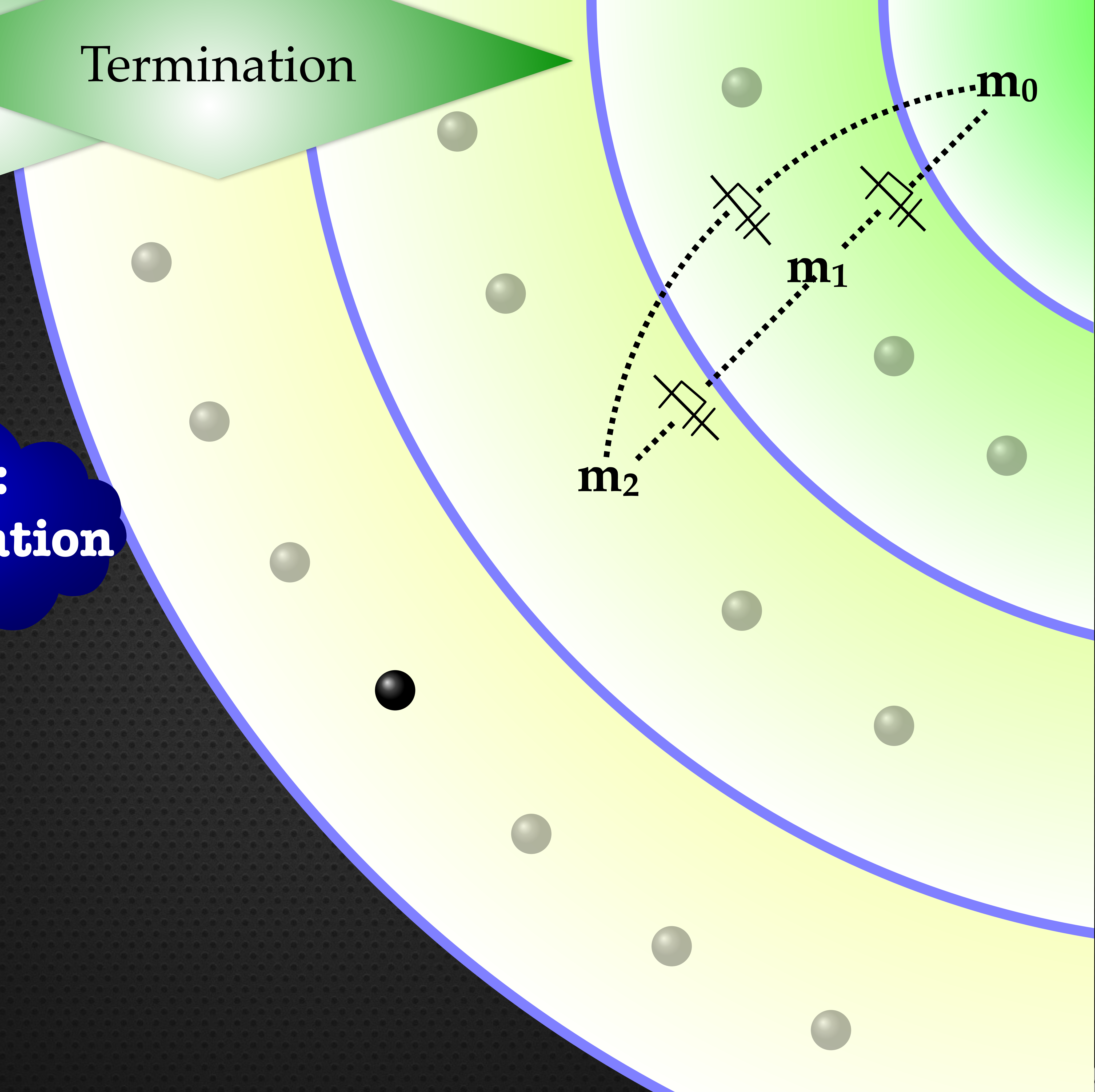


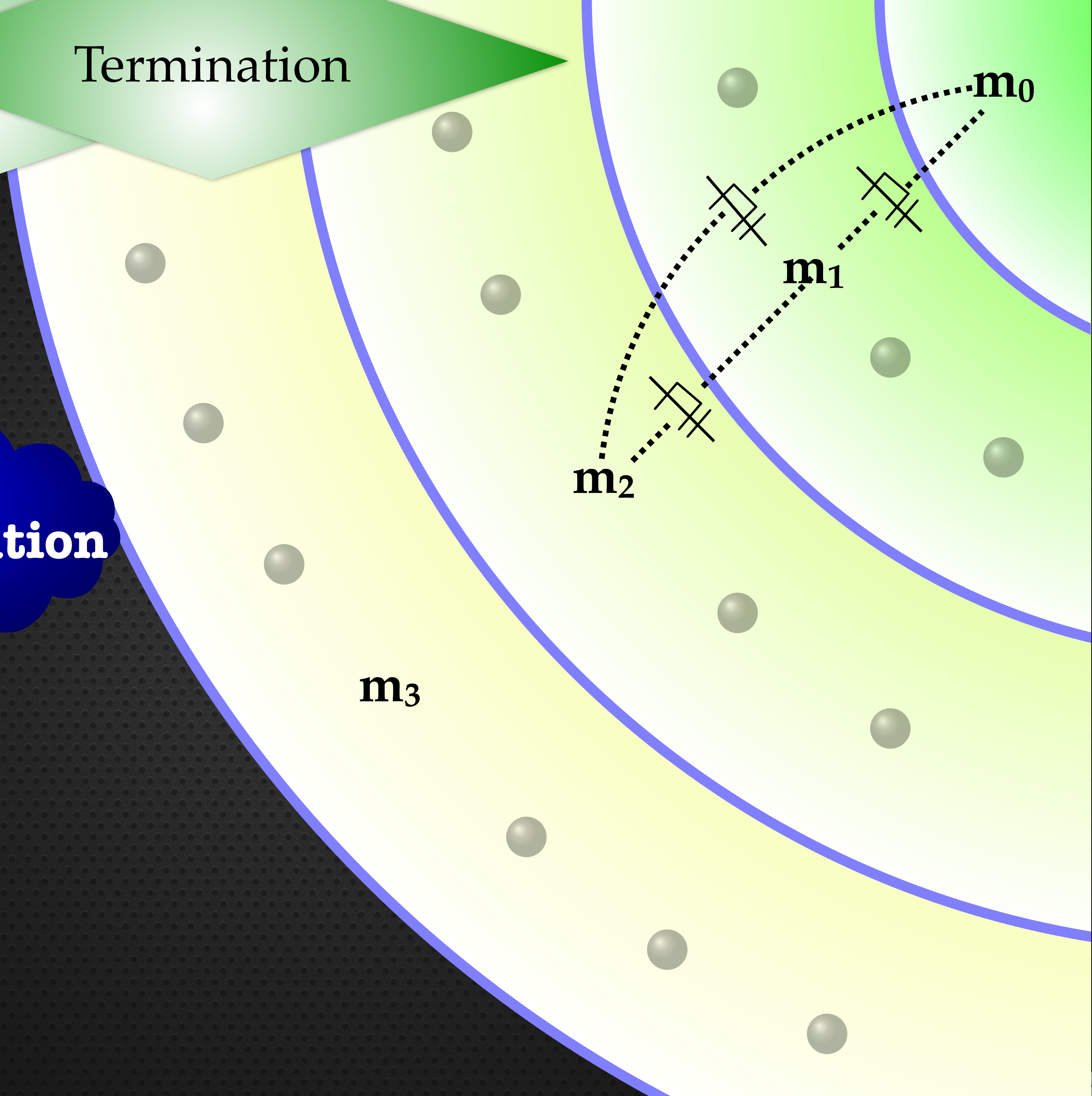
Assume: non-termination




$\mathbf{m}_0$
$\mathbf{m}_1$
$\mathbf{m}_2$
Assume:
non-termination

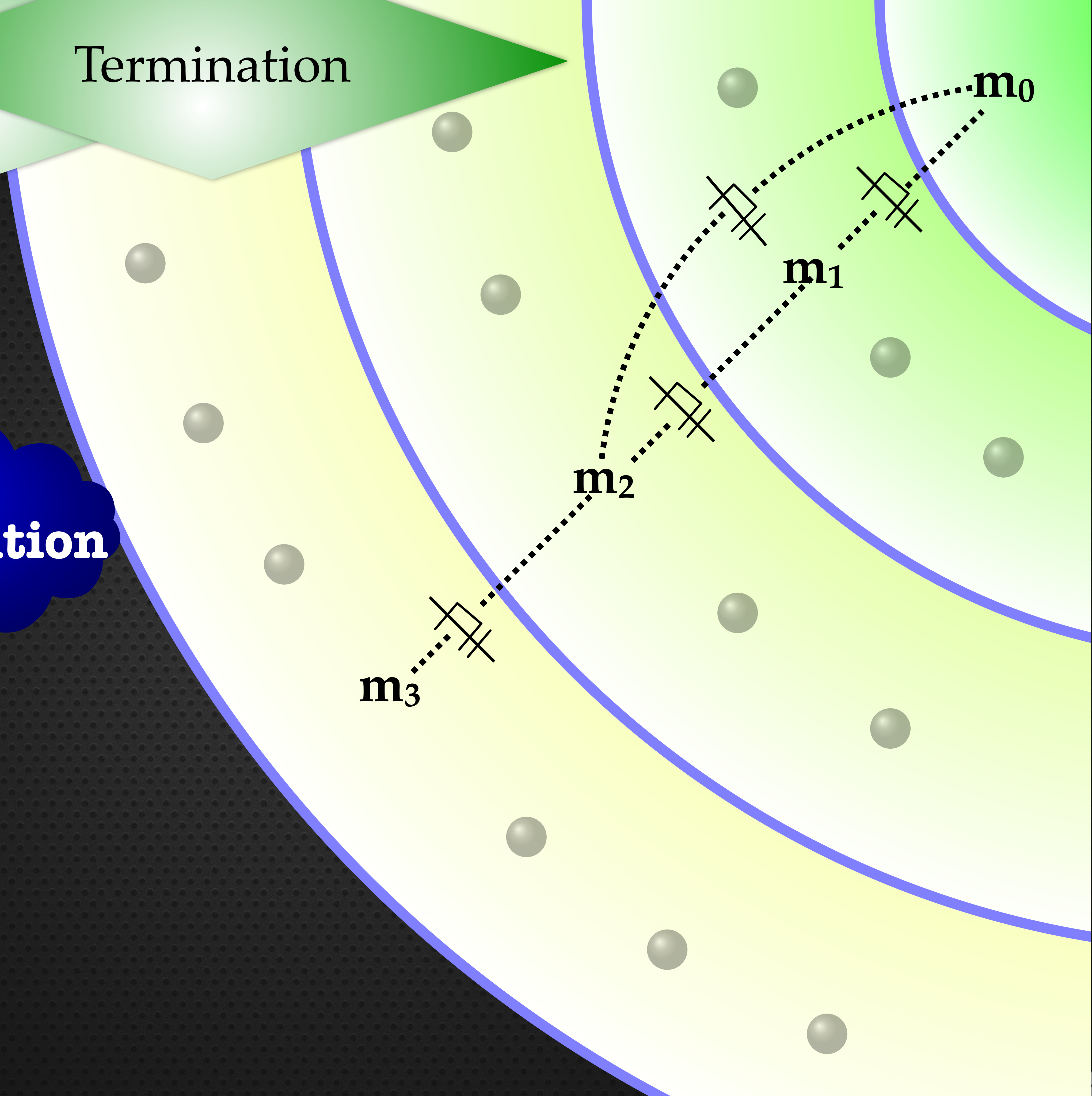
Assume: non-termination



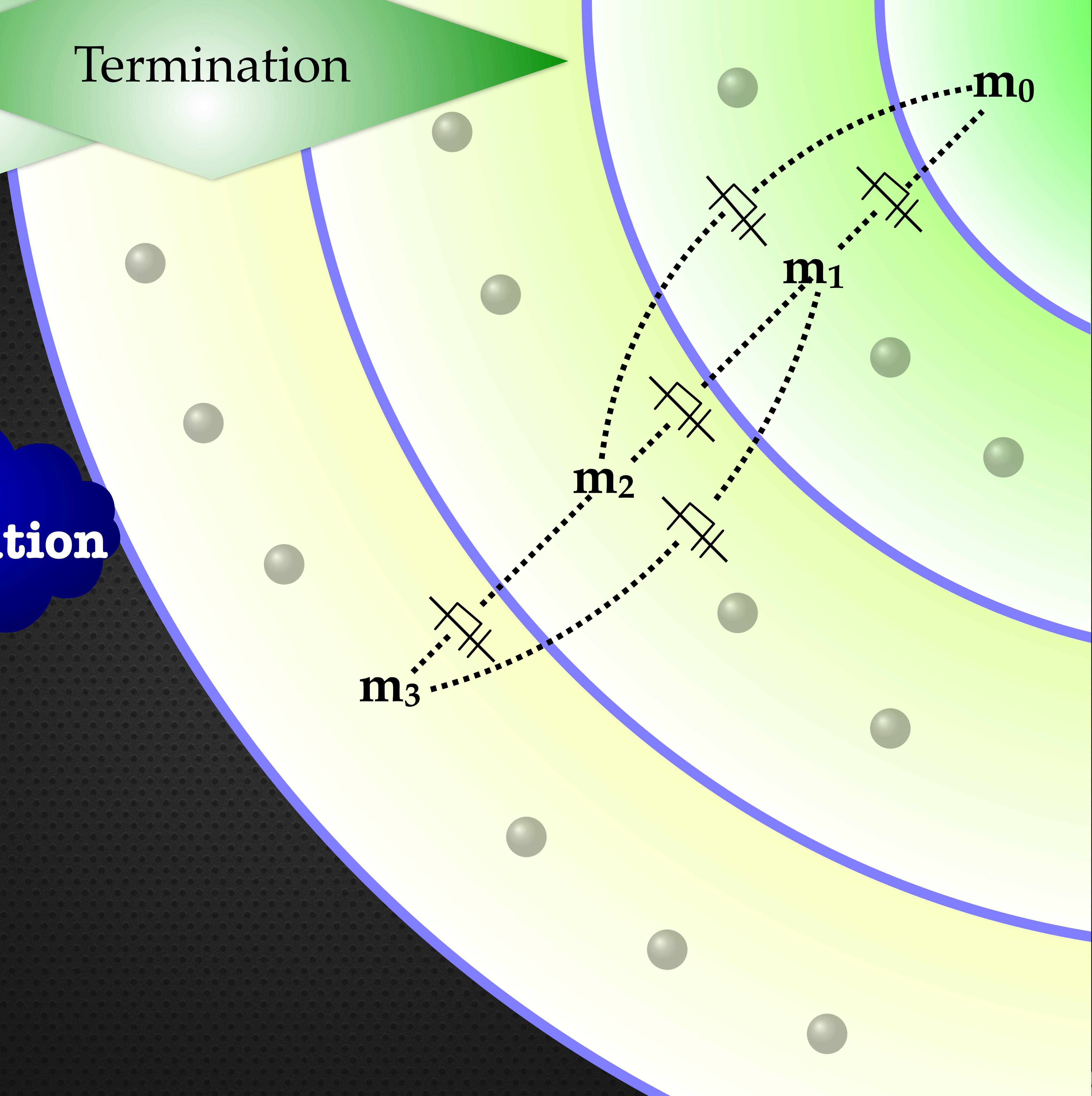


Assume: non-termination

Assume: non-termination

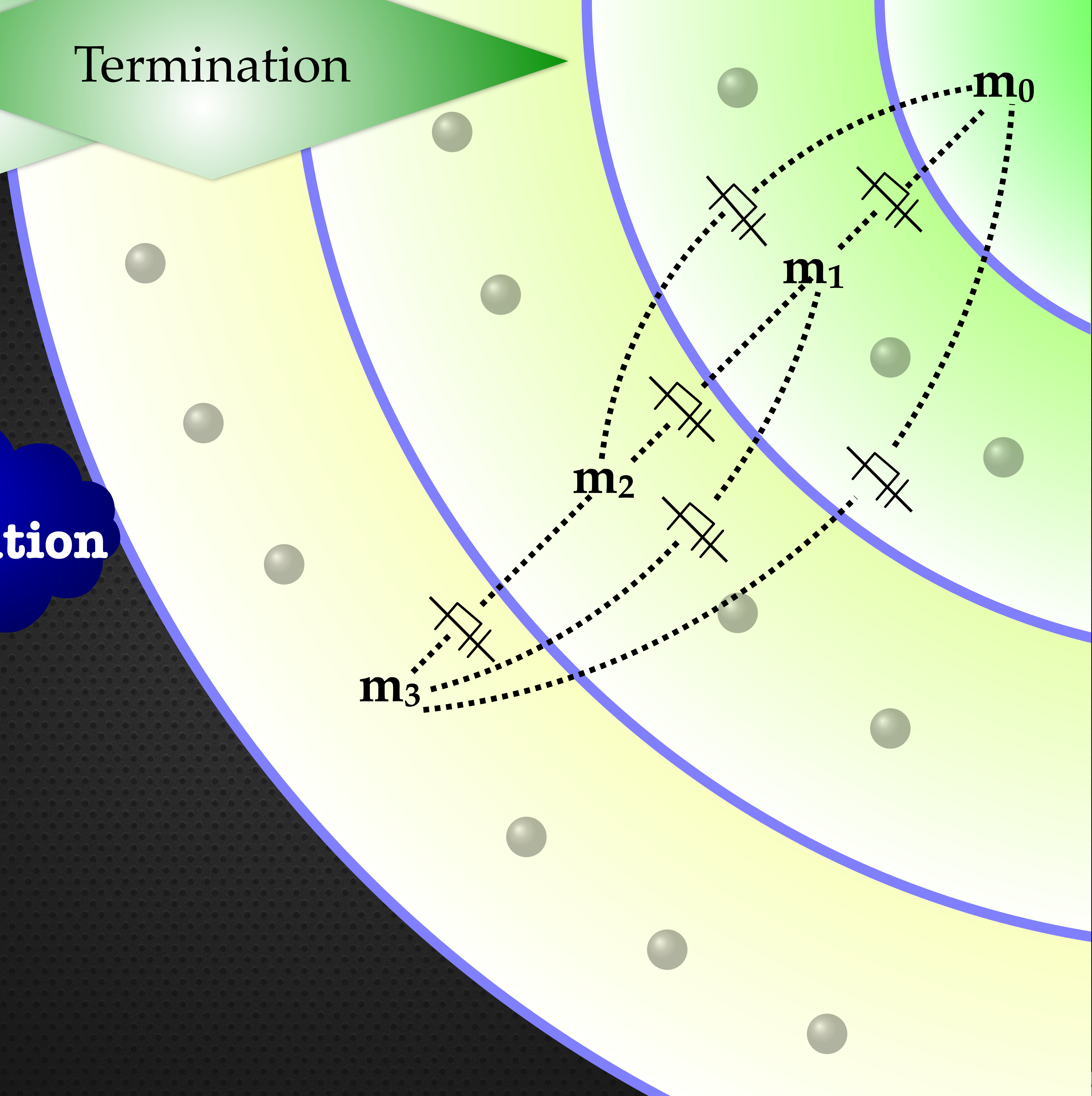


Assume: non-termination



Petri ... Backwa... Termination

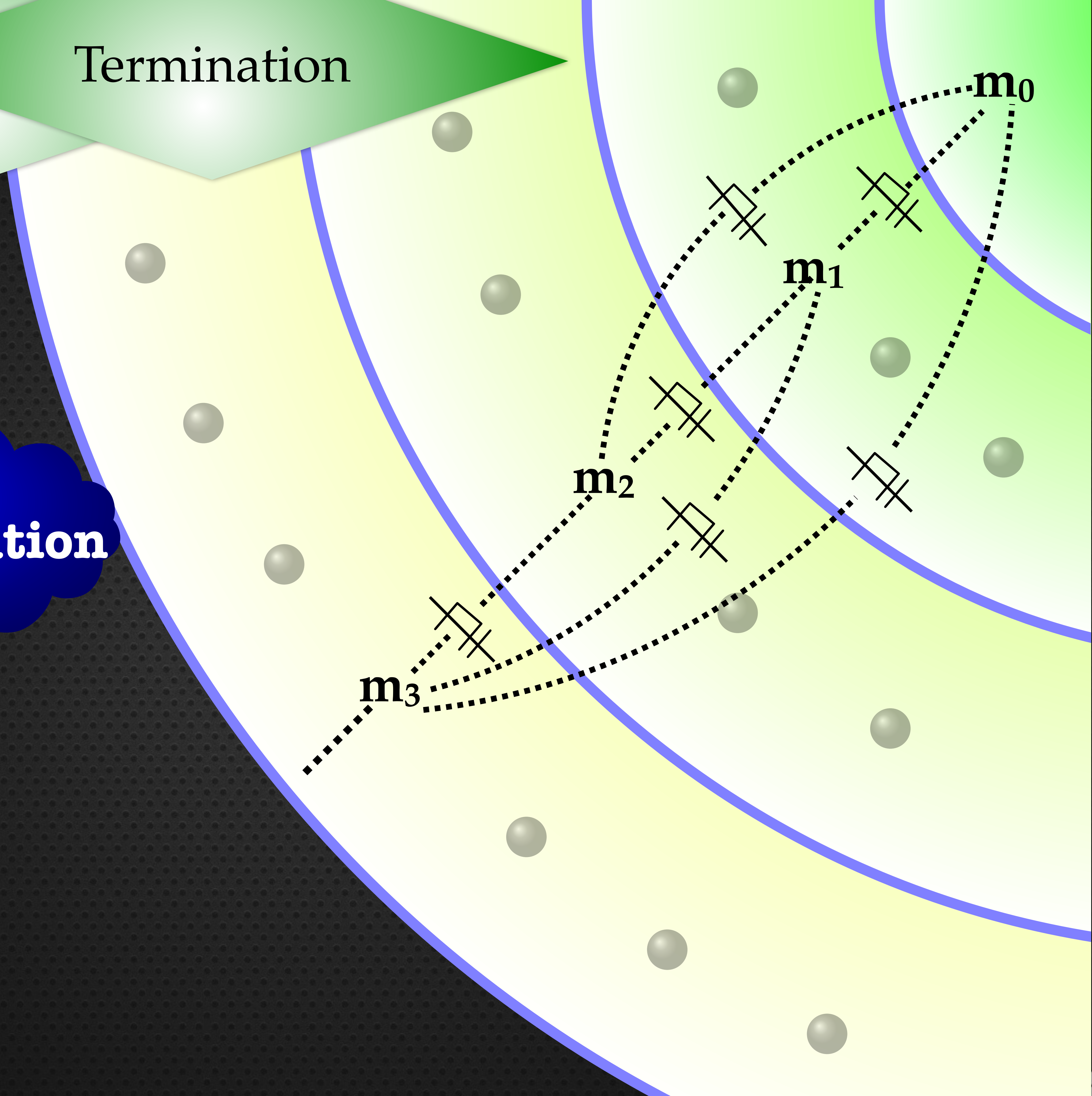
Assume: non-termination



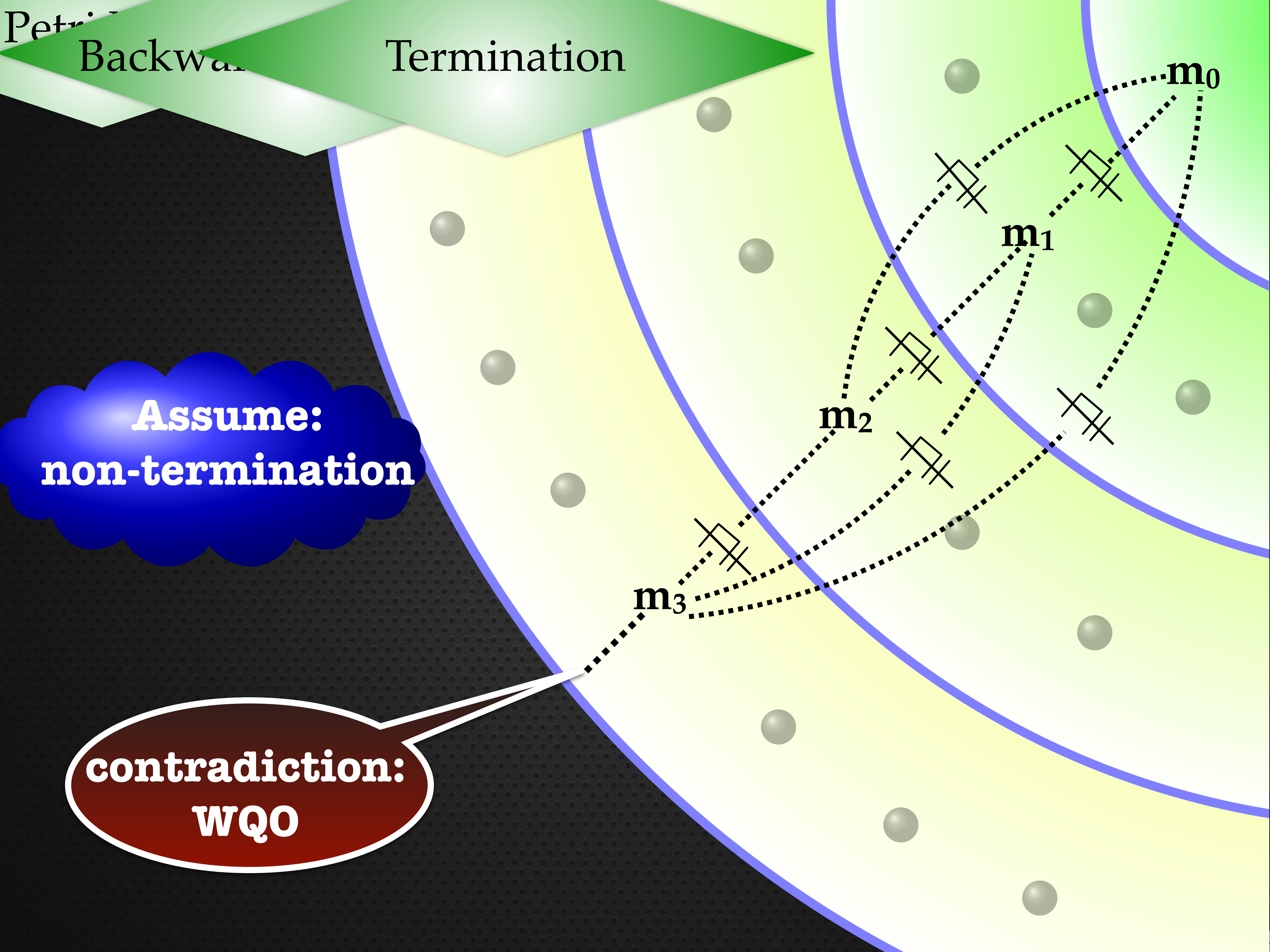
Petr

Backwa

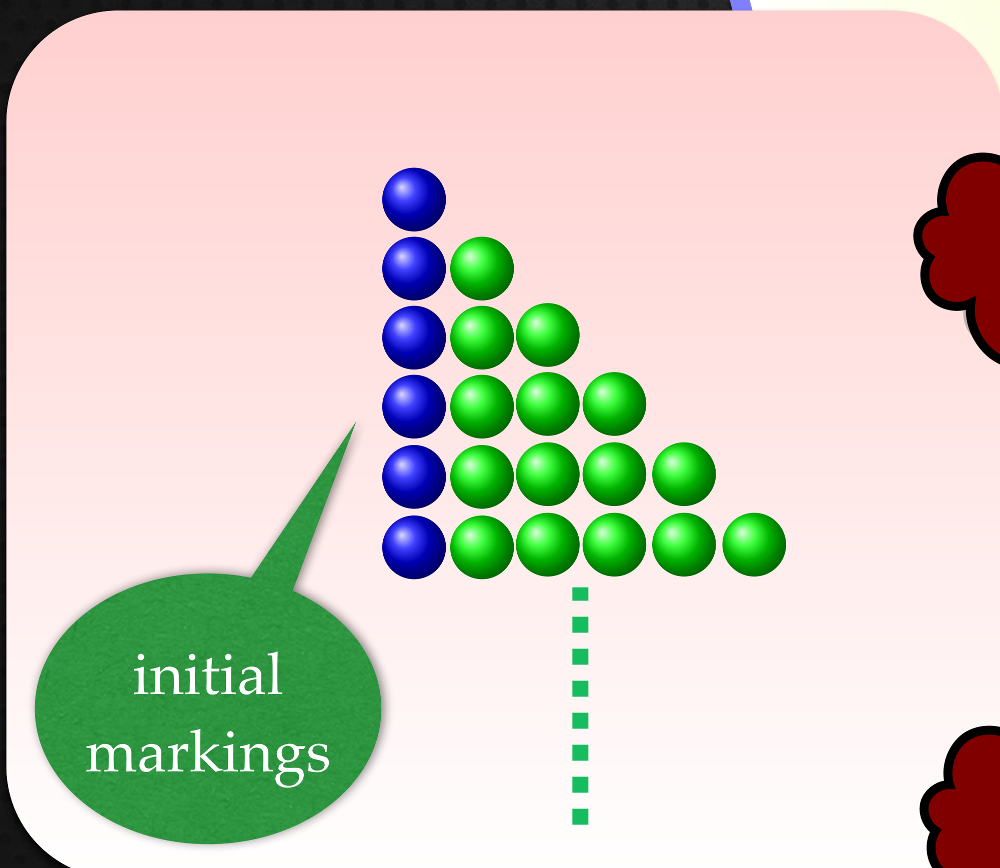
Termination



Assume: non-termination


Backwa
Termination
m_0
m_1
m_2
m_3
Assume:
non-termination
contradiction:
WQO

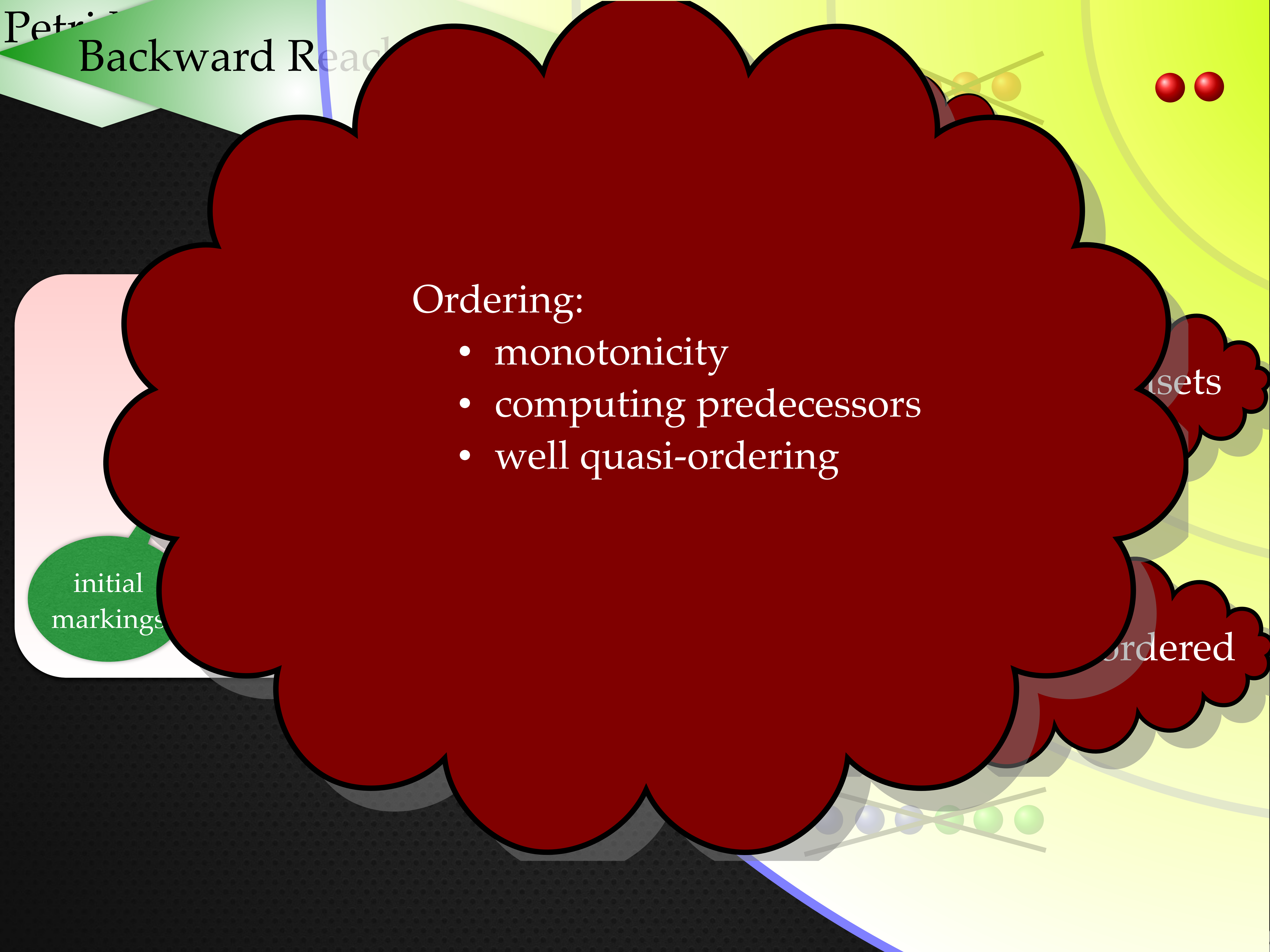
Backward Reachability



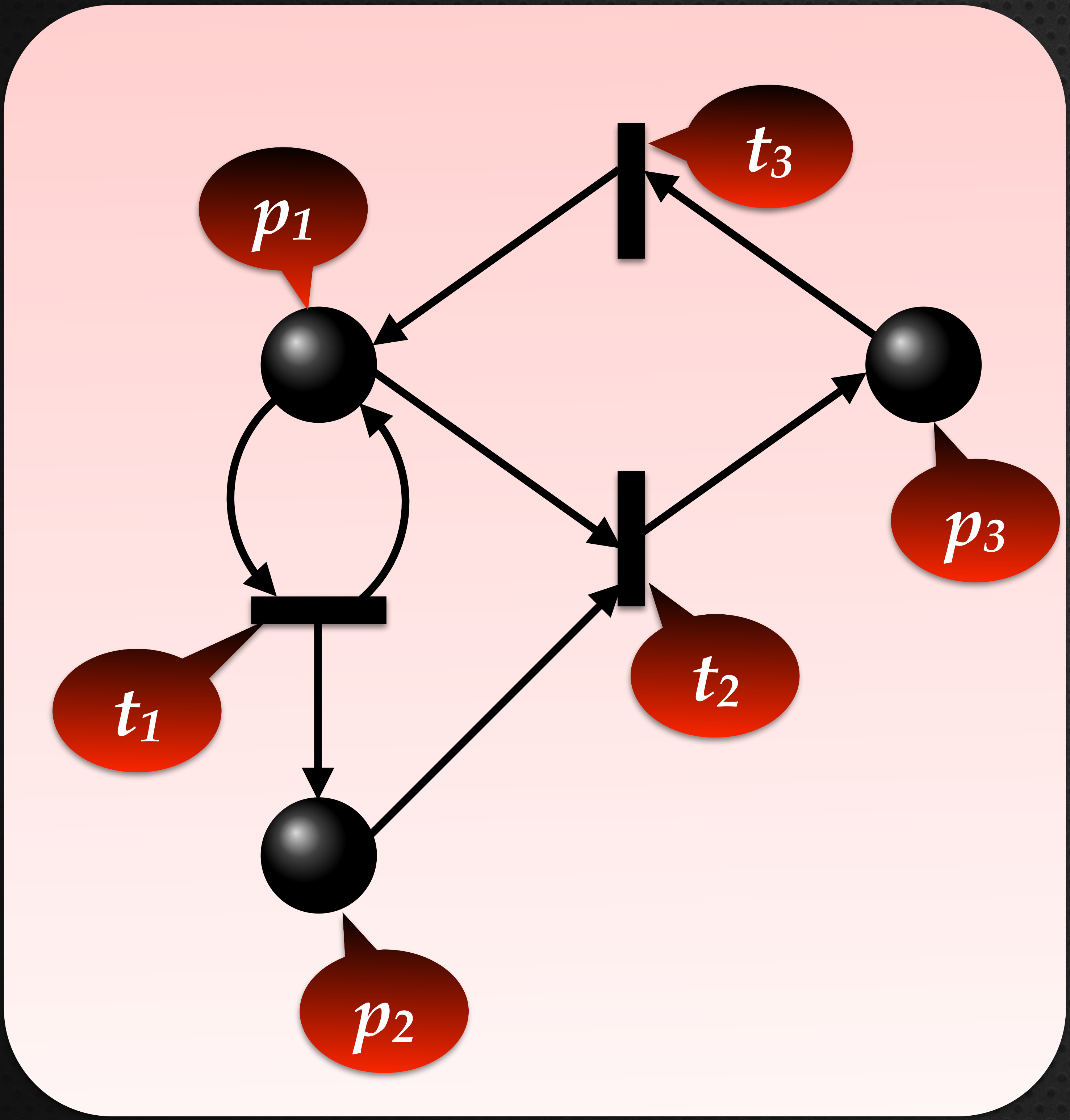
System Safe !

symbolic representation = finite multisets

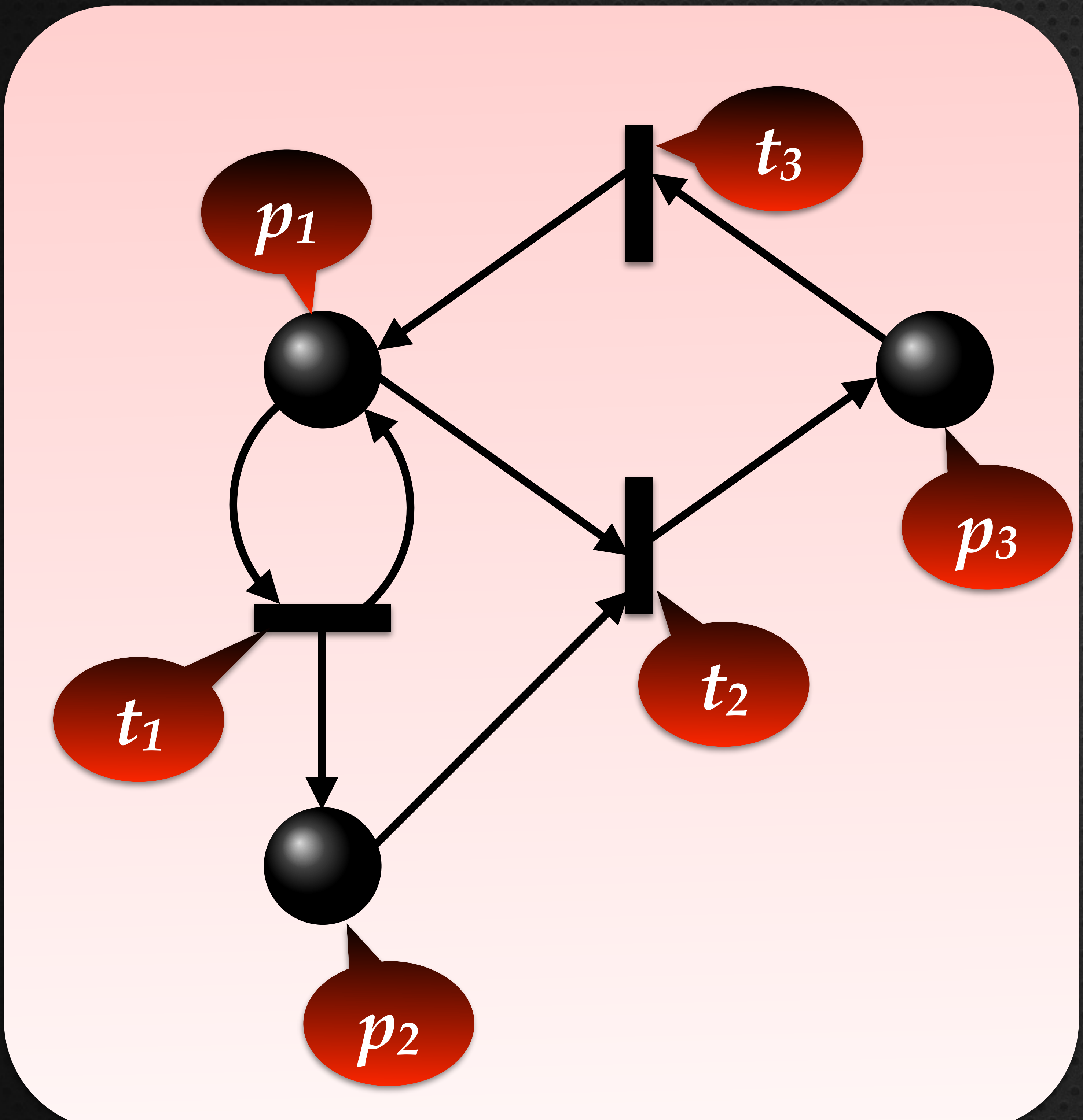
Termnination: multisets well quasi-ordered


Petri
Backward Reach
Ordering:
• monotonicity
• computing predecessors
• well quasi-ordering
initial
markings
sets
ordered

Petri Nets



Petri Nets



- **Perform backward reachability analysis from** *[p3,p3]*
- **Reachable from:**
  - *[p1,p1]?*
  - *[p1,p2]?*