

Stochastic Model Checking



UNIVERSITÄT
DES
SAARLANDES

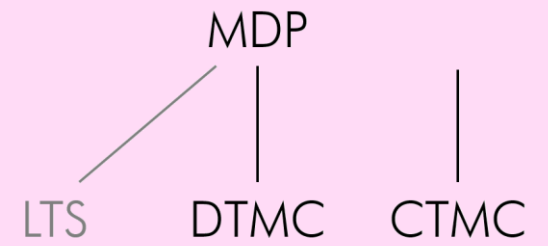
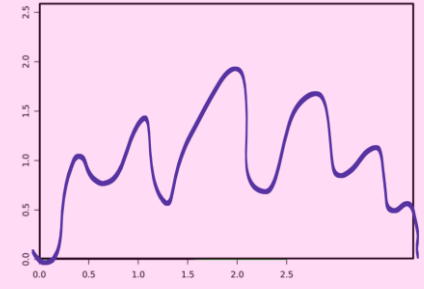
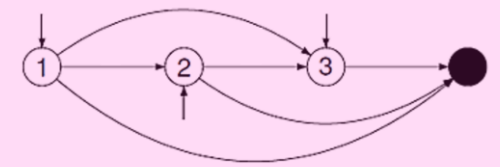
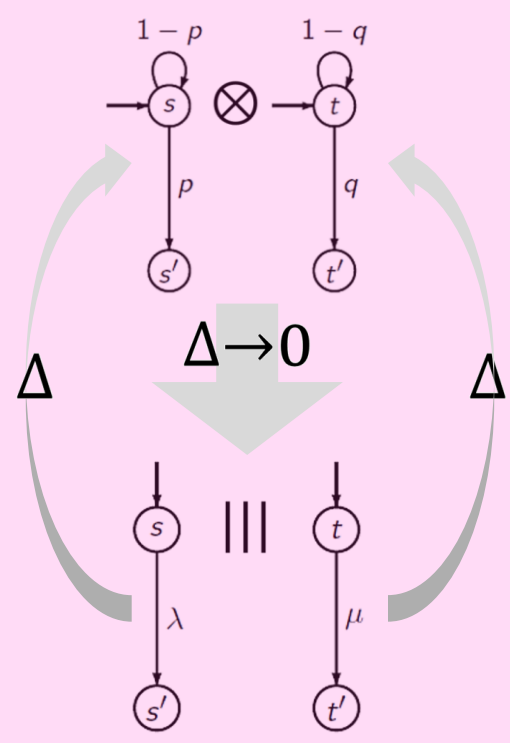
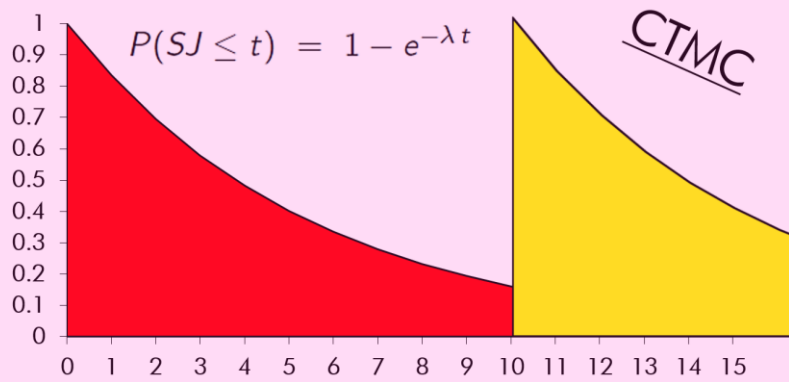
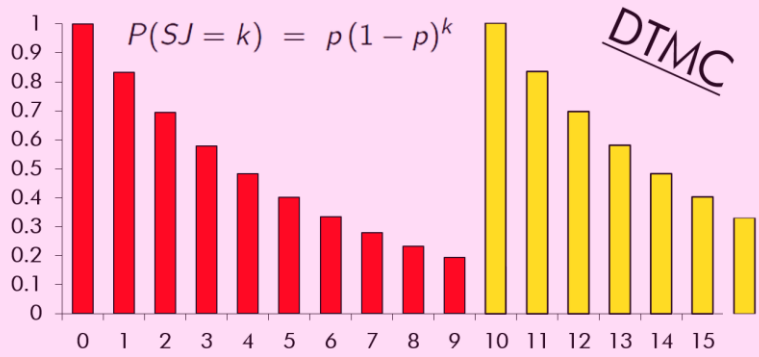
Arnd Hartmanns
University of Twente – Enschede, the Netherlands

Holger Hermanns
Saarland University – Saarbrücken, Germany
Institute of Intelligent Software – Guangzhou, China

Stochastic Model Checking

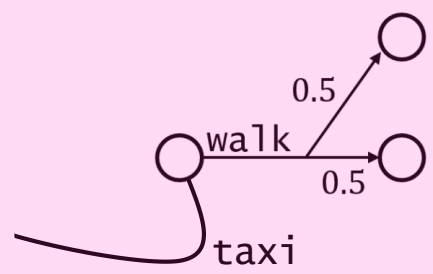
Part Two (contd.)

What Did You See Yesterday?



$\mathbb{P}_{\max}(\diamond \checkmark) = ?$

Bellman equation for MDP (\mathbb{P})



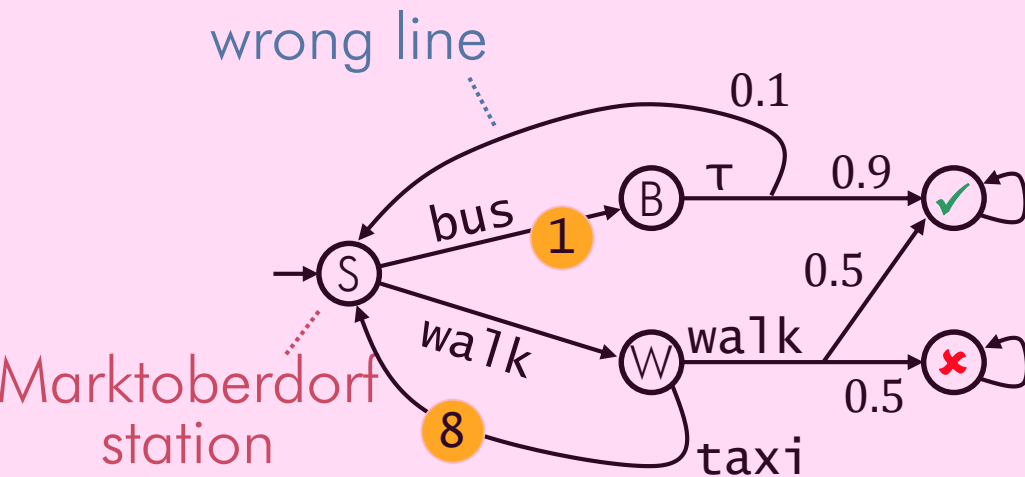
$\longrightarrow \subseteq S \times \text{Act} \times \text{Distr}(S)$

Probability distributions over states.

- $\mathbb{L}_J(up)$
- $\mathbb{P}_J(\diamond^{[t,t]} up)$
- $\mathbb{P}_J(\phi \mathcal{U}^{[t,t]} up)$
- $\mathbb{P}_J(\square^{[t,t']} up)$
- $\mathbb{L}_J(\mathbb{P}_J(\square^{[t,t']}))$
- $\mathbb{P}_J(\phi \mathcal{U}^{[t,t']} \mathbb{L}_J(up))$

Markov Decision Processes

Value iteration: dynamic programming



	S	B	W	✓	✗
$V_{\mathbb{P}}^0$	0	0	0	1	0
$V_{\mathbb{P}}^1$	0	0.9	0.5	1	0
$V_{\mathbb{P}}^2$	0.9	0.9	0.5	1	0
$V_{\mathbb{P}}^3$	0.9	0.99	0.9	1	0
$V_{\mathbb{P}}^4$	0.99	0.99	0.9	1	0

...

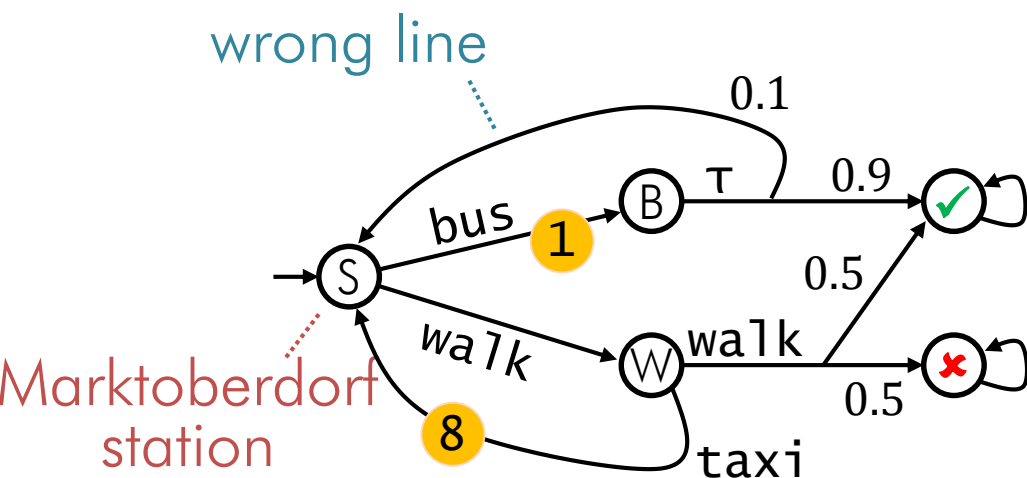
$$\mathbb{P}_{\max}(\diamond \checkmark) = ? \quad V_{\mathbb{P}}^i(s) = \max_{a \in A} \sum_{\mu \in T(s,a)} \underbrace{\mu(s')}_{\text{sum over } a\text{'s targets...}} \cdot \underbrace{V_{\mathbb{P}}^{i-1}(s')}_{\text{...of weighted target values}}$$

Bellman equation for MDP (\mathbb{P})

maximum over all actions a sum over a 's targets... ...of weighted target values

Markov Decision Processes

Value iteration: dynamic programming



	S	B	W	✓	✗
$V_{\mathbb{P}}^0$	0	0	0	1	0
$V_{\mathbb{P}}^1$	0	0.9	0.5	1	0
$V_{\mathbb{P}}^2$	0.9	0.9	0.5	1	0
$V_{\mathbb{P}}^3$	0.9	0.99	0.9	1	0
$V_{\mathbb{P}}^4$	0.99	0.99	0.9	1	0

...

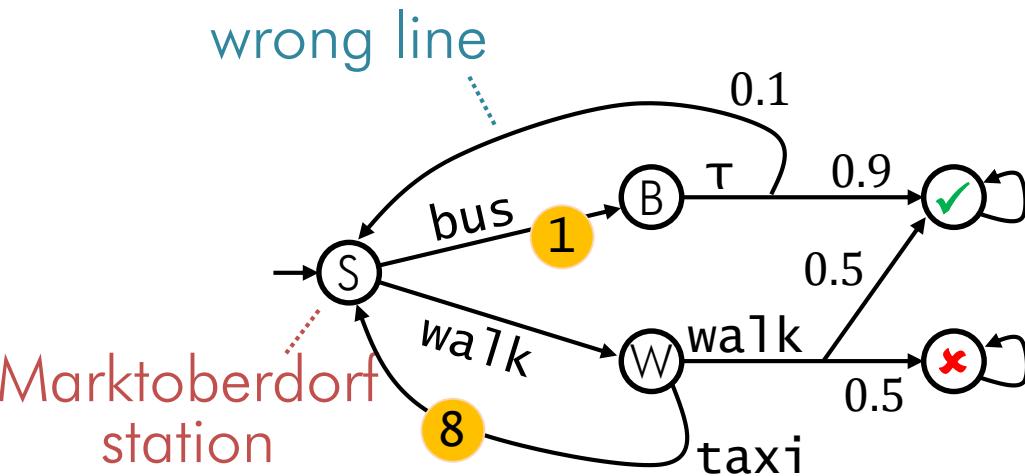
$$\mathbb{P}_{\max}(\diamond \checkmark) = ? \quad V_{\mathbb{P}}^i(s) = \max_{a \in A} \sum_{\mu \in T(s,a)} \underbrace{\mu(s')}_{\text{sum over } a\text{'s targets...}} \cdot \underbrace{V_{\mathbb{P}}^{i-1}(s')}_{\text{...of weighted target values}}$$

Bellman equation for MDP (\mathbb{P})

maximum over all actions a sum over a 's targets... ...of weighted target values

Markov Decision Processes

Value iteration: dynamic programming



	S	B	W	✓	✗
$V_{\mathbb{P}}^0$	0	0	0	1	0
$V_{\mathbb{P}}^1$	0	0.9	0.5	1	0
$V_{\mathbb{P}}^2$	0.9	0.9	0.5	1	0
$V_{\mathbb{P}}^3$	0.9	0.99	0.9	1	0
$V_{\mathbb{P}}^4$	0.99	0.99	0.9	1	0

...

$$\mathbb{P}_{\max}(\diamond \checkmark) = ?$$

Bellman equation for MDP (\mathbb{P})

$$V_{\mathbb{P}}^i(s) = \max_{\substack{a \\ s \rightarrow \mu}} \sum_{s'} \mu(s') \cdot V_{\mathbb{P}}^{i-1}(s')$$

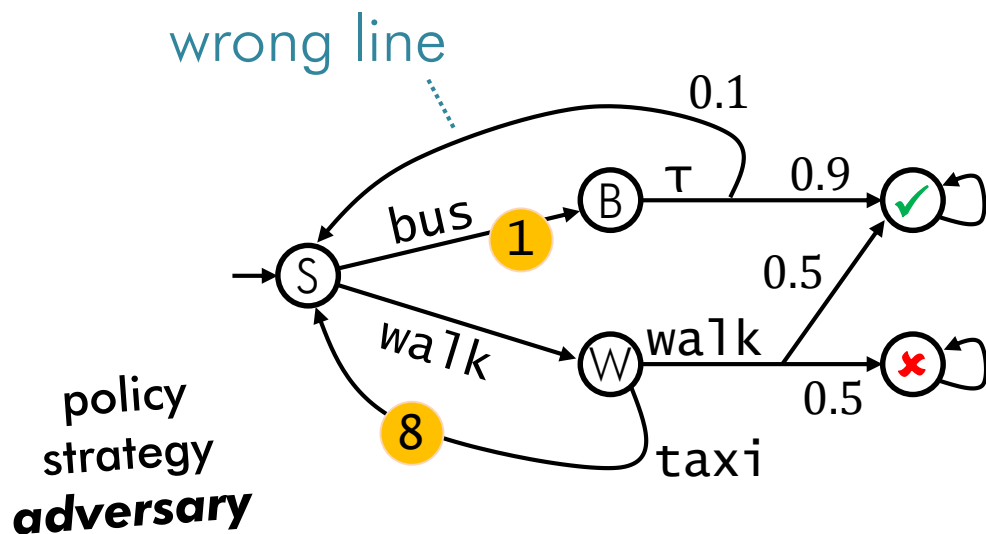
maximum over all transitions from s

$$\sum_{s'} \underbrace{\mu(s') \cdot V_{\mathbb{P}}^{i-1}(s')}_{\text{...of weighted target values}}$$

sum over μ 's targets... of weighted target values

Markov Decision Processes

Schedulers for MDP



$$V_{\mathbb{P}}(s) = \max_a \sum_{\mu \in T(s,a)} \mu(s') \cdot V_{\mathbb{P}}(s')$$

	S	B	W	✓	✗
$V_{\mathbb{P}}^0$	0	0	0	1	0
$V_{\mathbb{P}}^1$	0	0.9	0.5	1	0
$V_{\mathbb{P}}^2$	0.9	0.9	0.5	1	0
$V_{\mathbb{P}}^3$	0.9	0.99	0.9	1	0
$V_{\mathbb{P}}^4$	0.99	0.99	0.9	1	0

Memoryless: $S \rightarrow A$

Reward-positional:
 $S \times \mathbb{N} \rightarrow A$

History-dependent:
 $(S \times A)^* \times S \rightarrow A$
 or $(S \times A)^* \times S \rightarrow \text{Dist}(A)$

Scheduler \Downarrow
 $\{ S \mapsto \text{bus}, W \mapsto \text{taxi} \}$

memoryless: optimal for unbounded properties

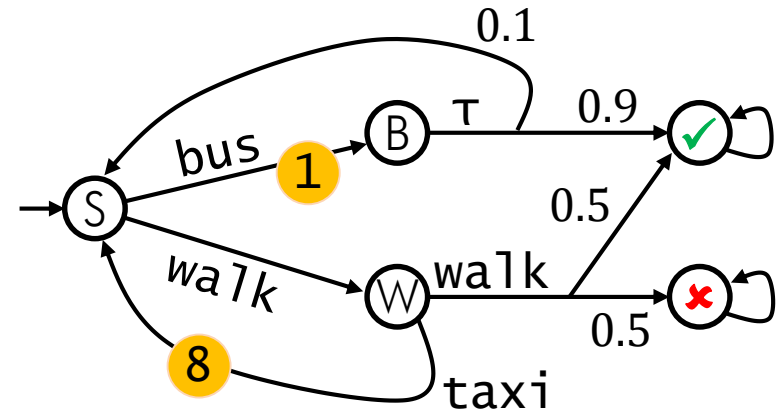
Markov Decision Processes

MDP in Modest

action `bus`, `walk`, `taxi`;

```
property PArrivedMin =  
  Pmin(<> arrived);
```

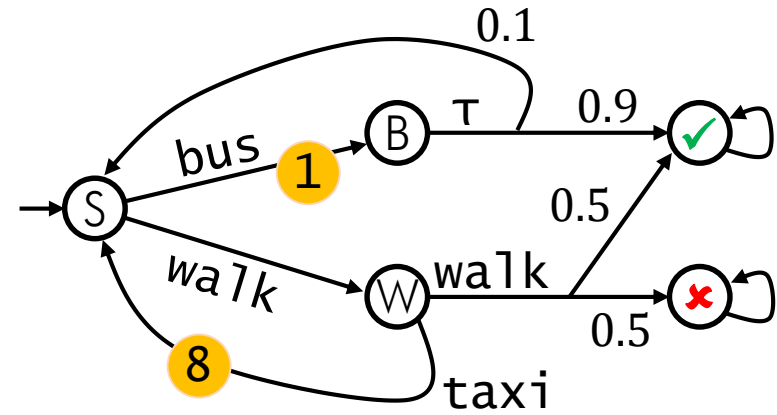
```
property PArrivedMax =  
  Pmax(<> arrived);
```



Markov Decision Processes

MDP in Modest

```
do {  
  :: bus {= cost = 1 =};  
  tau palt {  
    :0.1: {= =}  
    :0.9: {= arrived = true =}; stop  
  }  
  :: walk;  
  alt {  
    :: taxi {= cost = 8 =}  
    :: walk palt {  
      :0.5: {= arrived = true =}  
      :0.5: {= lost = true =}  
    }; stop  
  }  
}
```



alt: nondeterministic
choice of action

palt: probabilistic
choice of target
of chosen action

⇒ Try it with **mcsta**
and
get optimal strategies.

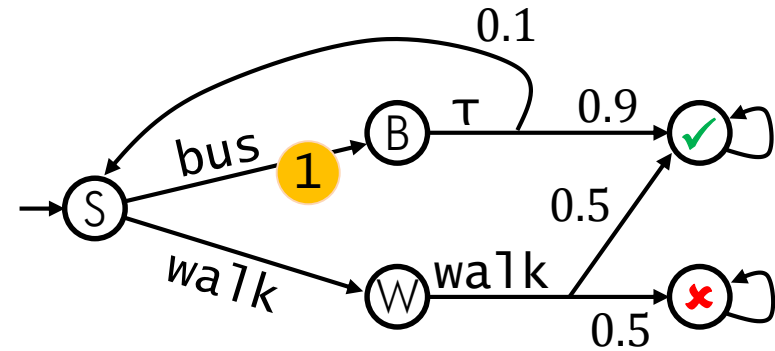
Markov Decision Processes

Bounded properties

$$\mathbb{P}_{\max}(\diamond \checkmark \wedge \text{🟡} \leq 0) =$$

$$\mathbb{P}_{\max}(\diamond \checkmark \wedge \text{🟡} \leq 1) =$$

$$\mathbb{P}_{\max}(\diamond \checkmark \wedge \text{🟡} \leq 2) =$$



Memoryless scheduler do not suffice,
need **reward-positional** schedulers!

$$S \times \mathbb{N} \rightarrow A$$

...and clever algorithms to compute values efficiently

A Comparison of Time- and Reward-Bounded
Probabilistic Model Checking Techniques*

Multi-Cost Bounded Reachability in MDP*

Hartmanns, Junges, Katoen, Quatmann: Multi-Cost
Bounded Reachability in MDP (TACAS 2018)

Hahn, Hartmanns: A Comparison of Time and Reward
Bounded Probabilistic Model Checking Techniques
(SETTA 2016)

¹ Institut
²

Abstract
concerning
are of cent
quirements

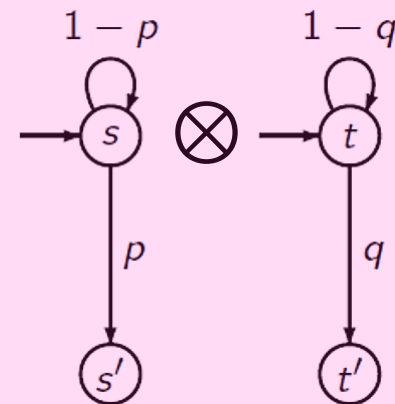
Arnd Hartmanns¹, Sebastian Junges²,
Joost-Pieter Katoen^{1,2}, and Tim Quatmann²

¹ University of Twente, Enschede, The Netherlands
² RWTH Aachen University, Aachen, Germany

Composition Operators on Discrete-Time Markov Chains

Only the synchronous product makes sense.

$$\frac{s_1 \xrightarrow{p} s'_1 \wedge s_2 \xrightarrow{q} s'_2}{\langle s_1, s_2 \rangle \xrightarrow{pq} \langle s'_1, s'_2 \rangle}$$



The MCs proceed in lock-step through discrete time.

What happens to the probabilities?

Synchronous Product

for parallel systems with fully synchronized processes

given TS $\mathcal{T}_1 = (S_1, \text{Act}_1, \longrightarrow_1, \dots)$,
 $\mathcal{T}_2 = (S_2, \text{Act}_2, \longrightarrow_2, \dots)$

$$\mathcal{T}_1 \otimes \mathcal{T}_2 = (S_1 \times S_2, \text{Act}, \longrightarrow, \dots)$$

$$\frac{s_1 \xrightarrow{\alpha} s'_1 \wedge s_2 \xrightarrow{\beta} s'_2}{\langle s_1, s_2 \rangle \xrightarrow{\alpha * \beta} \langle s'_1, s'_2 \rangle}$$

$\text{Act}_1 \times \text{Act}_2 \longrightarrow \text{Act}$
 $(\alpha, \beta) \mapsto \alpha * \beta$

Composition Operators on Markov Decision Processes

Anything goes!

$$\frac{s_1 \xrightarrow{\alpha}_1 \mathbf{P}_1}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \mathbf{Q}_1}$$

$$\frac{s_2 \xrightarrow{\alpha}_2 \mathbf{P}_2}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \mathbf{Q}_2}$$

where $\mathbf{Q}_1(\langle s'_1, s'_2 \rangle) = \mathbf{P}_1(s'_1)$

if $s'_2 = s_2$ and 0 otherwise,

and $\mathbf{Q}_2(\langle s'_1, s'_2 \rangle) = \dots$

Concurrency – Nondeterminism – Interleaving

$$\mathcal{T}_1 = (S_1, Act_1, \xrightarrow{\cdot}_1, S_{0,1})$$

$$\mathcal{T}_2 = (S_2, Act_2, \xrightarrow{\cdot}_2, S_{0,2})$$

$$\mathcal{T}_1 \parallel \mathcal{T}_2 = (S_1 \times S_2, Act_1 \cup Act_2, \xrightarrow{\cdot}, S_{0,1} \times S_{0,2})$$

where the transition relation $\xrightarrow{\cdot}$ is given by:

$$\frac{s_1 \xrightarrow{\alpha}_1 s'_1}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s'_1, s_2 \rangle}$$

only \mathcal{T}_1 moves

$$\frac{s_2 \xrightarrow{\alpha}_2 s'_2}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1, s'_2 \rangle}$$

only \mathcal{T}_2 moves

Composition Operators on Markov Decision Processes

Anything goes!

$$\frac{s_1 \xrightarrow{\alpha}_1 \mathbf{P}_1}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \mathbf{Q}_1}$$

$$\frac{s_2 \xrightarrow{\alpha}_2 \mathbf{P}_2}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \mathbf{Q}_2}$$

where $\mathbf{Q}_1(\langle s'_1, s'_2 \rangle) = \mathbf{P}_1(s'_1)$

if $s'_2 = s_2$ and 0 otherwise,

and $\mathbf{Q}_2(\langle s'_1, s'_2 \rangle) = \dots$

$$\frac{s_1 \xrightarrow{\alpha}_1 \mathbf{P}_1 \quad \wedge \quad s_2 \xrightarrow{\alpha}_2 \mathbf{P}_2}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \mathbf{P}_1 \mathbf{P}_2}$$

where $\mathbf{P}_1 \mathbf{P}_2(\langle s'_1, s'_2 \rangle) = \mathbf{P}_1(s'_1) \mathbf{P}_2(s'_2)$.

Handshake Communication

$\mathcal{T}_1 = (S_1, \text{Act}_1, \rightarrow_1, \dots)$, $\mathcal{T}_2 = (S_2, \text{Act}_2, \rightarrow_2, \dots)$

$\text{Syn} \subseteq \text{Act}_1 \cap \text{Act}_2$ (set of synchronization actions)

$\mathcal{T}_1 \parallel_{\text{Syn}} \mathcal{T}_2 = (S_1 \times S_2, \text{Act}_1 \cup \text{Act}_2, \rightarrow, \dots)$

interleaving for every action $\alpha \in \text{Act}_i \setminus \text{Syn}$:

$$\frac{s_1 \xrightarrow{\alpha}_1 s'_1}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s'_1, s_2 \rangle}$$

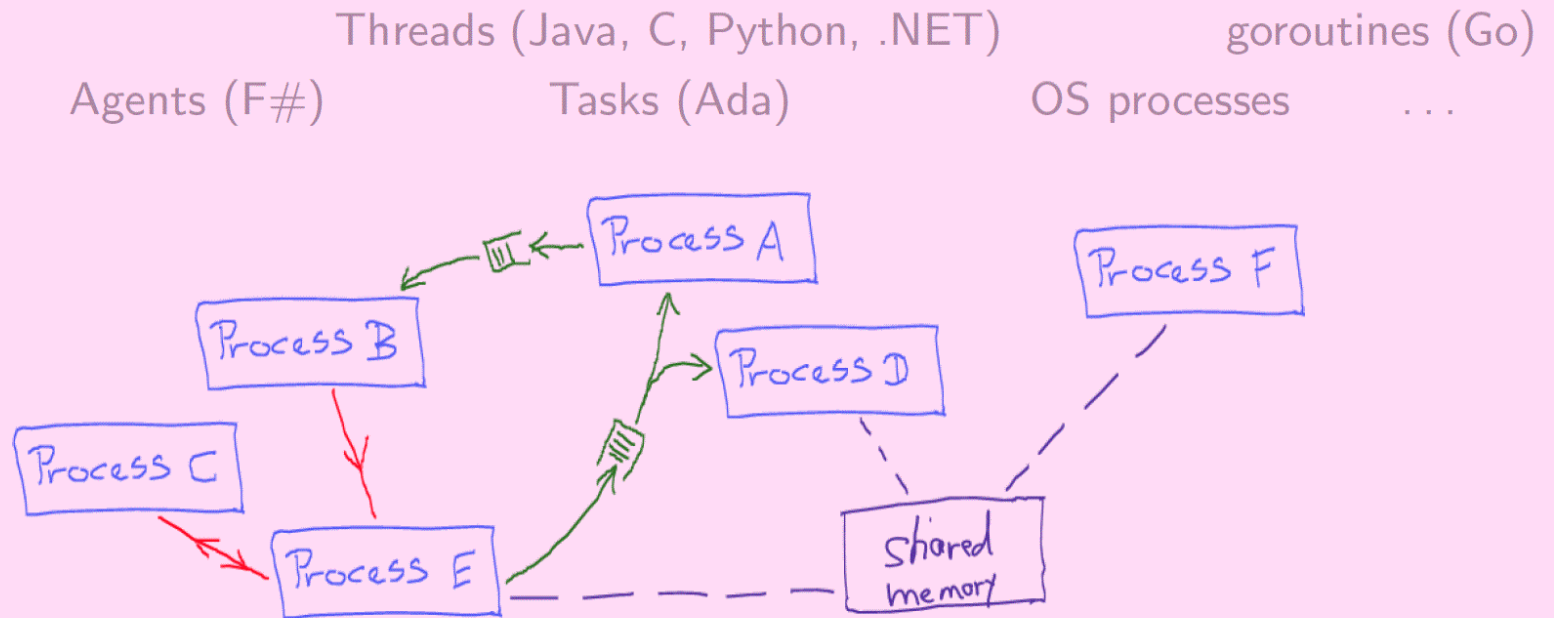
handshaking (rendezvous) for $\alpha \in \text{Syn}$:

$$\frac{s_1 \xrightarrow{\alpha}_1 s'_1 \quad \wedge \quad s_2 \xrightarrow{\alpha}_2 s'_2}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s'_1, s'_2 \rangle}$$

only \mathcal{T}_1 moves

only \mathcal{T}_2 moves

What is out there?



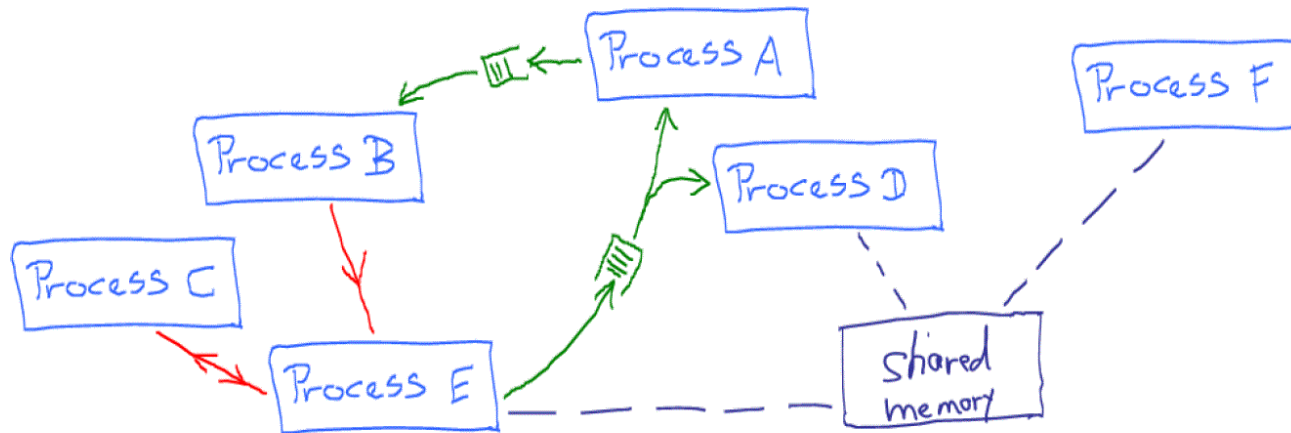
Base principles:

- Communication via shared variables
in shared memory
- Synchronous communication via messages
Channels with capacity 0
- Asynchronous communication via messages
Channels with capacity > 0 (Buffers!)

- ... can all be encoded into
- handshake communication + auxiliary LTS

What is out there?

Threads (Java, C, Python, .NET) goroutines (Go)
Agents (F#) Tasks (Ada) OS processes ...



Base principles:

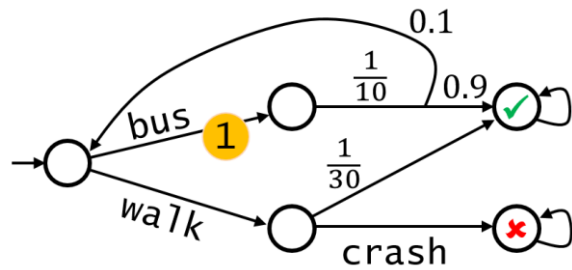
- Communication via shared variables
in shared memory
- Synchronous communication via messages
Channels with capacity 0
- Asynchronous communication via messages
Channels with capacity > 0 (Buffers!)

shared memory

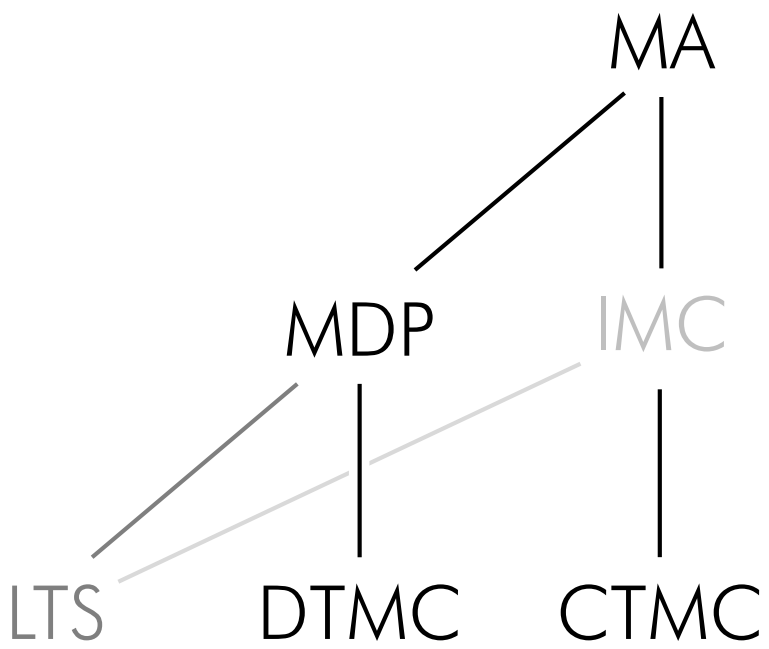
message passing

- ... can all be encoded into
- handshake communication
- + auxiliary
- MDP

Markov Automata



Markov Automata



Sojourn time distributions are geometric.

Sojourn time distributions are exponential.

$P(SJ \leq t) = 1 - e^{-\lambda t}$

Markov Decision Processes

The main difference: $\rightarrow \subseteq S \times Act \times Distr(S)$

Probability distributions over states

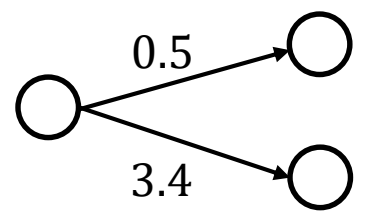
Concurrency Modelling Primer

Labelled transition systems:

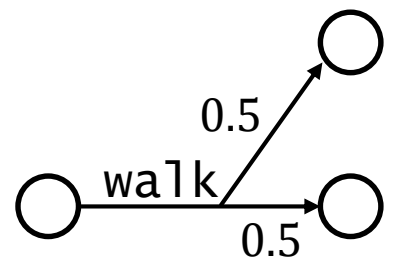
- states S
- actions Act
- transitions $\rightarrow \subseteq S \times Act \times S$
- state (set)

Markov Automata

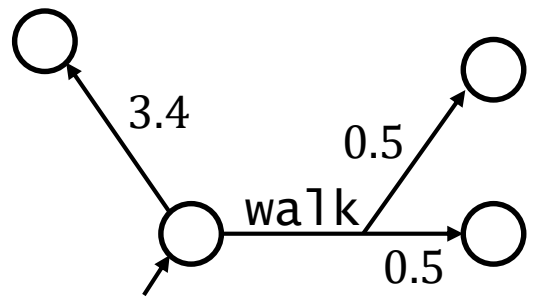
Two relations:



plus



plus "maximal progress"



Sojourn time distributions are geometric.

Sojourn time distributions are exponential.

CTMC

Markov Decision Processes

The main difference: $\rightarrow \subseteq S \times \text{Act} \times \text{Distr}(S)$

Probability distributions over states

```
graph LR; S(( )) -- walk --> T1(( )); S -- 0.5 --> T2(( ))
```

Concurrency Modelling Primer

Labelled transition systems:

- states S
- actions Act
- transitions $\rightarrow \subseteq S \times \text{Act} \times S$

state (set)

```
graph TD; S(( )) -- get_coke --> pay((pay))
```

Composition Operators on Markov Decision Processes

Anything goes!

$$\frac{s_1 \xrightarrow{\alpha}_1 \mathbf{P}_1}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \mathbf{Q}_1}$$

$$\frac{s_2 \xrightarrow{\alpha}_2 \mathbf{P}_2}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \mathbf{Q}_2}$$

where $\mathbf{Q}_1(\langle s'_1, s'_2 \rangle) = \mathbf{P}_1(s'_1)$

if $s'_2 = s_2$ and 0 otherwise,

and $\mathbf{Q}_2(\langle s'_1, s'_2 \rangle) = \dots$

$$\frac{s_1 \xrightarrow{\alpha}_1 \mathbf{P}_1 \quad \wedge \quad s_2 \xrightarrow{\alpha}_2 \mathbf{P}_2}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \mathbf{P}_1 \mathbf{P}_2}$$

where $\mathbf{P}_1 \mathbf{P}_2(\langle s'_1, s'_2 \rangle) = \mathbf{P}_1(s'_1) \mathbf{P}_2(s'_2)$.

Handshake Communication

$\mathcal{T}_1 = (S_1, \text{Act}_1, \rightarrow_1, \dots)$, $\mathcal{T}_2 = (S_2, \text{Act}_2, \rightarrow_2, \dots)$

$\text{Syn} \subseteq \text{Act}_1 \cap \text{Act}_2$ (set of synchronization actions)

$\mathcal{T}_1 \parallel_{\text{Syn}} \mathcal{T}_2 = (S_1 \times S_2, \text{Act}_1 \cup \text{Act}_2, \rightarrow, \dots)$

interleaving for every action $\alpha \in \text{Act}_i \setminus \text{Syn}$:

$$\frac{s_1 \xrightarrow{\alpha}_1 s'_1}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s'_1, s_2 \rangle}$$

$$\frac{s_2 \xrightarrow{\alpha}_2 s'_2}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1, s'_2 \rangle}$$

handshaking (rendevvous) for $\alpha \in \text{Syn}$:

$$\frac{s_1 \xrightarrow{\alpha}_1 s'_1 \quad \wedge \quad s_2 \xrightarrow{\alpha}_2 s'_2}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s'_1, s'_2 \rangle}$$

only \mathcal{T}_1 moves

only \mathcal{T}_2 moves

Composition Operators on Markov Automata

for $\alpha \notin \text{Syn}$:

$$\frac{s_1 \xrightarrow{\alpha}_1 P_1}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} Q_1} \quad \frac{s_2 \xrightarrow{\alpha}_2 P_2}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} Q_2}$$

$$\frac{s_1 \xrightarrow{\lambda}_1 s'_1}{\langle s_1, s_2 \rangle \xrightarrow{\lambda} \langle s'_1, s_2 \rangle}$$

for $\alpha \in \text{Syn}$:

$$\frac{s_1 \xrightarrow{\alpha}_1 P_1 \wedge s_2 \xrightarrow{\alpha}_2 P_2}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} P_1 P_2}$$

$$\frac{s_2 \xrightarrow{\mu}_2 s'_2}{\langle s_1, s_2 \rangle \xrightarrow{\mu} \langle s_1, s'_2 \rangle}$$

Anything goes, too!

Composition of Continuous-Time Markov Chains

Only the interleaving operator makes sense.

$$\frac{s_1 \xrightarrow{\lambda}_1 s'_1}{\langle s_1, s_2 \rangle \xrightarrow{\lambda} \langle s'_1, s_2 \rangle}$$

$$\frac{s_2 \xrightarrow{\mu}_2 s'_2}{\langle s_1, s_2 \rangle \xrightarrow{\mu} \langle s_1, s'_2 \rangle}$$

The MCs proceed in independently through continuous time.



What happens...

Composition Operators on Markov Decision Processes

Anything goes!

$$\frac{s_1 \xrightarrow{\alpha}_1 P_1}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} Q_1}$$

$$\frac{s_2 \xrightarrow{\alpha}_2 P_2}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} Q_2}$$

Concurrency – Nondete

$$\begin{aligned} T_1 &= (S_1, Act_1, \dots) \\ T_2 &= (S_2, Act_2, \dots) \end{aligned}$$

$$T_1 \parallel T_2 = (S_1 \times S_2, Act_1 \cup Act_2)$$

$$\frac{s_1 \xrightarrow{\alpha}_1 s'_1}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s'_1, s_2 \rangle}$$

only T_1 moves

where $Q_1(\langle s'_1, s'_2 \rangle) = P_1(s'_1)$
if $s'_2 = s_2$ and 0 otherwise,
and $Q_2(\langle s'_1, s'_2 \rangle) = \dots$

$$\frac{s_1 \xrightarrow{\alpha}_1 P_1 \wedge s_2 \xrightarrow{\alpha}_2 P_2}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} P_1 P_2}$$

where $P_1 P_2(\langle s'_1, s'_2 \rangle) = P_1(s'_1) P_2(s'_2)$.

Handshake Communication

$T_1 = (S_1, Act_1, \rightarrow_1, \dots)$, $T_2 = (S_2, Act_2, \rightarrow_2, \dots)$

$\text{Syn} \subseteq Act_1 \cap Act_2$ (set of synchronization actions)

$T_1 \parallel_{\text{Syn}} T_2 = (S_1 \times S_2, Act_1 \cup Act_2, \rightarrow)$

interleaving for every action $\alpha \in Act_i \setminus \text{Syn}$

$$\frac{s_1 \xrightarrow{\alpha}_1 s'_1}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s'_1, s_2 \rangle} \quad \frac{s_2 \xrightarrow{\alpha}_2 s'_2}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1, s'_2 \rangle}$$

handshaking (rendezvous) for $\alpha \in \text{Syn}$:

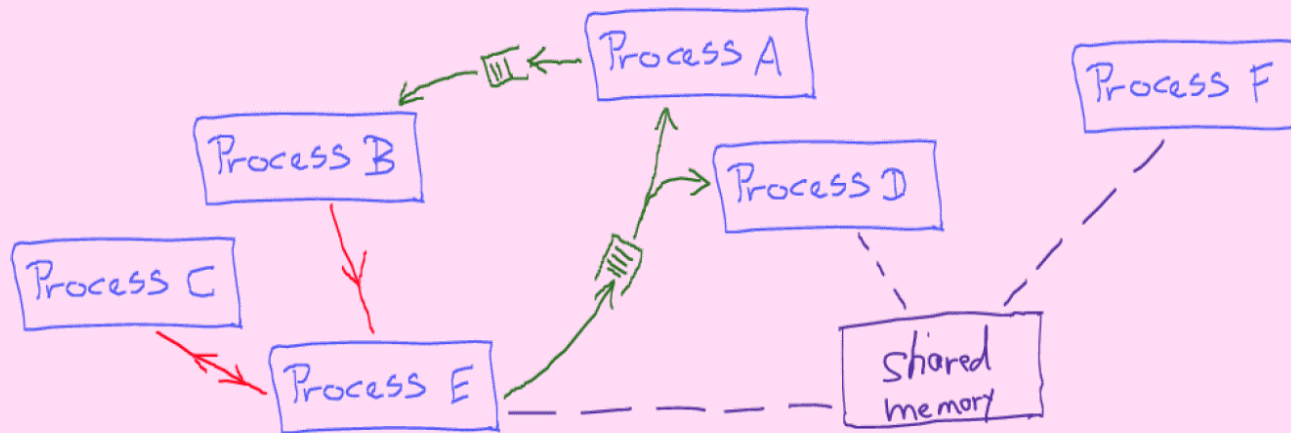
$$\frac{s_1 \xrightarrow{\alpha}_1 s'_1 \wedge s_2 \xrightarrow{\alpha}_2 s'_2}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s'_1, s'_2 \rangle}$$

only T_1 moves

only T_2 moves

What is out there?

Threads (Java, C, Python, .NET) goroutines (Go)
Agents (F#) Tasks (Ada) OS processes ...



Base principles:

- Communication via shared variables
in shared memory
- Synchronous communication via messages
Channels with capacity 0
- Asynchronous communication via messages
Channels with capacity > 0 (Buffers!)

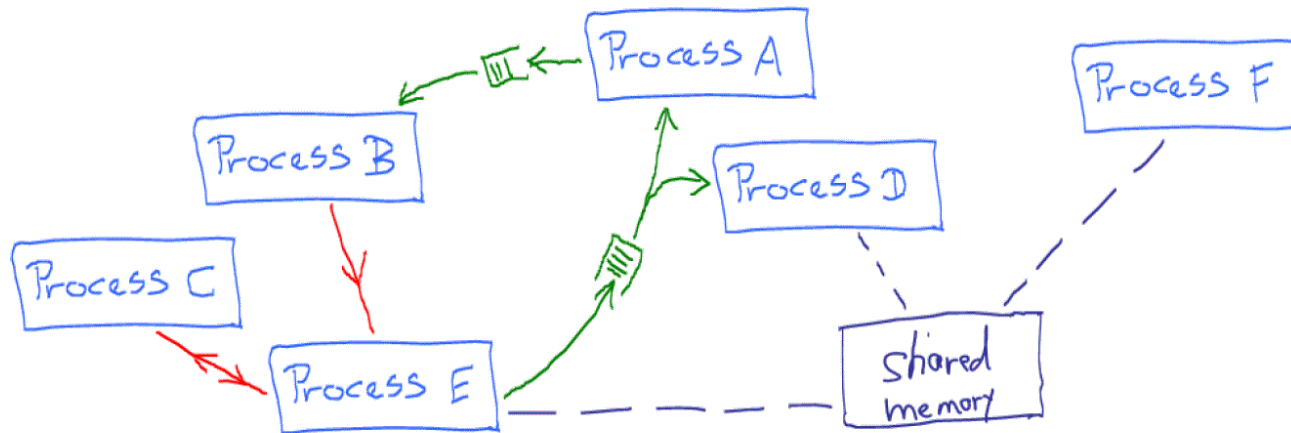
shared memory

message passing

- ... can all be encoded into
- handshake communication
- + auxiliary
- MDP

What is out there?

Threads (Java, C, Python, .NET) goroutines (Go)
Agents (F#) Tasks (Ada) OS processes ...



Base principles:

- Communication via shared variables
in shared memory
- Synchronous communication via messages
Channels with capacity 0
- Asynchronous communication via messages
Channels with capacity > 0 (Buffers!)

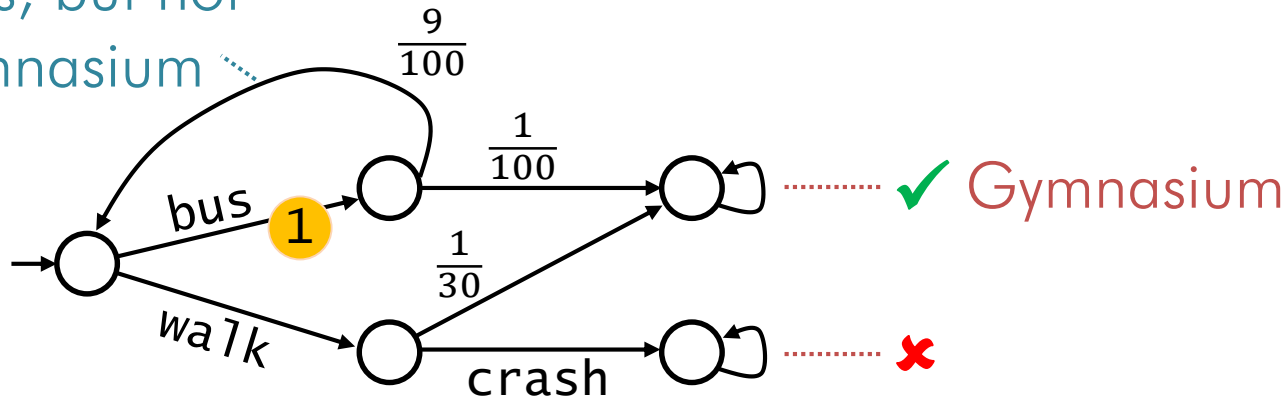
shared memory

message passing

- ... can all be encoded into
- handshake communication
- + auxiliary
- MA

Orthogonal combination of MDP and CTMC

bus ends, but not
at Gymnasium



*maximal
progress*

Action transitions might happen immediately.

$$\mathbb{P}(\diamond \checkmark) = ?$$

$$\mathbb{E}(\text{🟡 to } \checkmark \vee \text{✗}) = ?$$

$$\mathbb{E}(\text{🕒 to } \checkmark \vee \text{✗}) = ?$$

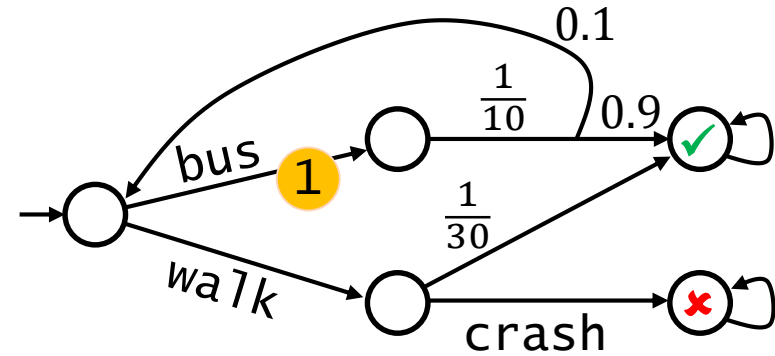
⇒ probabilities, time and costs in a single model!

$$\mathbb{E}(\text{🕒 to } (\checkmark \vee \text{✗}) \wedge \text{🟡} \leq 2)$$

Markov Automata

MA in Modest

```
do {  
  :: bus {= cost = 1 =};  
  rate(1/10) tau palt {  
    :0.1: {= =}  
    :0.9: {= arrived = true =};  
    stop  
  }  
  :: walk;  
  alt {  
    :: rate(1/30) tau {= arrived = true =}  
    :: crash  
  };  
  stop  
}
```

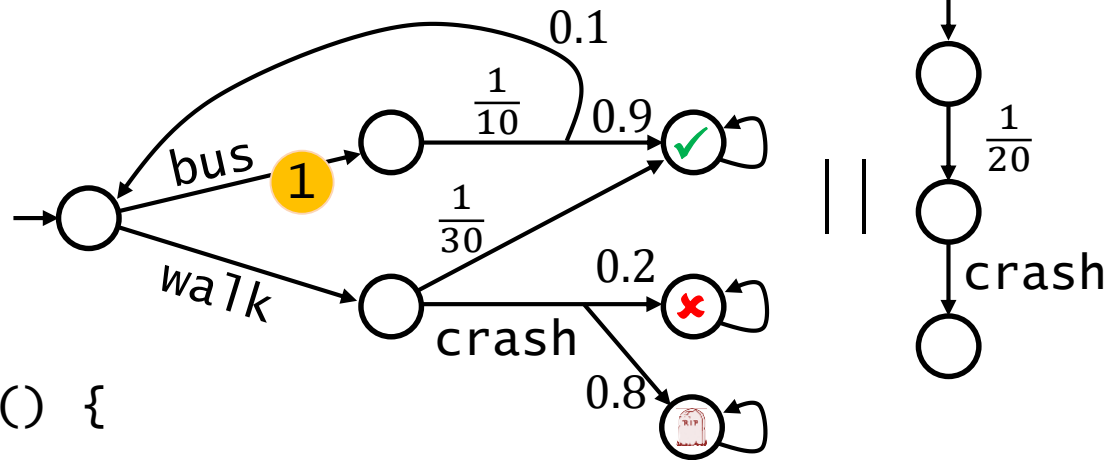


Compositional modelling in Modest

```
action crash;  
process Student() {  
  ... }  
}
```

```
process AutonomousCar() {  
  rate(1 / 20) tau;  
  crash  
}
```

```
par {  
  :: Student()  
  :: AutonomousCar()  
}
```



⇒ Try it with `mcsta`.

Markov Automata Model Checking

Long-run/unbounded properties } use the embedded MDP
Transition reward bounds } \Rightarrow MDP model checking

Time bounds or reward bounds accumulated over time:

Different approximation methods available.

Hatefi, H.: Model Checking Algorithms for Markov Automata (ECEASST 53, 2012)

Butkova, Hartmanns, H.: A Modest Approach to Modelling and Checking Markov Automata (QEST 2019)

Guck, Hatefi, H., Katoen, Timmer: Analysis of Timed and Long-Run Objectives for Markov Automata. Logical Methods in Computer Science 10(3) (2014)

Yuliya Butkova, Ralf Wimmer, Holger Hermanns: Long-Run Rewards for Markov Automata. (TACAS 2017)

Model Checking Algorithms for Markov Automata *
Hassan Hatefi¹ and Holger Hermanns²

ECEASST

A Modest Approach to Modelling and Checking Markov Automata*

Yuliya Butkova¹, Arnd Hartmanns², and Holger Hermanns¹
¹ Saarland University, Saarbrücken, Germany
² University of Bayreuth, Bayreuth, Germany

Logical Methods in Computer Science
Vol. 10(3:17)2014, pp. 1–29
www.lmcs-online.org

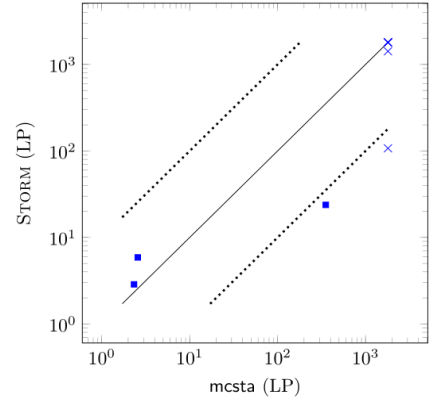
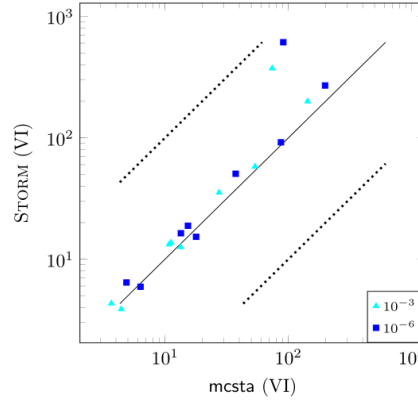
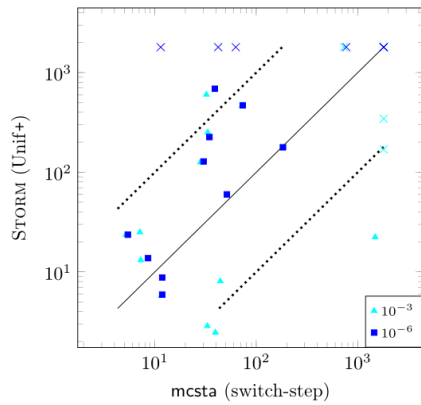
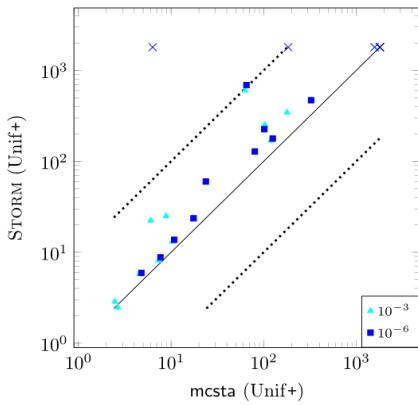
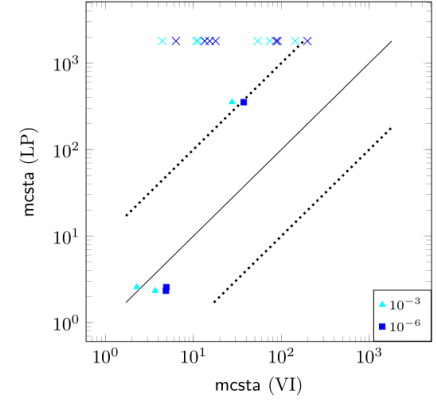
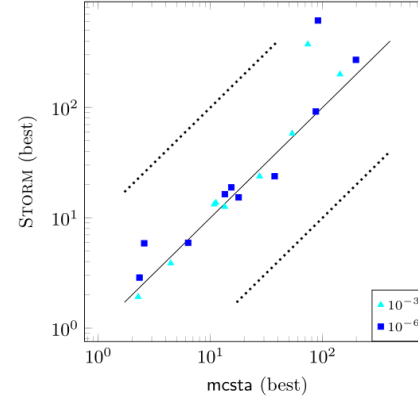
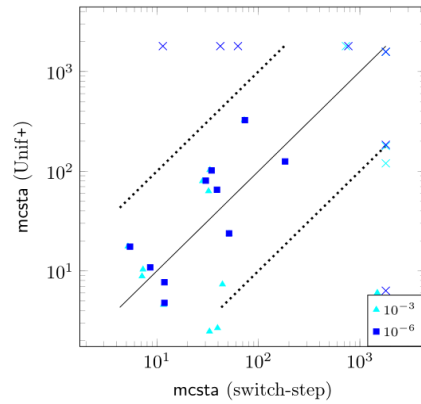
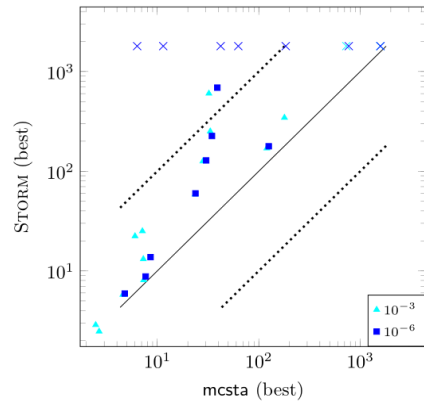
ANALYSIS OF TIMED AND LONG-RUN OBJECTIVES FOR MARKOV AUTOMATA *

DENNIS GUCK^a, HASSAN HATEFI^b, HOLGER HERMANNSS^c,
JOOST-PIETER KATOEN^d, AND MARK TIMMER^e

Long-Run Rewards for Markov Automata

Yuliya Butkova¹ (✉), Ralf Wimmer², and Holger Hermanns¹
¹ Saarland University, Saarbrücken, Germany
² University of Bayreuth, Bayreuth, Germany

Comparisons to STORM – another MA Model Checker



Time-bounded properties

Long-run average reward properties

In practice, the expressiveness of MA rarely harms analysis, but it eases modelling.

A Modest Approach to Modelling and Checking Markov Automata*

Yuliya Butkova¹, Arnd Hartmanns², and Holger Hermanns²

¹ Saarland University, Saarbrücken, Germany
² University of Bayreuth, Bayreuth, Germany

Butkova, Hartmanns, H.: A Modest Approach to Modelling and Checking Markov Automata (QEST 2019)

Markov Automata

Case study: an **attack on Bitcoin**

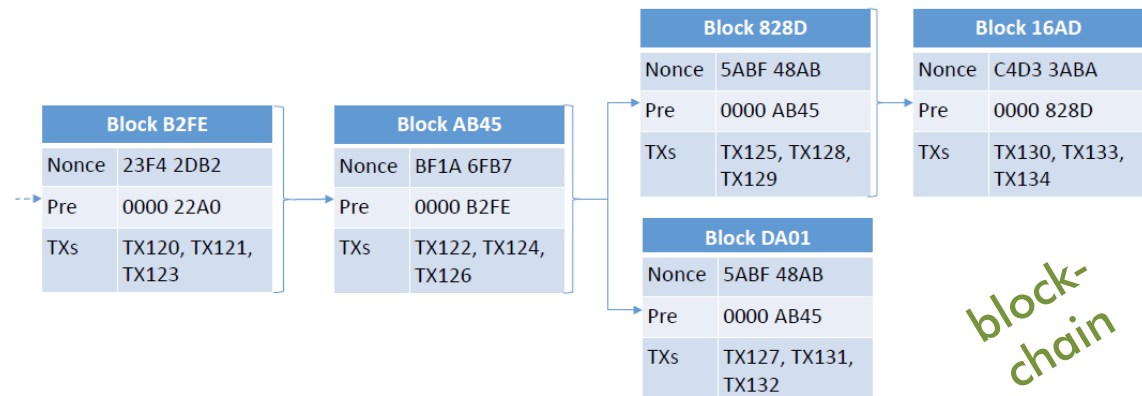
- every transaction recorded in block
- every block hashes the previous block
- longest chain of blocks is authoritative
- transaction confirmed when 6 blocks in

CD=6



self-adaptive

Hash rate: one new block discovered every 12 minutes



block-chain

Markov Automata

Case study: an **attack** on Bitcoin

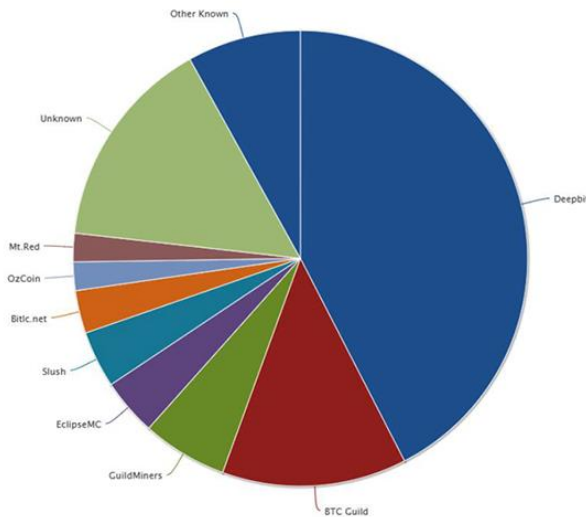
- every transaction recorded in block
- every block hashes the previous block
- longest chain of blocks is authoritative
- transaction confirmed when 6 blocks in

CD=6

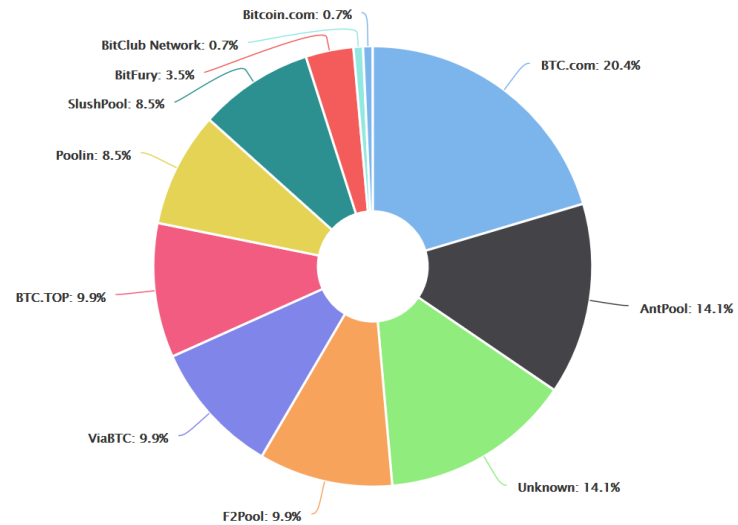


self-adaptive

What if **we** control a large amount of the hash rate?



2012



2019

Source: blockchain.com

Markov Automata

Case study: an **attack on Bitcoin**

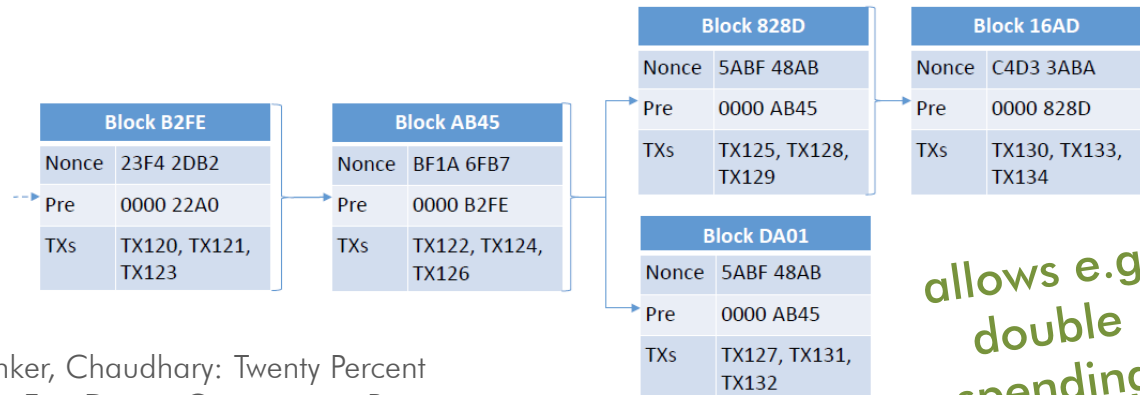
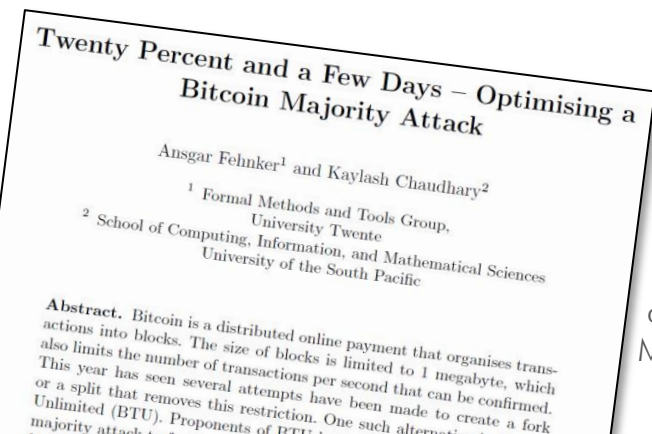
Attack: *fork* the chain, make fork longest

⇒ need to be lucky and create
6 blocks faster than everyone else

Assume we control 20% of the global hash rate.



20% of
"every
12 min"



Fehnker, Chaudhary: Twenty Percent and a Few Days – Optimising a Bitcoin Majority Attack (NASA Formal Methods 2018)

allows e.g.
double
spending

Case study: an **attack on Bitcoin**

Attack: *fork* the chain, make fork longest

⇒ need to be lucky and create
6 blocks faster than everyone else

Assume we control 20% of the global hash rate.

- One new block every 12 minutes.
- 20% of that rate is ours, 80% is "honest".
- We try to fork the chain,
make fork 6 blocks long, and
longer than the parallel honest chain.
- When the "honest" chain gets longer, do we
give up and start a new fork, or continue?
- How long until we win?



20% of
"every
12 min"

Modelling – the Honest Pool

```
const real M; // fraction of hash rate controlled by malicious mining pool
action sln; // indicates that the honest pool mined a new block

process HonestPool()
{
    rate(1/12 * (1 - M)) tau; // wait 12 / (1 - M) minutes on average
    sln; // signal that a new block was found
    HonestPool() // repeat
}
```

Handshaking communication

(abstracts from communication delays)

```
par {
    :: HonestPool()
    :: DoubleSpendingAttacker()
}
```

*Wants to spend coins again,
despite announcing transaction.*

Modelling – The DoubleSpendingAttacker

```
const int CD;           // confirmation depth required by victim
const int DB = CD;     // attacker gives up when this far behind
action cnt;           // indicates that the attacker continues
int(0..CD+1) m_len;    // length of the secret fork
int(-DB..CD+1) m_diff = 0; // length of secret fork minus honest fork
bool gup;

process DoubleSpendingAttacker()
{
  do {
    :: rate(1/12 * M) {= m_len = min(CD+1, m_len + 1), m_diff++ =}
    :: sln {= m_diff-- =}; // public fork extended
    if(m_diff <= -DB) { tau {= gup = true =}; stop } // give up
    else { cnt } // continue
  }
}
```

Wants to spend coins again,
despite announcing transaction.

Evaluating the Attack

```
function bool win() = m_len > CD && m_diff > 0; // winning condition
property P_Win = Pmin(<> win()); // attacker wins
property P_GiveUp = Pmin(!win() U gup); // attacker gives up
```

```
./modest mcsta bitcoin-ds.modest -E "M = 0.2, CD = 6"
```

```
+ Property P_Win
```

```
Probability: 0.008693946907961582
```

```
+ Property P_GiveUp
```

```
Probability: 0.9913060345101223
```

Modelling – The TrustAttacker

```
action rst; // indicates that the attacker restarts from the public fork
process TrustAttacker()
{
  do {
    :: rate((1/12) * M) {= m_len = min(CD, m_len + 1), m_diff++ =}
    :: sln {= m_diff-- =}; // public fork extended
    alt { // strategy choice: restart or continue malicious fork
      :: rst {= m_len = 0, m_diff = 0 =} // can always restart
      :: when(m_diff > -DB) cnt // can continue if not too far behind
    }
  }
}
```

Now: reach CD first.

```
function bool win() = m_len >= CD && m_diff > 0;
property P_Win = Pmax(<> win());
```

```
property T_WinMin = Xmin(T, win());
```

Wants to damage trust in system.

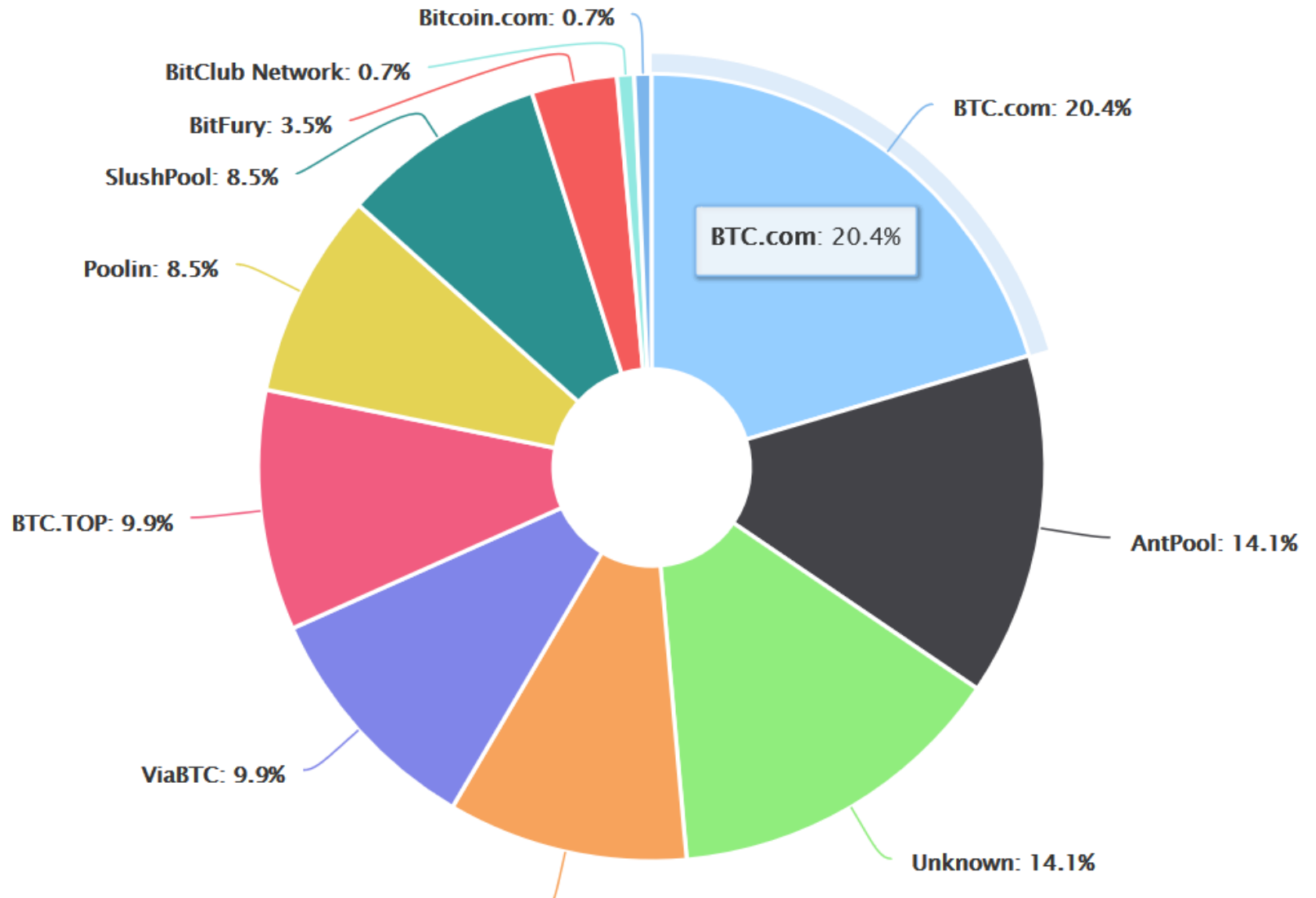
Analysis – The TrustAttacker

- + Property P_Win
Probability: 0.999999999999999994
- + Property T_WinMin
Value: 3736.59105869273
- + Property P_WinMax2
Probability: 0.535100786178178

```
function bool win() = m_len >= CD && m_diff > 0;  
property P_Win = Pmax(<> win());
```

```
property T_WinMin = Xmin(T, win());  
property P_WinMax2 = Pmax(<> [T<=2880] ( win()));
```

Current statistics



Source: blockchain.com

Modelling – The TrustAttacker

```
+ Property P_Win  
  Probability: 0.999999999999999994  
  
+ Property T_WinMin  
  Value: 3736.59105869273  
  
+ Property P_WinMax2  
  Probability: 0.535100786178178
```

```
function bool win() = m_len >= CD && m_diff > 0;  
property P_Win = Pmax(<> win());
```

```
property T_WinMin = Xmin(T, win());  
property P_WinMax2 = Pmax(<> [T<=2880] ( win()));
```

Strategy – The TrustAttacker

```
./modest mcsta bitcoin-attack.modest -E "M=0.2,CD=6" --scheduler sched.txt
```

```
•  
•  
•  
+ State: (HonestPool.location = loc_1, TrustAttacker.location = loc_10,  
          m_len = 1, m_diff = -2)  
Choice: rst
```

- ⇒ Your Homework: spell out the strategy!
- ⇒ Or: Extend by adding power costs!
- Or: Extend by communication delays!

Strategy – The TrustAttacker

```
./modest mcsta bitcoin-attack.modest -E "M=0.2,0
```

```
•  
•  
•  
+ State: (HonestPool.location = loc_1, TrustAtta  
m_len = 1, m_diff = -2)  
Choice: rst
```

- ⇒ Your Homework: spell out the strategy!
- ⇒ Or: Extend by adding power costs!
- Or: Extend by communication delays!

m_len	m_diff	choice
0	-1	rst
1	-2	rst
1	-1	cnt
1	0	cnt
2	-3	rst
2	-2	cnt
2	-1	cnt
2	0	cnt
2	1	cnt
3	-3	rst
3	-2	cnt
3	-1	cnt
3	0	cnt
3	1	cnt
3	2	cnt
4	-3	rst
4	-2	cnt
4	-1	cnt
4	0	cnt
4	1	cnt
4	2	cnt
4	3	cnt
5	-3	rst
5	-2	cnt
5	-1	cnt
5	0	cnt
5	1	cnt
5	2	cnt
5	3	cnt
5	4	cnt
6	-3	rst
6	-2	cnt
6	-1	cnt

txt

Where we stand

Discrete-time Markov chains:
discrete probabilistic choices

Continuous-time Markov chains:
continuous stochastic time

Markov decision processes:
decisions and uncertainty

Markov automata:
MDP plus CTMC,
compositionally

Probabilistic timed automata:
MDP plus hard real time

