Probabilistic Model Checking

Stefan Pranger

22. 06. 2023

Hausübung

So far...

- ... we have talked about:
  - Probabilistic Models: Markov Chains and Markov Decision Processes,

So far...

- … we have talked about:
  - Probabilistic Models: Markov Chains and Markov Decision Processes,
  - PCTL and how to compute probabilities;

So far...

- ... we have talked about:
  - Probabilistic Models: Markov Chains and Markov Decision Processes,
  - PCTL and how to compute probabilities;
  - Schedulers and

So far...

- ... we have talked about:
  - Probabilistic Models: Markov Chains and Markov Decision Processes,
  - PCTL and how to compute probabilities;
  - Schedulers and
  - Modelling in PRISM.

So far...

- ... we have talked about:
  - Probabilistic Models: Markov Chains and Markov Decision Processes,
  - PCTL and how to compute probabilities;
  - Schedulers and
  - Modelling in PRISM.

- Today we will round the topic off:
  - PCTL* for MCs (+ idea for MDPs)
  - Stochastic Games
  - Case Studies

PCTL* syntax

Subdivision into *state* (Φ)- and *path*-formulae (φ):

$$\begin{aligned} \Phi ::= {} & \mathit{true} \\ & \mid a \\ & \mid \Phi_1 \wedge \Phi_2 \\ & \mid \neg\Phi \\ & \mid \Pr_J(\varphi) \end{aligned} \qquad\qquad \begin{aligned} \varphi ::= {} & \Phi \\ & \mid \varphi_1 \wedge \varphi_2 \\ & \mid \neg\varphi \\ & \mid \mathbf{X}\varphi \\ & \mid \varphi_1 \;\mathbf{U}\; \varphi_2 \end{aligned}$$

where $a \in AP$ and $J \subseteq [0,1]$.

PCTL* syntax

Subdivision into *state* (Φ)- and *path*-formulae (φ):

$$
\begin{aligned}
\Phi ::= & \; \mathit{true} \\
& \mid a \\
& \mid \Phi_1 \wedge \Phi_2 \\
& \mid \neg \Phi \\
& \mid \Pr_J(\varphi)
\end{aligned}
\qquad\qquad
\begin{aligned}
\varphi ::= & \; \Phi \\
& \mid \varphi_1 \wedge \varphi_2 \\
& \mid \neg \varphi \\
& \mid \mathbf{X} \varphi \\
& \mid \varphi_1 \; \mathbf{U} \; \varphi_2
\end{aligned}
$$

where $a \in AP$ and $J \subseteq [0, 1]$.

We are now allowed to interchangly use state and path formulae as subformulae.

PCTL* syntax

Subdivision into *state* (Φ)- and *path*-formulae (φ):

$$
\begin{aligned}
\Phi ::= &\ \mathit{true} \\
\mid &\ a \\
\mid &\ \Phi_1 \wedge \Phi_2 \\
\mid &\ \neg\Phi \\
\mid &\ \Pr_J(\varphi)
\end{aligned}
\qquad\qquad
\begin{aligned}
\varphi ::= &\ \Phi \\
\mid &\ \varphi_1 \wedge \varphi_2 \\
\mid &\ \neg\varphi \\
\mid &\ \mathbf{X}\varphi \\
\mid &\ \varphi_1\ \mathbf{U}\ \varphi_2
\end{aligned}
$$

where $a \in AP$ and $J \subseteq [0,1]$.

We are now allowed to interchangly use state and path formulae as subformulae.

```
P=? [ GF "return_to_start" ];
P=? [ G(! (try = 1) | lost_count<4 U delivered=1 ) | delivered_count=MAX_COUNT ]
Pmax=? [ FG "hatch_closed" ]
...
```

Checking Linear Time Properties

- Last building block to model check PCTL*

Checking Linear Time Properties

- Last building block to model check PCTL*

Let $\mathcal{M}$ be a Markov Chain and φ be an LTL formula.

We are interested in:

$$Pr(\mathcal{M}, s \models \varphi) = Pr_s\{\pi \in Paths(\mathcal{M}) \mid \pi \models \varphi\}$$

Computing Probabilities for LT-Properties

- Recall that LT-properties can be expressed using automata.

Computing Probabilities for LT-Properties

- Recall that LT-properties can be expressed using automata.
- We employ an automata-based approach:
  - Convert φ into a *deterministic Rabin automata* $\mathcal{A}$.
  - Compute the Product Markov Chain $M \times \mathcal{A}$.
  - Compute the probability to satisfy φ using the product (*more on that later*).

Deterministic Rabin Automata

A *deterministic Rabin automatonisatuple* $\mathcal{A} = (Q, \Sigma, \delta, q_0, Acc)$, with

- Q a set of states and initial state q_0,
- Σ an alphabet,
- $\delta : Q \times \Sigma \to Q$ a transition function and
- $Acc \subseteq 2^Q \times 2^Q$.

An automaton $\mathcal{A}$ accepts a run $\pi = q_0 q_1 q_2 \ldots$ iff there exists a pair $(L, K) \in Acc$ s.t.:

$$(\exists n \geq 0. \forall m \geq n. q_m \notin L) \wedge (\exists^{\text{inf}} n \geq 0. q_n \in K)$$

Product Markov Chain

Let $\mathcal{M}$ be a Markov chain and $\mathcal{A}$ be a DFA. The product $\mathcal{M} \times \mathcal{A} = (S \times Q, \mathbb{P}', i, \{accept\}, L')$ is a Markov chain where:

- $L'(\langle s, q\rangle) = \{accept\}$ if $q \in F$,
- $i = \langle s_0, q_1 \rangle$ is the initial state with $q_1 = \delta(q_0, L(s))$ and
- $\mathbb{P}'(\langle s, q\rangle, \langle s', q'\rangle) = \mathbb{P}(s, s')$ if $q' = \delta(q, L(s'))$ and 0 otherwise.

Product Markov Chain

Let $\mathcal{M}$ be a Markov chain and $\mathcal{A}$ be a DFA. The product $\mathcal{M} \times \mathcal{A} = (S \times Q, \mathbb{P}', i, \{accept\}, L')$ is a Markov chain where:

- $L'(\langle s, q \rangle) = \{accept\}$ if $q \in F$,
- $i = \langle s_0, q_1 \rangle$ is the initial state with $q_1 = \delta(q_0, L(s))$ and
- $\mathbb{P}'(\langle s, q \rangle, \langle s', q' \rangle) = \mathbb{P}(s, s')$ if $q' = \delta(q, L(s'))$ and 0 otherwise.

  Post-Lecture-Note: This is the definition of a product with a DFA, the product with a DRA can be done in a similar way.

Product Markov Chain

Let $\mathcal{M}$ be a Markov chain and $\mathcal{A}$ be a DFA. The product $\mathcal{M} \times \mathcal{A} = (S \times Q, \mathbb{P}', i, \{accept\}, L')$ is a Markov chain where:

- $L'(\langle s, q \rangle) = \{accept\}$ if $q \in F$,
- $i = \langle s_0, q_1 \rangle$ is the initial state with $q_1 = \delta(q_0, L(s))$ and
- $\mathbb{P}'(\langle s, q \rangle, \langle s', q' \rangle) = \mathbb{P}(s, s')$ if $q' = \delta(q, L(s'))$ and 0 otherwise.

  Post-Lecture-Note: This is the definition of a product with a DFA, the product with a DRA can be done in a similar way.

Since $\mathcal{A}$ is deterministic it can be interpreted as a witness for its current state on the product trace:

$$\pi^+ = \langle s_0, q_1 \rangle, \langle s_1, q_2 \rangle, \langle s_2, q_3 \rangle, \ldots$$

Computing the Probability to Satisfy φ

- We want to use the product $\mathcal{M} \times \mathcal{A}$ and know
- $\mathcal{A}$'s acceptance condition:

$$(\exists n \geq 0. \forall m \geq n.\, q_m \notin L_i) \wedge (\exists^{\text{inf}} n \geq 0. q_n \in K_i)$$

- for a pair $L_i, K_i \in Acc$.

Computing the Probability to Satisfy φ

- We want to use the product $\mathcal{M} \times \mathcal{A}$ and know
- $\mathcal{A}$'s acceptance condition:

$$(\exists n \geq 0. \forall m \geq n. q_m \notin L_i) \wedge (\exists^{\text{inf}} n \geq 0. q_n \in K_i)$$

- for a pair $L_i, K_i \in Acc$.
- $\Rightarrow$ we need to compute the probability to see infinitely many labels from K_i and only finitely many labels from L_i for some i.

Bottom Strongly Connected Components

- Consider the underlying directed graph $G = (V, E)$ for a given Markov chain $\mathcal{M}$ and a component $C \in V$.
- C is *strongly connected* if $\forall s, t \in C$:
  - s is reachable from t and
  - t is reachable from s.

Bottom Strongly Connected Components

- Consider the underlying directed graph $G = (V, E)$ for a given Markov chain $\mathcal{M}$ and a component $C \in V$.
- C is *strongly connected* if $\forall s, t \in C$:
  - s is reachable from t and
  - t is reachable from s.
- C is *bottom* strongly connected if no state outside of C is reachable from C.

Bottom Strongly Connected Components

- Consider the underlying directed graph $G = (V, E)$ for a given Markov chain $\mathcal{M}$ and a component $C \in V$.
- C is *strongly connected* if $\forall s, t \in C$:
  - s is reachable from t and
  - t is reachable from s.
- C is *bottom* strongly connected if no state outside of C is reachable from C.
- For Markov chains we have that a bottom strongly connected component
  - cannot be left and
  - all states will be visited infinitely often with a probability of one.

Computing the Probability to Satisfy φ

According to the acceptance condition $Acc = \{(L_0, K_0), \ldots (L_m, K_m)\}$ of $\mathcal{A}$:

- Identify BSCCs C_j such that:
  - For some $i \in [0, m]$:

$$C_j \cap (S \times L_i) = \emptyset \text{ and } C_j \cap (S \times K_i) \neq \emptyset$$

- Let $U = \bigcup_{j,\, C_j\ accepting} C_j$

Computing the Probability to Satisfy φ

According to the acceptance condition $Acc = \{(L_0, K_0), \ldots (L_m, K_m)\}$ of $\mathcal{A}$:

- Identify BSCCs C_j such that:
  - For some $i \in [0, m]$:

$$C_j \cap (S \times L_i) = \emptyset \text{ and } C_j \cap (S \times K_i) \neq \emptyset$$

- Let $U = \bigcup_{j,\, C_j\ accepting} C_j$
- We then have the following:

$$\Pr(\mathcal{M}, s \models \varphi) = \Pr(\mathcal{M} \times \mathcal{A}, \langle s, q_i \rangle \models \mathbf{F} U)$$

LT-Properties over MDP $\mathcal{M}$

Some Remarks:

- The concept of BSCCs needs to be enriched to account for nondeterminism.
- $\Rightarrow$ Concept of *Maximum End Components*

LT-Properties over MDP $\mathcal{M}$

Some Remarks:

- The concept of BSCCs needs to be enriched to account for nondeterminism.
- $\Rightarrow$ Concept of *Maximum End Components*
- Memoryless schedulers do not suffice to realize LT properties.

LT-Properties over MDP $\mathcal{M}$

Some Remarks:

- The concept of BSCCs needs to be enriched to account for nondeterminism.
- $\Rightarrow$ Concept of *Maximum End Components*
- Memoryless schedulers do not suffice to realize LT properties.

Examples

Probabilistic Model Zoo

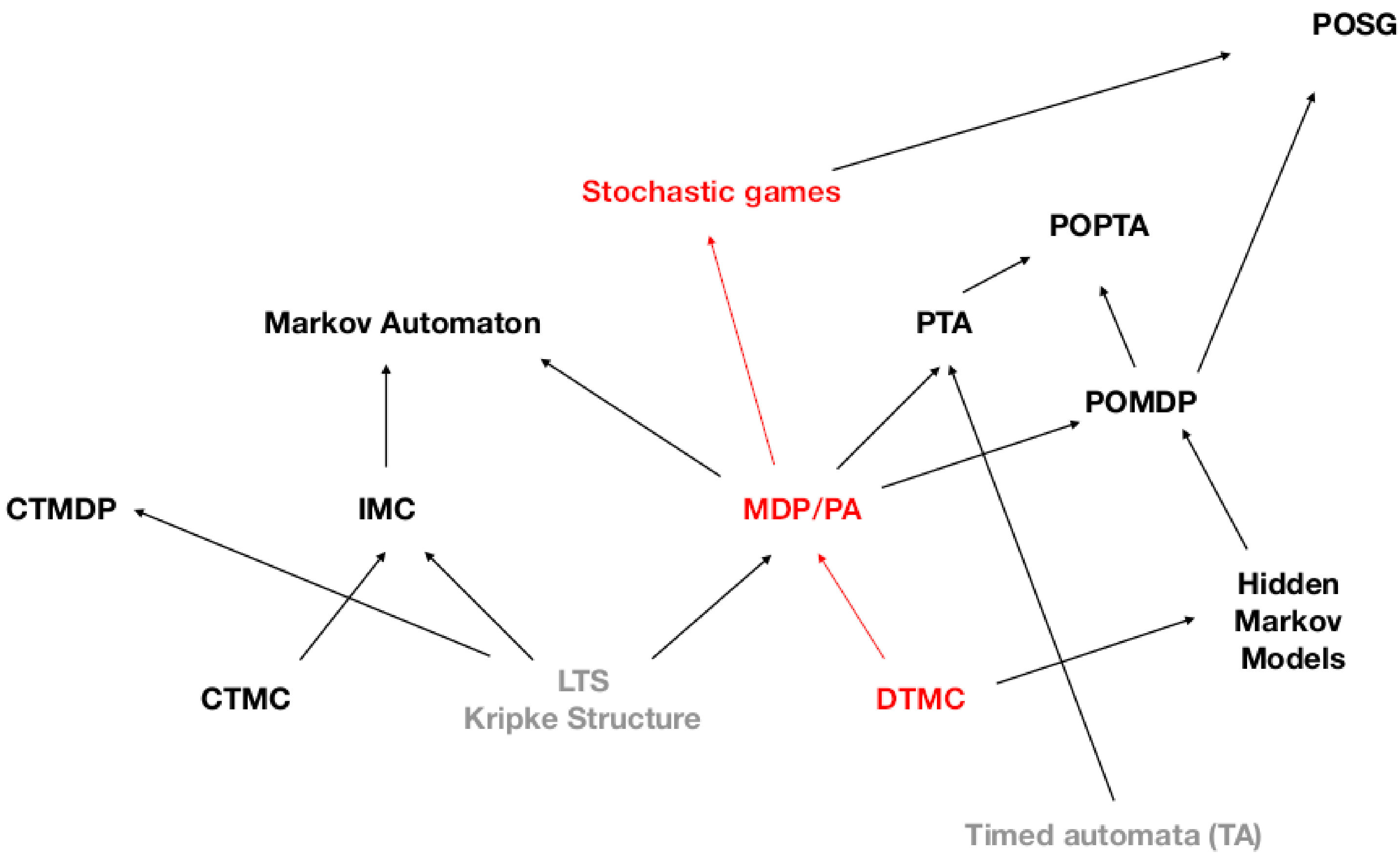


Figure from Sebastian Junge

Stochastic Games

- Generalization of MDPs
    - Multiple players decide on action in their respective states
    - They do this either turn-based or concurrently:
    - *Stochastic Multiplayer Games/Turn-based Stochastic Games*
    - *Concurrent Stochastic Games*

Stochastic Games

- Generalization of MDPs
  - Multiple players decide on action in their respective states
  - They do this either turn-based or concurrently:
  - *Stochastic Multiplayer Games/Turn-based Stochastic Games*
  - *Concurrent Stochastic Games*
- Different properties:
  - Zero-sum: A single value that is maximized by player 1, minimized by player 2.
  - Nonzero-sum: Players cooperate to achieve their individual goals.

Stochastic Multiplayer Game - Definition

Stochastic Multiplayer Game $\mathcal{G} = (S, \Pi, Act, \mathbb{P}, s_0, AP, L)$

- S a set of states and initial state s_0,
- Π a set of players,
- Act a set of actions,
- $\mathbb{P} : S \times Act \times S \to [0,1]$, s.t.

$$\sum_{s' \in S} \mathbb{P}(s, a, s') = 1 \; \forall (s, a) \in S \times Act$$

- AP set of atomic states and $L : S \to 2^{AP}$ a labelling function.

Solving Zero-Sum Reachability in Turn-based Stochastic Games

- Which players minimize and which players should maximize?

Solving Zero-Sum Reachability in Turn-based Stochastic Games

- Which players minimize and which players should maximize?
- $\Rightarrow$ this will become part of the property, by using
- a different logic : Probabilistic Alternating-time Temporal Logic (PATL)

Solving Zero-Sum Reachability in Turn-based Stochastic Games

- Which players minimize and which players should maximize?
- $\Rightarrow$ this will become part of the property, by using
- a different logic : Probabilistic Alternating-time Temporal Logic (PATL)
- For the purposes of this course:

```
player robot1
robotModule
endplayer
...
<<robot1>> Pmax=? [ G !"crash"]
```

  - The player "robot1" controls all actions defined in 'robotModule' *

Solution Method

- Adapt the Value Iteration approach from the MDP problem:
- Let S_{P1} and S_{P2} be the sets of states of the maximizer and and minimizer resp.

$$
\begin{aligned}
x_s^{(0)} &= 1, \forall s \in B \\
x_s^{(n)} &= 0, \forall s \in S_{=0} \\
x_s^{(0)} &= 0, \qquad \forall s \in S \setminus S_{=0} \\
\hline
x_s^{(n+1)} &= \max\{\textstyle\sum_{s' \in S} \mathbb{P}(s,a,s') \cdot x_{s'} | a \in Act(s)\}, \ \forall s \in (S \cap S_{P1}) \setminus S_{=0} \\
x_s^{(n+1)} &= \min\{\textstyle\sum_{s' \in S} \mathbb{P}(s,a,s') \cdot x_{s'} | a \in Act(s)\}, \ \forall s \in (S \cap S_{P2}) \setminus S_{=0}
\end{aligned}
$$

Shields

Recap:

- We can compute schedulers that maximize the probability to stay safe in an uncertain environment.

Shields

Recap:

- We can compute schedulers that maximize the probability to stay safe in an uncertain environment.
- In planning/reinforcement learning there are often goals that are beyond the scope of safety.
  - $\Rightarrow$ Need to ensure safety while hindering exploration as little as possible.

Shields

Recap:

- We can compute schedulers that maximize the probability to stay safe in an uncertain environment.
- In planning/reinforcement learning there are often goals that are beyond the scope of safety.
    - $\Rightarrow$ Need to ensure safety while hindering exploration as little as possible.
- A shield ensures that the probability to stay safe never drops beyond a certain threshold

Shields

- We can use the computation results from probabilistic model checking to construct a shield:

Shields

- We can use the computation results from probabilistic model checking to construct a shield:

When using an absolute threshold:

- Action is s *allowed* if: $\sum_{s' \in S} \mathbb{P}(s, a, s') \cdot x_{s'} > \gamma$

Shields

- We can use the computation results from probabilistic model checking to construct a shield:

When using an absolute threshold:

- Action is s *allowed* if: $\sum_{s' \in S} \mathbb{P}(s, a, s') \cdot x_{s'} > \gamma$

When using a relative threshold:

- Action is s *allowed* if: $\sum_{s' \in S} \mathbb{P}(s, a, s') \cdot x_{s'} > \lambda \cdot x_s$

Shields

- We distinguish between:
  - Absolute thresholds for safety and
  - Relative thresholds for safety,
- and:
  - Post-Shielding and
  - Pre-Shielding.

Shields

- We distinguish between:
  - Absolute thresholds for safety and
  - Relative thresholds for safety,
- and:
  - Post-Shielding and
  - Pre-Shielding.
- Let's look at some examples:

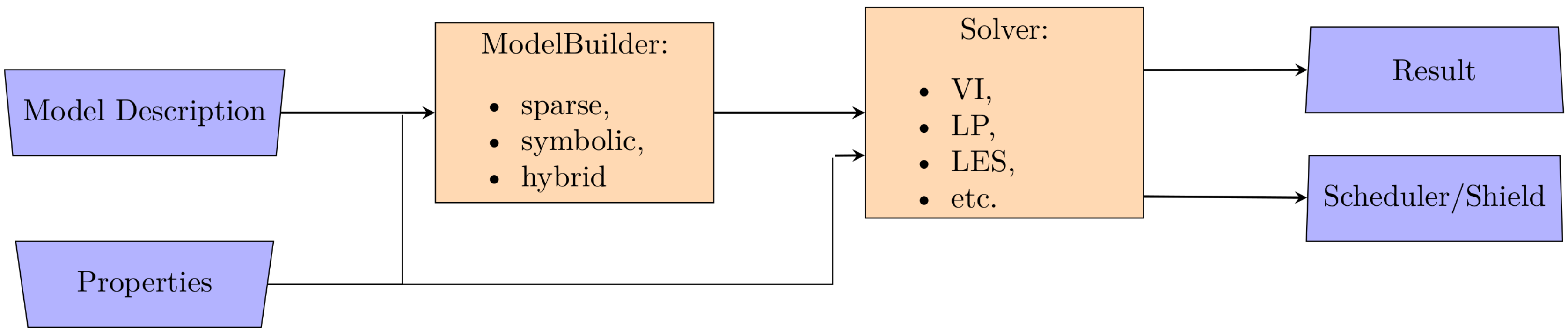
```
Pre-Safety-Shield with absolute comparison (gamma = 0.8):
state id [label]: 'allowed actions' [<value>: (<action id label)>]:

0 [move =0 & x1=0 & y1=0 & x2=4 & y2 =4]: 1.0:(0 {e}); 1:(1 {s})
3 [move =0 & x1=1 & y1=0 & x2=3 & y2 =4]: 0.9:(0 {e}); 1:(2 {w})
4 [move =0 & x1=1 & y1=0 & x2=4 & y2 =4]: 0.9:(1 {s}); 1:(3 {n})

Post-Safety-Shield with relative comparison (lambda = 0.95):
state id [label]: 'forwarded actions' [<action id> label: <forwarded action id> label]:

0 [move =0 & x1=0 & y1=0 & x2=4 & y2 =4]: 0{e}:0{e}; 1{s}:1{s}
3 [move =0 & x1=1 & y1=0 & x2=3 & y2 =4]: 0{e}:2{w}; 2{w}:2{w}
4 [move =0 & x1=1 & y1=0 & x2=4 & y2 =4]: 1{s}:3{n}; 3{n}:3{n}
```

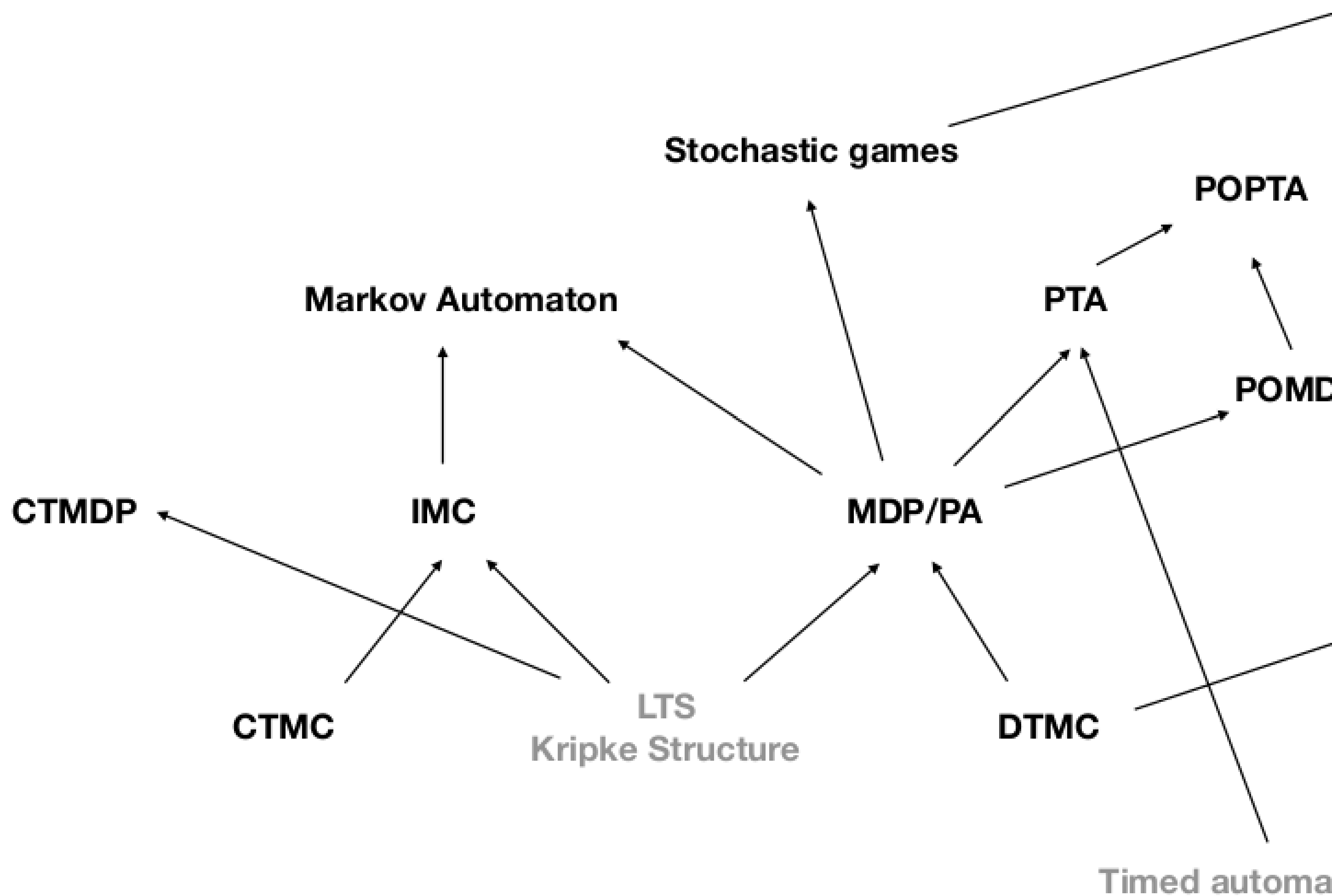
Summary Slide



Revisit: The Probabilistic Model Zoo

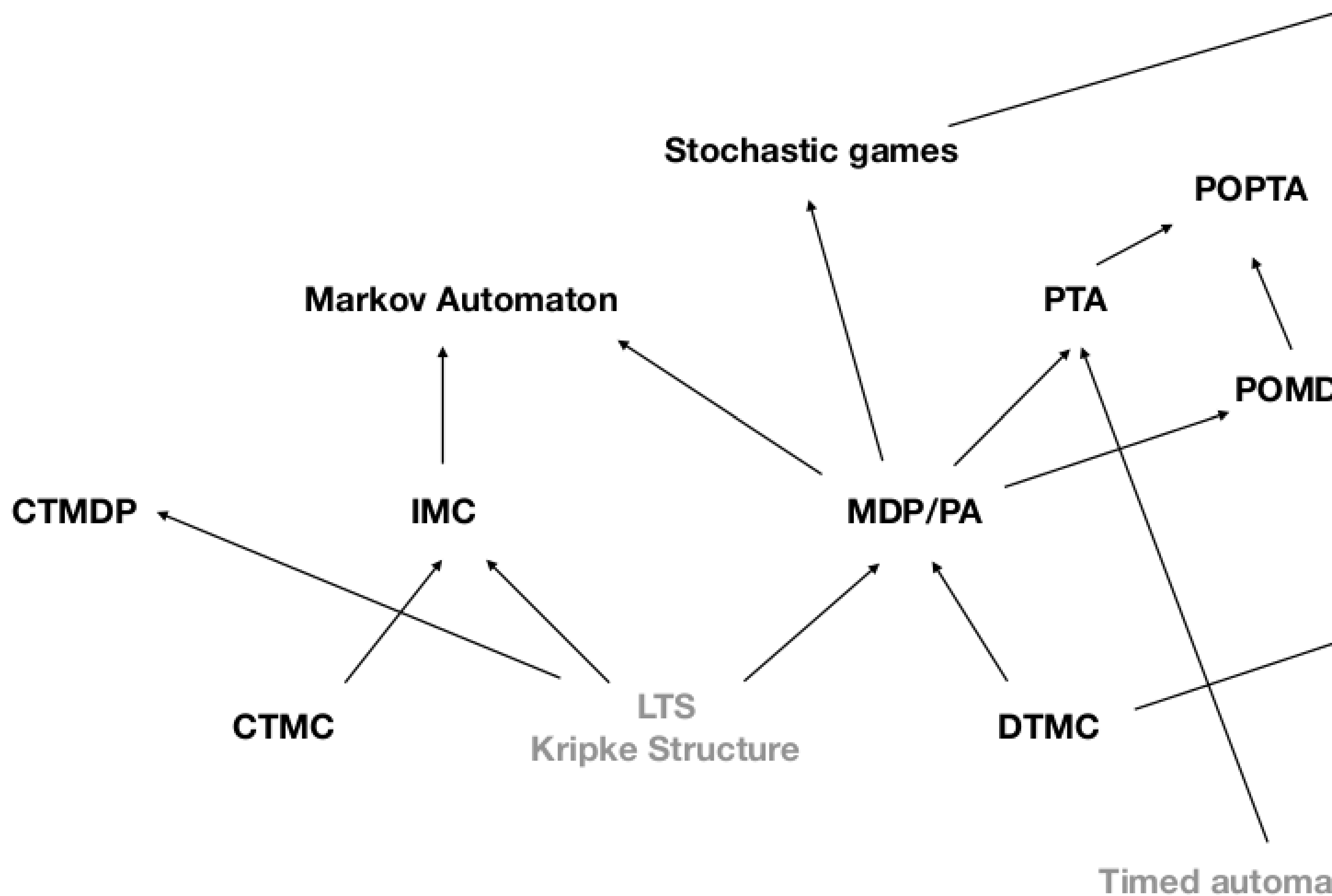
Two more primitives:

- Partial Observability
- Continuous-Time



Revisit: The Probabilistic Model Zoo

Two more primitives:
- Partial Observability
- Continuous-Time



Homework