ACI «Sécurité Informatique» CORTOS

- CORTOS = Control and Observation of Real-Time Open Systems
- Participants: LSV + VERIMAG + IRCCyN
- Web: `http://www.lsv.ens-cachan.fr/aci-cortos/`

Thèmes du projet

- Algorithmes de synthèse de contrôleur
- Observation et détection de fautes
- Logiques pour exprimer le contrôle
- Contrôle optimal

Session Invitée

1. Introduction au contrôle des systèmes temps-réel
2. Observation partielle des systèmes temporisés
3. Implémentabilité des automates temporisés

Control of Timed Systems

K. Altisen[1], P. Bouyer[2], T. Cachat[3], F. Cassez[4], G. Gardey[4]

[1]VERIMAG Grenoble
[2]LSV Cachan
[3]LIAFA Paris
[4]IRCCyN Nantes

MSR'05

October 2005, Autrans, France

Outline of the talk

- Verification & Control
- Control of Finite Automata
- Timed Game Automata
- Symbolic Algorithms for Timed Game Automata
- Conclusion

Outline of the talk

- Verification & Control
- Control of Finite Automata
- Timed Game Automata
- Symbolic Algorithms for Timed Game Automata
- Conclusion

Outline of the talk

- Verification & Control
- Control of Finite Automata
- Timed Game Automata
- Symbolic Algorithms for Timed Game Automata
- Conclusion

Outline of the talk

- Verification & Control
- Control of Finite Automata
- Timed Game Automata
- Symbolic Algorithms for Timed Game Automata
- Conclusion

Outline of the talk

- Verification & Control
- Control of Finite Automata
- Timed Game Automata
- Symbolic Algorithms for Timed Game Automata
- Conclusion

Outline

- Verification & Control
- Control of Finite Automata
- Timed Game Automata
- Symbolic Algorithms for Timed Game Automata
- Conclusion

Verification and Control

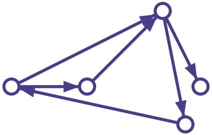
Verification and Control

Does the system meet the specification ?

Verification and Control

Does the system meet the specification ?

Modelling



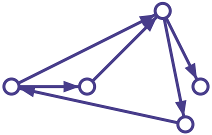
$\square$ (not bad)

$S \models \phi$

Verification and Control

Does the system meet the specification ?

Modelling



$\Box$ (not bad)

$S \models \phi$

Model Checking Problem

Does the closed system S satisfy ϕ ?

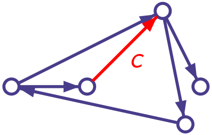
Verification and Control

Can we enforce the system to meet the specification ?

Verification and Control

Can we enforce the system to meet the specification ?

Modelling



$\square$ (not bad)

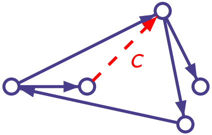
S

ϕ

Verification and Control

Can we enforce the system to meet the specification ?

Modelling



$\Box$ (not bad)

S

ϕ

Control Problem

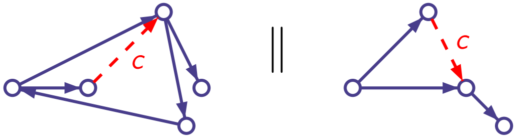
Can the open system S be restricted to satisfy ϕ ?

Is there a Controller C s.t. $(S \parallel C) \models \phi$?

Verification and Control

Can we enforce the system to meet the specification ?

Modelling



$\|$

$\square$ (not bad)

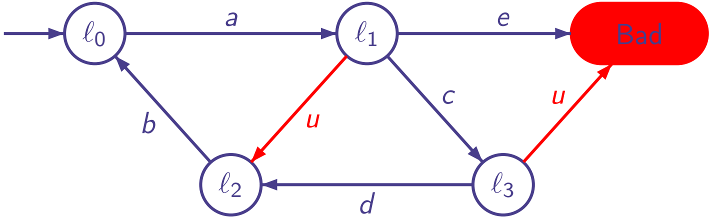
$S \quad \| \quad C \quad \models \quad \phi$

Control Problem

Can the open system S be restricted to satisfy ϕ ?

Is there a Controller C s.t. $(S \parallel C) \models \phi$?

Control of Discrete Event Systems

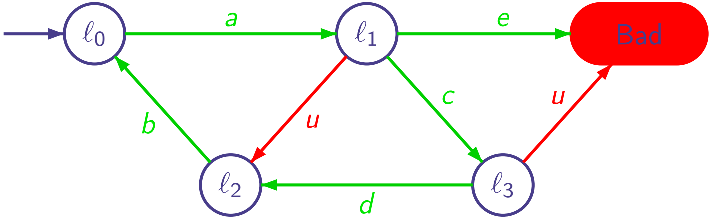


- Introduced by Ramadge & Wonham [Ramadge, 87]
- Discrete Event System = Finite Automaton with

> Controllable (Act_c) and Uncontrollable (Act_u) actions

- Example of Control Objective: "avoid state Bad"
- Means: disable some controllable transitions at the right time
  Ramadge & Wonham Theory is based on Language Theory [Ramadge, 89, Thistle, 94]

Control and Game

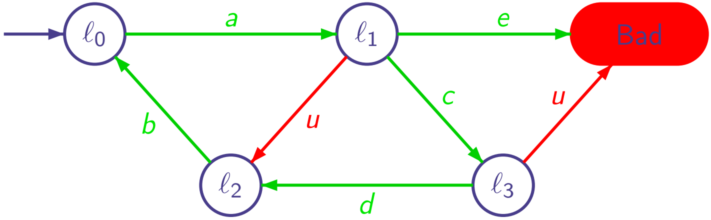


Open System = 2-player game, Controller (C) vs Environment (E)

- Controller does Act_c moves, Environment does Act_u moves
- Control Objective = Winning condition on the game

  "Avoid bad states" (safety) or "Enforce good states" (reachability)
- Control Problem: find a strategy for the controller to win the game
- Various types of game models for C and E
  - Finite or pushdown or counter automata ...
  - Timed or hybrid automata

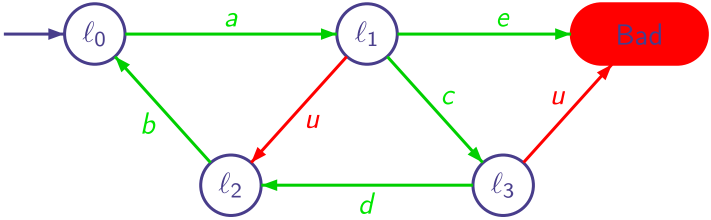
Control and Game



Open System = 2-player game, Controller (C) vs Environment (E)

- Controller does Act_c moves, Environment does Act_u moves
- Control Objective = Winning condition on the game
  "Avoid bad states" (safety) or "Enforce good states" (reachability)
- Control Problem: find a strategy for the controller to win the game
- Various types of game models for C and E
  - Finite or pushdown or counter automata ...
  - Timed or hybrid automata

Control and Game

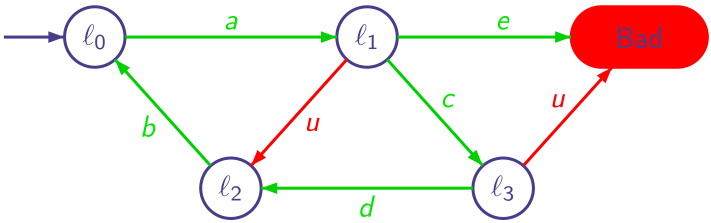


Open System = 2-player game, Controller (C) vs Environment (E)

- Controller does Act_c moves, Environment does Act_u moves
- Control Objective = Winning condition on the game

  "Avoid bad states" (safety) or "Enforce good states" (reachability)
- Control Problem: find a strategy for the controller to win the game
- Various types of game models for C and E
  - Finite or pushdown or counter automata ...
  - Timed or hybrid automata

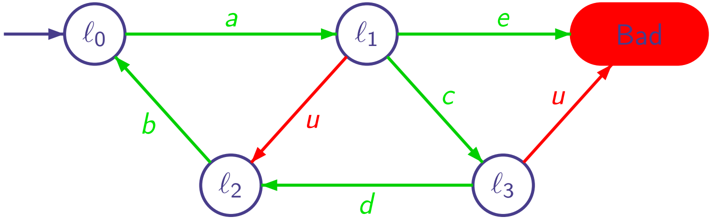
Control and Game



Open System = 2-player game, Controller (C) vs Environment (E)

- Controller does Act_c moves, Environment does Act_u moves
- Control Objective = Winning condition on the game
  "Avoid bad states" (safety) or "Enforce good states" (reachability)
- Control Problem: find a strategy for the controller to win the game
- Various types of game models for C and E
  - Finite or pushdown or counter automata ...
  - Timed or hybrid automata

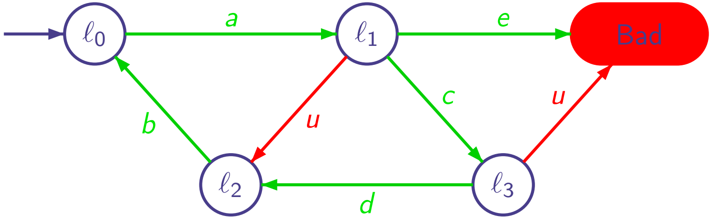
Control and Game



Open System = 2-player game, Controller (C) vs Environment (E)

- Controller does Act_c moves, Environment does Act_u moves
- Control Objective = Winning condition on the game
  "Avoid bad states" (safety) or "Enforce good states" (reachability)
- Control Problem: find a strategy for the controller to win the game
- Various types of game models for C and E
  - Finite or pushdown or counter automata ...
  - Timed or hybrid automata

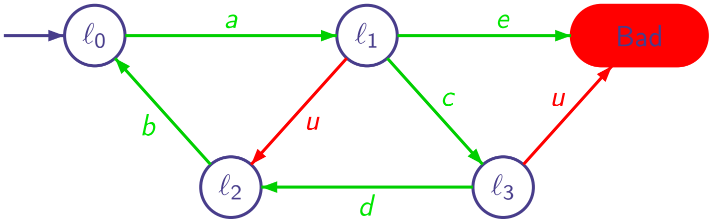
Control and Game



Open System = 2-player game, Controller (C) vs Environment (E)

- Controller does Act_c moves, Environment does Act_u moves
- Control Objective = Winning condition on the game
  "Avoid bad states" (safety) or "Enforce good states" (reachability)
- Control Problem: find a strategy for the controller to win the game
- Various types of game models for C and E
  - Finite or pushdown or counter automata ...
  - Timed or hybrid automata

Control and Game

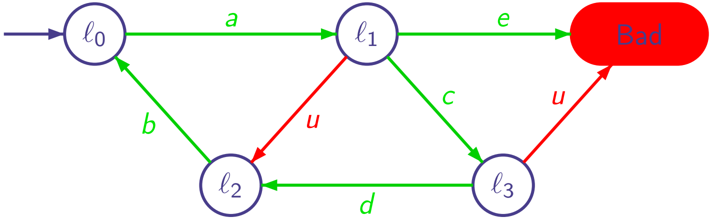


Open System = 2-player game, Controller (C) vs Environment (E)

- Controller does Act_c moves, Environment does Act_u moves
- Control Objective = Winning condition on the game

  "Avoid bad states" (safety) or "Enforce good states" (reachability)
- Control Problem: find a strategy for the controller to win the game
- Various types of game models for C and E
  - Finite or pushdown or counter automata . . .
  - Timed or hybrid automata

Control and Game



Open System = 2-player game, Controller (C) vs Environment (E)

- Controller does Act_c moves, Environment does Act_u moves
- Control Objective = Winning condition on the game
  "Avoid bad states" (safety) or "Enforce good states" (reachability)
- Control Problem: find a strategy for the controller to win the game
- Various types of game models for C and E
  - Finite or pushdown or counter automata . . .
  - Timed or hybrid automata

Problems of Interest

Verification Problem (or Model Checking Problem)

Input: a model of the closed system S and a property φ
Problem: Does S satisfy φ ?

Control Problem (CP)

Input: a model of the open system (game) G and a property φ
Problem: Is there a controller (strategy) C s.t. $(C \parallel G)$ satisfy φ ?

Control Synthesis Problem (CSP)

Input: a model of the open system (game) G and a property φ
Problem: If the answer to the CP(G, φ) is "yes", can we effectively compute a witness controller ?

Problems of Interest

Verification Problem (or Model Checking Problem)

Input: a model of the closed system S and a property φ

Problem: Does S satisfy φ ?

Control Problem (CP)

Input: a model of the open system (game) G and a property φ

Problem: Is there a controller (strategy) C s.t. $(C \parallel G)$ satisfy φ ?

Control Synthesis Problem (CSP)

Input: a model of the open system (game) G and a property φ

Problem: If the answer to the CP(G, φ) is “yes”, can we effectively compute a witness controller ?

Problems of Interest

Verification Problem (or Model Checking Problem)

Input: a model of the closed system S and a property φ
Problem: Does S satisfy φ ?

Control Problem (CP)

Input: a model of the open system (game) G and a property φ
Problem: Is there a controller (strategy) C s.t. $(C \parallel G)$ satisfy φ ?

Control Synthesis Problem (CSP)

Input: a model of the open system (game) G and a property φ
Problem: If the answer to the CP(G, φ) is "yes", can we effectively compute a witness controller ?

Problems of Interest

Verification Problem (or Model Checking Problem)

Input: a model of the closed system S and a property φ
Problem: Does S satisfy φ ?

Control Problem (CP)

Input: a model of the open system (game) G and a property φ
Problem: Is there a controller (strategy) C s.t. $(C \parallel G)$ satisfy φ ?

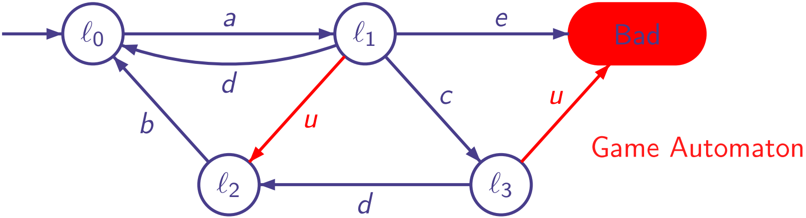
Control Synthesis Problem (CSP)

Input: a model of the open system (game) G and a property φ
Problem: If the answer to the CP(G, φ) is "yes", can we effectively compute a witness controller ?

Outline

- Verification & Control
- Control of Finite Automata
- Timed Game Automata
- Symbolic Algorithms for Timed Game Automata
- Conclusion

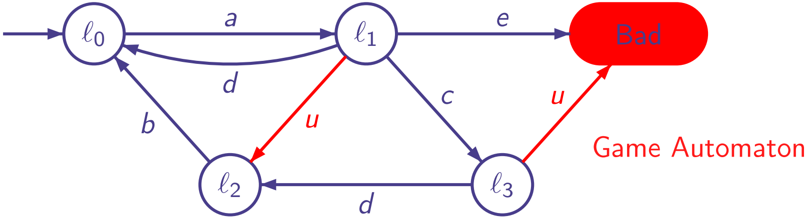
Game Automata, Strategies & Winning States



Strategy

- A strategy f gives for each finite run the controllable action to take. We assume full observability of the system

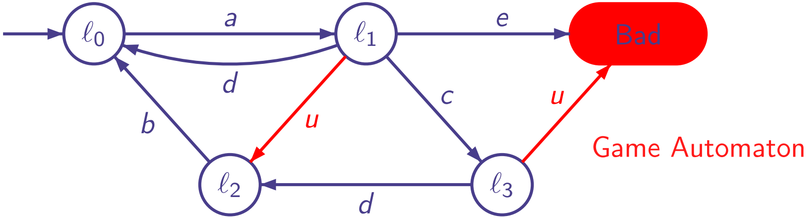
Game Automata, Strategies & Winning States



Strategy

- A strategy f gives for each finite run the controllable action to take. We assume full observability of the system

Game Automata, Strategies & Winning States



Strategy

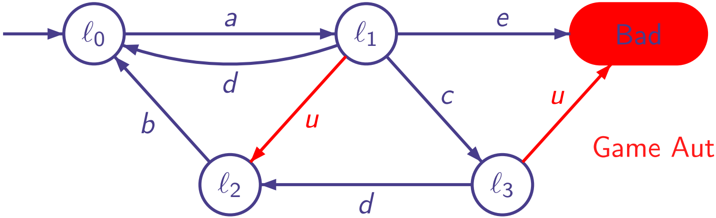
- A strategy f gives for each finite run the controllable action to take. We assume full observability of the system

Example of Strategies:

$$f(\ell_0) = a$$
$$f(\ell_0 \xrightarrow{a} \ell_1) = c$$
$$f(\ell_0 \xrightarrow{a} \ell_1 \xrightarrow{u} \ell_2) = b$$
$$f(\ell_0 \xrightarrow{a} \ell_1 \xrightarrow{u} \ell_2 \xrightarrow{b} \ell_0 \xrightarrow{a} \ell_1) = e$$

$$f'(\cdots \ell_0) = a$$
$$f'(\cdots \ell_1) = c$$
$$f'(\cdots \ell_2) = b$$
$$f'(\cdots \ell_3) = d$$

Game Automata, Strategies & Winning States



Strategy

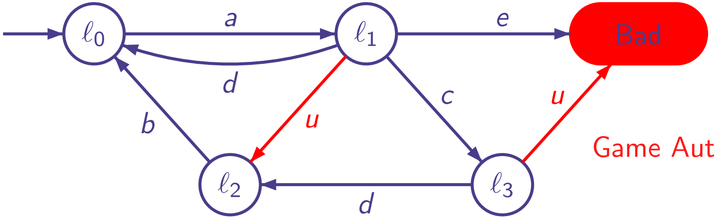
- A strategy f gives for each finite run the controllable action to take. We assume full observability of the system

Example of Strategies:

$$f(\ell_0) = a$$
$$f(\ell_0 \xrightarrow{a} \ell_1) = c$$
$$f(\ell_0 \xrightarrow{a} \ell_1 \xrightarrow{u} \ell_2) = b$$
$$f(\ell_0 \xrightarrow{a} \ell_1 \xrightarrow{u} \ell_2 \xrightarrow{b} \ell_0 \xrightarrow{a} \ell_1) = e$$

$$f'(\cdots \ell_0) = a$$
$$f'(\cdots \ell_1) = c$$
$$f'(\cdots \ell_2) = b$$
$$f'(\cdots \ell_3) = d$$

Game Automata, Strategies & Winning States

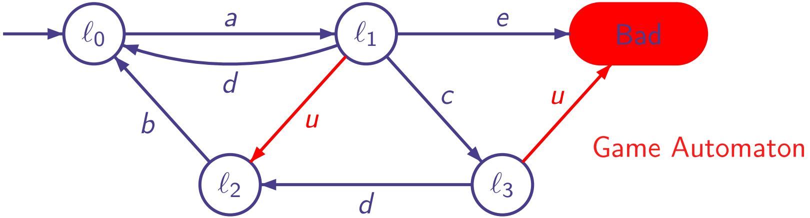


Strategy

- A strategy f gives for each finite run the controllable action to take. We assume full observability of the system
- A strategy restricts the set of runs of the system.

  from a state s it generates of subset of the runs of the initial game

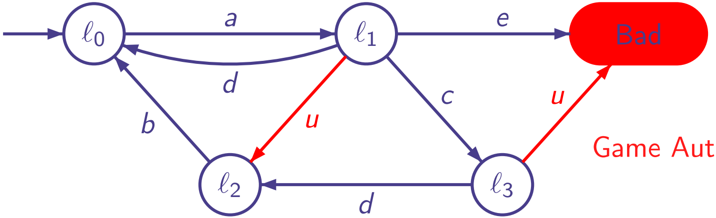
Game Automata, Strategies & Winning States



Strategy

- A strategy f gives for each finite run the controllable action to take. We assume full observability of the system
- A strategy restricts the set of runs of the system.
  from a state s it generates of subset of the runs of the initial game
- A strategy is winning if it generates only good runs.

Game Automata, Strategies & Winning States



Game Automaton

Strategy

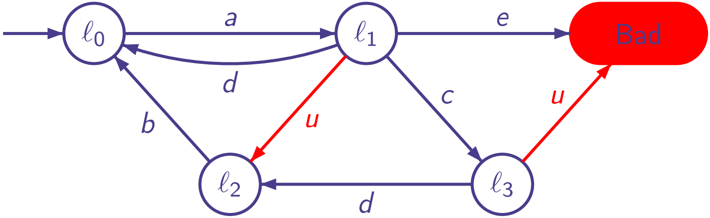
- A strategy f gives for each finite run the controllable action to take. We assume full observability of the system
- A strategy restricts the set of runs of the system.

  from a state s it generates of subset of the runs of the initial game
- A strategy is winning if it generates only good runs.

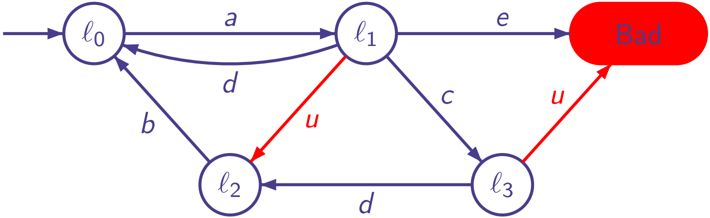
Winning States

A state s is winning if there exists a winning strategy from s.

Controllable Predecessors



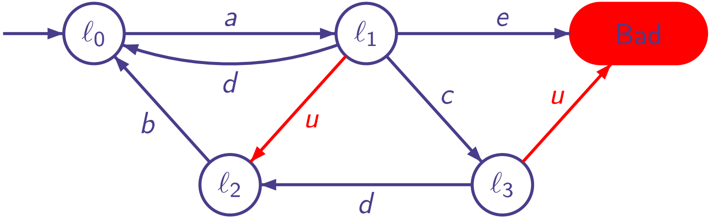
Controllable Predecessors



$\pi(X)$ = states from which one can enforce X with a controllable action

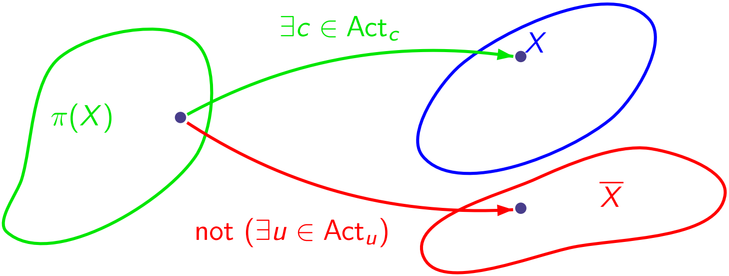
$$\pi(X) = \mathrm{Pred}^{\mathrm{Act}_c}(X) \setminus \mathrm{Pred}^{\mathrm{Act}_u}(\overline{X})$$

Controllable Predecessors

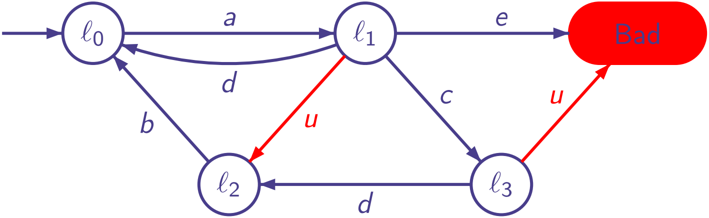


$\pi(X)$ = states from which one can enforce X with a controllable action

$$\pi(X) = \mathrm{Pred}^{\mathrm{Act}_c}(X) \setminus \mathrm{Pred}^{\mathrm{Act}_u}(\overline{X})$$



Controllable Predecessors



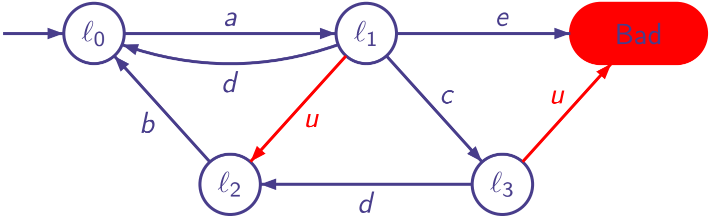
$\pi(X)$ = states from which one can enforce X with a controllable action

$$\pi(X) = \mathrm{Pred}^{\mathrm{Act}_c}(X) \setminus \mathrm{Pred}^{\mathrm{Act}_u}(\overline{X})$$

Some Values of the π Operator

- $\pi(\{\ell_3\}) = \emptyset$
- $\pi(\{\ell_1\}) = \{\ell_0\}$
- $\pi(\{\ell_0, \ell_1\}) = \{\ell_0, \ell_2\}$
- $\pi(\{\ell_0, \ell_1, \ell_2\}) = \{\ell_0, \ell_1, \ell_2\}$

Controllable Predecessors

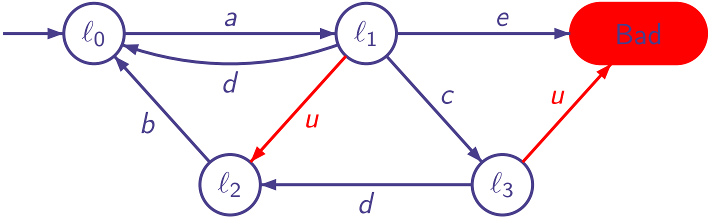


$\pi(X)$ = states from which one can enforce X with a controllable action

A Fixpoint Characterization of Winning States:

1. let φ be a set of safe (good) states and G a game
2. let W^* be the greatest fixpoint of $h(X) = \varphi \cap \pi(X)$
3. W^* is the set of winning states for (G, φ)

Controllable Predecessors

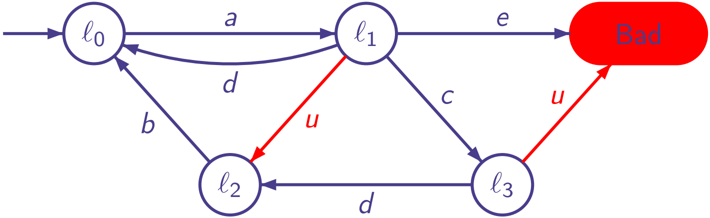


$\pi(X)$ = states from which one can enforce X with a controllable action

A Fixpoint Characterization of Winning States:

1. let φ be a set of safe (good) states and G a game
2. let W^* be the greatest fixpoint of $h(X) = \varphi \cap \pi(X)$
3. W^* is the set of winning states for (G, φ)

Controllable Predecessors

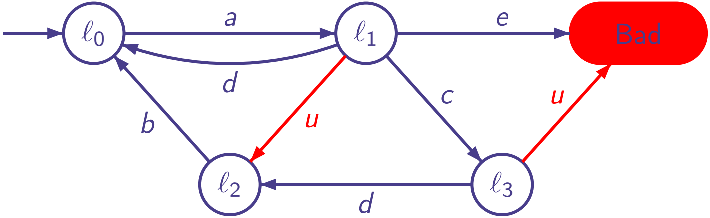


$\pi(X)$ = states from which one can enforce X with a controllable action

A Fixpoint Characterization of Winning States:

1. let φ be a set of safe (good) states and G a game
2. let W^* be the greatest fixpoint of $h(X) = \varphi \cap \pi(X)$
3. W^* is the set of winning states for (G, φ)

Controllable Predecessors

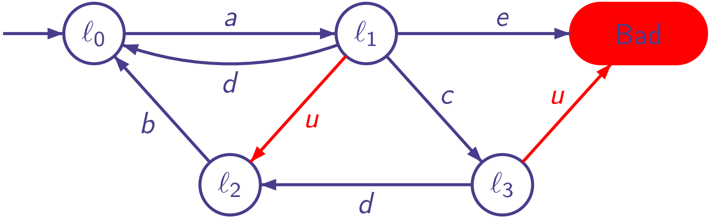


$\pi(X)$ = states from which one can enforce X with a controllable action

A Fixpoint Characterization of Winning States:

1. let φ be a set of safe (good) states and G a game
2. let W^* be the greatest fixpoint of $h(X) = \varphi \cap \pi(X)$
3. W^* is the set of winning states for (G, φ)

Controllable Predecessors



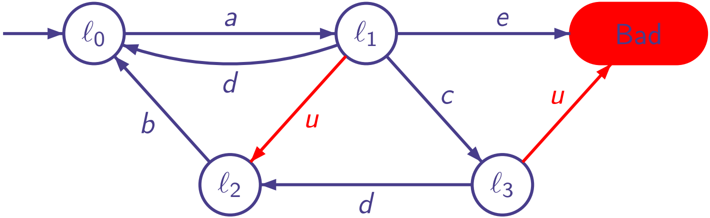
$\pi(X)$ = states from which one can enforce X with a controllable action

A Fixpoint Characterization of Winning States:

1. let φ be a set of safe (good) states and G a game
2. let W^* be the greatest fixpoint of $h(X) = \varphi \cap \pi(X)$
3. W^* is the set of winning states for (G, φ)

- CP: check that $\ell_0 \in W^*$

Controllable Predecessors



$\pi(X)$ = states from which one can enforce X with a controllable action

A Fixpoint Characterization of Winning States:

1. let φ be a set of safe (good) states and G a game
2. let W^* be the greatest fixpoint of $h(X) = \varphi \cap \pi(X)$
3. W^* is the set of winning states for (G, φ)

- CP: check that $\ell_0 \in W^*$
- CSP: Given W^* and G, we can build a winning strategy

Results for Finite Games

Given G a finite game, φ a control objective

Theorem (Positional Strategies are Sufficient)

Positional (or memoryless) strategies suffice to win ω-regular games.
The number of states of C is $\leq$ number of states of G.

Results for Finite Games

Given G a finite game, φ a control objective

The fixpoint computation of W^* terminates

Theorem (Positional Strategies are Sufficient)

Positional (or memoryless) strategies suffice to win ω-regular games.
The number of states of C is $\leq$ number of states of G.

Results for Finite Games

Given G a finite game, φ a control objective

Theorem (CP is Decidable)

CP is decidable for ω-regular objectives.

Theorem (Positional Strategies are Sufficient)

Positional (or memoryless) strategies suffice to win ω-regular games. The number of states of C is $\leq$ number of states of G.

Results for Finite Games

Given G a finite game, φ a control objective

Theorem (CP is Decidable)

CP is decidable for ω-regular objectives.

Theorem (Effectiveness of CSP)

Strategy synthesis is effective. We can build a finite automaton (controller) C that specifies a winning strategy.

Theorem (Positional Strategies are Sufficient)

Positional (or memoryless) strategies suffice to win ω-regular games. The number of states of C is $\leq$ number of states of G.

Results for Finite Games

Given G a finite game, φ a control objective

Theorem (CP is Decidable)

CP is decidable for ω-regular objectives.

Theorem (Effectiveness of CSP)

Strategy synthesis is effective. We can build a finite automaton (controller) C that specifies a winning strategy.

Theorem (Positional Strategies are Sufficient)

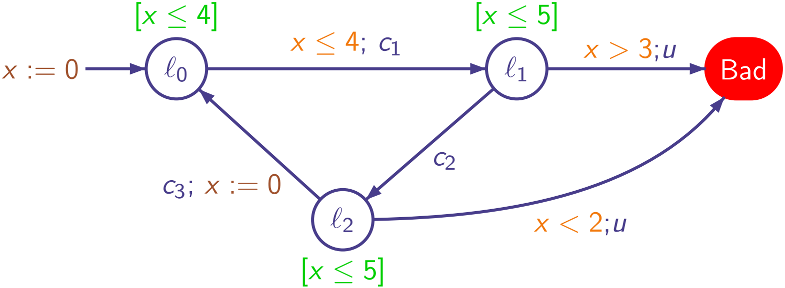
Positional (or memoryless) strategies suffice to win ω-regular games. The number of states of C is $\leq$ number of states of G.

Add Dense Time ... CP and CSP ?

Outline

- Verification & Control
- Control of Finite Automata
- **Timed Game Automata**
- Symbolic Algorithms for Timed Game Automata
- Conclusion

Timed Automata [Alur & Dill'94]

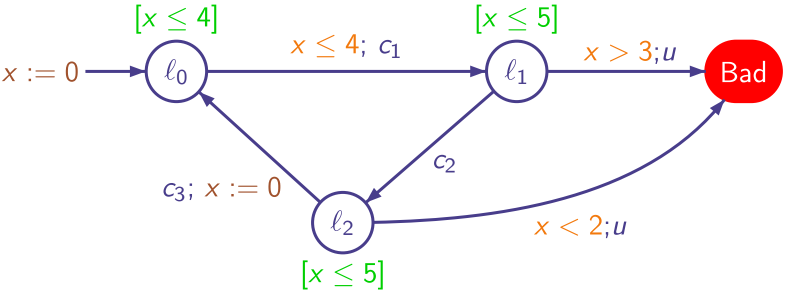


Runs = sequence of discrete and time steps

$$\rho_1 : \quad (\ell_0, 0) \xrightarrow{1.55} (\ell_0, 1.55) \xrightarrow{c_1} (\ell_1, 1.55) \xrightarrow{1.67} (\ell_1, 3.22) \xrightarrow{u} (\mathrm{Bad}, 3.22)$$

$$\rho_2 : \quad (\ell_0, 0) \xrightarrow{1.1} (\ell_0, 1.1) \xrightarrow{c_1} (\ell_1, 1.1) \xrightarrow{2.1} (\ell_1, 3.2) \xrightarrow{c_2} (\ell_2, 3.2)$$
$$\xrightarrow{0.1} (\ell_2, 3.3) \xrightarrow{u} (\ell_0, 0) \quad \cdots\cdots\cdots$$

Timed Automata [Alur & Dill'94]

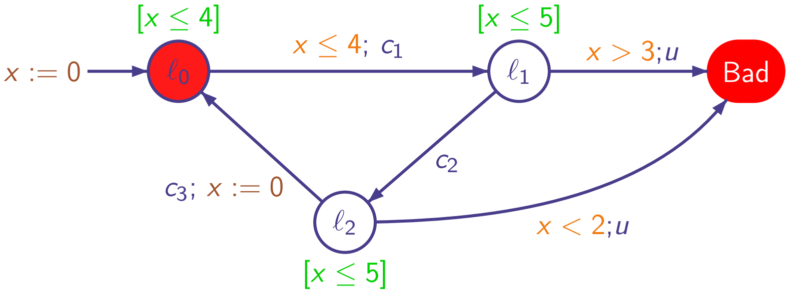


Runs = sequence of discrete and time steps

$$\rho_1 : \quad (\ell_0, 0) \xrightarrow{1.55} (\ell_0, 1.55) \xrightarrow{c_1} (\ell_1, 1.55) \xrightarrow{1.67} (\ell_1, 3.22) \xrightarrow{u} (\mathrm{Bad}, 3.22)$$

$$\rho_2 : \quad (\ell_0, 0) \xrightarrow{1.1} (\ell_0, 1.1) \xrightarrow{c_1} (\ell_1, 1.1) \xrightarrow{2.1} (\ell_1, 3.2) \xrightarrow{c_2} (\ell_2, 3.2) \xrightarrow{0.1} (\ell_2, 3.3) \xrightarrow{u} (\ell_0, 0) \quad \cdots\cdots\cdots$$

Timed Automata [Alur & Dill'94]



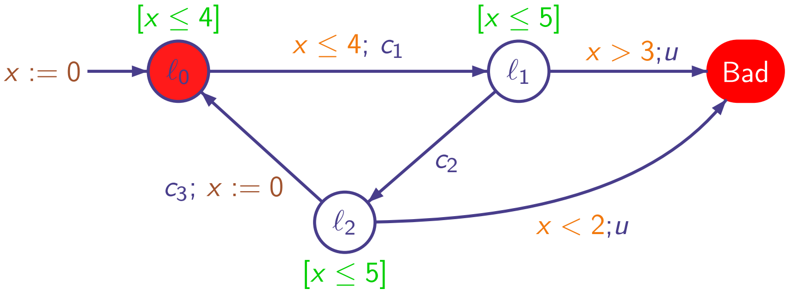
Runs = sequence of discrete and time steps

ρ_1 : $(\ell_0, 0) \xrightarrow{1.55} (\ell_0, 1.55) \xrightarrow{c_1} (\ell_1, 1.55) \xrightarrow{1.67} (\ell_1, 3.22) \xrightarrow{u} (\mathrm{Bad}, 3.22)$

ρ_2 : $(\ell_0, 0) \xrightarrow{1.1} (\ell_0, 1.1) \xrightarrow{c_1} (\ell_1, 1.1) \xrightarrow{2.1} (\ell_1, 3.2) \xrightarrow{c_2} (\ell_2, 3.2)$

$\xrightarrow{0.1} (\ell_2, 3.3) \xrightarrow{u} (\ell_0, 0) \quad \cdots\cdots\cdots$

Timed Automata [Alur & Dill'94]



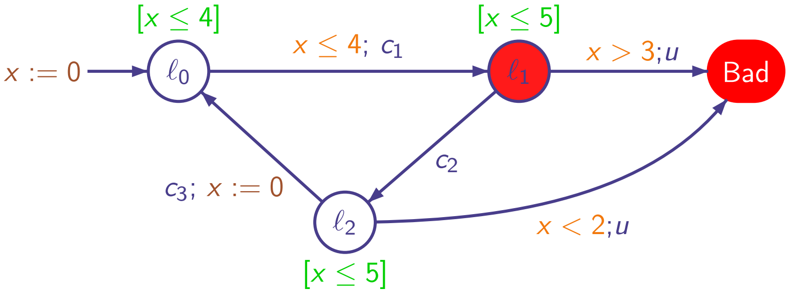
Runs = sequence of discrete and time steps

$\rho_1: \quad (\ell_0, 0) \xrightarrow{1.55} (\ell_0, 1.55) \xrightarrow{c_1} (\ell_1, 1.55) \xrightarrow{1.67} (\ell_1, 3.22) \xrightarrow{u} (\mathrm{Bad}, 3.22)$

$\rho_2: \quad (\ell_0, 0) \xrightarrow{1.1} (\ell_0, 1.1) \xrightarrow{c_1} (\ell_1, 1.1) \xrightarrow{2.1} (\ell_1, 3.2) \xrightarrow{c_2} (\ell_2, 3.2)$

$\xrightarrow{0.1} (\ell_2, 3.3) \xrightarrow{u} (\ell_0, 0) \quad \cdots\cdots\cdots$

Timed Automata [Alur & Dill'94]

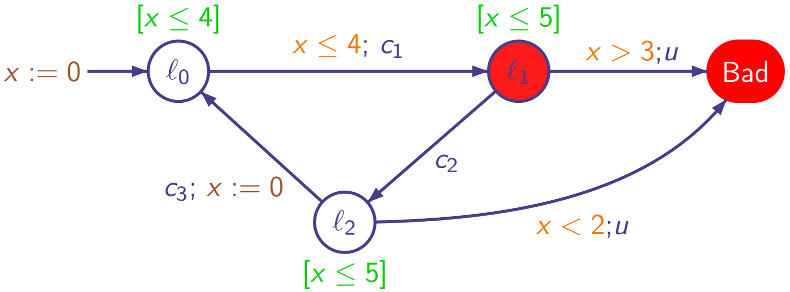


Runs = sequence of discrete and time steps

$\rho_1 : \quad (\ell_0, 0) \xrightarrow{1.55} (\ell_0, 1.55) \xrightarrow{c_1} (\ell_1, 1.55) \xrightarrow{1.67} (\ell_1, 3.22) \xrightarrow{u} (\mathrm{Bad}, 3.22)$

$\rho_2 : \quad (\ell_0, 0) \xrightarrow{1.1} (\ell_0, 1.1) \xrightarrow{c_1} (\ell_1, 1.1) \xrightarrow{2.1} (\ell_1, 3.2) \xrightarrow{c_2} (\ell_2, 3.2)$
$\xrightarrow{0.1} (\ell_2, 3.3) \xrightarrow{u} (\ell_0, 0) \quad \cdots\cdots\cdots$

Timed Automata [Alur & Dill'94]

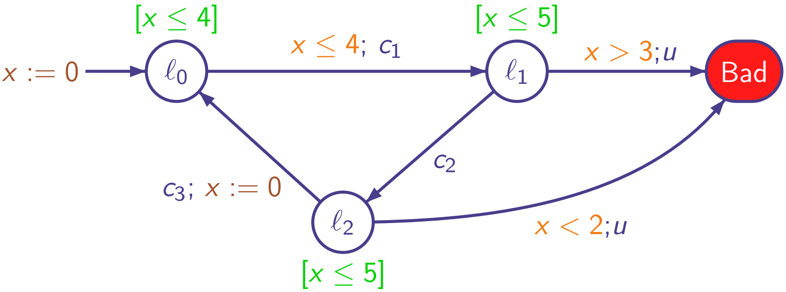


Runs = sequence of discrete and time steps

$\rho_1: \quad (\ell_0, 0) \xrightarrow{1.55} (\ell_0, 1.55) \xrightarrow{c_1} (\ell_1, 1.55) \xrightarrow{1.67} (\ell_1, 3.22) \xrightarrow{u} (\text{Bad}, 3.22)$

$\rho_2: \quad (\ell_0, 0) \xrightarrow{1.1} (\ell_0, 1.1) \xrightarrow{c_1} (\ell_1, 1.1) \xrightarrow{2.1} (\ell_1, 3.2) \xrightarrow{c_2} (\ell_2, 3.2)$
$\xrightarrow{0.1} (\ell_2, 3.3) \xrightarrow{u} (\ell_0, 0) \quad \cdots\cdots\cdots$

Timed Automata [Alur & Dill'94]

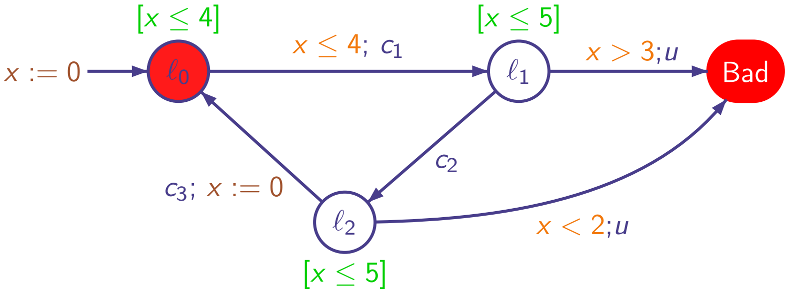


Runs = sequence of discrete and time steps

$\rho_1: \quad (\ell_0, 0) \xrightarrow{1.55} (\ell_0, 1.55) \xrightarrow{c_1} (\ell_1, 1.55) \xrightarrow{1.67} (\ell_1, 3.22) \xrightarrow{u} (\text{Bad}, 3.22)$

$\rho_2: \quad (\ell_0, 0) \xrightarrow{1.1} (\ell_0, 1.1) \xrightarrow{c_1} (\ell_1, 1.1) \xrightarrow{2.1} (\ell_1, 3.2) \xrightarrow{c_2} (\ell_2, 3.2)$
$\xrightarrow{0.1} (\ell_2, 3.3) \xrightarrow{u} (\ell_0, 0) \quad \cdots\cdots\cdots$

Timed Automata [Alur & Dill'94]

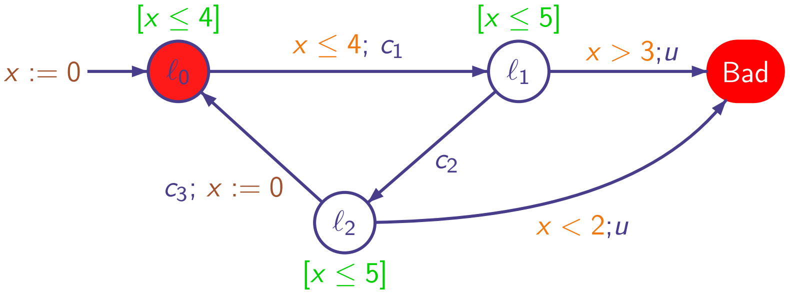


Runs = sequence of discrete and time steps

ρ_1 : $(\ell_0, 0) \xrightarrow{1.55} (\ell_0, 1.55) \xrightarrow{c_1} (\ell_1, 1.55) \xrightarrow{1.67} (\ell_1, 3.22) \xrightarrow{u} (\text{Bad}, 3.22)$

ρ_2 : $(\ell_0, 0) \xrightarrow{1.1} (\ell_0, 1.1) \xrightarrow{c_1} (\ell_1, 1.1) \xrightarrow{2.1} (\ell_1, 3.2) \xrightarrow{c_2} (\ell_2, 3.2)$
$\xrightarrow{0.1} (\ell_2, 3.3) \xrightarrow{u} (\ell_0, 0) \quad \cdots\cdots\cdots$

Timed Automata [Alur & Dill'94]

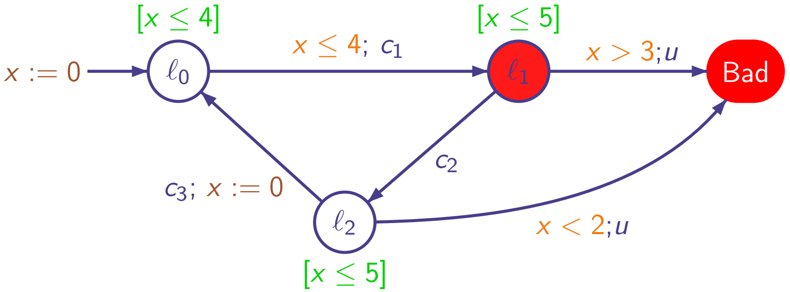


Runs = sequence of discrete and time steps

ρ_1 : $(\ell_0, 0) \xrightarrow{1.55} (\ell_0, 1.55) \xrightarrow{c_1} (\ell_1, 1.55) \xrightarrow{1.67} (\ell_1, 3.22) \xrightarrow{u} (\text{Bad}, 3.22)$

ρ_2 : $(\ell_0, 0) \xrightarrow{1.1} (\ell_0, 1.1) \xrightarrow{c_1} (\ell_1, 1.1) \xrightarrow{2.1} (\ell_1, 3.2) \xrightarrow{c_2} (\ell_2, 3.2)$
$\xrightarrow{0.1} (\ell_2, 3.3) \xrightarrow{u} (\ell_0, 0)$ ·········

Timed Automata [Alur & Dill'94]

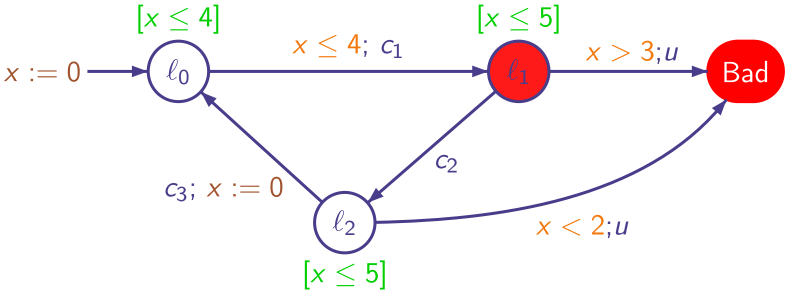


Runs = sequence of discrete and time steps

ρ_1 : $(\ell_0, 0) \xrightarrow{1.55} (\ell_0, 1.55) \xrightarrow{c_1} (\ell_1, 1.55) \xrightarrow{1.67} (\ell_1, 3.22) \xrightarrow{u} (\text{Bad}, 3.22)$

ρ_2 : $(\ell_0, 0) \xrightarrow{1.1} (\ell_0, 1.1) \xrightarrow{c_1} (\ell_1, 1.1) \xrightarrow{2.1} (\ell_1, 3.2) \xrightarrow{c_2} (\ell_2, 3.2)$
$\xrightarrow{0.1} (\ell_2, 3.3) \xrightarrow{u} (\ell_0, 0)$ $\cdots\cdots\cdots$

Timed Automata [Alur & Dill'94]

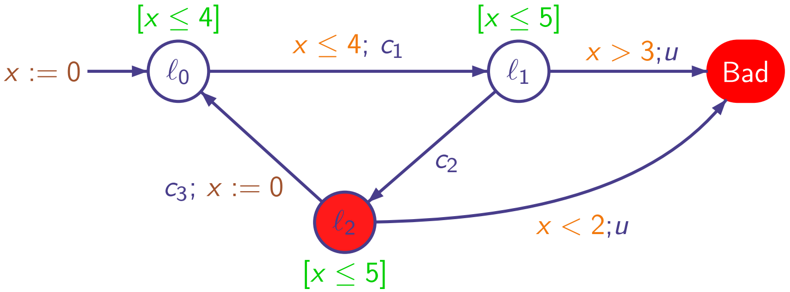


Runs = sequence of discrete and time steps

ρ_1 : $(\ell_0, 0) \xrightarrow{1.55} (\ell_0, 1.55) \xrightarrow{c_1} (\ell_1, 1.55) \xrightarrow{1.67} (\ell_1, 3.22) \xrightarrow{u} (\text{Bad}, 3.22)$

ρ_2 : $(\ell_0, 0) \xrightarrow{1.1} (\ell_0, 1.1) \xrightarrow{c_1} (\ell_1, 1.1) \xrightarrow{2.1} (\ell_1, 3.2) \xrightarrow{c_2} (\ell_2, 3.2)$
$\xrightarrow{0.1} (\ell_2, 3.3) \xrightarrow{u} (\ell_0, 0)$ ··········

Timed Automata [Alur & Dill'94]



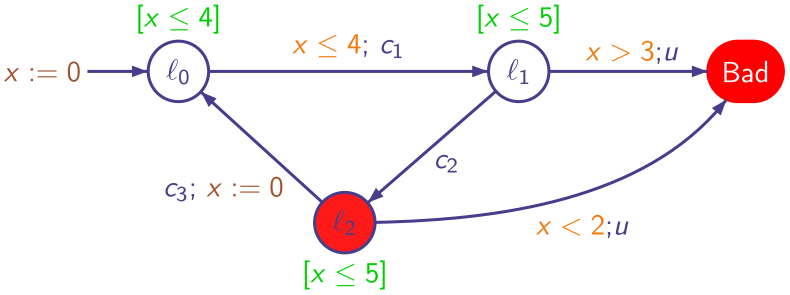
Runs = sequence of discrete and time steps

$$\rho_1 : \quad (\ell_0, 0) \xrightarrow{1.55} (\ell_0, 1.55) \xrightarrow{c_1} (\ell_1, 1.55) \xrightarrow{1.67} (\ell_1, 3.22) \xrightarrow{u} (\text{Bad}, 3.22)$$

$$\rho_2 : \quad (\ell_0, 0) \xrightarrow{1.1} (\ell_0, 1.1) \xrightarrow{c_1} (\ell_1, 1.1) \xrightarrow{2.1} (\ell_1, 3.2) \xrightarrow{c_2} (\ell_2, 3.2)$$

$$\xrightarrow{0.1} (\ell_2, 3.3) \xrightarrow{u} (\ell_0, 0) \quad \cdots\cdots\cdots$$

Timed Automata [Alur & Dill'94]

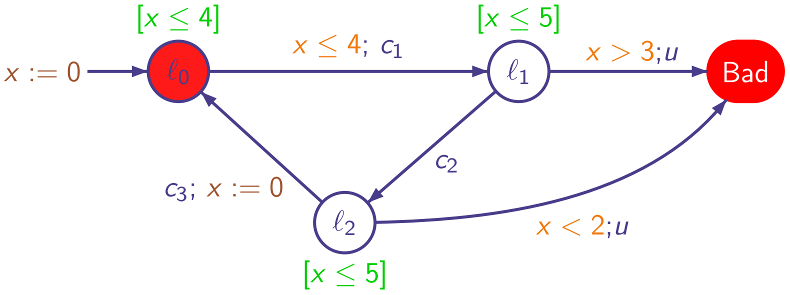


Runs = sequence of discrete and time steps

ρ_1 : $(\ell_0, 0) \xrightarrow{1.55} (\ell_0, 1.55) \xrightarrow{c_1} (\ell_1, 1.55) \xrightarrow{1.67} (\ell_1, 3.22) \xrightarrow{u} (\text{Bad}, 3.22)$

ρ_2 : $(\ell_0, 0) \xrightarrow{1.1} (\ell_0, 1.1) \xrightarrow{c_1} (\ell_1, 1.1) \xrightarrow{2.1} (\ell_1, 3.2) \xrightarrow{c_2} (\ell_2, 3.2)$
$\xrightarrow{0.1} (\ell_2, 3.3) \xrightarrow{u} (\ell_0, 0)$ ....

Timed Automata [Alur & Dill'94]

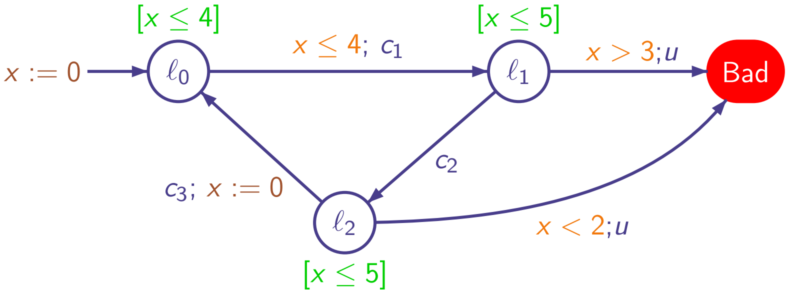


Runs = sequence of discrete and time steps

$$\rho_1: \quad (\ell_0, 0) \xrightarrow{1.55} (\ell_0, 1.55) \xrightarrow{c_1} (\ell_1, 1.55) \xrightarrow{1.67} (\ell_1, 3.22) \xrightarrow{u} (\text{Bad}, 3.22)$$

$$\rho_2: \quad (\ell_0, 0) \xrightarrow{1.1} (\ell_0, 1.1) \xrightarrow{c_1} (\ell_1, 1.1) \xrightarrow{2.1} (\ell_1, 3.2) \xrightarrow{c_2} (\ell_2, 3.2)$$
$$\xrightarrow{0.1} (\ell_2, 3.3) \xrightarrow{u} (\ell_0, 0) \quad \cdots\cdots\cdots$$

Timed Automata [Alur & Dill'94]

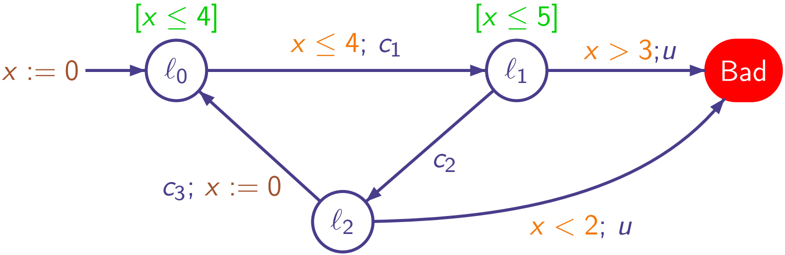


Runs = sequence of discrete and time steps

$\rho_1: \quad (\ell_0, 0) \xrightarrow{1.55} (\ell_0, 1.55) \xrightarrow{c_1} (\ell_1, 1.55) \xrightarrow{1.67} (\ell_1, 3.22) \xrightarrow{u} (\text{Bad}, 3.22)$

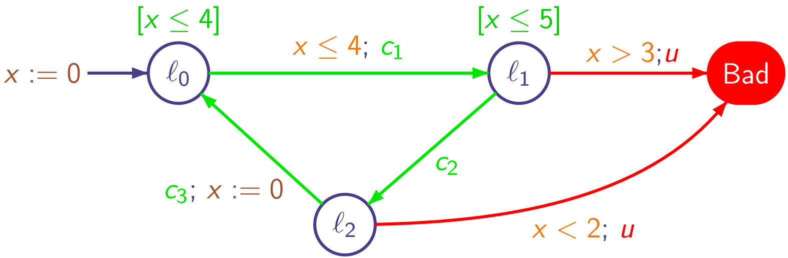
$\rho_2: \quad (\ell_0, 0) \xrightarrow{1.1} (\ell_0, 1.1) \xrightarrow{c_1} (\ell_1, 1.1) \xrightarrow{2.1} (\ell_1, 3.2) \xrightarrow{c_2} (\ell_2, 3.2)$
$\xrightarrow{0.1} (\ell_2, 3.3) \xrightarrow{u} (\ell_0, 0) \quad \cdots\cdots\cdots$

Timed Game Automata



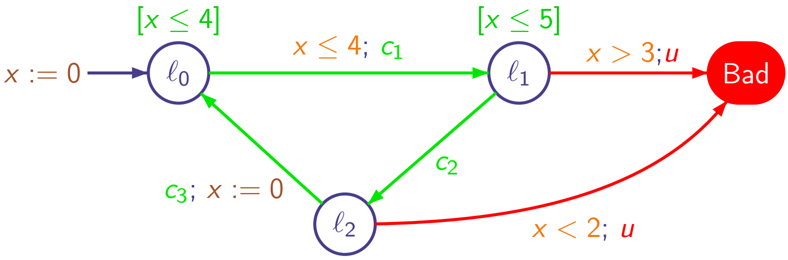
- Introduced by Maler, Pnueli, Sifakis [Maler, 95]
- The controller continuously observes the system
  time elapsing and discrete moves are observable
- It has the choice between two types of moves:
  - "do nothing"
  - "do a controllable action" (among the ones that are possible)
- It can stop time from elapsing by taking a controllable move

Timed Game Automata



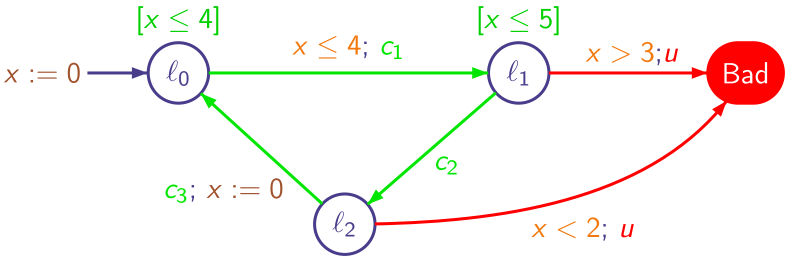
- Introduced by Maler, Pnueli, Sifakis [Maler, 95]
- The controller continuously observes the system time elapsing and discrete moves are observable
- It has the choice between two types of moves:
  - "do nothing"
  - "do a controllable action" (among the ones that are possible)
- It can stop time from elapsing by taking a controllable move

Timed Game Automata



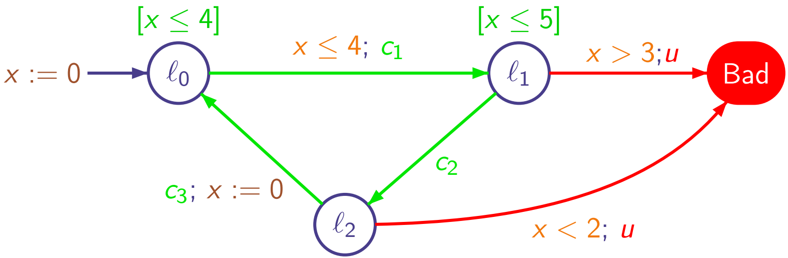
- Introduced by Maler, Pnueli, Sifakis [Maler, 95]
- The controller continuously observes the system
  time elapsing and discrete moves are observable
- It has the choice between two types of moves:
  - "do nothing"
  - "do a controllable action" (among the ones that are possible)
- It can stop time from elapsing by taking a controllable move

Timed Game Automata



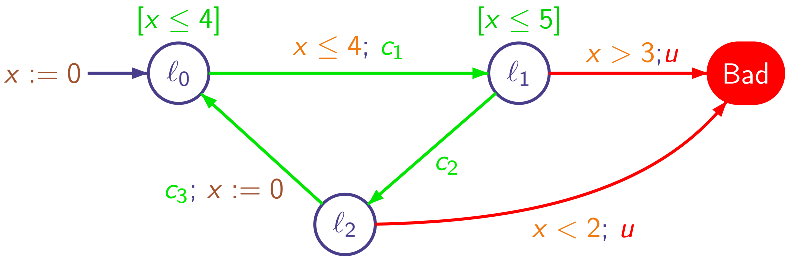
- Introduced by Maler, Pnueli, Sifakis [Maler, 95]
- The controller continuously observes the system
  time elapsing and discrete moves are observable
- It has the choice between two types of moves:
  - "do nothing"
  - "do a controllable action" (among the ones that are possible)
- It can stop time from elapsing by taking a controllable move

Timed Game Automata



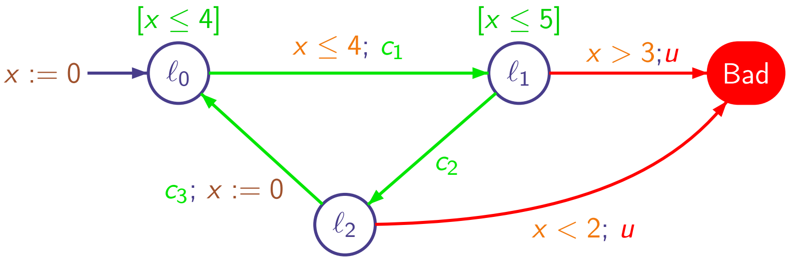
- Introduced by Maler, Pnueli, Sifakis [Maler, 95]
- The controller continuously observes the system
  time elapsing and discrete moves are observable
- It has the choice between two types of moves:
  - "do nothing"
  - "do a controllable action" (among the ones that are possible)
- It can stop time from elapsing by taking a controllable move

Timed Game Automata



- Introduced by Maler, Pnueli, Sifakis [Maler, 95]
- The controller continuously observes the system
  time elapsing and discrete moves are observable
- It has the choice between two types of moves:
  - "do nothing"
  - "do a controllable action" (among the ones that are possible)
- It can stop time from elapsing by taking a controllable move

Timed Game Automata



- Introduced by Maler, Pnueli, Sifakis [Maler, 95]
- The controller continuously observes the system
  time elapsing and discrete moves are observable
- It has the choice between two types of moves:
  - "do nothing"
  - "do a controllable action" (among the ones that are possible)
- It can stop time from elapsing by taking a controllable move

How to Deal with Dense-Time ?

- Infinite state systems

Symbolic representation of states

- A strategy (or controller) can choose to wait

Add a special wait action

- Dense time $\cdots$ the controller can be unfair
  - block time
  - do infinitely many actions in a bounded time
  - do arbitrarily closed (in time) discrete actions

Implementation Issues

How to Deal with Dense-Time ?

- Infinite state systems

> Symbolic representation of states

- A strategy (or controller) can choose to wait

> Add a special wait action

- Dense time · · · the controller can be unfair
  - block time
  - do infinitely many actions in a bounded time
  - do arbitrarily closed (in time) discrete actions

> Implementation Issues

How to Deal with Dense-Time ?

- Infinite state systems

> Symbolic representation of states

- A strategy (or controller) can choose to wait

> Add a special wait action

- Dense time · · · the controller can be unfair
  - block time
  - do infinitely many actions in a bounded time
  - do arbitrarily closed (in time) discrete actions

> Implementation Issues

How to Deal with Dense-Time ?

- Infinite state systems

  Symbolic representation of states

- A strategy (or controller) can choose to wait

  Add a special wait action

- Dense time $\cdots$ the controller can be unfair
  - block time
  - do infinitely many actions in a bounded time
  - do arbitrarily closed (in time) discrete actions

  Implementation Issues

How to Deal with Dense-Time ?

- Infinite state systems

  Symbolic representation of states

- A strategy (or controller) can choose to wait

  Add a special wait action

- Dense time $\cdots$ the controller can be unfair
  - block time
  - do infinitely many actions in a bounded time
  - do arbitrarily closed (in time) discrete actions

  Implementation Issues

How to Deal with Dense-Time ?

- Infinite state systems

Symbolic representation of states

- A strategy (or controller) can choose to wait

Add a special wait action

- Dense time $\cdots$ the controller can be unfair
  - block time
  - do infinitely many actions in a bounded time
  - do arbitrarily closed (in time) discrete actions

Implementation Issues

How to Deal with Dense-Time ?

- Infinite state systems

  Symbolic representation of states

- A strategy (or controller) can choose to wait

  Add a special wait action

- Dense time $\cdots$ the controller can be unfair
  - block time
  - do infinitely many actions in a bounded time
  - do arbitrarily closed (in time) discrete actions

  Implementation Issues

How to Deal with Dense-Time ?

- Infinite state systems

Symbolic representation of states

- A strategy (or controller) can choose to wait

Add a special wait action

- Dense time ··· the controller can be unfair
  - block time
  - do infinitely many actions in a bounded time
  - do arbitrarily closed (in time) discrete actions

Implementation Issues

How to Deal with Dense-Time ?

- Infinite state systems

> Symbolic representation of states

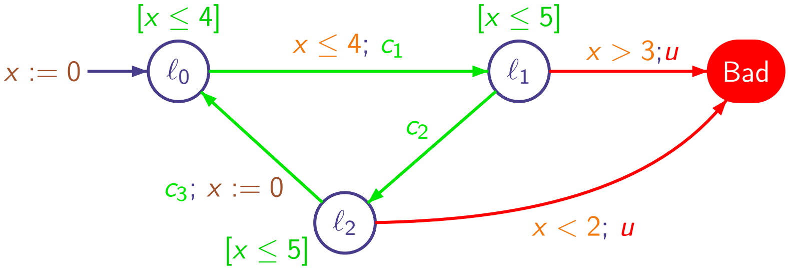
- A strategy (or controller) can choose to wait

> Add a special wait action

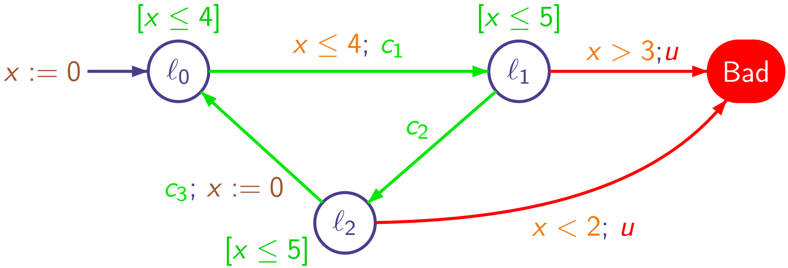
- Dense time $\cdots$ the controller can be unfair
  - block time
  - do infinitely many actions in a bounded time
  - do arbitrarily closed (in time) discrete actions

> Implementation Issues

Strategies and Winning States



Strategies and Winning States

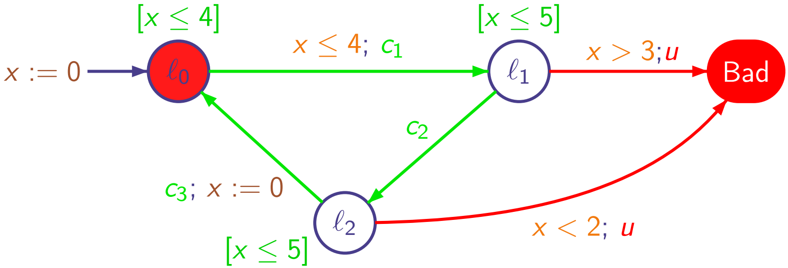


The strategy f: "Always wait as long as the system permits"

ρ_1 : $(\ell_0, 0) \xrightarrow{4} (\ell_0, 4) \xrightarrow{c_1} (\ell_1, 4) \xrightarrow{0.5} (\ell_1, 4.5) \xrightarrow{u} (\text{Bad}, 4.5)$

ρ_2 : $(\ell_0, 0) \xrightarrow{4} (\ell_0, 4) \xrightarrow{c_1} (\ell_1, 4) \xrightarrow{1.0} (\ell_1, 5) \xrightarrow{c_2} (\ell_2, 5) \xrightarrow{c_3} (\ell_0, 0) \cdots$

Strategies and Winning States

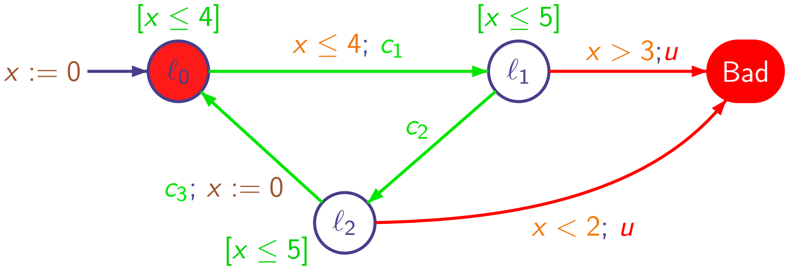


The strategy f: "Always wait as long as the system permits"

ρ_1 : $(\ell_0, 0) \xrightarrow{4} (\ell_0, 4) \xrightarrow{c_1} (\ell_1, 4) \xrightarrow{0.5} (\ell_1, 4.5) \xrightarrow{u} (\mathrm{Bad}, 4.5)$

ρ_2 : $(\ell_0, 0) \xrightarrow{4} (\ell_0, 4) \xrightarrow{c_1} (\ell_1, 4) \xrightarrow{1.0} (\ell_1, 5) \xrightarrow{c_2} (\ell_2, 5) \xrightarrow{c_3} (\ell_0, 0) \cdots$

Strategies and Winning States

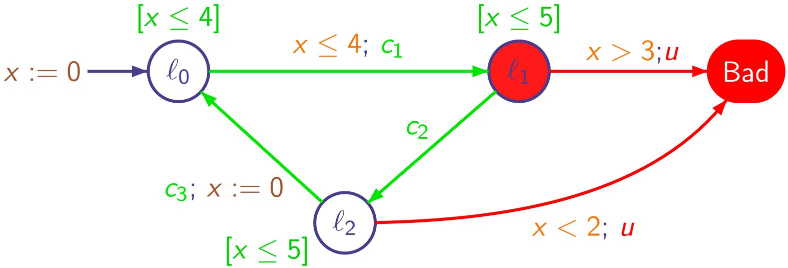


The strategy f: "Always wait as long as the system permits"

ρ_1 : $(\ell_0, 0) \xrightarrow{4} (\ell_0, 4) \xrightarrow{c_1} (\ell_1, 4) \xrightarrow{0.5} (\ell_1, 4.5) \xrightarrow{u} (\text{Bad}, 4.5)$

ρ_2 : $(\ell_0, 0) \xrightarrow{4} (\ell_0, 4) \xrightarrow{c_1} (\ell_1, 4) \xrightarrow{1.0} (\ell_1, 5) \xrightarrow{c_2} (\ell_2, 5) \xrightarrow{c_3} (\ell_0, 0) \cdots$

Strategies and Winning States

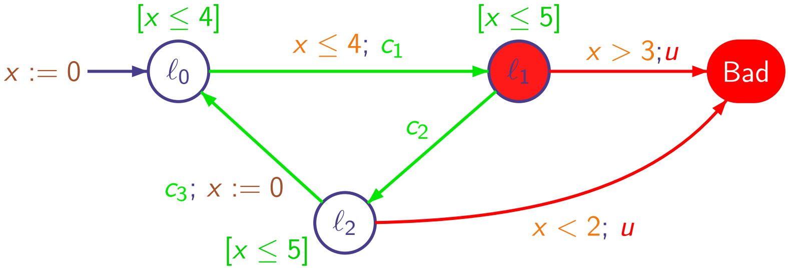


The strategy f: "Always wait as long as the system permits"

$\rho_1 :\quad (\ell_0, 0) \xrightarrow{4} (\ell_0, 4) \xrightarrow{c_1} (\ell_1, 4) \xrightarrow{0.5} (\ell_1, 4.5) \xrightarrow{u} (\mathrm{Bad}, 4.5)$

$\rho_2 :\quad (\ell_0, 0) \xrightarrow{4} (\ell_0, 4) \xrightarrow{c_1} (\ell_1, 4) \xrightarrow{1.0} (\ell_1, 5) \xrightarrow{c_2} (\ell_2, 5) \xrightarrow{c_3} (\ell_0, 0) \cdots$

Strategies and Winning States

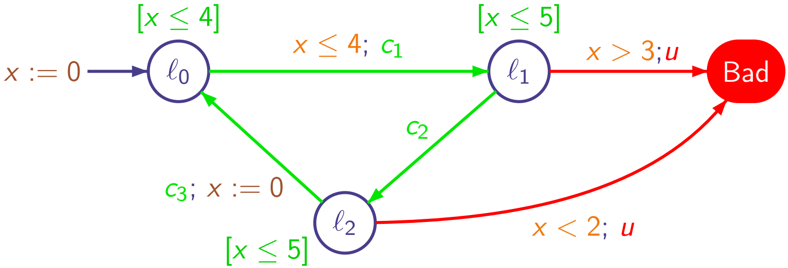


The strategy f: "Always wait as long as the system permits"

ρ_1 : $(\ell_0, 0) \xrightarrow{4} (\ell_0, 4) \xrightarrow{c_1} (\ell_1, 4) \xrightarrow{0.5} (\ell_1, 4.5) \xrightarrow{u} (\text{Bad}, 4.5)$

ρ_2 : $(\ell_0, 0) \xrightarrow{4} (\ell_0, 4) \xrightarrow{c_1} (\ell_1, 4) \xrightarrow{1.0} (\ell_1, 5) \xrightarrow{c_2} (\ell_2, 5) \xrightarrow{c_3} (\ell_0, 0) \cdots$

Strategies and Winning States

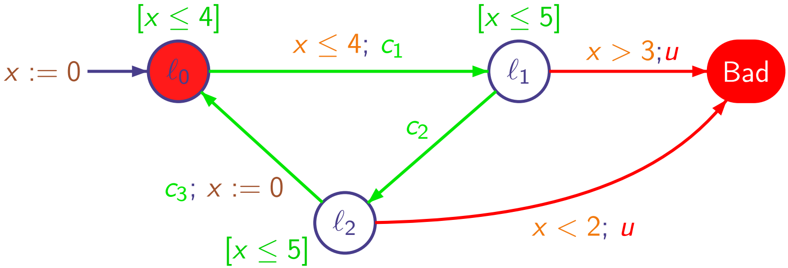


The strategy f: "Always wait as long as the system permits"

$\rho_1: \quad (\ell_0, 0) \xrightarrow{4} (\ell_0, 4) \xrightarrow{c_1} (\ell_1, 4) \xrightarrow{0.5} (\ell_1, 4.5) \xrightarrow{u} (\text{Bad}, 4.5)$

$\rho_2: \quad (\ell_0, 0) \xrightarrow{4} (\ell_0, 4) \xrightarrow{c_1} (\ell_1, 4) \xrightarrow{1.0} (\ell_1, 5) \xrightarrow{c_2} (\ell_2, 5) \xrightarrow{c_3} (\ell_0, 0) \cdots$

Strategies and Winning States

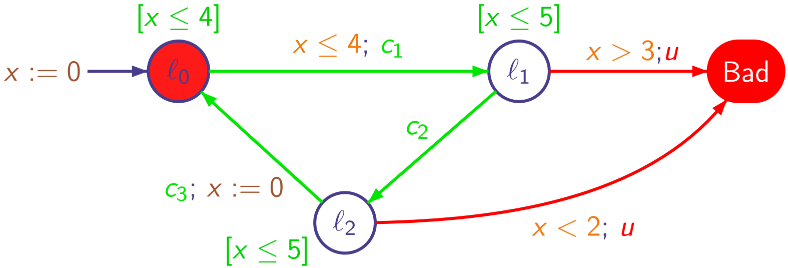


The strategy f: "Always wait as long as the system permits"

ρ_1 : $(\ell_0, 0) \xrightarrow{4} (\ell_0, 4) \xrightarrow{c_1} (\ell_1, 4) \xrightarrow{0.5} (\ell_1, 4.5) \xrightarrow{u} (\text{Bad}, 4.5)$

ρ_2 : $(\ell_0, 0) \xrightarrow{4} (\ell_0, 4) \xrightarrow{c_1} (\ell_1, 4) \xrightarrow{1.0} (\ell_1, 5) \xrightarrow{c_2} (\ell_2, 5) \xrightarrow{c_3} (\ell_0, 0) \cdots$

Strategies and Winning States

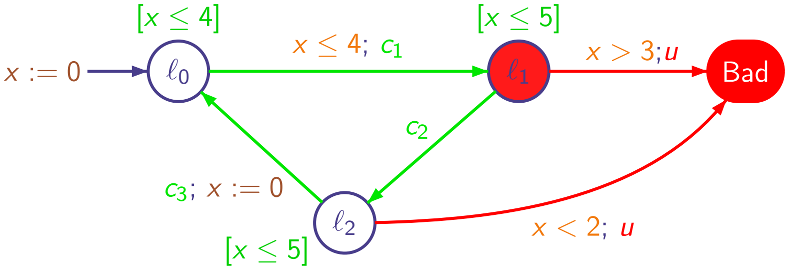


The strategy f: "Always wait as long as the system permits"

ρ_1 : $(\ell_0, 0) \xrightarrow{4} (\ell_0, 4) \xrightarrow{c_1} (\ell_1, 4) \xrightarrow{0.5} (\ell_1, 4.5) \xrightarrow{u} (\text{Bad}, 4.5)$

ρ_2 : $(\ell_0, 0) \xrightarrow{4} (\ell_0, 4) \xrightarrow{c_1} (\ell_1, 4) \xrightarrow{1.0} (\ell_1, 5) \xrightarrow{c_2} (\ell_2, 5) \xrightarrow{c_3} (\ell_0, 0) \cdots$

Strategies and Winning States

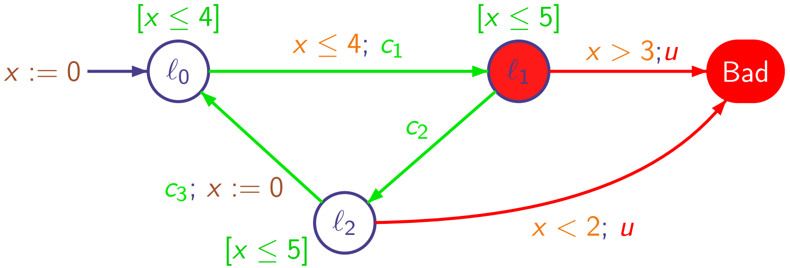


The strategy f: "Always wait as long as the system permits"

$\rho_1 :\quad (\ell_0, 0) \xrightarrow{4} (\ell_0, 4) \xrightarrow{c_1} (\ell_1, 4) \xrightarrow{0.5} (\ell_1, 4.5) \xrightarrow{u} (\mathrm{Bad}, 4.5)$

$\rho_2 :\quad (\ell_0, 0) \xrightarrow{4} (\ell_0, 4) \xrightarrow{c_1} (\ell_1, 4) \xrightarrow{1.0} (\ell_1, 5) \xrightarrow{c_2} (\ell_2, 5) \xrightarrow{c_3} (\ell_0, 0) \cdots$

Strategies and Winning States

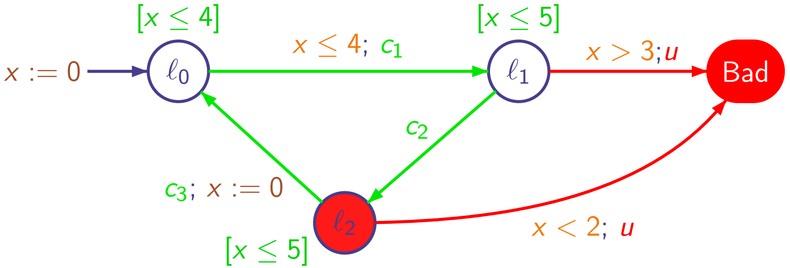


The strategy f: "Always wait as long as the system permits"

ρ_1 : $(\ell_0, 0) \xrightarrow{4} (\ell_0, 4) \xrightarrow{c_1} (\ell_1, 4) \xrightarrow{0.5} (\ell_1, 4.5) \xrightarrow{u} (\text{Bad}, 4.5)$

ρ_2 : $(\ell_0, 0) \xrightarrow{4} (\ell_0, 4) \xrightarrow{c_1} (\ell_1, 4) \xrightarrow{1.0} (\ell_1, 5) \xrightarrow{c_2} (\ell_2, 5) \xrightarrow{c_3} (\ell_0, 0) \cdots$

Strategies and Winning States

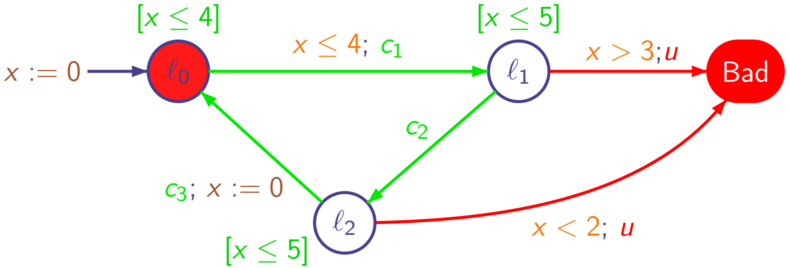


The strategy f: "Always wait as long as the system permits"

ρ_1 : $(\ell_0, 0) \xrightarrow{4} (\ell_0, 4) \xrightarrow{c_1} (\ell_1, 4) \xrightarrow{0.5} (\ell_1, 4.5) \xrightarrow{u} (\mathrm{Bad}, 4.5)$

ρ_2 : $(\ell_0, 0) \xrightarrow{4} (\ell_0, 4) \xrightarrow{c_1} (\ell_1, 4) \xrightarrow{1.0} (\ell_1, 5) \xrightarrow{c_2} (\ell_2, 5) \xrightarrow{c_3} (\ell_0, 0) \cdots$

Strategies and Winning States

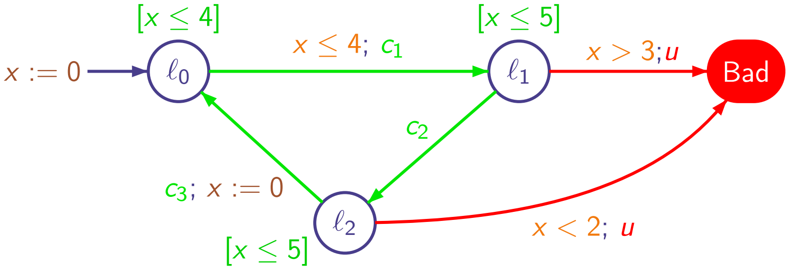


The strategy f: "Always wait as long as the system permits"

ρ_1 : $(\ell_0, 0) \xrightarrow{4} (\ell_0, 4) \xrightarrow{c_1} (\ell_1, 4) \xrightarrow{0.5} (\ell_1, 4.5) \xrightarrow{u} (\mathrm{Bad}, 4.5)$

ρ_2 : $(\ell_0, 0) \xrightarrow{4} (\ell_0, 4) \xrightarrow{c_1} (\ell_1, 4) \xrightarrow{1.0} (\ell_1, 5) \xrightarrow{c_2} (\ell_2, 5) \xrightarrow{c_3} (\ell_0, 0) \cdots$

Strategies and Winning States

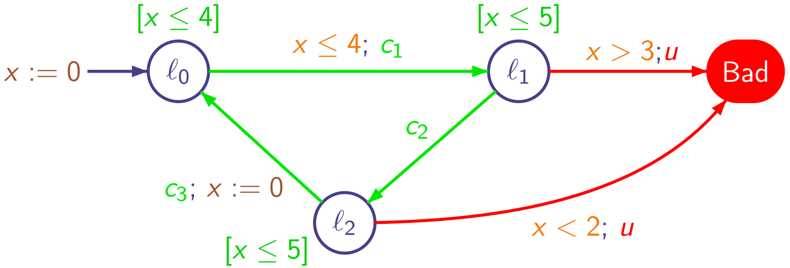


The strategy f: "Always wait as long as the system permits"

ρ_1 : $(\ell_0, 0) \xrightarrow{4} (\ell_0, 4) \xrightarrow{c_1} (\ell_1, 4) \xrightarrow{0.5} (\ell_1, 4.5) \xrightarrow{u} (\text{Bad}, 4.5)$

ρ_2 : $(\ell_0, 0) \xrightarrow{4} (\ell_0, 4) \xrightarrow{c_1} (\ell_1, 4) \xrightarrow{1.0} (\ell_1, 5) \xrightarrow{c_2} (\ell_2, 5) \xrightarrow{c_3} (\ell_0, 0) \cdots$

Strategies and Winning States

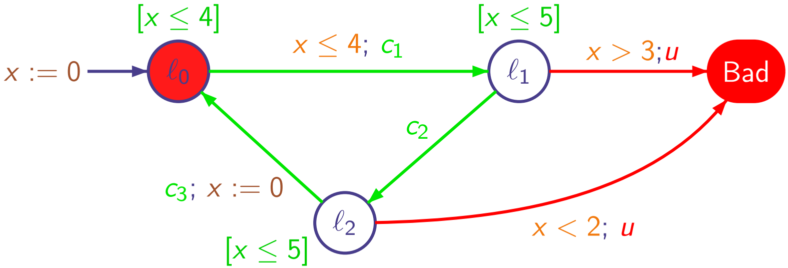


A winning strategy f'

in ℓ_0 at $x = 2$ do c_1; in ℓ_1 at $x = 2.5$ do c_2; in ℓ_2 at $x = 4$ do c_3

$\rho: \quad (\ell_0, 0) \xrightarrow{2} (\ell_0, 2) \xrightarrow{c_1} (\ell_1, 2)$

Strategies and Winning States

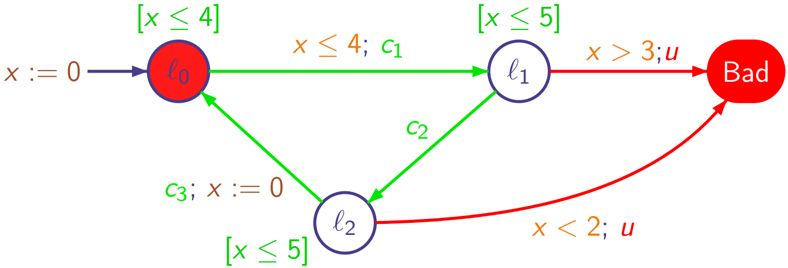


A winning strategy f'

in ℓ_0 at $x = 2$ do c_1; in ℓ_1 at $x = 2.5$ do c_2; in ℓ_2 at $x = 4$ do c_3

$\rho: \quad (\ell_0, 0) \xrightarrow{2} (\ell_0, 2) \xrightarrow{c_1} (\ell_1, 2)$

Strategies and Winning States

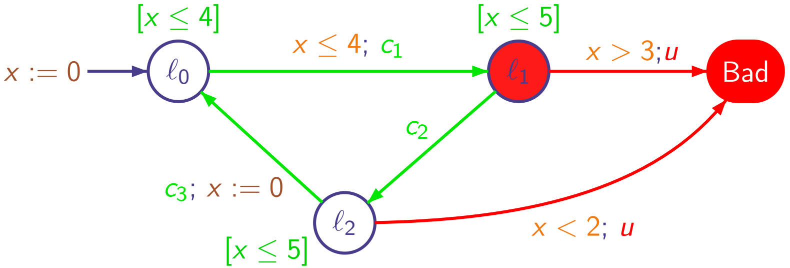


A winning strategy f'

in ℓ_0 at $x = 2$ do c_1; in ℓ_1 at $x = 2.5$ do c_2; in ℓ_2 at $x = 4$ do c_3

$\rho : \quad (\ell_0, 0) \xrightarrow{2} (\ell_0, 2) \xrightarrow{c_1} (\ell_1, 2)$

Strategies and Winning States

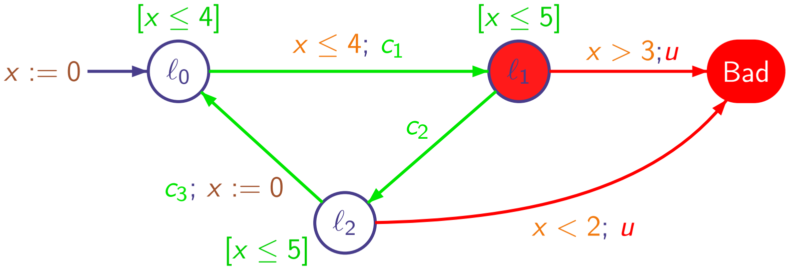


A winning strategy f'

in ℓ_0 at $x = 2$ do c_1; in ℓ_1 at $x = 2.5$ do c_2; in ℓ_2 at $x = 4$ do c_3

$\rho : \quad (\ell_0, 0) \xrightarrow{2} (\ell_0, 2) \xrightarrow{c_1} (\ell_1, 2)$

Strategies and Winning States

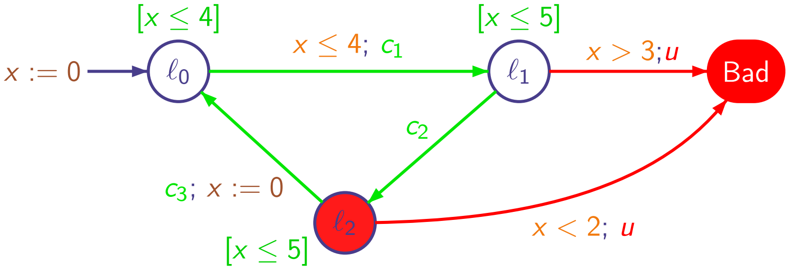


A winning strategy f'

in ℓ_0 at $x = 2$ do c_1; in ℓ_1 at $x = 2.5$ do c_2; in ℓ_2 at $x = 4$ do c_3

$\rho: \quad (\ell_0, 0) \xrightarrow{2} (\ell_0, 2) \xrightarrow{c_1} (\ell_1, 2) \xrightarrow{0.5} (\ell_1, 2.5)$

Strategies and Winning States

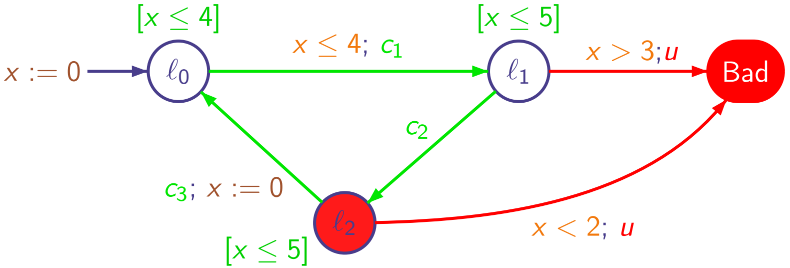


A winning strategy f'

in ℓ_0 at $x = 2$ do c_1; in ℓ_1 at $x = 2.5$ do c_2; in ℓ_2 at $x = 4$ do c_3

$\rho: \quad (\ell_0, 0) \xrightarrow{2} (\ell_0, 2) \xrightarrow{c_1} (\ell_1, 2) \xrightarrow{0.5} (\ell_1, 2.5) \xrightarrow{c_2} (\ell_2, 2.5)$

Strategies and Winning States

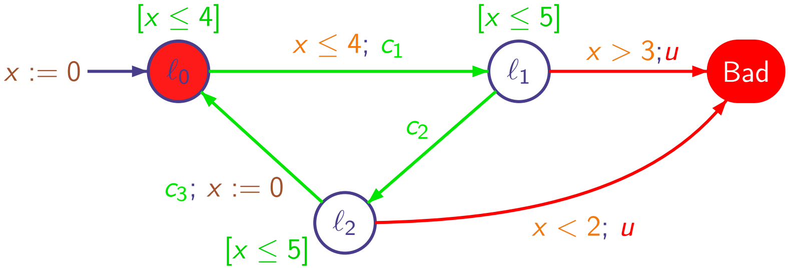


A winning strategy f'

in ℓ_0 at $x = 2$ do c_1; in ℓ_1 at $x = 2.5$ do c_2; in ℓ_2 at $x = 4$ do c_3

$\rho : \quad (\ell_0, 0) \xrightarrow{2} (\ell_0, 2) \xrightarrow{c_1} (\ell_1, 2) \xrightarrow{0.5} (\ell_1, 2.5) \xrightarrow{c_2} (\ell_2, 2.5) \xrightarrow{1.5} (\ell_2, 4)$

Strategies and Winning States

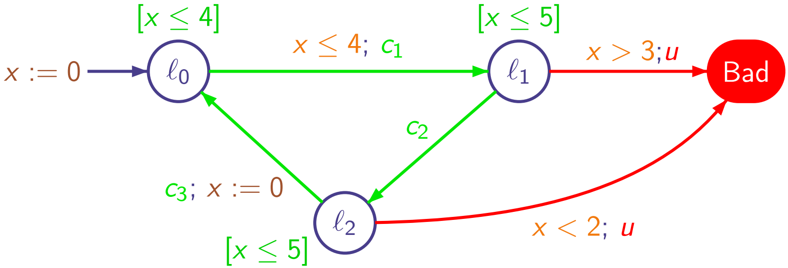


A winning strategy f'

in ℓ_0 at $x = 2$ do c_1; in ℓ_1 at $x = 2.5$ do c_2; in ℓ_2 at $x = 4$ do c_3

$$\rho: \quad (\ell_0, 0) \xrightarrow{2} (\ell_0, 2) \xrightarrow{c_1} (\ell_1, 2) \xrightarrow{0.5} (\ell_1, 2.5) \xrightarrow{c_2} (\ell_2, 2.5) \xrightarrow{1.5} (\ell_2, 4) \xrightarrow{c_3} (\ell_0, 0)$$

Strategies and Winning States

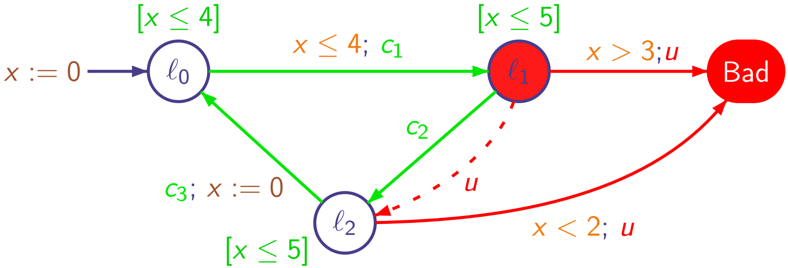


A winning strategy f'

in ℓ_0 at $x = 2$ do c_1; in ℓ_1 at $x = 2.5$ do c_2; in ℓ_2 at $x = 4$ do c_3

$$\rho : \quad (\ell_0, 0) \xrightarrow{2} (\ell_0, 2) \xrightarrow{c_1} (\ell_1, 2) \xrightarrow{0.5} (\ell_1, 2.5) \xrightarrow{c_2} (\ell_2, 2.5) \xrightarrow{1.5} (\ell_2, 4) \xrightarrow{c_3} (\ell_0, 0) \cdots$$

Strategies and Winning States

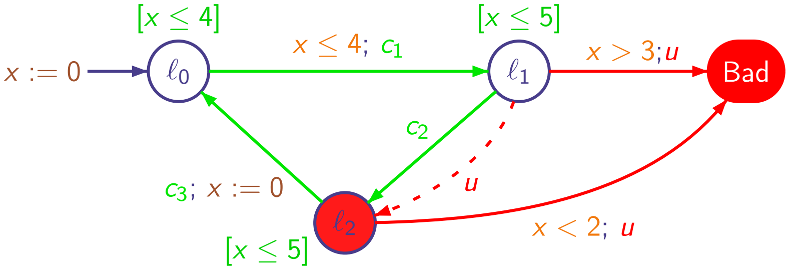


A winning strategy f'

in ℓ_0 at $x = 2$ do c_1; in ℓ_1 at $x = 2.5$ do c_2; in ℓ_2 at $x = 4$ do c_3

$\rho : \quad (\ell_0, 0) \xrightarrow{2} (\ell_0, 2) \xrightarrow{c_1} (\ell_1, 2)$

Strategies and Winning States

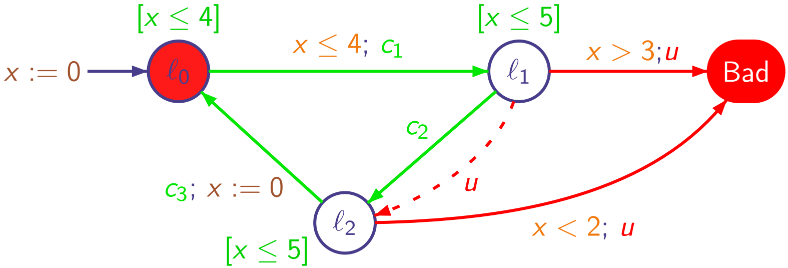


A winning strategy f'

in ℓ_0 at $x = 2$ do c_1; in ℓ_1 at $x = 2.5$ do c_2; in ℓ_2 at $x = 4$ do c_3

$\rho: \quad (\ell_0, 0) \xrightarrow{2} (\ell_0, 2) \xrightarrow{c_1} (\ell_1, 2) \xrightarrow{u \text{ at } \delta \leq 0.5} (\ell_2, 2 + \delta)$

Strategies and Winning States

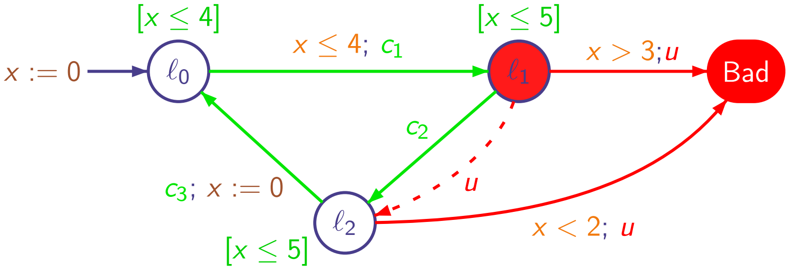


A winning strategy f'

in ℓ_0 at $x = 2$ do c_1; in ℓ_1 at $x = 2.5$ do c_2; in ℓ_2 at $x = 4$ do c_3

$$\rho : \quad (\ell_0, 0) \xrightarrow{2} (\ell_0, 2) \xrightarrow{c_1} (\ell_1, 2) \xrightarrow{u \text{ at } \delta \leq 0.5} (\ell_2, 2 + \delta) \xrightarrow{c_3 \text{ at } 2-\delta} (\ell_0, 0) \cdots$$

Strategies and Winning States



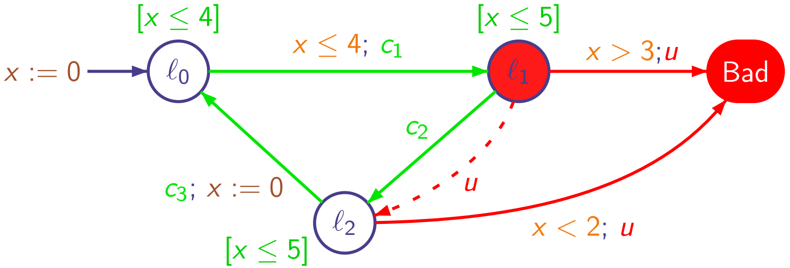
A winning strategy f'

in ℓ_0 at $x = 2$ do c_1; in ℓ_1 at $x = 2.5$ do c_2; in ℓ_2 at $x = 4$ do c_3

$\rho: \quad (\ell_0, 0) \xrightarrow{2} (\ell_0, 2) \xrightarrow{c_1} (\ell_1, 2) \xrightarrow{u \text{ at } \delta \leq 0.5} (\ell_2, 2+\delta) \xrightarrow{c_3 \text{ at } 2-\delta} (\ell_0, 0) \cdots$

$\rho': \quad (\ell_0, 0) \xrightarrow{2} (\ell_0, 2) \xrightarrow{c_1} (\ell_1, 2)$

Strategies and Winning States



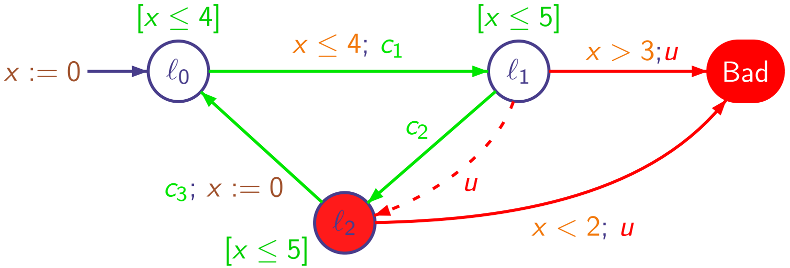
A winning strategy f'

in ℓ_0 at $x = 2$ do c_1; in ℓ_1 at $x = 2.5$ do c_2; in ℓ_2 at $x = 4$ do c_3

$\rho: \quad (\ell_0, 0) \xrightarrow{2} (\ell_0, 2) \xrightarrow{c_1} (\ell_1, 2) \xrightarrow{u \text{ at } \delta \leq 0.5} (\ell_2, 2+\delta) \xrightarrow{c_3 \text{ at } 2-\delta} (\ell_0, 0) \cdots$

$\rho': \quad (\ell_0, 0) \xrightarrow{2} (\ell_0, 2) \xrightarrow{c_1} (\ell_1, 2) \xrightarrow{0.5} (\ell_1, 2.5)$

Strategies and Winning States



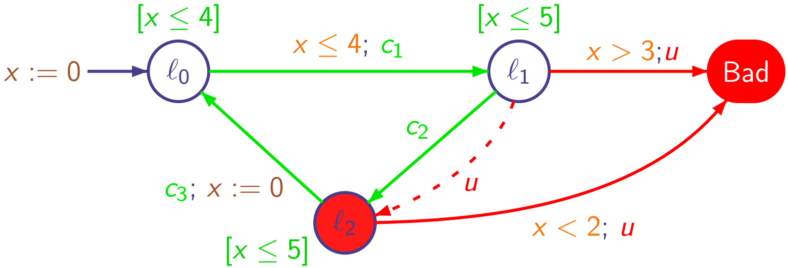
A winning strategy f'

in ℓ_0 at $x = 2$ do c_1; in ℓ_1 at $x = 2.5$ do c_2; in ℓ_2 at $x = 4$ do c_3

$$\rho: \quad (\ell_0, 0) \xrightarrow{2} (\ell_0, 2) \xrightarrow{c_1} (\ell_1, 2) \xrightarrow{u \text{ at } \delta \leq 0.5} (\ell_2, 2 + \delta) \xrightarrow{c_3 \text{ at } 2-\delta} (\ell_0, 0) \cdots$$

$$\rho': \quad (\ell_0, 0) \xrightarrow{2} (\ell_0, 2) \xrightarrow{c_1} (\ell_1, 2) \xrightarrow{0.5} (\ell_1, 2.5) \xrightarrow{c_2} (\ell_2, 2.5)$$

Strategies and Winning States



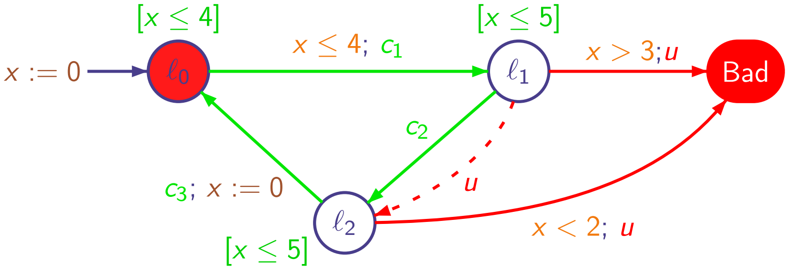
A winning strategy f'

in ℓ_0 at $x = 2$ do c_1; in ℓ_1 at $x = 2.5$ do c_2; in ℓ_2 at $x = 4$ do c_3

$$\rho : \quad (\ell_0, 0) \xrightarrow{2} (\ell_0, 2) \xrightarrow{c_1} (\ell_1, 2) \xrightarrow{u \text{ at } \delta \leq 0.5} (\ell_2, 2+\delta) \xrightarrow{c_3 \text{ at } 2-\delta} (\ell_0, 0) \cdots$$

$$\rho' : \quad (\ell_0, 0) \xrightarrow{2} (\ell_0, 2) \xrightarrow{c_1} (\ell_1, 2) \xrightarrow{0.5} (\ell_1, 2.5) \xrightarrow{c_2} (\ell_2, 2.5) \xrightarrow{1.5} (\ell_2, 4)$$

Strategies and Winning States



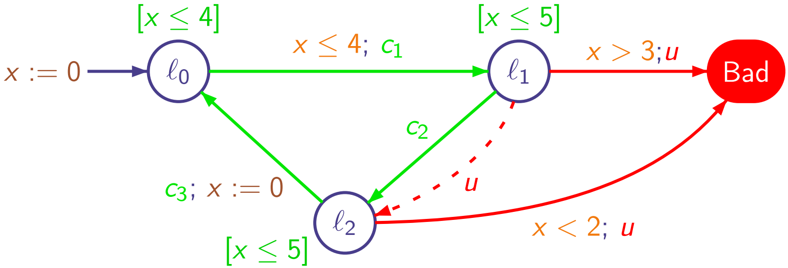
A winning strategy f'

in ℓ_0 at $x = 2$ do c_1; in ℓ_1 at $x = 2.5$ do c_2; in ℓ_2 at $x = 4$ do c_3

ρ : $(\ell_0, 0) \xrightarrow{2} (\ell_0, 2) \xrightarrow{c_1} (\ell_1, 2) \xrightarrow{u \text{ at } \delta \leq 0.5} (\ell_2, 2 + \delta) \xrightarrow{c_3 \text{ at } 2-\delta} (\ell_0, 0) \cdots$

ρ' : $(\ell_0, 0) \xrightarrow{2} (\ell_0, 2) \xrightarrow{c_1} (\ell_1, 2) \xrightarrow{0.5} (\ell_1, 2.5) \xrightarrow{c_2} (\ell_2, 2.5) \xrightarrow{1.5} (\ell_2, 4)$
$\xrightarrow{c_3} (\ell_0, 0) \cdots$

Strategies and Winning States



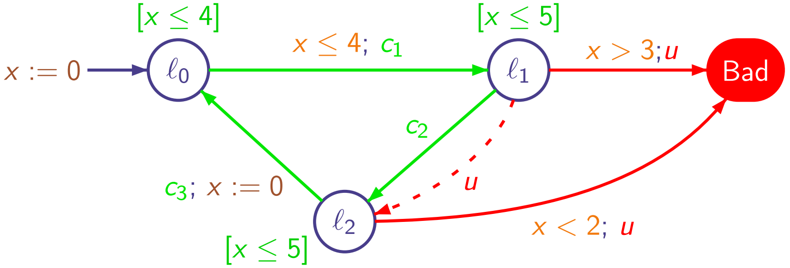
A winning strategy f'

in ℓ_0 at $x = 2$ do c_1; in ℓ_1 at $x = 2.5$ do c_2; in ℓ_2 at $x = 4$ do c_3

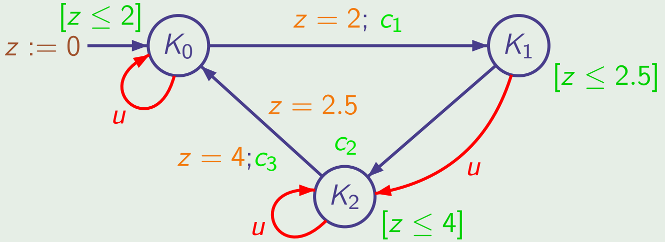
$$\rho : \quad (\ell_0, 0) \xrightarrow{2} (\ell_0, 2) \xrightarrow{c_1} (\ell_1, 2) \xrightarrow{u \text{ at } \delta \leq 0.5} (\ell_2, 2 + \delta) \xrightarrow{c_3 \text{ at } 2-\delta} (\ell_0, 0) \cdots$$

$$\rho' : \quad (\ell_0, 0) \xrightarrow{2} (\ell_0, 2) \xrightarrow{c_1} (\ell_1, 2) \xrightarrow{0.5} (\ell_1, 2.5) \xrightarrow{c_2} (\ell_2, 2.5) \xrightarrow{1.5} (\ell_2, 4) \xrightarrow{c_3} (\ell_0, 0) \cdots$$

Strategies and Winning States



The Strategy f' as a Timed Automaton



Outline

- Verification & Control
- Control of Finite Automata
- Timed Game Automata
- **Symbolic Algorithms for Timed Game Automata**
- Conclusion

Symbolic States

- $Q = L \times \mathbb{R}_{\geq 0}^{Clock}$ a the set of states of the TGA $\qquad q = (\ell, v) \in Q$
- Discrete predecessors of $X \subseteq Q$ by an action a:
$$\mathrm{Pred}^a(X) = \{q \in Q \mid q \xrightarrow{a} q' \text{ and } q' \in X\}$$
- Time predecessors of $X \subseteq Q$:
$$\mathrm{Pred}^\delta(X) = \{q \in Q \mid \exists t \geq 0 \mid q \xrightarrow{t} q' \text{ and } q' \in X\}$$
- Zone = conjunction of triangular constraints
$x - y < 3$, $x \geq 2 \wedge 1 < y - x < 2$
- State predicate (SP) $P = \cup_{i \in [1..n]}(\ell_{j_i}, Z_i)$, $\ell_i \in L$, Z_i is a zone
$(\ell_1, 2 \leq x < 4)$ or $(\ell_0, x < 1 \wedge y - x \geq 2)$ or $(\ell_0, x \leq 2) \cup (\ell_2, x > 0)$

Effectiveness of Pred^a and Pred^δ

If P is a SP then $\mathrm{Pred}^a(P), \mathrm{Pred}^\delta(P)$ are SP and can be computed effectively. (There is a symbolic version of Pred^a and Pred^δ.)

Symbolic States

- $Q = L \times \mathbb{R}_{\geq 0}^{Clock}$ a the set of states of the TGA $\qquad q = (\ell, v) \in Q$
- Discrete predecessors of $X \subseteq Q$ by an action a:
$$\text{Pred}^a(X) = \{q \in Q \mid q \xrightarrow{a} q' \text{ and } q' \in X\}$$
- Time predecessors of $X \subseteq Q$:
$$\text{Pred}^\delta(X) = \{q \in Q \mid \exists t \geq 0 \mid q \xrightarrow{t} q' \text{ and } q' \in X\}$$
- Zone = conjunction of triangular constraints
$x - y < 3$, $x \geq 2 \wedge 1 < y - x < 2$
- State predicate (SP) $P = \cup_{i \in [1..n]}(\ell_{j_i}, Z_i)$, $\ell_i \in L$, Z_i is a zone
$(\ell_1, 2 \leq x < 4)$ or $(\ell_0, x < 1 \wedge y - x \geq 2)$ or $(\ell_0, x \leq 2) \cup (\ell_2, x > 0)$

Effectiveness of Pred^a and Pred^δ

If P is a SP then $\text{Pred}^a(P), \text{Pred}^\delta(P)$ are SP and can be computed effectively. (There is a symbolic version of Pred^a and Pred^δ.)

Symbolic States

- $Q = L \times \mathbb{R}_{\geq 0}^{Clock}$ a the set of states of the TGA $\qquad q = (\ell, v) \in Q$
- Discrete predecessors of $X \subseteq Q$ by an action a:

$$\text{Pred}^a(X) = \{q \in Q \mid q \xrightarrow{a} q' \text{ and } q' \in X\}$$

- Time predecessors of $X \subseteq Q$:

$$\text{Pred}^\delta(X) = \{q \in Q \mid \exists t \geq 0 \mid q \xrightarrow{t} q' \text{ and } q' \in X\}$$

- Zone = conjunction of triangular constraints
  $x - y < 3,\ x \geq 2 \wedge 1 < y - x < 2$
- State predicate (SP) $P = \cup_{i \in [1..n]}(\ell_{j_i}, Z_i)$, $\ell_i \in L$, Z_i is a zone
  $(\ell_1, 2 \leq x < 4)$ or $(\ell_0, x < 1 \wedge y - x \geq 2)$ or $(\ell_0, x \leq 2) \cup (\ell_2, x > 0)$

Effectiveness of Pred^a and Pred^δ

If P is a SP then $\text{Pred}^a(P), \text{Pred}^\delta(P)$ are SP and can be computed effectively. (There is a symbolic version of Pred^a and Pred^δ.)

Symbolic States

- $Q = L \times \mathbb{R}_{\geq 0}^{Clock}$ a the set of states of the TGA $\qquad q = (\ell, v) \in Q$
- Discrete predecessors of $X \subseteq Q$ by an action a:
$$\text{Pred}^a(X) = \{q \in Q \mid q \xrightarrow{a} q' \text{ and } q' \in X\}$$
- Time predecessors of $X \subseteq Q$:
$$\text{Pred}^\delta(X) = \{q \in Q \mid \exists t \geq 0 \mid q \xrightarrow{t} q' \text{ and } q' \in X\}$$
- Zone = conjunction of triangular constraints
$x - y < 3,\ x \geq 2 \wedge 1 < y - x < 2$
- State predicate (SP) $P = \cup_{i \in [1..n]}(\ell_{j_i}, Z_i)$, $\ell_i \in L$, Z_i is a zone
$(\ell_1, 2 \leq x < 4)$ or $(\ell_0, x < 1 \wedge y - x \geq 2)$ or $(\ell_0, x \leq 2) \cup (\ell_2, x > 0)$

Effectiveness of Pred^a and Pred^δ

If P is a SP then $\text{Pred}^a(P), \text{Pred}^\delta(P)$ are SP and can be computed effectively. (There is a symbolic version of Pred^a and Pred^δ.)

Symbolic States

- $Q = L \times \mathbb{R}_{\geq 0}^{Clock}$ a the set of states of the TGA $\qquad q = (\ell, v) \in Q$
- Discrete predecessors of $X \subseteq Q$ by an action a:
$$\mathrm{Pred}^a(X) = \{q \in Q \mid q \xrightarrow{a} q' \text{ and } q' \in X\}$$
- Time predecessors of $X \subseteq Q$:
$$\mathrm{Pred}^\delta(X) = \{q \in Q \mid \exists t \geq 0 \mid q \xrightarrow{t} q' \text{ and } q' \in X\}$$
- Zone = conjunction of triangular constraints
$x - y < 3,\ x \geq 2 \wedge 1 < y - x < 2$
- State predicate (SP) $P = \cup_{i \in [1..n]}(\ell_{j_i}, Z_i)$, $\ell_i \in L$, Z_i is a zone
$(\ell_1, 2 \leq x < 4)$ or $(\ell_0, x < 1 \wedge y - x \geq 2)$ or $(\ell_0, x \leq 2) \cup (\ell_2, x > 0)$

Effectiveness of Pred^a and Pred^δ

If P is a SP then $\mathrm{Pred}^a(P), \mathrm{Pred}^\delta(P)$ are SP and can be computed effectively. (There is a symbolic version of Pred^a and Pred^δ.)

Symbolic States

- $Q = L \times \mathbb{R}_{\geq 0}^{Clock}$ a the set of states of the TGA $\qquad q = (\ell, v) \in Q$
- Discrete predecessors of $X \subseteq Q$ by an action a:
$$\text{Pred}^a(X) = \{q \in Q \mid q \xrightarrow{a} q' \text{ and } q' \in X\}$$
- Time predecessors of $X \subseteq Q$:
$$\text{Pred}^\delta(X) = \{q \in Q \mid \exists t \geq 0 \mid q \xrightarrow{t} q' \text{ and } q' \in X\}$$
- Zone = conjunction of triangular constraints
$x - y < 3$, $x \geq 2 \wedge 1 < y - x < 2$
- State predicate (SP) $P = \cup_{i \in [1..n]}(\ell_{j_i}, Z_i)$, $\ell_i \in L$, Z_i is a zone
$(\ell_1, 2 \leq x < 4)$ or $(\ell_0, x < 1 \wedge y - x \geq 2)$ or $(\ell_0, x \leq 2) \cup (\ell_2, x > 0)$

Effectiveness of Pred^a and Pred^δ

If P is a SP then $\text{Pred}^a(P), \text{Pred}^\delta(P)$ are SP and can be computed effectively. (There is a symbolic version of Pred^a and Pred^δ.)

Symbolic Computation For Timed Games

X is a state predicate

- $\mathrm{cPred}(X) = \bigcup_{c \in \mathrm{Act}_c} \mathrm{Pred}^c(X)$ $\qquad$ $\mathrm{uPred}(X) = \bigcup_{u \in \mathrm{Act}_u} \mathrm{Pred}^u(X)$
  cPred and uPred are effectively computable
- $\mathrm{Pred}_\delta(X, Y)$: Time controllable predecessors of X avoiding Y:

q $\qquad\qquad$ $q' \in X$

$\mathrm{Pred}_\delta(X, Y)$ is effectively computable for state predicates X, Y

- Controllable Predecessors Operator:

$$\pi_\delta(X) = \mathrm{Pred}_\delta\left(\mathrm{cPred}(X), \mathrm{uPred}(\overline{X})\right)$$

$\pi_\delta(X)$ is effectively computable for state predicate X.

Symbolic Computation For Timed Games

X is a state predicate

- $\mathrm{cPred}(X) = \bigcup_{c \in \mathrm{Act}_c} \mathrm{Pred}^c(X)$ $\qquad$ $\mathrm{uPred}(X) = \bigcup_{u \in \mathrm{Act}_u} \mathrm{Pred}^u(X)$
  cPred and uPred are effectively computable
- $\mathrm{Pred}_\delta(X, Y)$: Time controllable predecessors of X avoiding Y:

q $\qquad$ $q' \in X$

$\mathrm{Pred}_\delta(X, Y)$ is effectively computable for state predicates X, Y

- Controllable Predecessors Operator:

$$\pi_\delta(X) = \mathrm{Pred}_\delta\left(\mathrm{cPred}(X), \mathrm{uPred}(\overline{X})\right)$$

$\pi_\delta(X)$ is effectively computable for state predicate X.

Symbolic Computation For Timed Games

X is a state predicate

- $\mathrm{cPred}(X) = \bigcup_{c \in \mathrm{Act}_c} \mathrm{Pred}^c(X)$ $\mathrm{uPred}(X) = \bigcup_{u \in \mathrm{Act}_u} \mathrm{Pred}^u(X)$
  cPred and uPred are effectively computable
- $\mathrm{Pred}_\delta(X, Y)$: Time controllable predecessors of X avoiding Y:

q $q' \in X$

$\mathrm{Pred}_\delta(X, Y)$ is effectively computable for state predicates X, Y

- Controllable Predecessors Operator:

$$\pi_\delta(X) = \mathrm{Pred}_\delta\left(\mathrm{cPred}(X), \mathrm{uPred}(\overline{X})\right)$$

$\pi_\delta(X)$ is effectively computable for state predicate X.

Symbolic Computation For Timed Games

X is a state predicate

- $\mathrm{cPred}(X) = \bigcup_{c \in \mathrm{Act}_c} \mathrm{Pred}^c(X)$ $\qquad$ $\mathrm{uPred}(X) = \bigcup_{u \in \mathrm{Act}_u} \mathrm{Pred}^u(X)$
  cPred and uPred are effectively computable
- $\mathrm{Pred}_\delta(X, Y)$: Time controllable predecessors of X avoiding Y:

$$q \xrightarrow{\quad t \quad} q' \in X$$

$\mathrm{Pred}_\delta(X, Y)$ is effectively computable for state predicates X, Y

- Controllable Predecessors Operator:

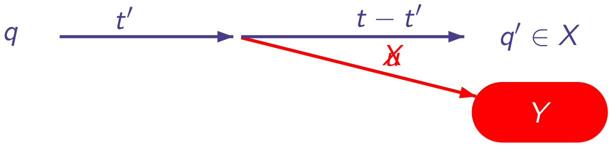
$$\pi_\delta(X) = \mathrm{Pred}_\delta\left(\mathrm{cPred}(X), \mathrm{uPred}(\overline{X})\right)$$

$\pi_\delta(X)$ is effectively computable for state predicate X.

Symbolic Computation For Timed Games

X is a state predicate

- $\mathrm{cPred}(X) = \bigcup_{c \in \mathrm{Act}_c} \mathrm{Pred}^c(X)$ $\qquad$ $\mathrm{uPred}(X) = \bigcup_{u \in \mathrm{Act}_u} \mathrm{Pred}^u(X)$
  cPred and uPred are effectively computable
- $\mathrm{Pred}_\delta(X, Y)$: Time controllable predecessors of X avoiding Y:



$\mathrm{Pred}_\delta(X, Y)$ is effectively computable for state predicates X, Y

- Controllable Predecessors Operator:

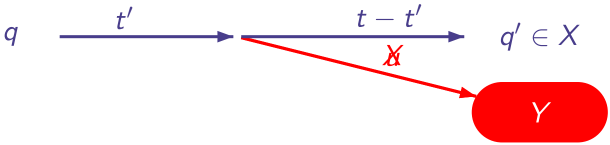
$$\pi_\delta(X) = \mathrm{Pred}_\delta\left(\mathrm{cPred}(X), \mathrm{uPred}(\overline{X})\right)$$

$\pi_\delta(X)$ is effectively computable for state predicate X.

Symbolic Computation For Timed Games

X is a state predicate

- $\mathrm{cPred}(X) = \bigcup_{c \in \mathrm{Act}_c} \mathrm{Pred}^c(X)$ $\qquad$ $\mathrm{uPred}(X) = \bigcup_{u \in \mathrm{Act}_u} \mathrm{Pred}^u(X)$
  cPred and uPred are effectively computable
- $\mathrm{Pred}_\delta(X, Y)$: Time controllable predecessors of X avoiding Y:



> $\mathrm{Pred}_\delta(X, Y)$ is effectively computable for state predicates X, Y

- Controllable Predecessors Operator:

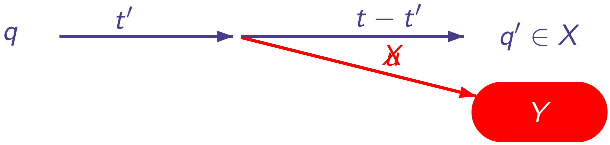
$$\pi_\delta(X) = \mathrm{Pred}_\delta\left(\mathrm{cPred}(X), \mathrm{uPred}(\overline{X})\right)$$

> $\pi_\delta(X)$ is effectively computable for state predicate X.

Symbolic Computation For Timed Games

X is a state predicate

- $\mathrm{cPred}(X) = \bigcup_{c \in \mathrm{Act}_c} \mathrm{Pred}^c(X)$ $\qquad$ $\mathrm{uPred}(X) = \bigcup_{u \in \mathrm{Act}_u} \mathrm{Pred}^u(X)$
  cPred and uPred are effectively computable
- $\mathrm{Pred}_\delta(X, Y)$: Time controllable predecessors of X avoiding Y:



$\mathrm{Pred}_\delta(X, Y)$ is effectively computable for state predicates X, Y

- Controllable Predecessors Operator:

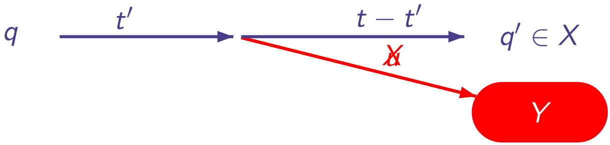
$$\pi_\delta(X) = \mathrm{Pred}_\delta\left(\mathrm{cPred}(X), \mathrm{uPred}(\overline{X})\right)$$

$\pi_\delta(X)$ is effectively computable for state predicate X.

Symbolic Computation For Timed Games

X is a state predicate

- $\mathrm{cPred}(X) = \bigcup_{c \in \mathrm{Act}_c} \mathrm{Pred}^c(X)$ $\qquad$ $\mathrm{uPred}(X) = \bigcup_{u \in \mathrm{Act}_u} \mathrm{Pred}^u(X)$
  cPred and uPred are effectively computable
- $\mathrm{Pred}_\delta(X, Y)$: Time controllable predecessors of X avoiding Y:



> $\mathrm{Pred}_\delta(X, Y)$ is effectively computable for state predicates X, Y

- Controllable Predecessors Operator:

$$\pi_\delta(X) = \mathrm{Pred}_\delta\left(\mathrm{cPred}(X), \mathrm{uPred}(\overline{X})\right)$$

> $\pi_\delta(X)$ is effectively computable for state predicate X.

Solving CP and CSP for Safety Timed Games

Symbolic Algorithm for Safety Timed Games

1. let φ be a State Predicate, G a timed game
2. let W^* be the greatest fixpoint of $h(X) = \varphi \cap \pi_\delta(X)$
3. W^* is the set of winning states for (G, φ)

Solving CP and CSP for Safety Timed Games

Symbolic Algorithm for Safety Timed Games

1. let φ be a State Predicate, G a timed game
2. let W^* be the greatest fixpoint of $h(X) = \varphi \cap \pi_\delta(X)$
3. W^* is the set of winning states for (G, φ)

- CP: check that $(\ell_0, 0) \in W^*$
- CSP: by def. of π_δ there is a strategy

Solving CP and CSP for Safety Timed Games

Symbolic Algorithm for Safety Timed Games

1. let φ be a State Predicate, G a timed game
2. let W^* be the greatest fixpoint of $h(X) = \varphi \cap \pi_\delta(X)$
3. W^* is the set of winning states for (G, φ)

Theorem (Termination [Maler, 95, De Alfaro, 01])

The iterative computation of W^* terminates for (G, φ) with G a timed game automaton φ a ω-regular control objective.

Solving CP and CSP for Safety Timed Games

Symbolic Algorithm for Safety Timed Games

1. let φ be a State Predicate, G a timed game
2. let W^* be the greatest fixpoint of $h(X) = \varphi \cap \pi_\delta(X)$
3. W^* is the set of winning states for (G, φ)

Theorem (Termination [Maler, 95, De Alfaro, 01])

The iterative computation of W^* terminates for (G, φ) with G a timed game automaton φ a ω-regular control objective.

Theorem (Decidability of CP [Maler, 95, De Alfaro, 01])

The (Safety) Control Problem is decidable for Timed Game Automata.

Solving CP and CSP for Safety Timed Games

Symbolic Algorithm for Safety Timed Games

1. let φ be a State Predicate, G a timed game
2. let W^* be the greatest fixpoint of $h(X) = \varphi \cap \pi_\delta(X)$
3. W^* is the set of winning states for (G, φ)

Theorem (Termination [Maler, 95, De Alfaro, 01])

The iterative computation of W^* terminates for (G, φ) with G a timed game automaton φ a ω-regular control objective.

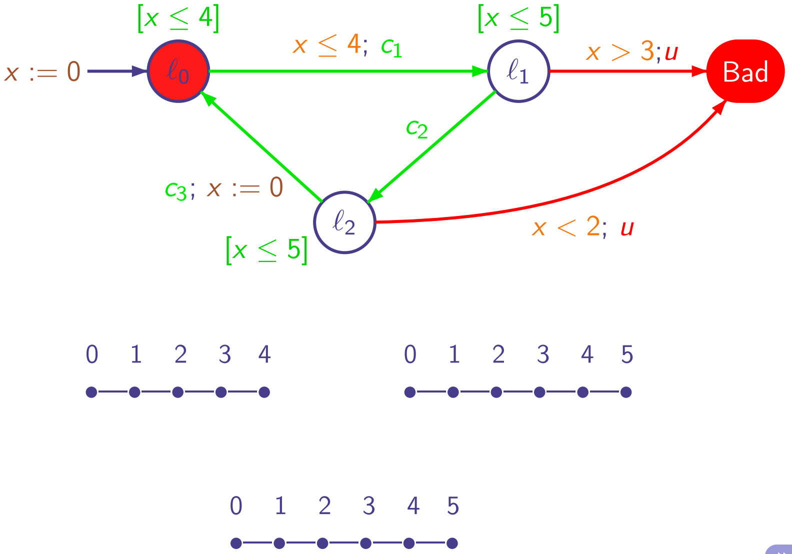
Theorem (Decidability of CP [Maler, 95, De Alfaro, 01])

The (Safety) Control Problem is decidable for Timed Game Automata.

Theorem (Effectiveness of CSP)

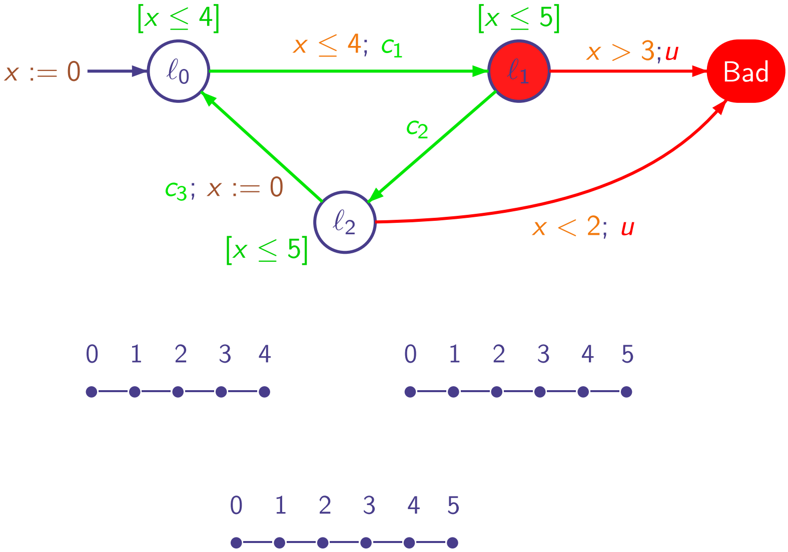
If $(\ell_0, 0) \in W^*$ we can compute a positional winning strategy.

Example of Computation

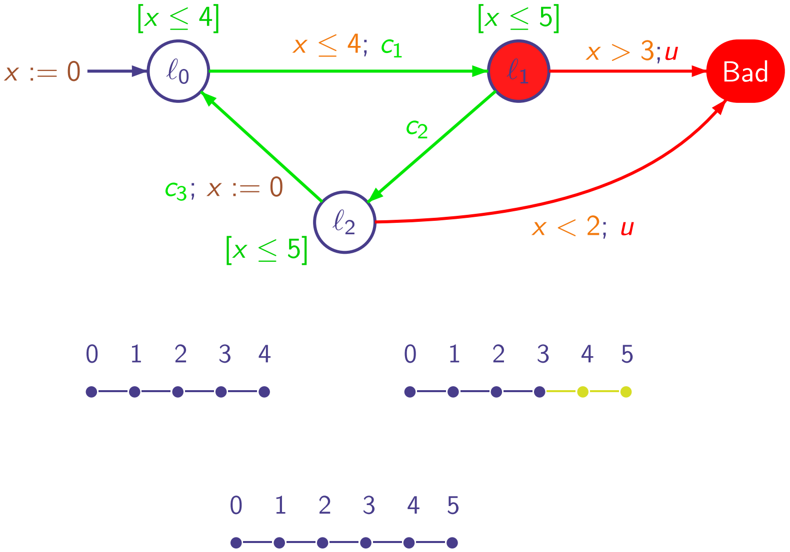


Skip

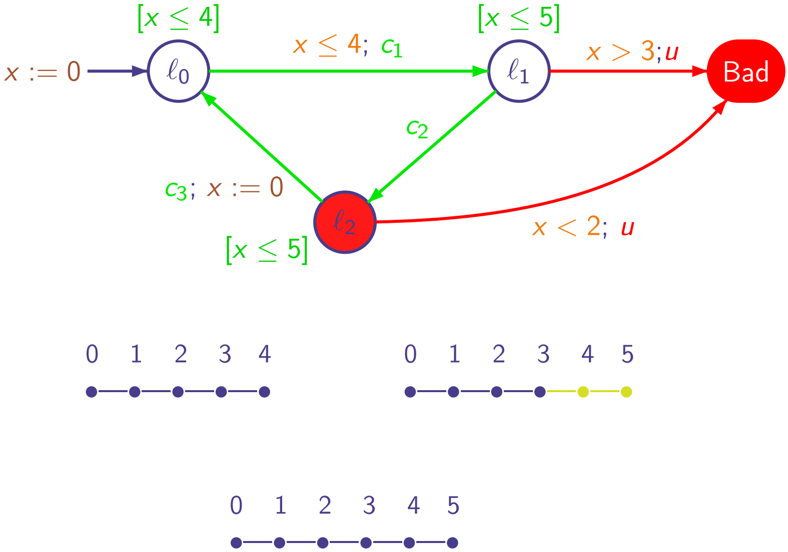
Example of Computation



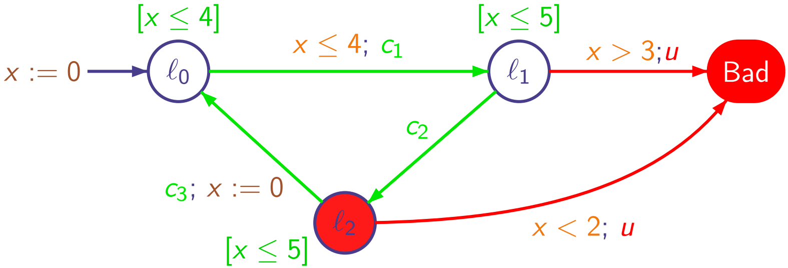
Example of Computation

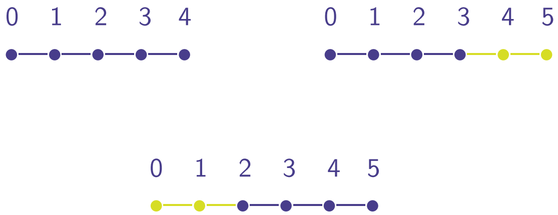


Example of Computation

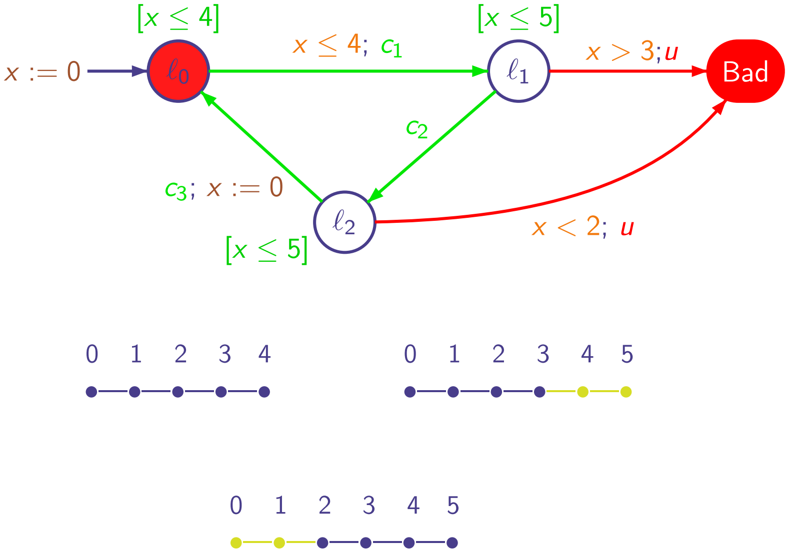


Example of Computation

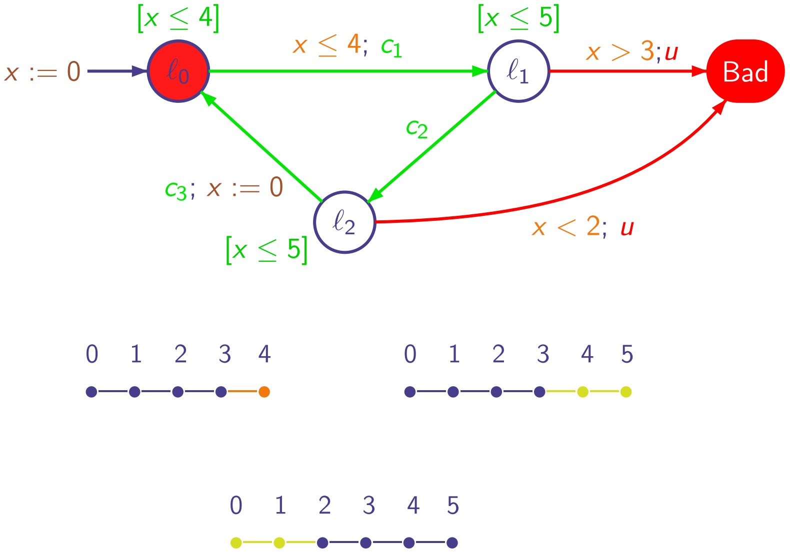




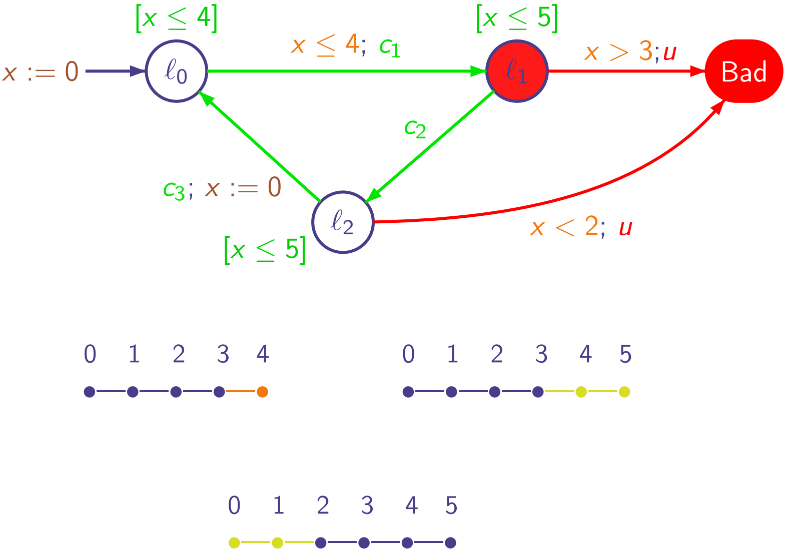
Example of Computation



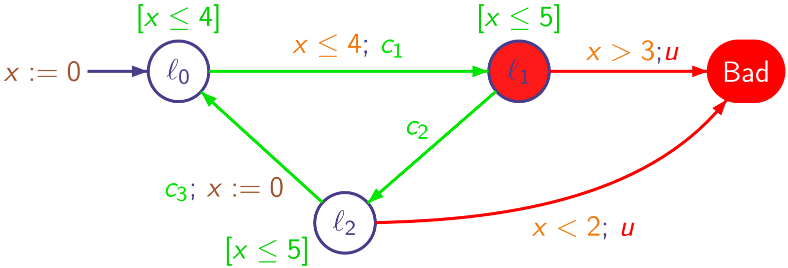
Example of Computation

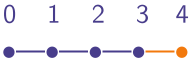


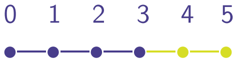
Example of Computation

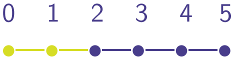


Example of Computation

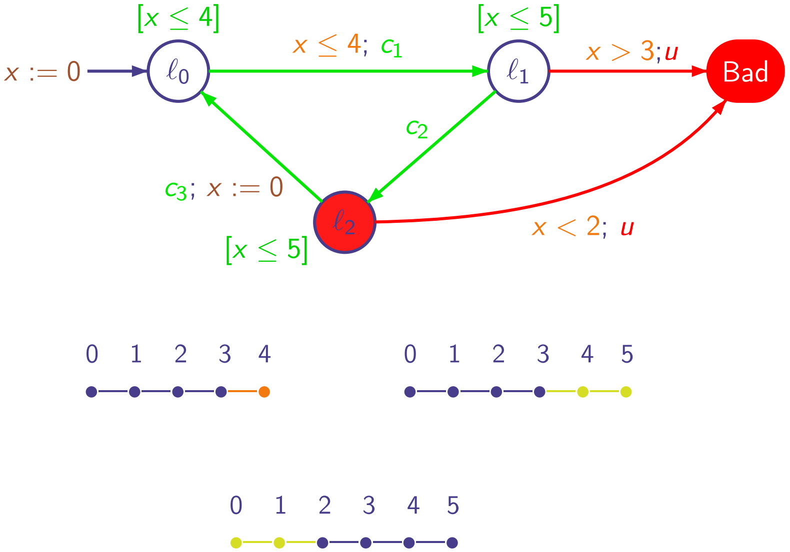




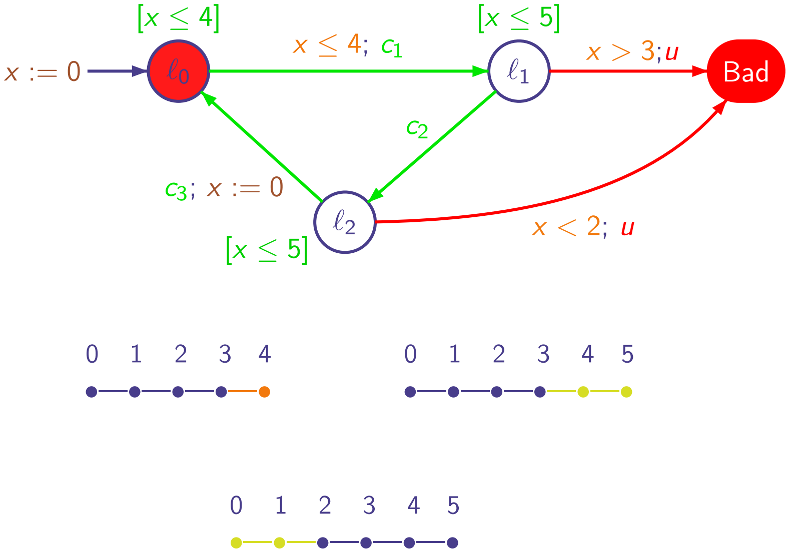




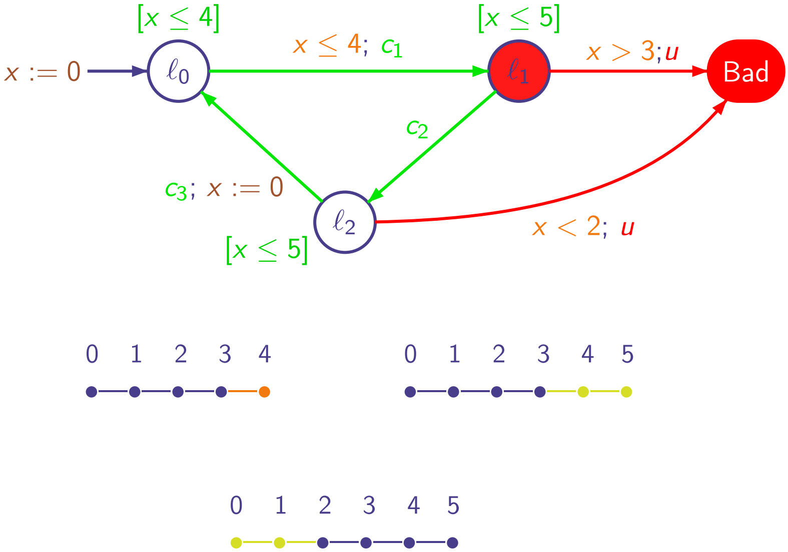
Example of Computation



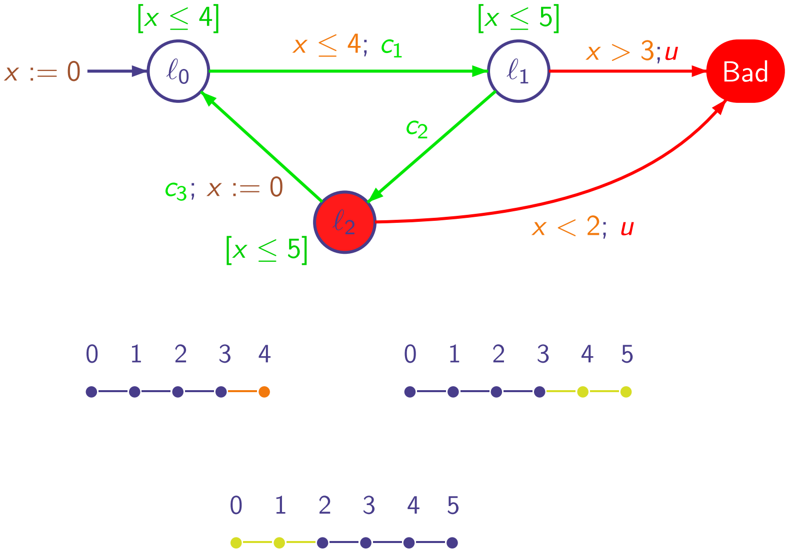
Example of Computation



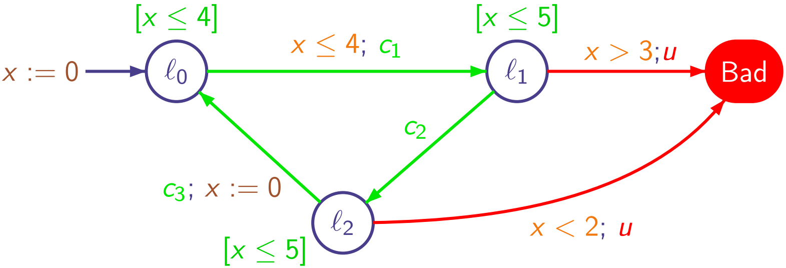
Example of Computation



Example of Computation



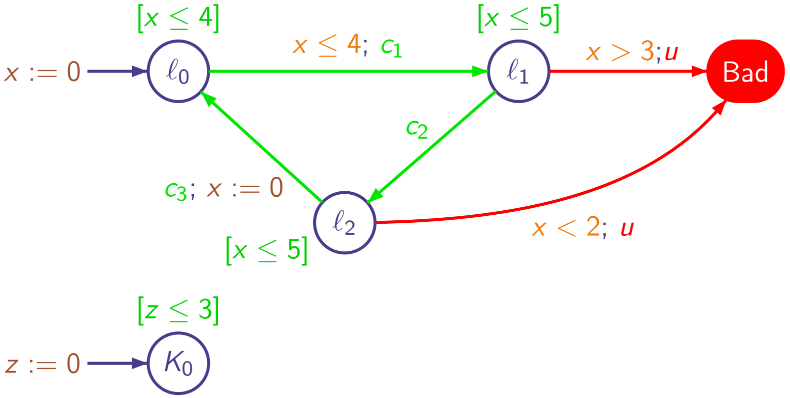
Example of Computation



$z := 0$

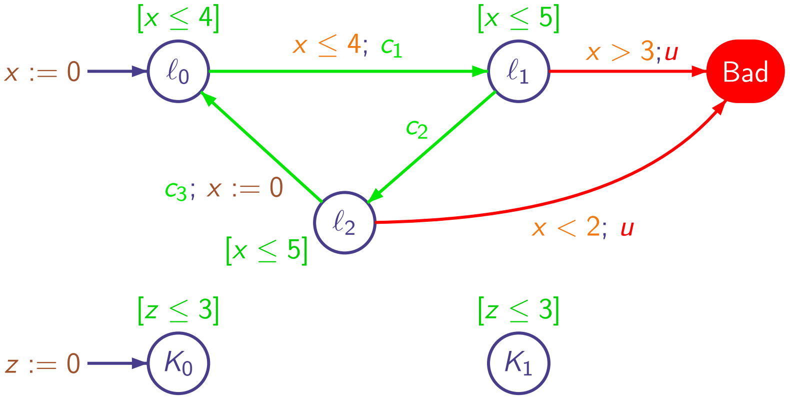
$(\ell_0, 0 \leq x \leq 3)$
$(\ell_1, 0 \leq x \leq 3)$
$(\ell_2, 2 \leq x \leq 5)$

Example of Computation



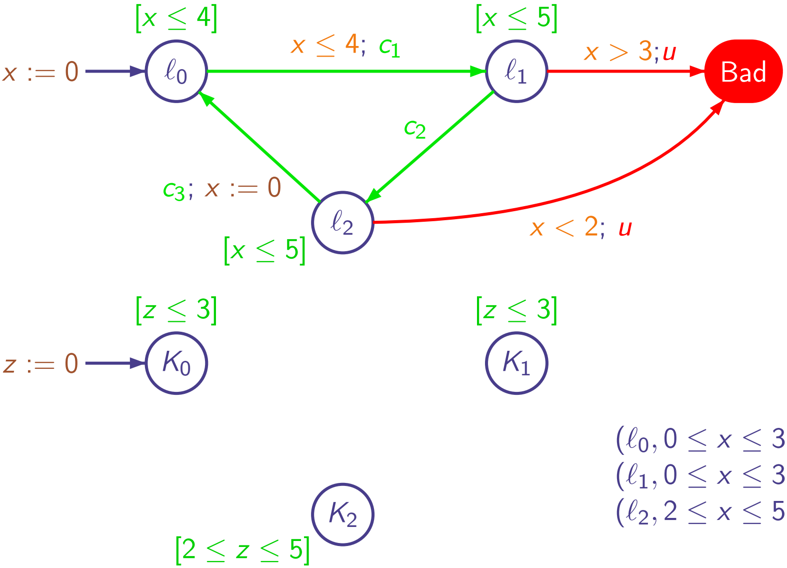
$(\ell_0, 0 \leq x \leq 3)$
$(\ell_1, 0 \leq x \leq 3)$
$(\ell_2, 2 \leq x \leq 5)$

Example of Computation



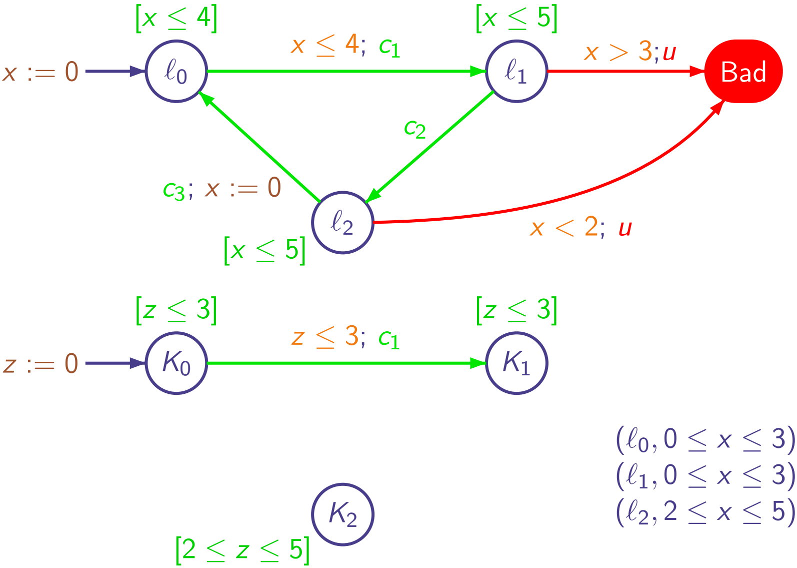
$(\ell_0, 0 \leq x \leq 3)$
$(\ell_1, 0 \leq x \leq 3)$
$(\ell_2, 2 \leq x \leq 5)$

Example of Computation



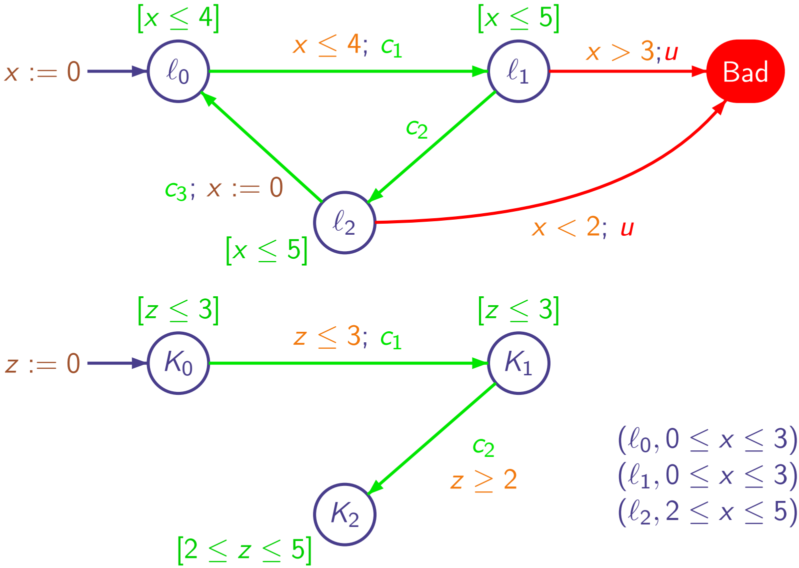
$(\ell_0, 0 \leq x \leq 3)$
$(\ell_1, 0 \leq x \leq 3)$
$(\ell_2, 2 \leq x \leq 5)$

Example of Computation



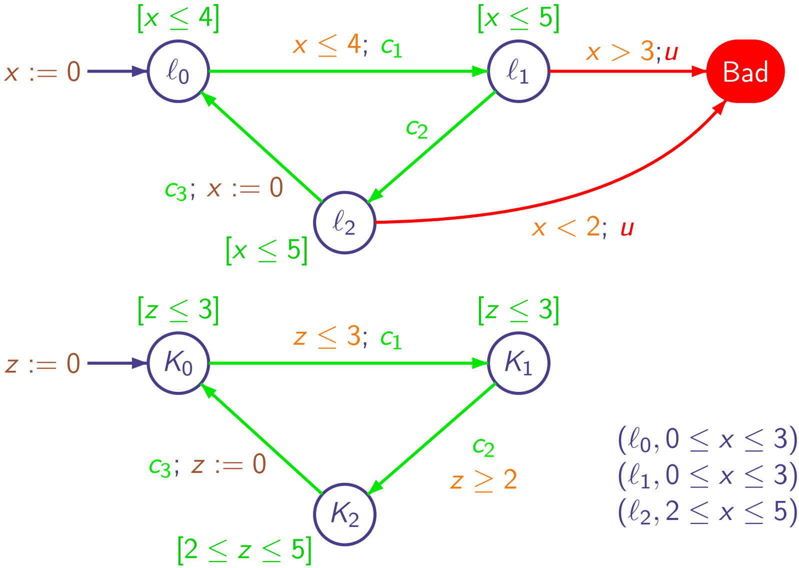
$(\ell_0, 0 \leq x \leq 3)$
$(\ell_1, 0 \leq x \leq 3)$
$(\ell_2, 2 \leq x \leq 5)$

Example of Computation



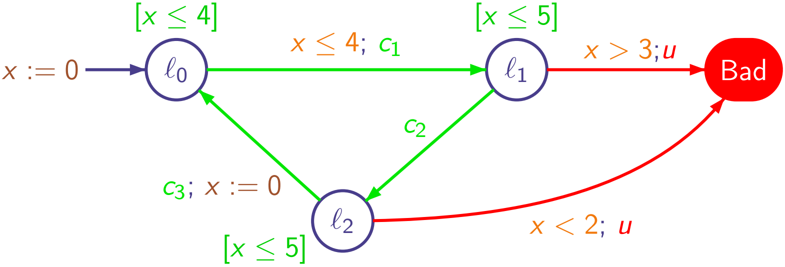
$(\ell_0, 0 \leq x \leq 3)$
$(\ell_1, 0 \leq x \leq 3)$
$(\ell_2, 2 \leq x \leq 5)$

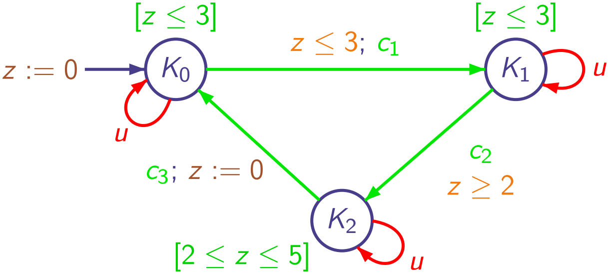
Example of Computation



$(\ell_0, 0 \leq x \leq 3)$
$(\ell_1, 0 \leq x \leq 3)$
$(\ell_2, 2 \leq x \leq 5)$

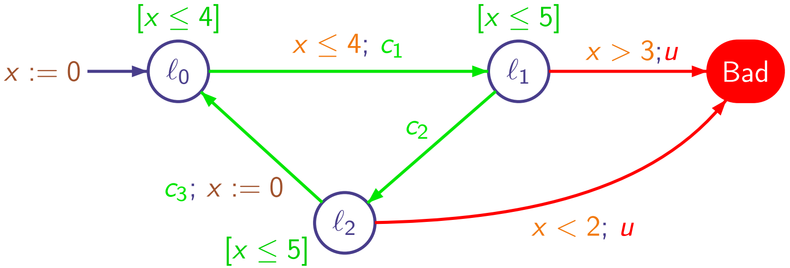
Example of Computation

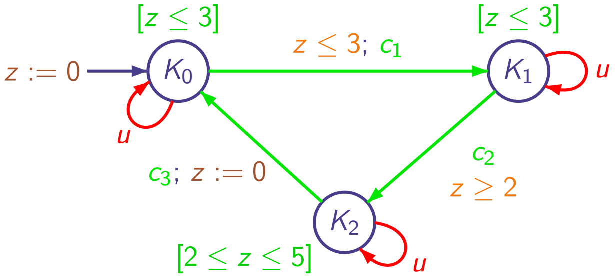




$(\ell_0, 0 \leq x \leq 3)$
$(\ell_1, 0 \leq x \leq 3)$
$(\ell_2, 2 \leq x \leq 5)$

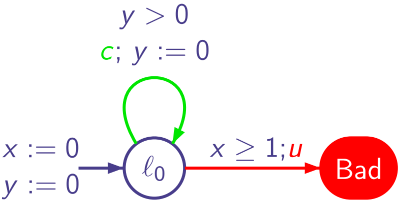
Example of Computation





The Most Liberal Controller

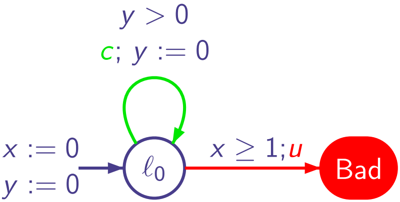
Problems with Dense-Time Control (I)



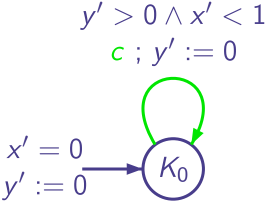
The System

The Controller is Zeno !!!

Problems with Dense-Time Control (I)



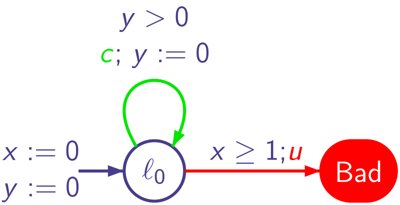
The System



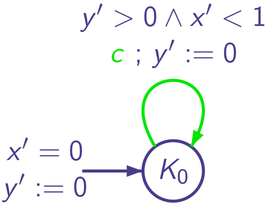
The Controller

The Controller is Zeno !!!

Problems with Dense-Time Control (I)



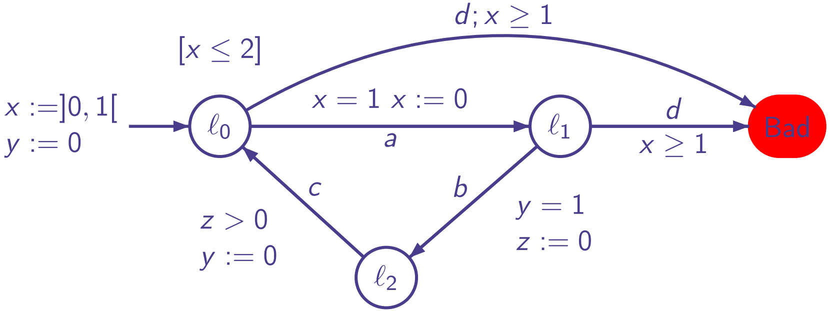
The System



The Controller

The Controller is Zeno !!!

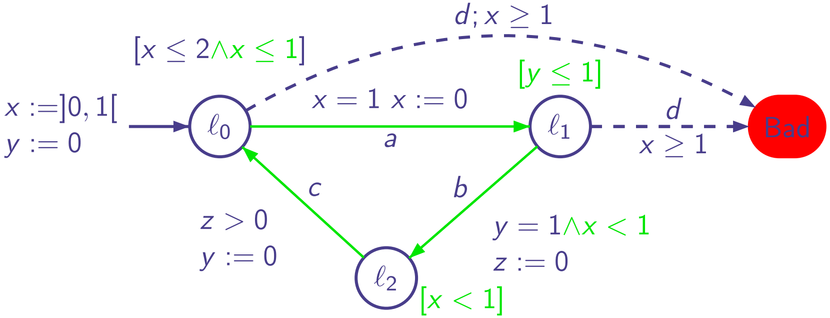
Problems with Dense-Time Control (II)



- Let δ_i : time spent in ℓ_2 on loop i
- The controller must ensure: $\sum_{i=0}^{i=+\infty} \delta_i < x_0 - y_0$

The Controller is Non-Zeno but not Implementable !!!

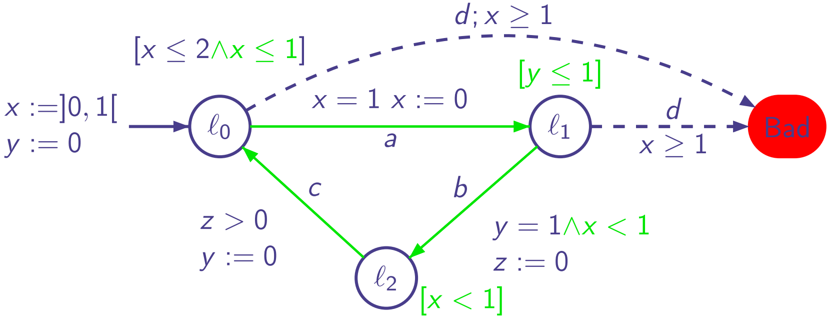
Problems with Dense-Time Control (II)



- Let δ_i : time spent in ℓ_2 on loop i
- The controller must ensure: $\sum_{i=0}^{i=+\infty} \delta_i < x_0 - y_0$

The Controller is Non-Zeno but not Implementable !!!

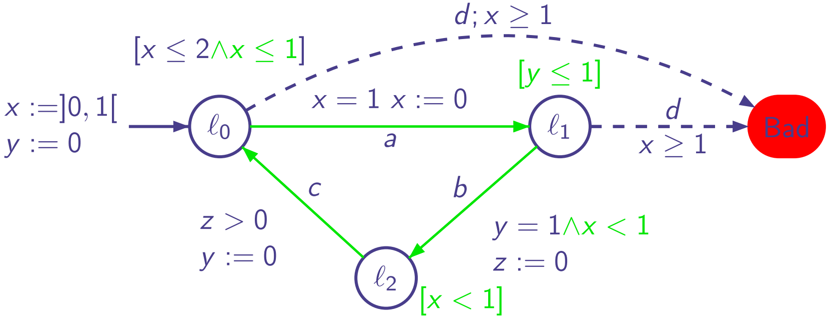
Problems with Dense-Time Control (II)



- Let δ_i : time spent in ℓ_2 on loop i
- The controller must ensure: $\sum_{i=0}^{i=+\infty} \delta_i < x_0 - y_0$

The Controller is Non-Zeno but not Implementable !!!

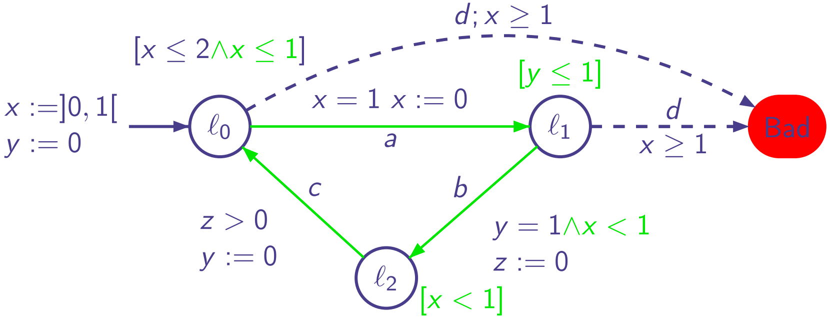
Problems with Dense-Time Control (II)



- Let δ_i : time spent in ℓ_2 on loop i
- The controller must ensure: $\sum_{i=0}^{i=+\infty} \delta_i < x_0 - y_0$

The Controller is Non-Zeno but not Implementable !!!

Problems with Dense-Time Control (II)



- Let δ_i : time spent in ℓ_2 on loop i
- The controller must ensure: $\sum_{i=0}^{i=+\infty} \delta_i < x_0 - y_0$

The Controller is Non-Zeno but not Implementable !!!

Outline

- Verification & Control
- Control of Finite Automata
- Timed Game Automata
- Symbolic Algorithms for Timed Game Automata
- Conclusion

Partial Conclusion

Assumptions:

- Timed systems with full observation
- Ideal controller that operates in dense-time

Results:

- Control Problem is decidable for ω-regular objectives
- Control Synthesis Problem is effective
- Positional (or Memoryless) strategies are sufficient

Advanced Topics:

- Partial Observability
  Patricia
- Implementation
  Karine

References

R. Alur and D. Dill.
A theory of timed automata.
Theoretical Computer Science B, 126:183–235, 1994.

Luca De Alfaro, Thomas A. Henzinger, and Rupak Majumdar.
Symbolic algorithms for infinite-state games.
In *Proc. 12th International Conference on Concurrency Theory (CONCUR'01)*, volume 2154 of *LNCS*, pages 536–550. Springer, 2001.

Eugene Asarin, Oded Maler, Amir Pnueli, and Joseph Sifakis.
Controller synthesis for timed automata.
In *Proc. IFAC Symposium on System Structure and Control*, pages 469–474. Elsevier Science, 1998.

AndrÃ© Arnold, Aymeric Vincent, and Igor Walukiewicz.
Games for synthesis of controllers with partial observation.
Theoretical Computer Science, 303(1):7–34,2003.

References (cont.)

- J.R. Büchi and L.H. Landweber.
  Solving sequential conditions by finite-state operators.
  Trans. of the AMS; 138:295–311.

- Franck Cassez, Thomas A. Henzinger, and Jean-FranÃ§ois Raskin.
  A comparison of control problems for timed and hybrid systems.
  In *Proc. 5th Int. Workshop on Hybrid Systems: Computation and Control (HSCC'02)*, volume 2289 of *LNCS*, pages 134–148. Springer, 2002.

- Oded Maler, Amir Pnueli, and Joseph Sifakis.
  On the synthesis of discrete controllers for timed systems.
  In *Proc. 12th Annual Symposium on Theoretical Aspects of Computer Science (STACS'95)*, volume 900, pages 229–242. Springer, 1995.

References (cont.)

P.J. Ramadge and W.M. Wonham.
Supervisory control of a class of discrete event processes.
SIAM J. of Control and Optimization, 25:206–230, 1987

P.J. Ramadge and W.M. Wonham.
The control of discrete event processes.
Proc. of IEEE, 77:81–98, 1989

J.G. Thistle and W.M. Wonham.
Control of infinite behavior of finite automata.
SIAM J. of Control and Optimization, 32:1075–1097, 1994

Timed Automata [Alur & Dill'94]

A Timed Automaton $\mathcal{A}$ is a tuple $(L, \ell_0, \mathrm{Act}, X, \mathrm{inv}, \longrightarrow)$ where:

- L is a finite set of locations
- ℓ_0 is the initial location
- X is a finite set of clocks
- Act is a finite set of actions
- $\longrightarrow$ is a set of transitions of the form $\ell \xrightarrow{g,\,a,\,R} \ell'$ with:
  - $\ell, \ell' \in L$,
  - $a \in \mathrm{Act}$
  - a guard g which is a clock constraint over X
  - a reset set R which is the set of clocks to be reset to 0

Clock constraints are boolean combinations of $x \sim k$ with $x \in C$ and $k \in \mathbb{Z}$ and $\sim \in \{\leq, <\}$.

Back

Semantics of Timed Automata

Let $\mathcal{A} = (L, \ell_0, \mathrm{Act}, X, \mathrm{inv}, \longrightarrow)$ be a Timed Automaton.

A state (ℓ, v) of $\mathcal{A}$ is in $L \times \mathbb{R}_{\geq 0}^X$

The semantics of $\mathcal{A}$ is a Timed Transition System $S_{\mathcal{A}} = (Q, q_0, \mathrm{Act} \cup \mathbb{R}_{\geq 0}, \longrightarrow)$ with:

- $Q = L \times \mathbb{R}_{\geq 0}^X$
- $q_0 = (\ell_0, \overline{0})$
- $\longrightarrow$ consists in:

$$\textbf{discrete transition: } (\ell, v) \xrightarrow{a} (\ell', v') \iff \begin{cases} \exists\, \ell \xrightarrow{g\,,\,a\,,\,r} \ell' \in \mathcal{A} \\ v \models g \\ v' = v[r \leftarrow 0] \\ v' \models \mathrm{inv}(\ell') \end{cases}$$

$$\textbf{delay transition: } (\ell, v) \xrightarrow{d} (\ell, v + d) \iff d \in \mathbb{R}_{\geq 0} \wedge v + d \models \mathrm{inv}(\ell)$$

Back

Definition (Outcome in Timed Games)

Let $G = (L, \ell_0, \mathrm{Act}, X, E, \mathrm{inv})$ be a TGA and f a strategy over G. The outcome $\mathrm{Outcome}((\ell, v), f)$ of f from configuration (ℓ, v) in G is the subset of $\mathrm{Runs}((\ell, v), G)$ defined inductively by:

- $(\ell, v) \in \mathrm{Outcome}((\ell, v), f)$,
- if $\rho \in \mathrm{Outcome}((\ell, v), f)$ then $\rho' = \rho \xrightarrow{e} (\ell', v') \in \mathrm{Outcome}((\ell, v), f)$ if $\rho' \in \mathrm{Runs}((\ell, v), G)$ and one of the following three conditions hold:
  1. $e \in \mathrm{Act}_u$,
  2. $e \in \mathrm{Act}_c$ and $e = f(\rho)$,
  3. $e \in \mathbb{R}_{\geq 0}$ and $\forall 0 \leq e' < e, \exists (\ell'', v'') \in (L \times \mathbb{R}^X_{\geq 0})$ s.t. $last(\rho) \xrightarrow{e'} (\ell'', v'') \wedge f(\rho \xrightarrow{e'} (\ell'', v'')) = \lambda$.
- an infinite run ρ is in $\in \mathrm{Outcome}((\ell, v), f)$ if all the finite prefixes of ρ are in $\mathrm{Outcome}((\ell, v), f)$.