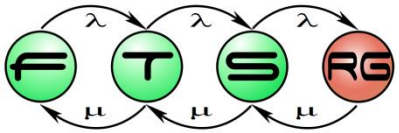
Reliability modeling with stochastic model checking

Kristóf Marussy

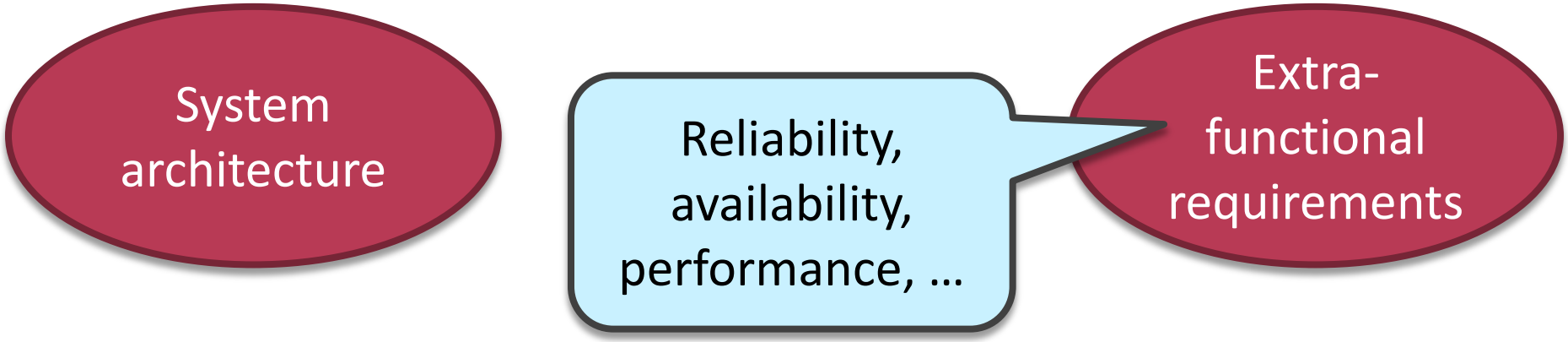
Budapest University of Technology and Economics
Fault Tolerant Systems Research Group



References

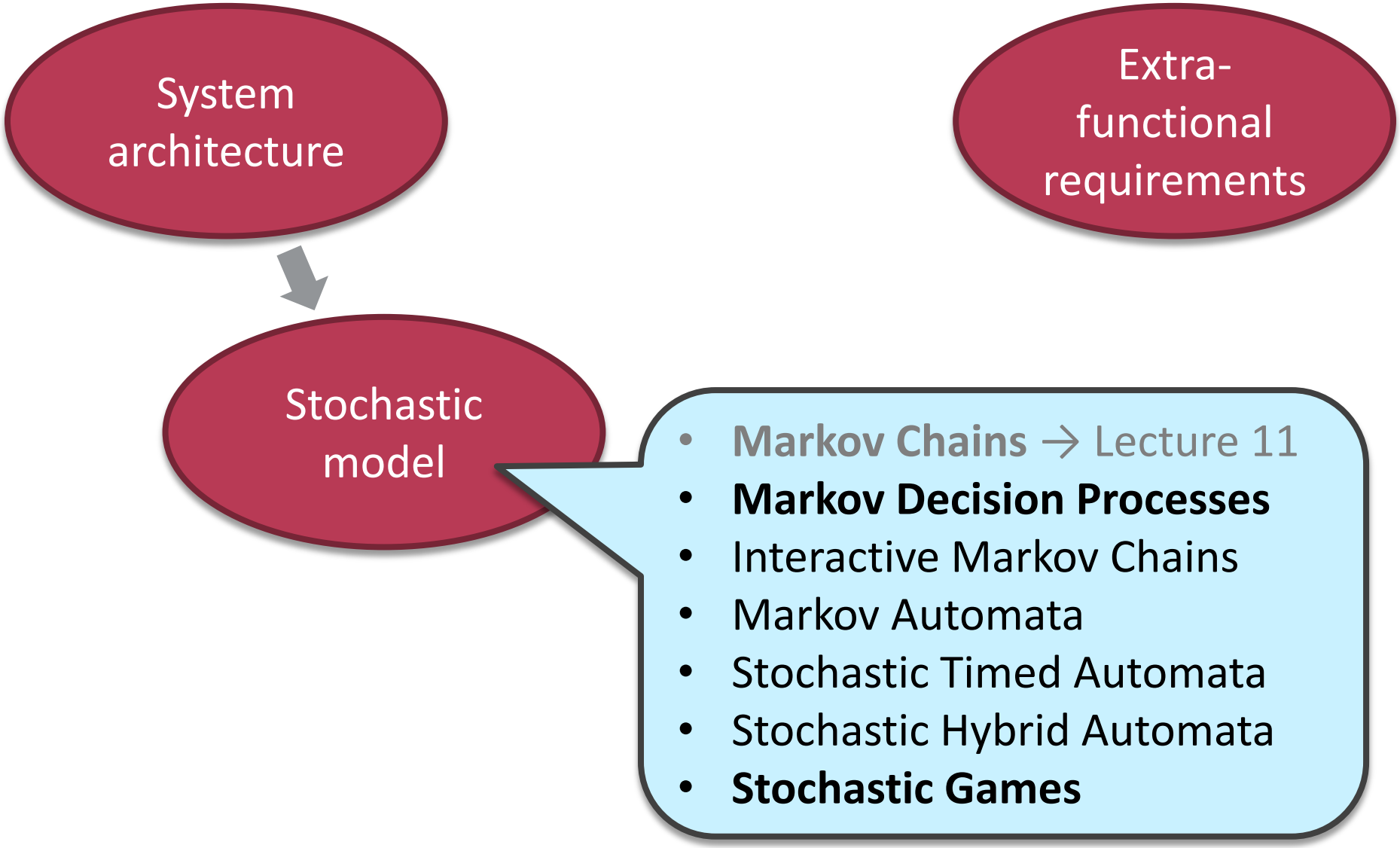
1. Katoen, Joost-Pieter: “The Probabilistic Model Checking Landscape”. In: *LICS '16,* pp. 31-45.
   DOI: 10.1145/2933575.2934574
2. Kwiatkowska, Marta and David Parker: “Advances in Probabilistic Model Checking”. In: *Marktoberdorf Summer School '11.*
   URL: http://qav.comlab.ox.ac.uk/papers/marktoberdorf11.pdf
3. Budde, Carlos E., Christian Dehnert, Ernst Moritz Hahn, Arnd Hartmanns, Sebastian Junges and Andrea Turrini: “JANI: Quantitative Model and Tool Interaction”. In: *TACAS '17,* pp. 151-168.
   DOI: 10.1007/978-3-662-54580-5_9

Dependability and performability modeling

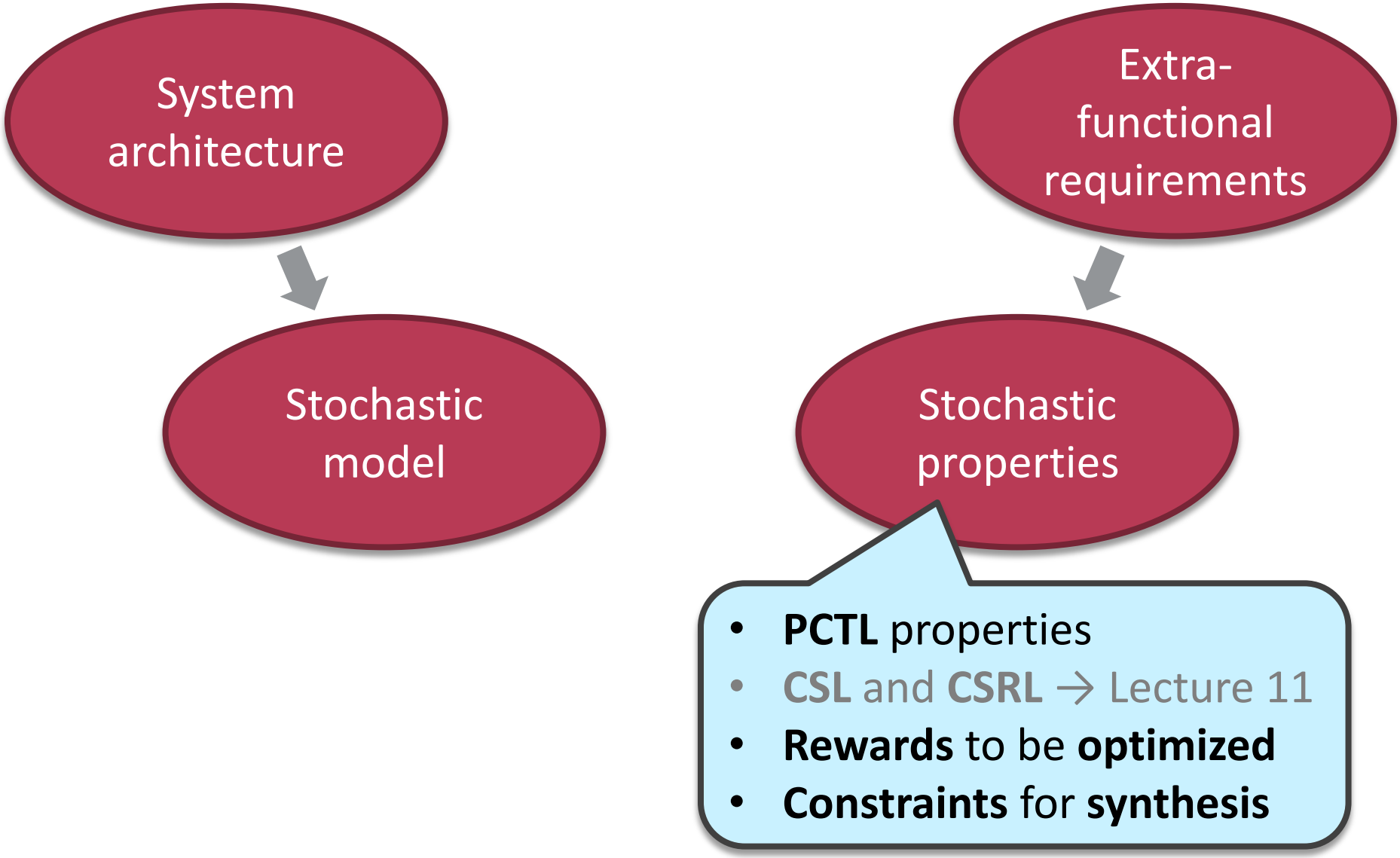


- **Check** whether the system
  - satisfies its service-level agreement, or
  - has tolerable hazard rate
- Choose between architecture **alternatives** by **optimization**
- **Synthesize** correct configurations

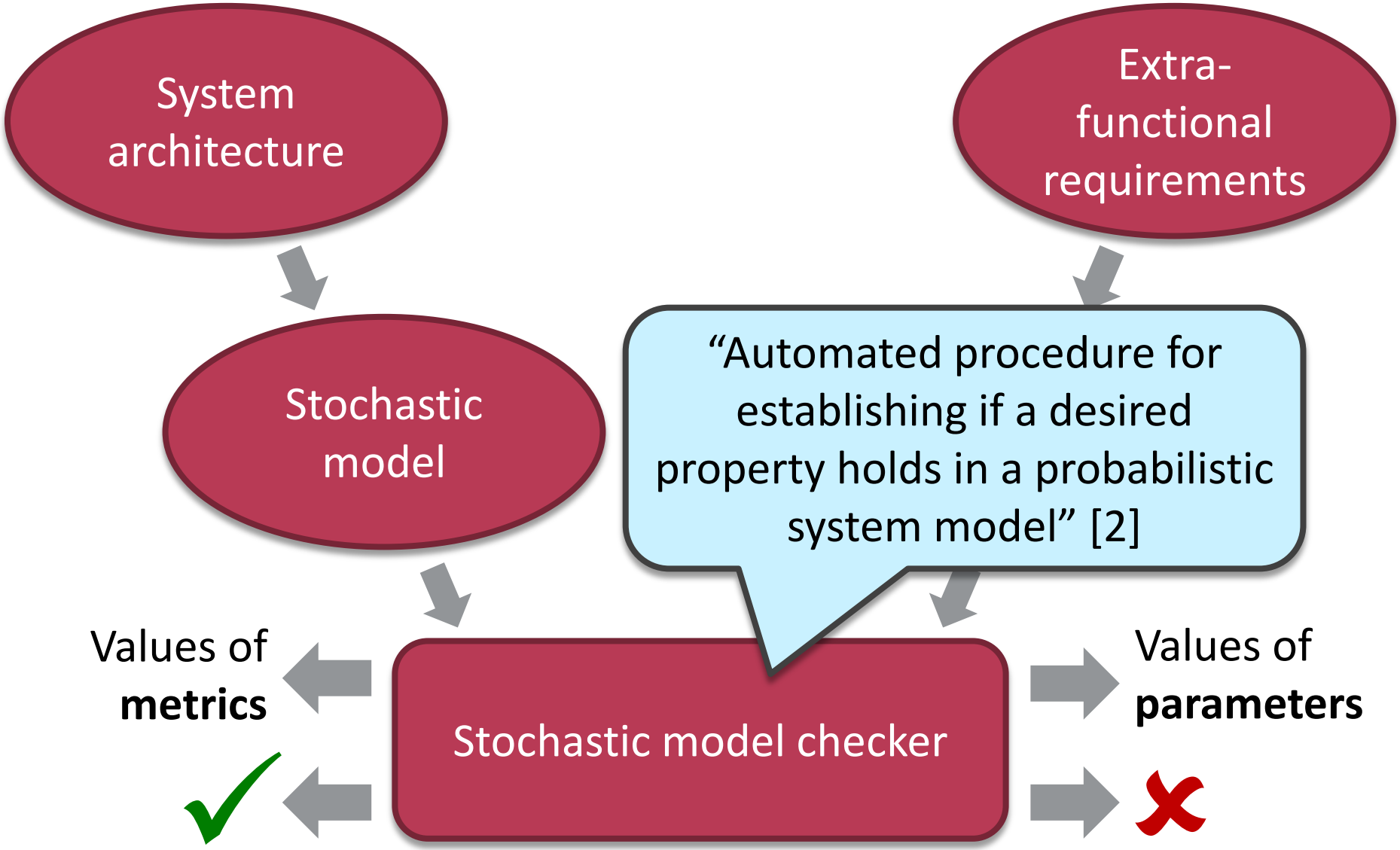
Lower-level formalisms



Stochastic property descriptions



Analysis goals



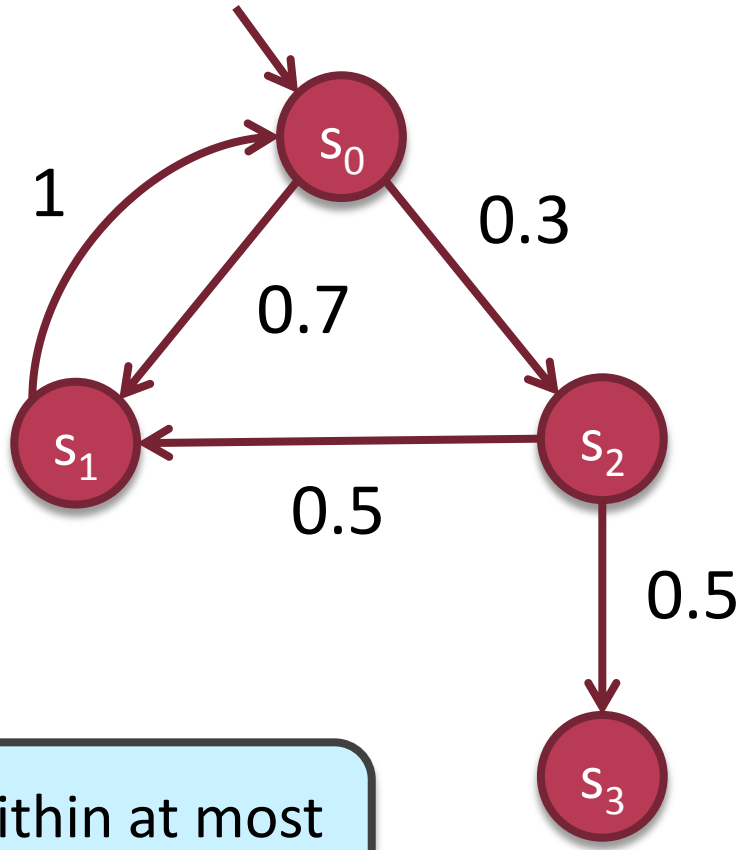
THE PROBABILISTIC MODELS LANDSCAPE

[1, Section 2]

Markov Chains

- **Markovian** models: behavior only depends on the **active state**
  - **Discrete** time: Transition probabilities
  - **Continuous** time: Transition rates
- **PCTL** properties, e.g.,

$$P^{\leq 0.5}\,\mathrm{F}^{[0,3]}\,s_3$$



The probability of reaching s_3 within at most 3 units of time is not larger than 0.5

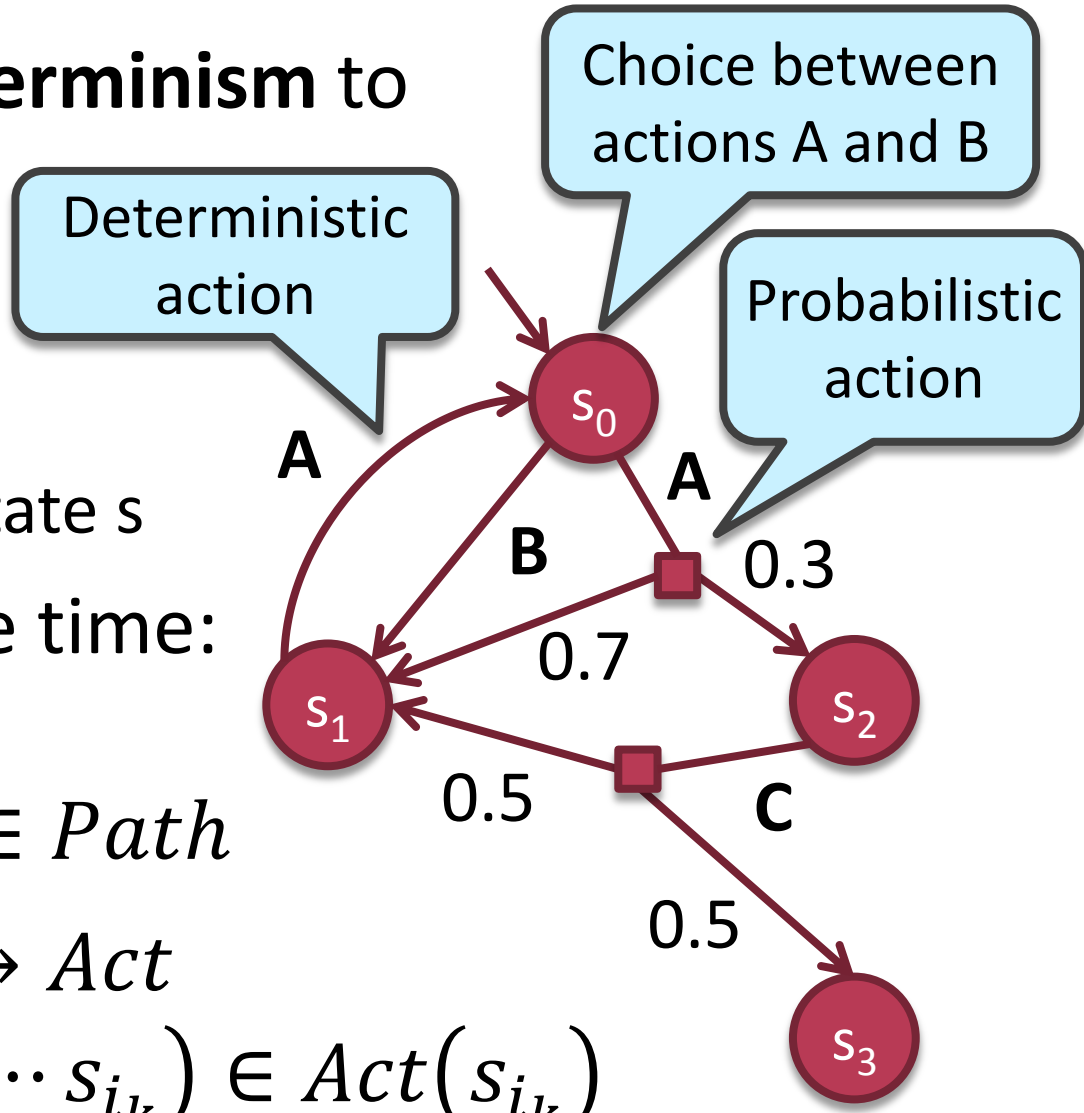
Markov Decision Processes (MDPs)

- Introducing **non-determinism** to Markovian models
- **Action** labels *Act*
  - *Act*(s) ⊆ *Act*
    actions available is state s
- **Trajectory** in discrete time: Sequence of states

$$\tau = s_{i_1} s_{i_2} \cdots s_{i_k} \in Path$$

- **Scheduler** σ: $Path \rightarrow Act$

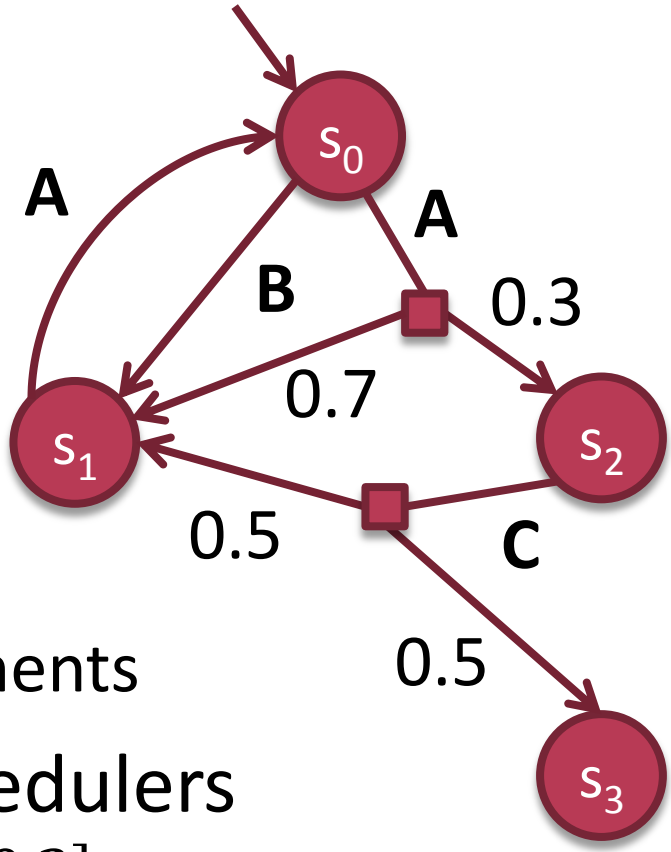
$$\sigma(\tau) = \sigma\left(s_{i_1} s_{i_2} \cdots s_{i_k}\right) \in Act\left(s_{i_k}\right)$$



Markov Decision Processes (MDPs)

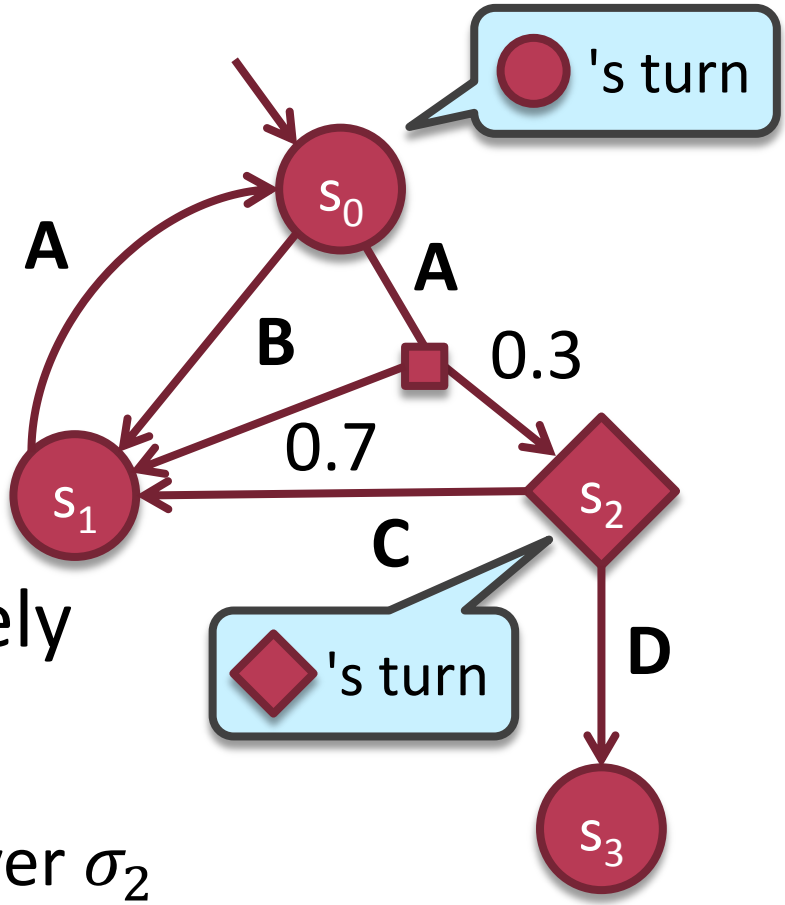
- Introducing **non-determinism** to Markovian models
- Actions: either **environmental** effects or **control**
  - Environment hampers satisfaction of extra-functional requirements
  - Control aids in satisfying requirements
- **Maximize** or **minimize** over schedulers

$$P^{\max=?}\mathrm{F}^{[0,3]}{}_{s_3} \qquad P^{\min=?}\mathrm{F}^{[0,3]}{}_{s_3}$$



Stochastic Games

- Model interactions with the environment and control at the same time
- Two **players**
  - Player ● – control
  - Player ◆ – environment
- Schedulers σ_1 and σ_2 for players ● and ◆, respectively
- $P^{=?}\mathrm{F}^{[0,3]}s_3$
  - Minimize over σ_1, maximize over σ_2



The solution landscape

- Continuous-Time Markov Chains
  - Transient and steady-state solution by **linear equation solvers**
- Markov Decision Processes
  - **Bellman equations** characterize extremal policies
  - Value iteration, policy iteration, linear programming
- Challenge: growth of time and memory costs due to **state space explosion**

THE ABSTRACTION LANDSCAPE

[1, Section 4]

Stochastic bisimulation minimization

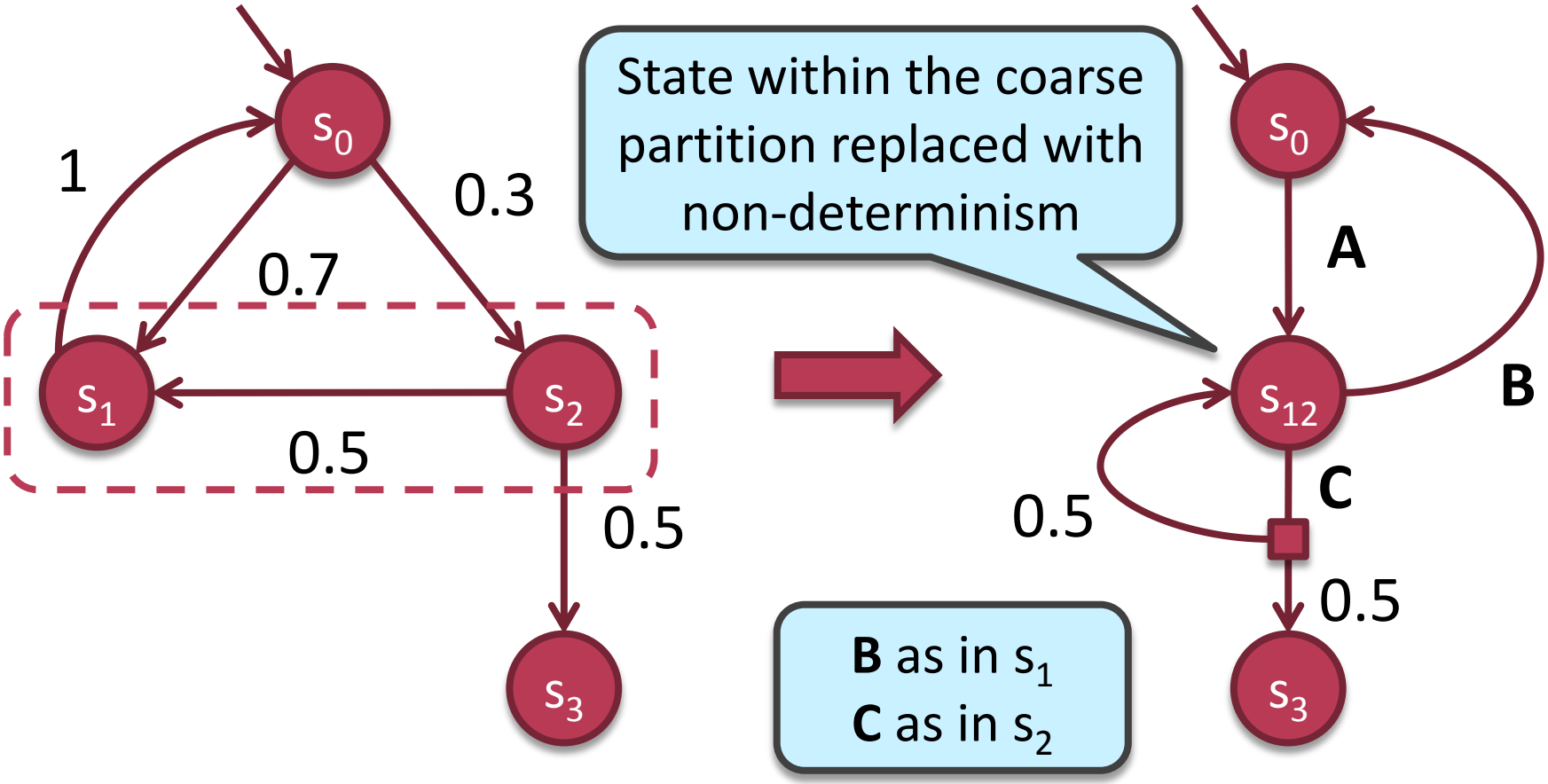
- Partition the states of the Markov chain into **equivalence classes** $B_1, B_2, \ldots, B_m$
  - States s_i and s_j belong to the same equivalence class if

$$\sum_{s' \in B_k} P(s_i, s') = \sum_{s' \in B_k} P(s_j, s') \quad \text{for all } B_k$$

    i.e., from these states, the probabilities of reaching another equivalence class are equal -- for all equivalence classes
  - Defined analogously for MDPs
  - Equivalent states can be merged to **reduce state space**

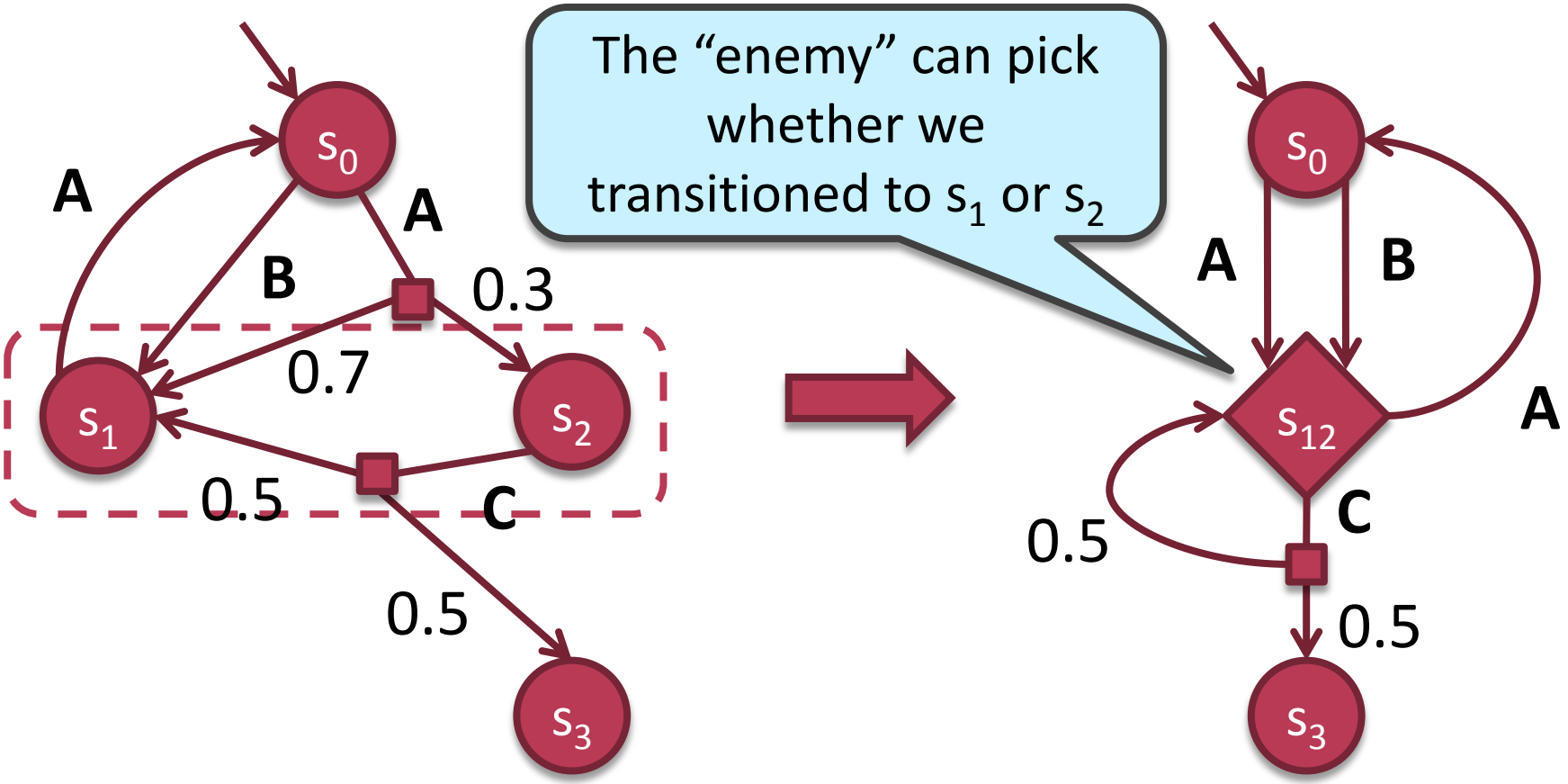
More aggressive merging

- Merge (possibly unrelated) states in an **MC** by turning it into an **MDP**



More aggressive merging

- Merge (possibly unrelated) states in an **MDP** by turning it into a **Stochastic Game**



OPEN SOURCE TOOLS

PRISM Model Checker

- http://www.prismmodelchecker.org/
- **PRISM** language: de facto Standard

```
dtmc

module die
    // local state
    s : [0..7] init 0;
    // value of the die
    d : [0..6] init 0;
    [] s=0 -> 0.5 : (s'=1) + 0.5 : (s'=2);
    [] s=1 -> 0.5 : (s'=3) + 0.5 : (s'=4);
    [] s=2 -> 0.5 : (s'=5) + 0.5 : (s'=6);
    [] s=3 -> 0.5 : (s'=1) + 0.5 : (s'=7) & (d'=1);
    [] s=4 -> 0.5 : (s'=7) & (d'=2) + 0.5 : (s'=7) & (d'=3);
    [] s=5 -> 0.5 : (s'=7) & (d'=4) + 0.5 : (s'=7) & (d'=5);
    [] s=6 -> 0.5 : (s'=2) + 0.5 : (s'=7) & (d'=6);
    [] s=7 -> (s'=7);
endmodule
```

PRISM Model Checker

- http://www.prismmodelchecker.org/
- **PRISM** language: de facto Standard
  - discrete-time Markov chains (DTMCs)
  - continuous-time Markov chains (CTMCs)
  - Markov decision processes (MDPs)
  - probabilistic automata (PAs)
  - probabilistic timed automata (PTAs)
  - PRISM-games extensions for Stochastic Games
- **Hybrid** (symbolic + explicit) analysis backend

Storm: modern probabilistic model checker

- Developed at RWTH Aachen University since 2012
  - Open source since 2017
- Various input formats
  - PRISM, JANI, GSPN, DFT, cpGCL, explicit
- Command line, C++, Python interfaces
- Standard model format: **JANI** [3]
  - JSON-based format for probabilistic models
  - Simplifies integration of probabilistic model checking and abstraction into toolchains
  - Trigger analysis tasks over WebSocket

Summary

- Analysis of stochastic and probabilistic models
  - **Model checking** extra-functional requirements
  - Calculation of **metrics**
  - Synthesizing optimal **parameters**
- Markov chains and **non-deterministic** extension
- **Abstraction** techniques
  - **Bisimulation** minimization
  - Merge states by introducing non-determinism
- Use in toolchains by invoking open-source tools

See [1, 2, 3] for more info