

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

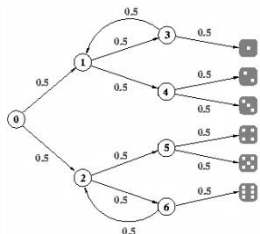
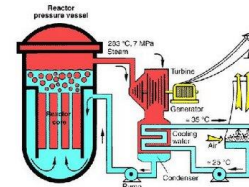
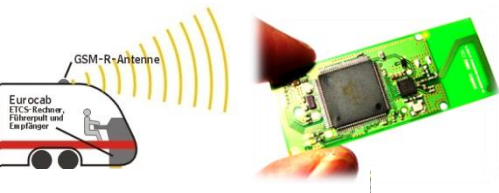
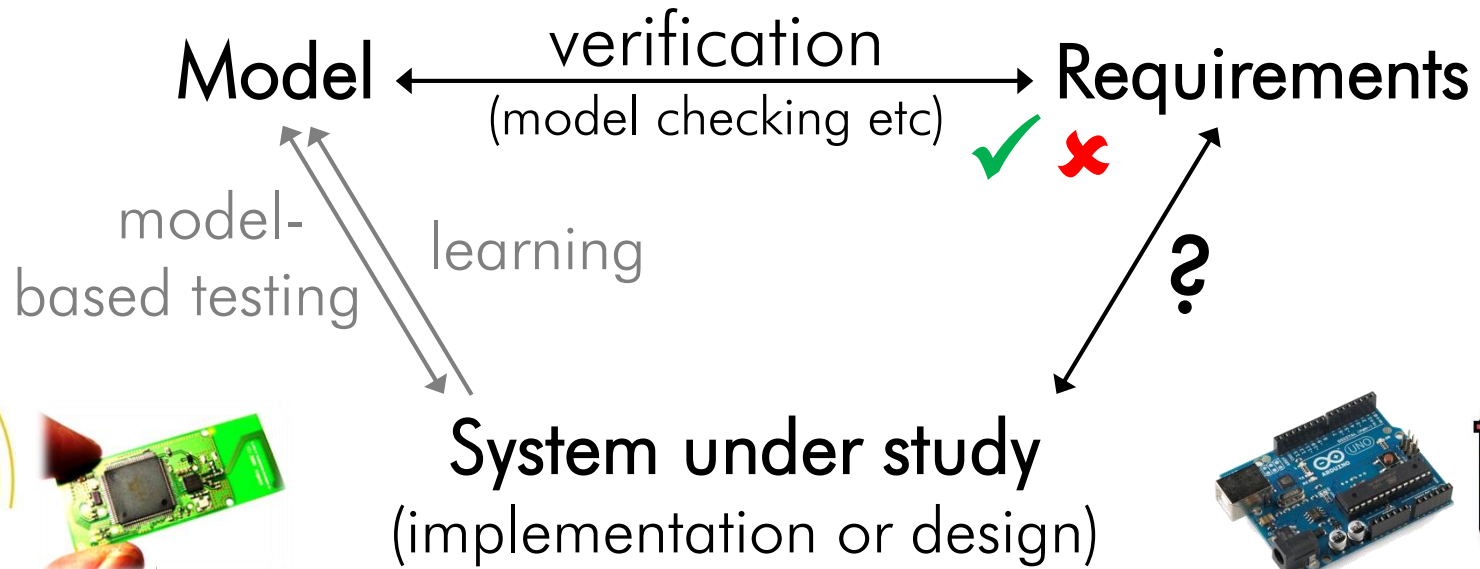
Stochastic Model Checking

Part Zero

Setting the Stage



Modelling and Analysis



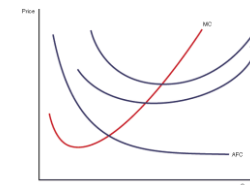
correctness



safety



performance



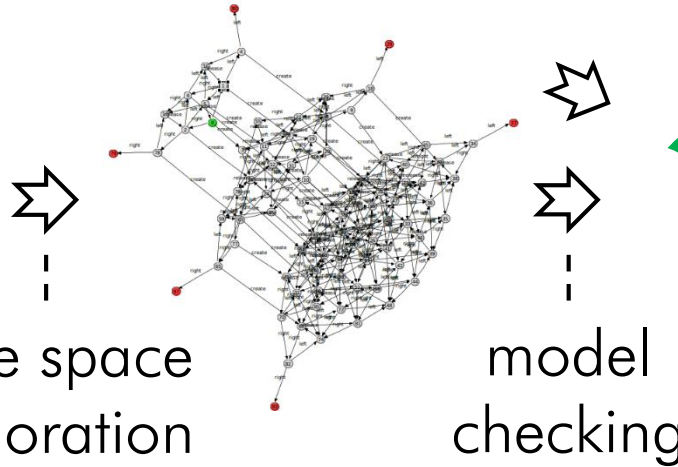
costs

Model Checking

"Can we reach certain states?"

good states final states
bad states ...

```
process P() {  
  alt {  
    :: stop {= fail = true =}  
    :: send; alt {  
      :: {= done = true =}  
      :: reset; P()  
    }  
  }  
}
```



+ witness
or
+ counter-example



randomised algorithms, uncertain environments,
machine-learnt functionalities, resource sharing ...

⇒ stochastic modelling and model checking

"What is the **probability**
of reaching certain states?"

probabilistic reachability

"What is the **expected** cost
until we reach certain states?"

reward expectations

All models are wrong,
but some are useful.

George E. P. Box

Models

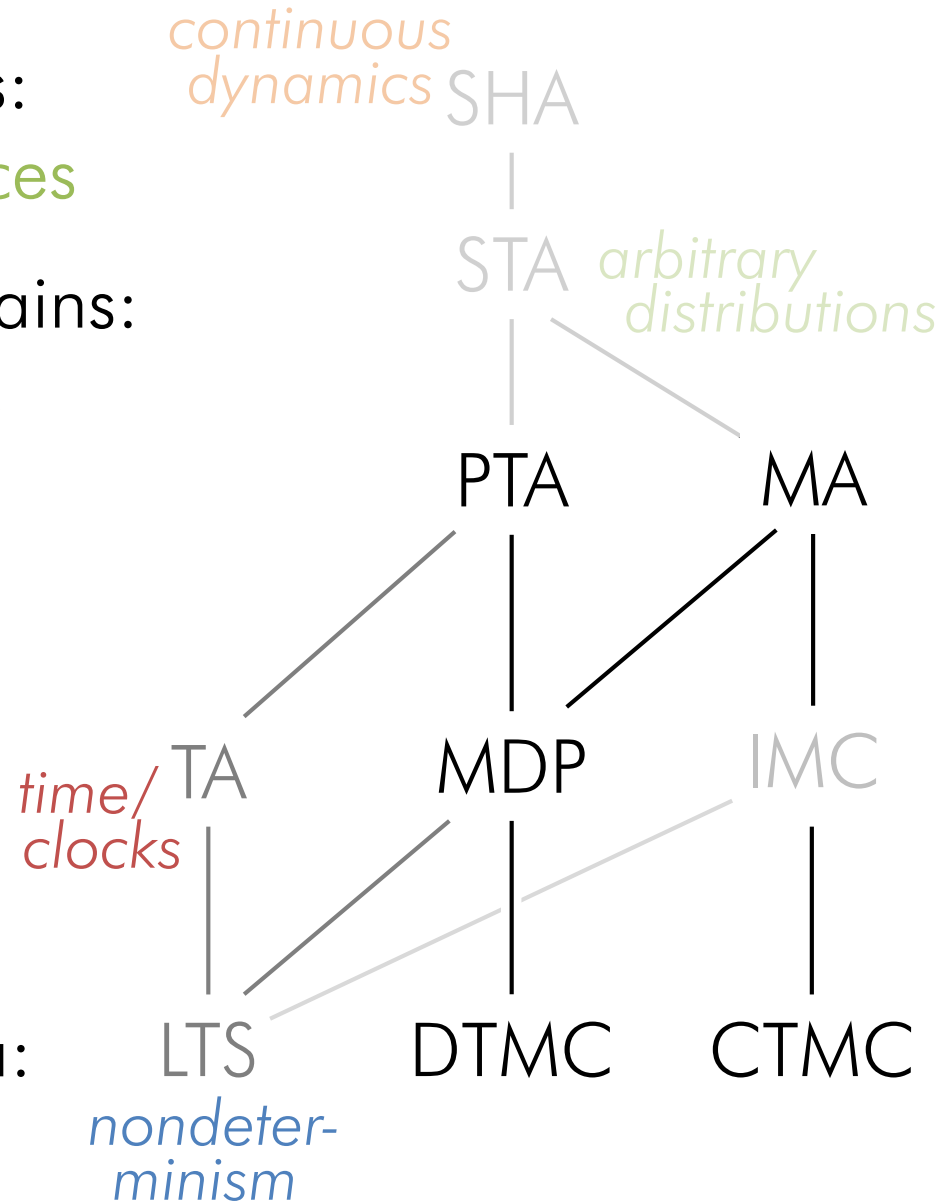
Discrete-time Markov chains:
discrete probabilistic choices

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



Concurrency Modelling Primer

Labelled transition systems:

states S

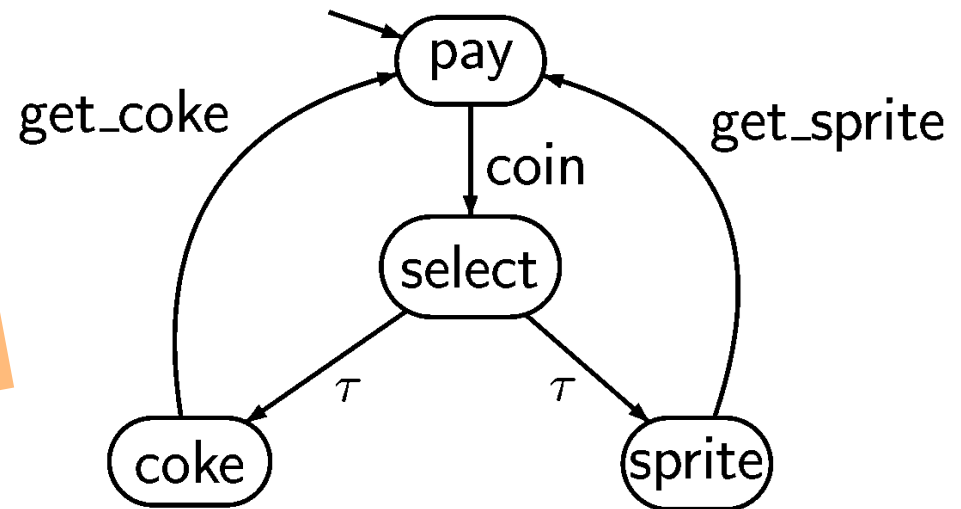
actions Act

transitions $\longrightarrow \subseteq S \times Act \times S$

initial state (set)

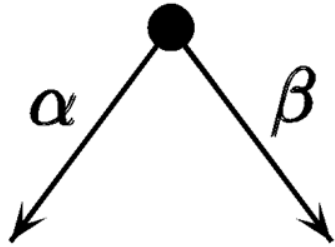
actions:
coin,
 τ
get_sprite,
get_coke

τ is the silent action "tau"

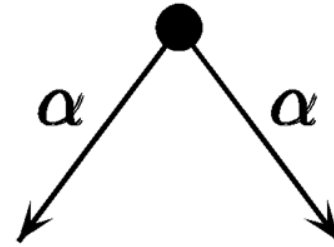


Nondeterminism

finite automata: language acceptors

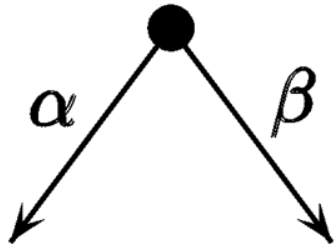


deterministic



nondeterministic

transition systems: operational system models



nondeterministic

decide nondeterministically to execute action α or β

worst-case analysis: consider all resolutions of the nondeterminism

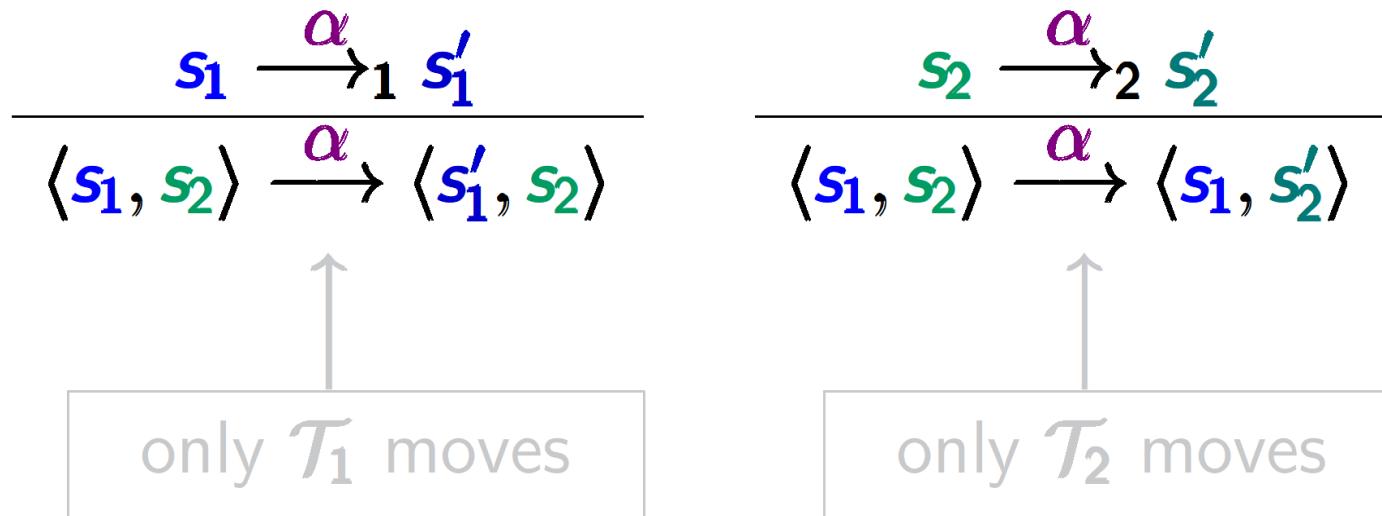
Concurrency – Nondeterminism – Interleaving

$$\mathcal{T}_1 = (S_1, Act_1, \longrightarrow_1, \dots)$$

$$\mathcal{T}_2 = (S_2, Act_2, \longrightarrow_2, \dots)$$

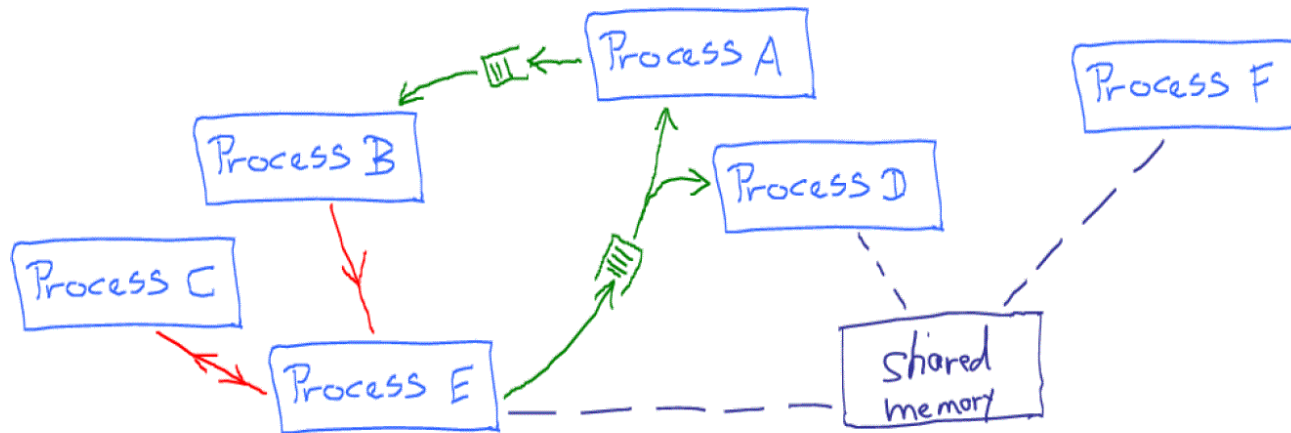
$$\mathcal{T}_1 \parallel \mathcal{T}_2 = (S_1 \times S_2, Act_1 \cup Act_2, \longrightarrow, \dots)$$

where the transition relation \longrightarrow is given by:



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 LTS

Handshake Communication

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

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

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

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

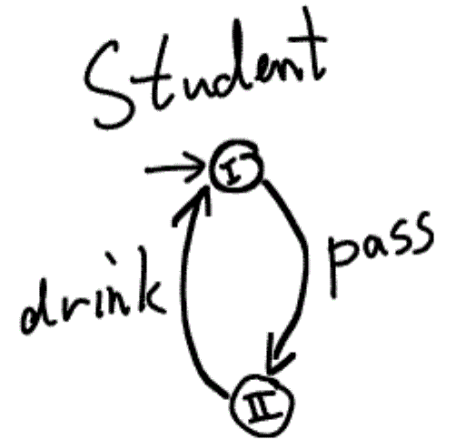
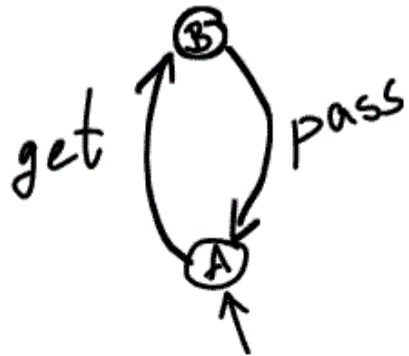
$$\frac{s_1 \xrightarrow{\alpha}_1 s'_1}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s'_1, s_2 \rangle} \qquad \frac{s_2 \xrightarrow{\alpha}_2 s'_2}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1, s'_2 \rangle}$$

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

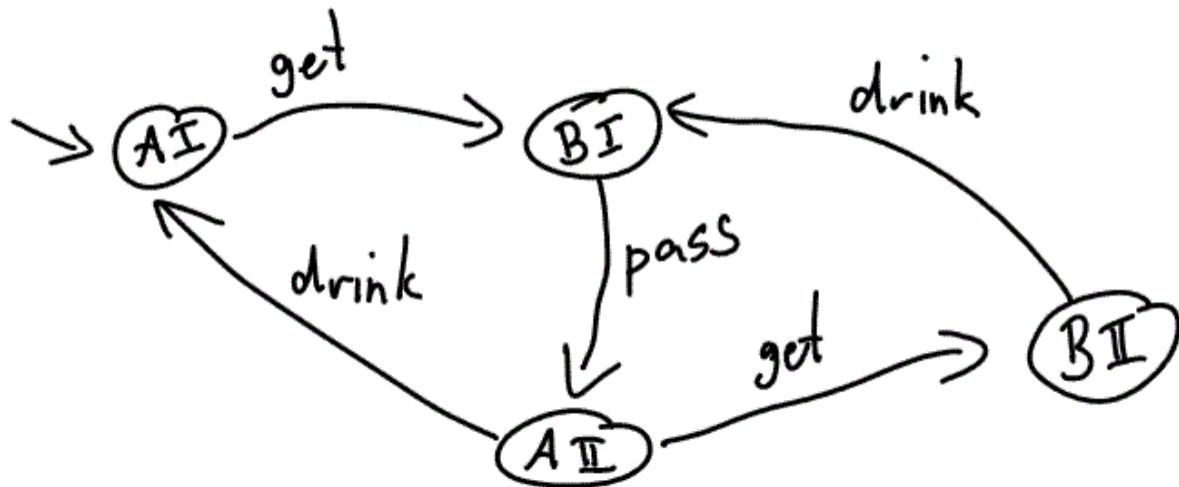
$$\frac{s_1 \xrightarrow{\alpha}_1 s'_1 \quad \wedge \quad s_2 \xrightarrow{\alpha}_2 s'_2}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s'_1, s'_2 \rangle}$$

Getting Real

Lecturer



Lecturer $\parallel_{\{\text{pass}\}}$ Student



Multiple Handshakes

$$\begin{array}{l} \text{transition systems } \mathcal{T}_1 = (S_1, \text{Act}_1, \rightarrow_1, \dots) \\ \mathcal{T}_2 = (S_2, \text{Act}_2, \rightarrow_2, \dots) \\ \mathcal{T}_3 = (S_3, \text{Act}_3, \rightarrow_3, \dots) \\ \mathcal{T}_4 = (S_4, \text{Act}_4, \rightarrow_4, \dots) \\ \vdots \end{array}$$

suppose that $\text{Act}_i \cap \text{Act}_j \cap \text{Act}_k = \emptyset$ if i, j, k are pairwise distinct. Then:

$$\begin{array}{l} \mathcal{T}_1 \parallel \mathcal{T}_2 \parallel \mathcal{T}_3 \parallel \mathcal{T}_4 \parallel \dots := \\ \left(\left(\left(\mathcal{T}_1 \parallel_{\text{Syn}_{1,2}} \mathcal{T}_2 \right) \parallel_{\text{Syn}_{1,2,3}} \mathcal{T}_3 \right) \parallel_{\text{Syn}_{1,2,3,4}} \mathcal{T}_4 \right) \dots \end{array}$$

where

$$\begin{array}{l} \text{Syn}_{1,2} = \text{Act}_1 \cap \text{Act}_2 \\ \text{Syn}_{1,2,3} = (\text{Act}_1 \cup \text{Act}_2) \cap \text{Act}_3 \\ \text{Syn}_{1,2,3,4} = (\text{Act}_1 \cup \text{Act}_2 \cup \text{Act}_3) \cap \text{Act}_4 \\ \dots \end{array}$$

Synchronous Product

for parallel systems with fully synchronized processes

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

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

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

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

Time to Play

- You pick a die.
- I pick one of the remaining.
- Then we throw.
- The higher number wins.

Which one
you pick?


$$P(A < B) =$$

$$P(B < C) =$$

$$P(C < D) =$$

$$P(D < A) =$$

$$\frac{2}{3}$$



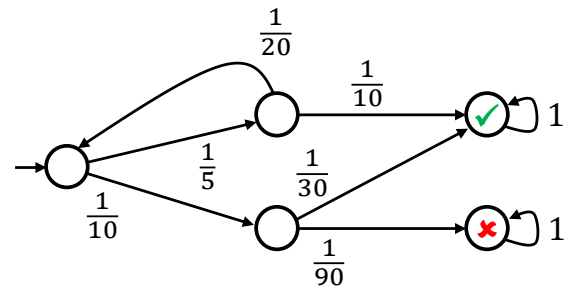
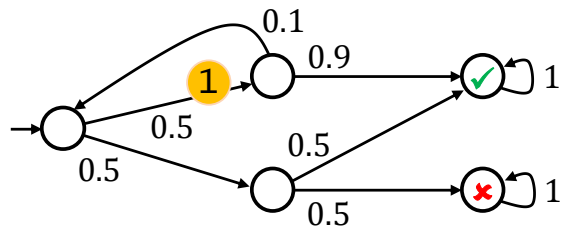
A: 0 0 4 4 4 4

B: 1 1 1 5 5 5

C: 2 2 2 2 6 6

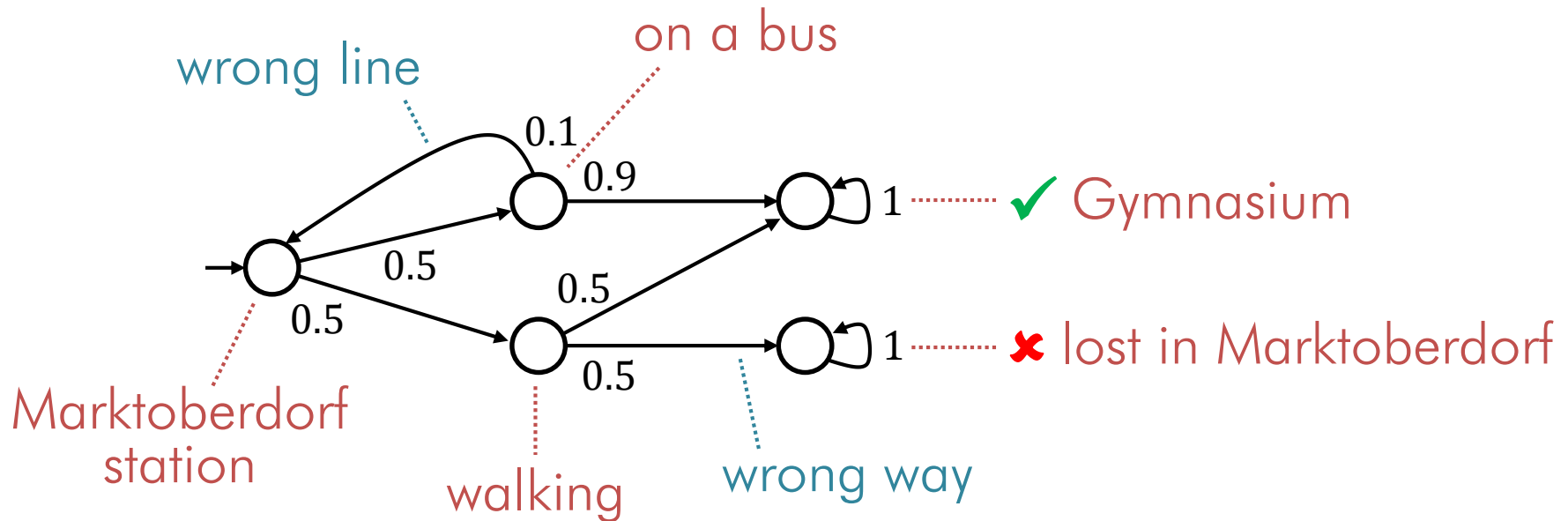
D: 3 3 3 3 3 3

Markov Chains



Markov Chains

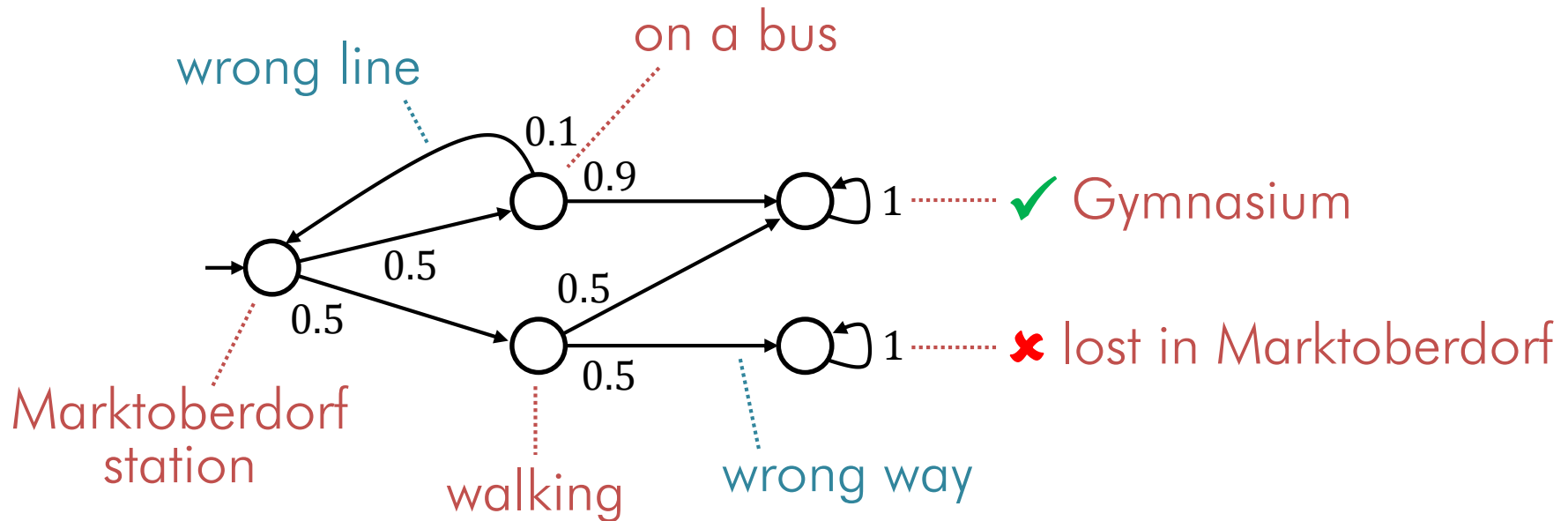
Discrete-time Markov chains



$$\mathbb{P}(\diamond \checkmark) = \underbrace{0.5^2}_{\text{walking}} + \underbrace{0.5 \cdot (0.9 + 0.1 \cdot (0.5^2 + 0.5 \cdot (0.9 + \dots))}_{\text{first bus}} \underbrace{\quad}_{\text{walking}} \underbrace{\quad}_{\text{second bus}}$$

Markov Chains

Discrete-time Markov chains

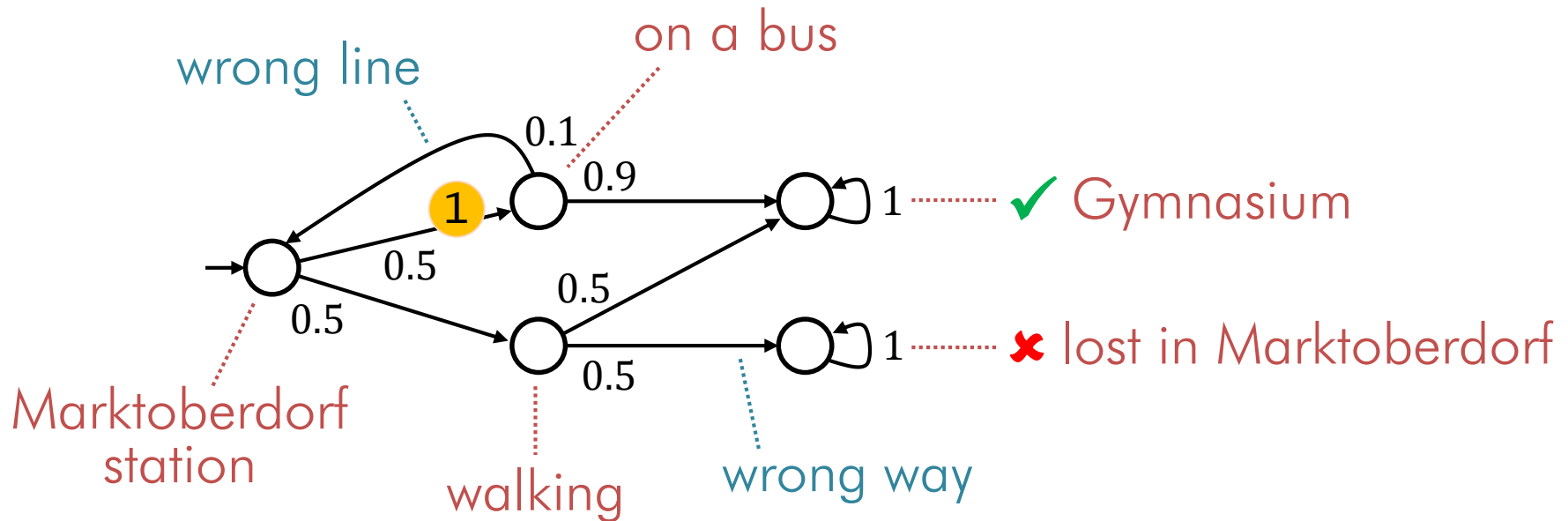


Value function

$$V_{\mathbb{P}} \in \mathcal{S} \rightarrow \mathbb{R}: V_{\mathbb{P}}(s) = \begin{cases} 1 & \text{if target} \\ \sum_{s' \in \mathcal{S}} P(s, s') \cdot V_{\mathbb{P}}(s') & \text{otherwise} \end{cases}$$

⇒ reachability probability is least fixpoint of $V_{\mathbb{P}}$ → iterative algorithm

Discrete-time Markov chains



$$\mathbb{E}(\text{yellow circles to } \underbrace{\checkmark \vee \times}_{\text{must have probability 1!}}) = 0.5 \cdot (1 + 0.1 \cdot 0.5 \cdot (1 + \dots$$

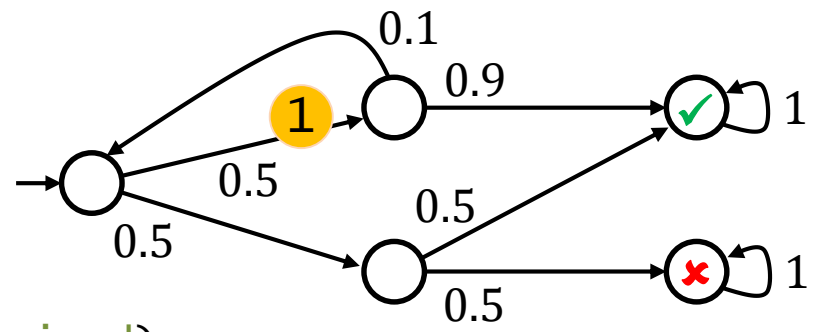
$$\text{Values: } V_{\mathbb{E}}(s) = \sum_{s' \in S} P(s, s') \cdot (\text{cost}(s, s') + V_{\mathbb{E}}(s'))$$

Markov Chains

```
bool lost, arrived;  
transient real cost;
```

```
property PLost = Pmax(<> lost);  
property PArrived = Pmax(<> arrived);  
property ECost = Xmax(S(cost), lost || arrived);
```

```
do {  
  :: tau palt {  
    :0.5: {= cost = 1 =};  
    tau palt {  
      :0.1: {= =}  
      :0.9: {= arrived = true =}; stop  
    }  
    :0.5: tau palt {  
      :0.5: {= arrived = true =}  
      :0.5: {= lost = true =}  
    };  
    stop  
  }  
}
```



DTMC in *Modest*

Modest

The Modest **Language**

*a modelling and
description language for
stochastic timed systems*

⇒ high-level language for
compositional modelling
with an SHA semantics

2001: PAPM-PROBMIV

2006: IEEE TSE (STA)

2013: FMSD (SHA)

The Modest **Toolset**

*a multiple-formalism,
multiple-solution toolkit
driven by Modest features*

⇒ model checking,
simulation (SMC),
model-based testing

2003: DSN (predecessor MoToR)

2009: QEST (mcpta tool)

2014: TACAS (toolset)

Form Methods Syst Des (2013) 43:191–232
DOI 10.1007/s10703-012-0167-z

**compositional modelling and analysis framework
for stochastic hybrid systems**

st Moritz Hahn · Arnd Hartmanns ·
Holger Hermanns · Joost-Pieter Katoen

Hartmanns, H.: The Modest Toolset:
An Integrated Environment for Quantitative
Modelling and Verification (TACAS 2014)

Hahn, Hartmanns, H., Katoen: A Compositional
Modelling and Analysis Framework for
Stochastic Hybrid Systems (FMSD 2013)

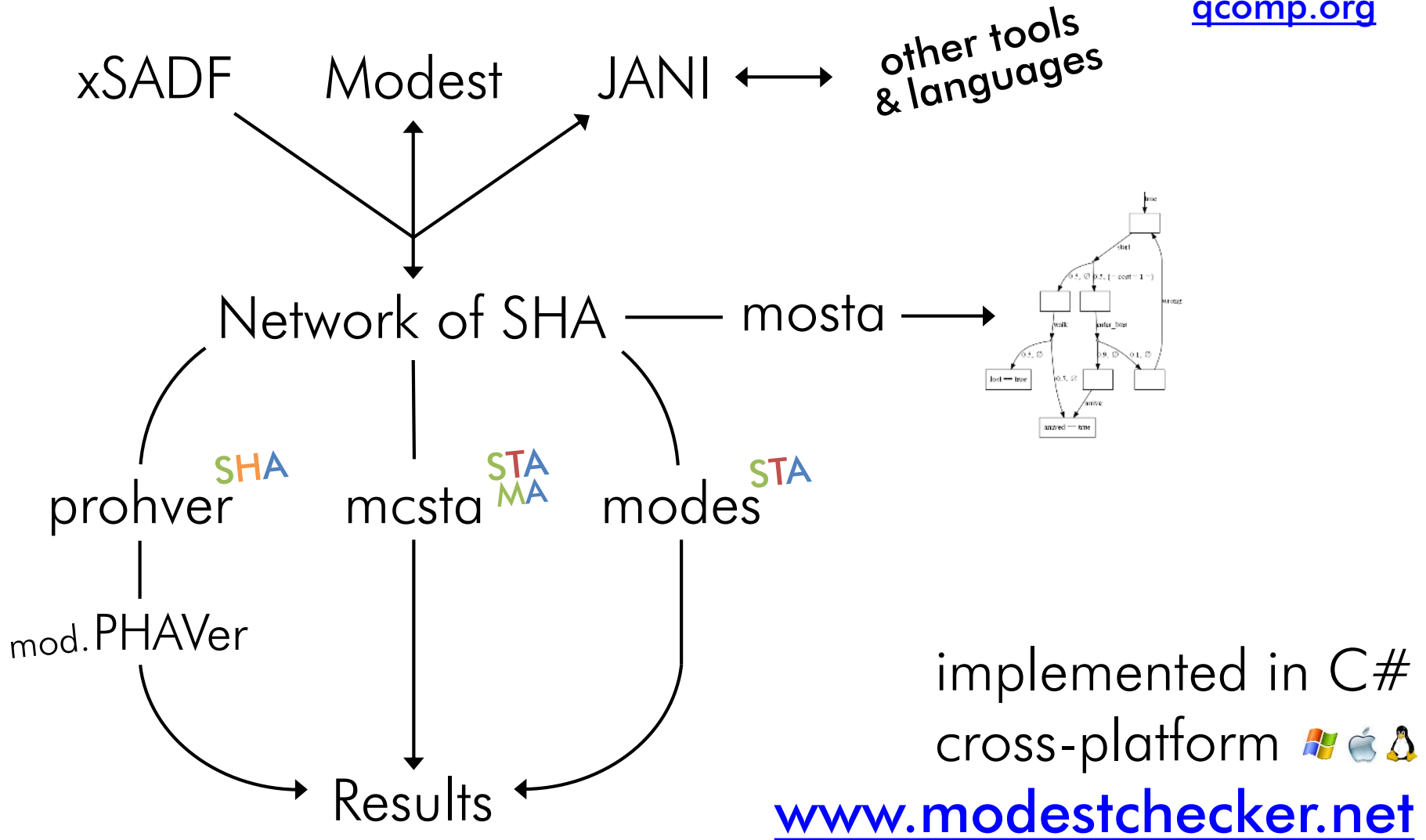
**The Modest Toolset: An Integrated Environment
for Quantitative Modelling and Verification***

Arnd Hartmanns and Holger Hermanns
Saarland University – Computer Science, Saarbrücken, Germany

Abstract Probabilities, real-time behaviour and continuous dynamics
are the key ingredients of quantitative models enabling formal studies of
non-functional properties such as dependability and performance. The
modest TOOLSET is based on networks of stochastic hybrid automata
with a compositional semantic foundation. Many existing automata-
based verification techniques for SHA. The toolset aims to facilitate
model-based compositional

The Modest Toolset

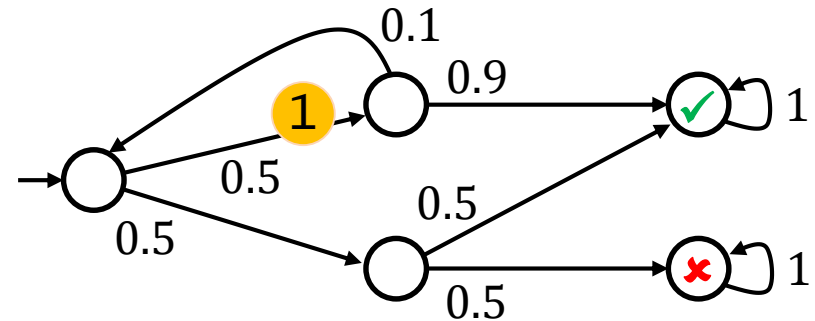
qcomp.org



Markov Chains

DTMC in Modest

```
do {  
  :: tau palt {  
    :0.5: {= cost = 1 =};  
    tau palt {  
      :0.1: {= =}  
      :0.9: {= arrived = true =}; stop  
    }  
    :0.5: tau palt {  
      :0.5: {= arrived = true =}  
      :0.5: {= lost = true =}  
    };  
    stop  
  }  
}
```



What are the values
of $P_{Arrived}$ & E_{Cost} ?

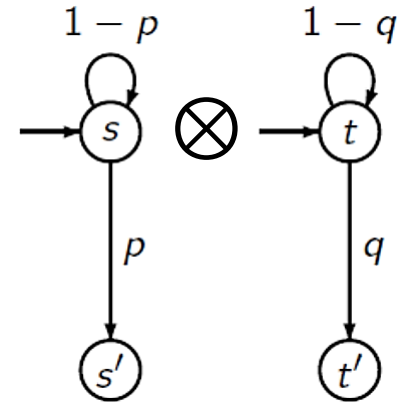
⇒ Your Homework!

Feel free to check
with `mcsta`!

Composition Operators on 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 $\mathcal{T}_1 = (S_1, \text{Act}_1, \xrightarrow{1}, \dots)$,
 $\mathcal{T}_2 = (S_2, \text{Act}_2, \xrightarrow{2}, \dots)$

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

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

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