unibz

SOFTWARE AND SYSTEMS ENGINEERING research group

Model-based testing under uncertainty

L1: Theoretical aspects & practical applications

Matteo Camilli
matteo.camilli@unibz.it
https://matteocamilli.github.io

Formal Methods at Work @
Gran Sasso Science Institute (GSSI)
A.Y. 2020/21

G S S I

Outline

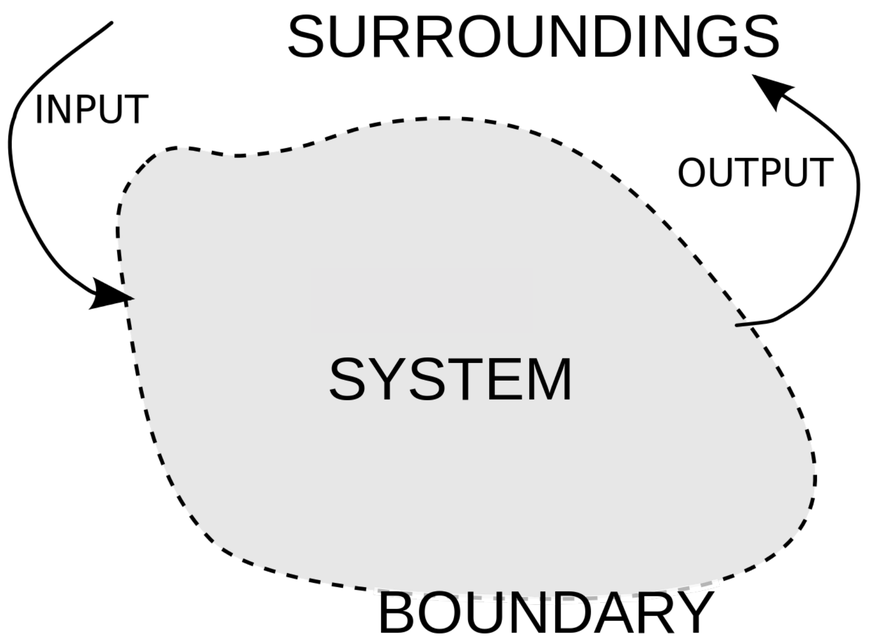
- Introduction
  - The (uncertain) world and the machine
- Markov Decision Process
  - Structure, Rewards
  - Policy, best policy, value iteration
- Model-based testing (MBT)
  - Offline vs online approaches
  - Conformance relation
  - Probabilistic alternating simulation and refinement
- Online MBT under uncertainty
  - Problem statement
  - Uncertain model paramenters
  - Bayesian inference
  - Framework and test case generation strategies

Introduction

- The world and the machine
- Requirements vs Specification vs Implementation
- Uncertain assumptions

Characterizations of Systems

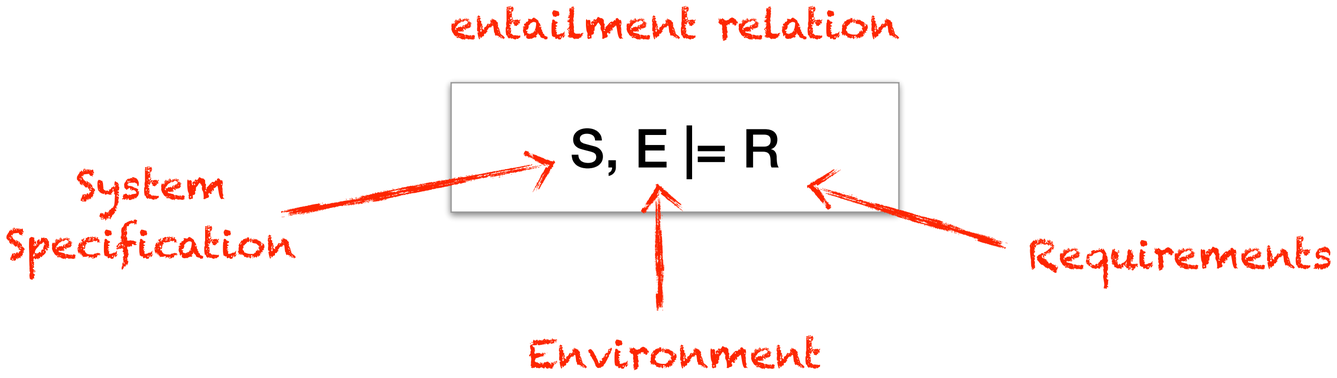
- System
  - Collection of interconnected elements that exist within and interact with an environment



System Boundaries

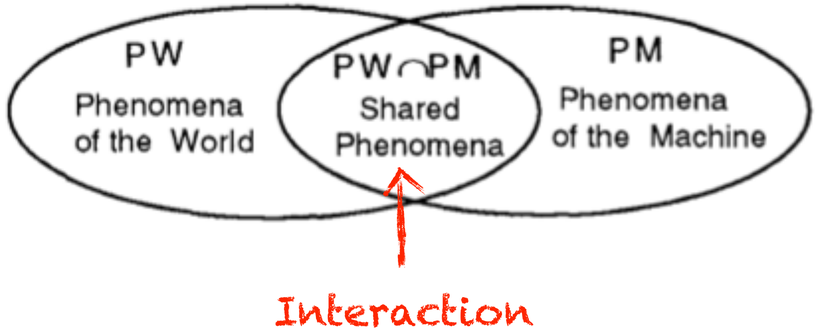
- System boundaries
  - Collection of elements that exist within the system and directly interact with one or more environments (i.e., part of the external world of interest)
  - Often, sensors + actuators

- [M. Jackson et al., ICSE ’95] “The world and the Machine”
  - Engineers are concerned with:
    - The world in which the machine (i.e., the system) serves useful purpose
    - The machine itself



System Boundaries — Interface

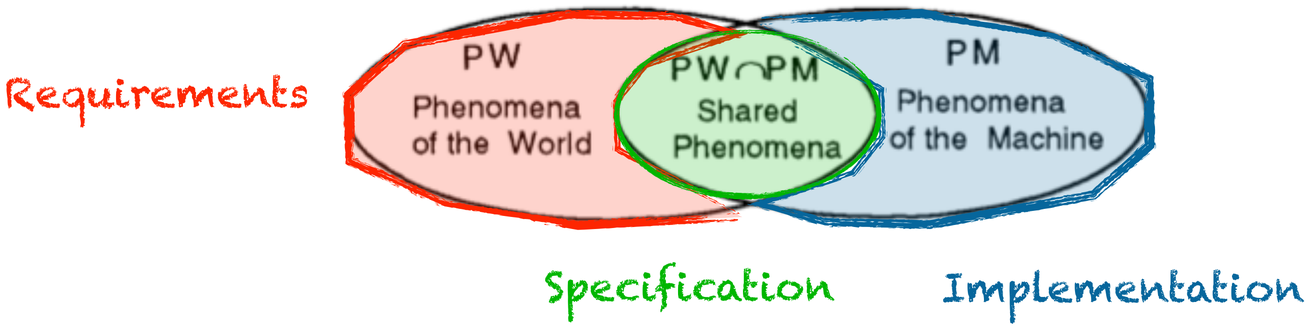
- Interface between the world and the machine
  - The problem exists in E
  - the solution to the problem is provided by S
  - Interface allow interactions between S and E



- Interaction: participation in shared phenomena (states + events)
  - Often not symmetrical —> E might initiate an event, but S might not have the power to inhibit the event
  - Example (Lift — simple Cyber-Physical System)
    - Lift has buttons (hw) + controller (sw) + on/off switches at floors to recognize the lift car (sensors)
    - Shared state controlled by E
      - Lift at 3rd floor (E) —> Switch at 3rd floor is on —> should map to the “knowledge” (S)

System Boundaries — Engineering

- Engineering facet
  - Intersecting phenomena sets suggest systematic usage of terms
    - Requirements
    - Specification
    - Implementation



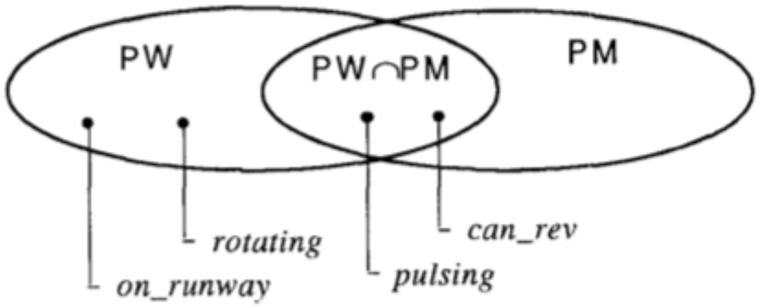
- Requirements: are concerned solely with PW
  - Defined by stakeholders needs
  - Stakeholders want us to engineer effects in the world (not in the machine)
- Implementation: concerned solely with PM
  - Properties and behavior of the machine engineered from the specification
- Specification: concerned with PW ∩ PM
  - staging area of the production from requirements to implementation
  - Abstract description of interactions between world and machine

System Boundaries — Engineering

- Example: avionic system
  - Requirement: reverse thrust can be engaged iff the plane is landing and already on the runway
    - REQ: can_rev <—> on_runway

Controlled by E, Shared with S

How does S know??



- Interface: sensors on landing wheels generate pulses when they rotate
  - State pulsing is now shared with the machine (rotating is not!)
- Engineers decide that the following assumptions hold
  - WORLD1: pulsing <—> rotating
  - WORLD2: rotating <—> on_runway
- Relying on this assumptions, developers can derive the spec
  - SPEC: can_rev <—> pulsing

WORLD1, WORLD2, SPEC |= REQ

Assumptions

System Boundaries — Engineering

- Are the world assumptions reasonable?
  - E.g., does WORLD2 always hold?
  - Heavy rain condition
    - Runway covered with water —> aquaplaning —> on_runway & !rotating

It holds because we decided that the world assumptions hold

WORLD1, WORLD2, SPEC |= REQ

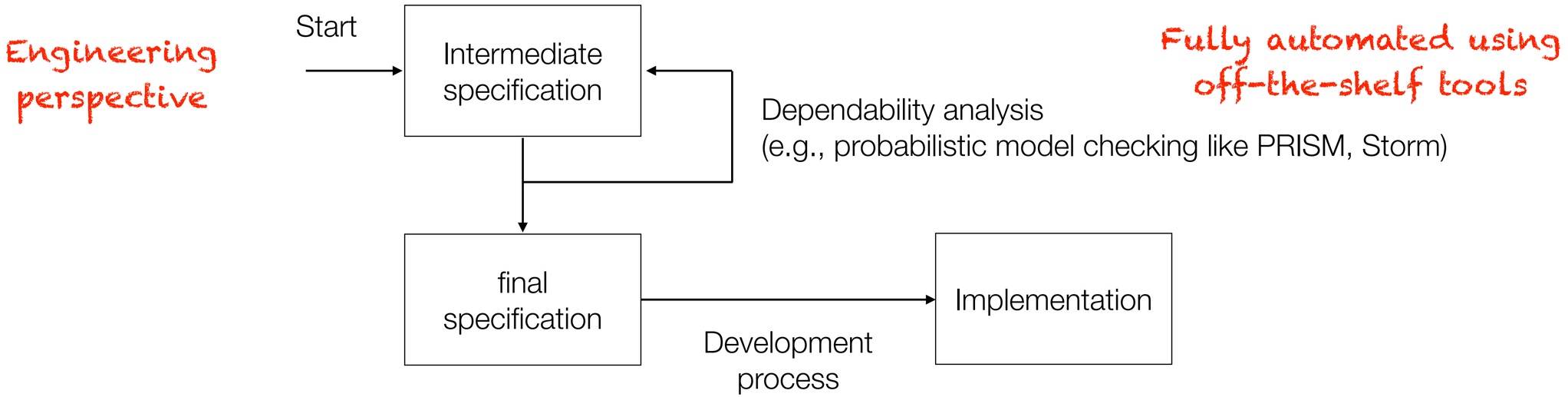
- Problem: engineers shall fully understand the properties of the world
  - full knowledge at design time is hard, or even impossible
  - The phenomena of the world can be uncertain (e.g., precise whether conditions)
  - Uncertainty affect in turn the behavior of the implementation

Markov decision processes

- Structure
- Examples
- Rewards
- Policy, best policy
- Value iteration algorithms

Markov Models

- Basic notions
  - Modeling formalism to describe **partially/fully stochastic** phenomena of interest
    - the modeled phenomenon meets the **Markov property**[1,2] (memoryless)
  - Formal framework for **performance** and **dependability** (reliability, availability, safety) analysis
  - **Dependability modeling** (upfront) at design time improves the quality of the system eventually produced

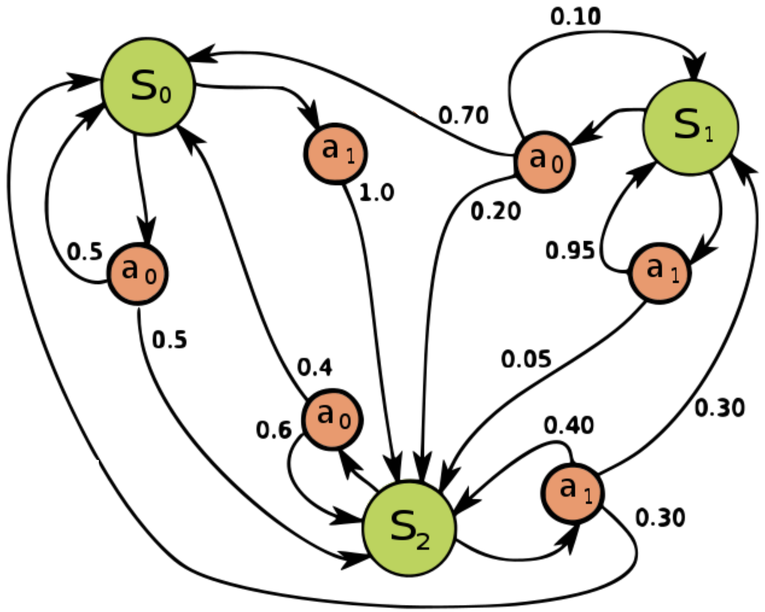


1. The the probability of moving to the next state only depends on the current state, not on the history that lead to that state.
2. R. C. Cheung, A user-oriented software reliability model, IEEE TSE, no. 2, pp. 118–125, 1980

Markov Decision Process

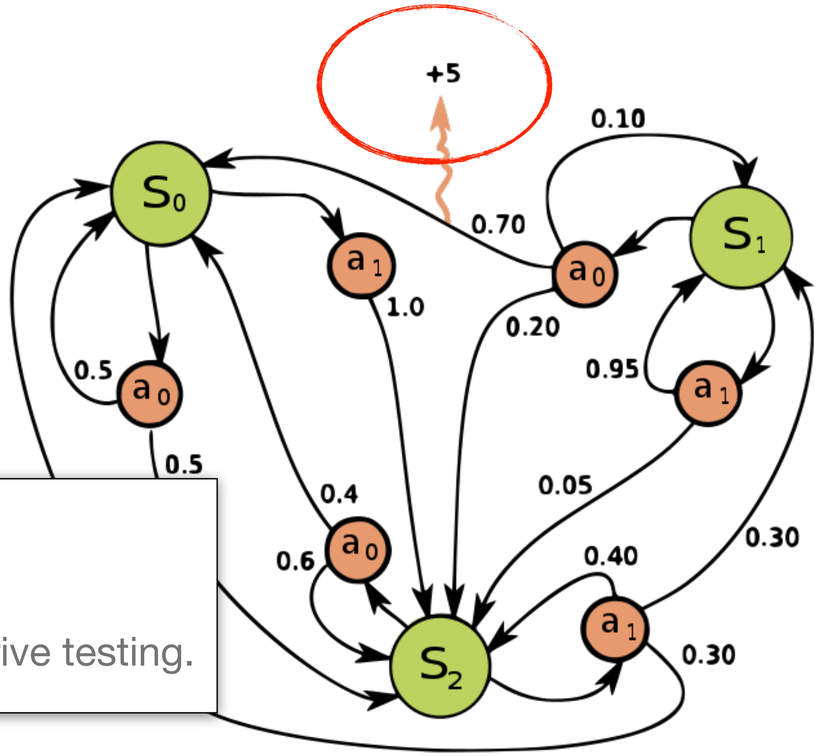
- Mathematical framework for modeling systems whose behavior is partially
  - Nondeterministic — actions (external stimuli) under the control of a decision maker
  - Stochastic — random outcome out of an executed action

- Formal structure
  - S: set of states (finite/infinite)
  - s_0: initial state
  - A: set of actions (alphabet)
  - P: S x A x S —> [0,1], $P(s, a, s') = P(s_{t+1} = s' \mid s_t = s, a_t = a)$



MDP with rewards

- An MDP model can be augmented with one or more reward structures
- Reward structure
  - R: S x A x S —> $\mathbb{R}$, R(s, a, s') reward for (s_{t+1} = s', s_t = s, a_t =a)
  - Describe nonfunctional aspects (e.g., energy consumption, computational cost, response time, …)

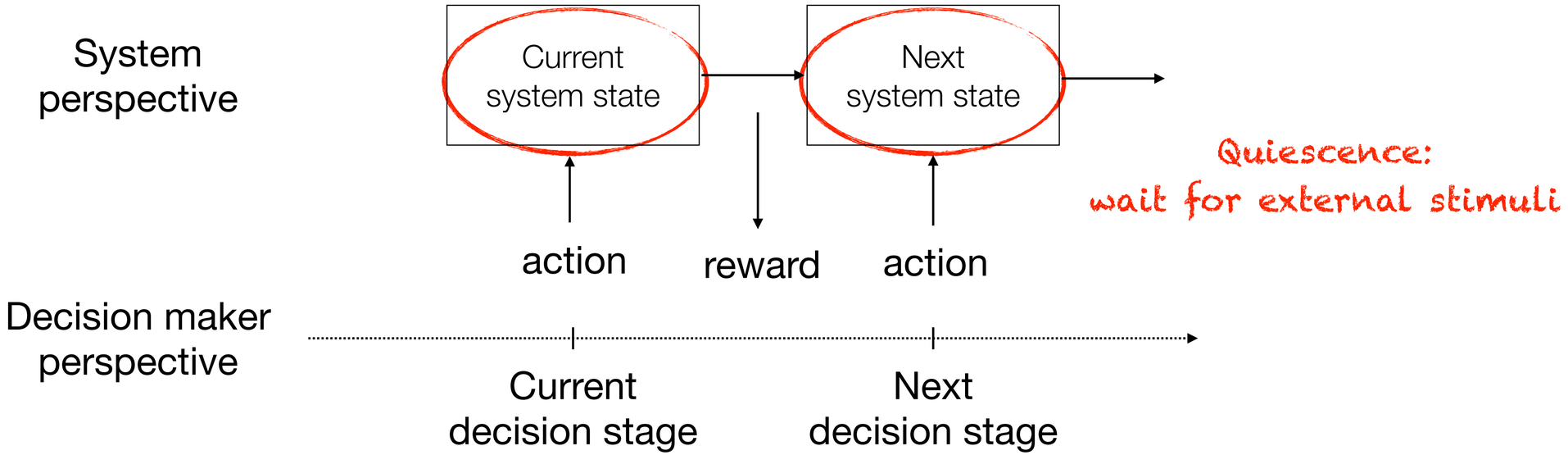


Mental note

This is the usual interpretation of a reward structure.
We'll see how to leverage this notion in a "unconventional" way to drive testing.

MDP behavior

- How does the model operate
  - The system must be in **one** of the **states** (finite countable set) at a time
  - The system makes a transition **s —> s'** when **one** of the available **actions** is selected



MDP behavior (2)

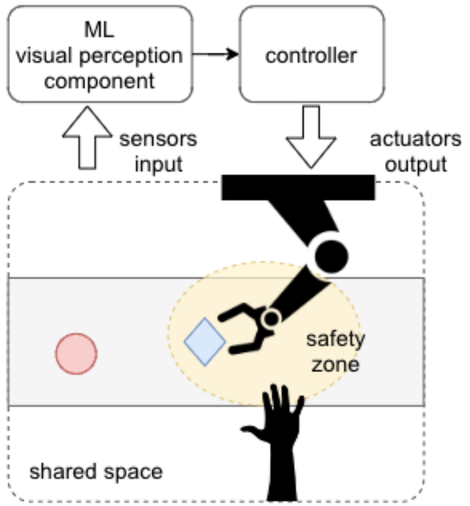
- States
  - Describe configurations or status of the shared phenomena
  - Instances where
    - Specific tasks are carried out
    - Operating in a fully-functioning mode, degraded mode, faulty, etc.
    - Undergoing recover/repair
    - Shared state initiated by the world (or the other way around)
- Actions
  - Possible inputs or external events
- Transitions
  - Define whether is possible to go from one state to another
  - Transition probability —> governs the likelihood of observing the transition

MDP examples

- Possible scenarios
  - Service-based systems
  - Web applications
  - Mobile applications
  - Cyber physical systems (CPSs)
  - Control policies in robotics
  - Security protocols
  - etc.
- Selected example
  - Human-robot collaborative system — example of cyber-physical system

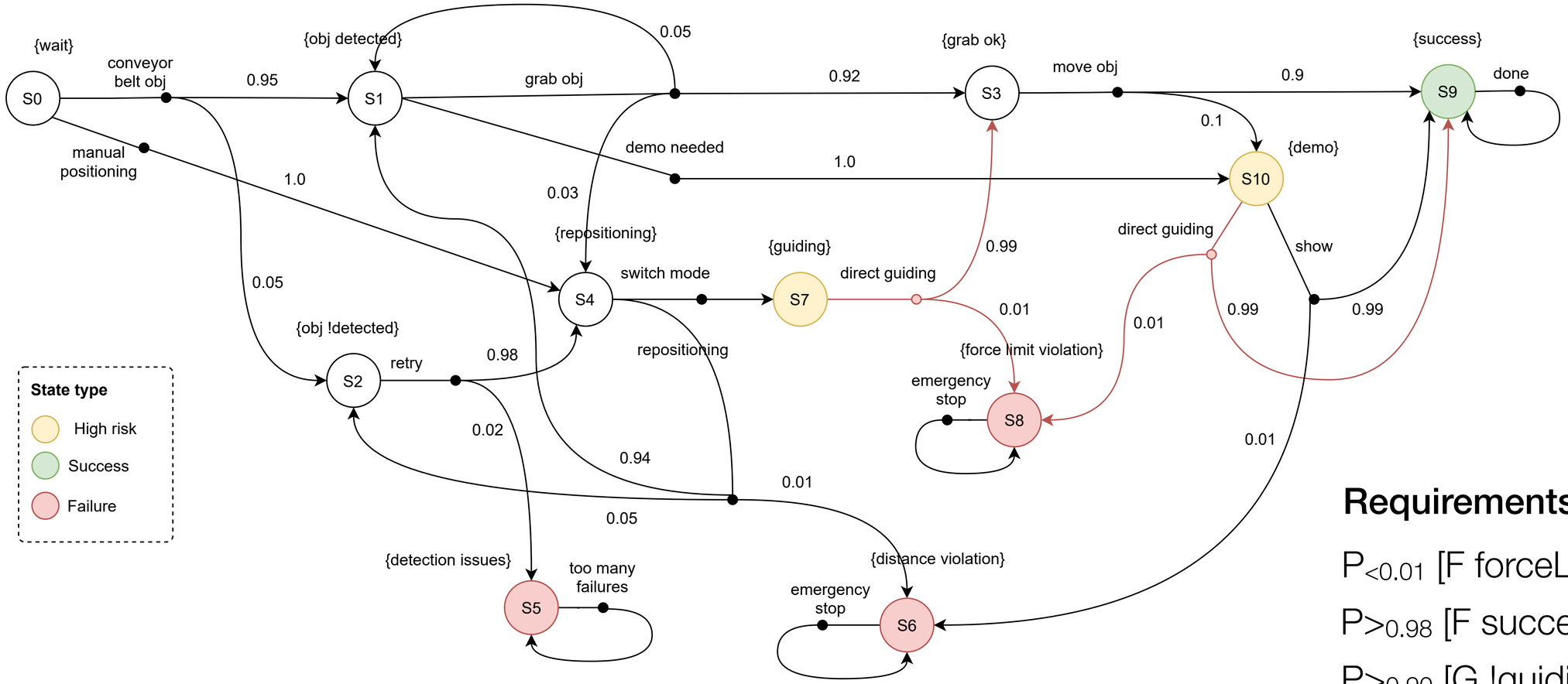
MDP example — human-robot collaboration

- Robotic arm detects and moves object from a conveyor belt into the right bucket
- The system is trained online by a human operator that can either:
  - show how to handle objects (protective separation shall be maintained); or
  - guide directly the movements of the arm (power and force limiting shall be applied)
- Industry 4.0 demo running at the ARENA Lab[1] (Fraunhofer Italia)



1. https://www.fraunhofer.it/de/leistungsangebot/fraunhofer-italia-application-center-arena-.html

MDP example — human-robot collaboration



Requirements (PCTL)

$P_{<0.01}$ [F forceLimitViolation]

$P_{>0.98}$ [F success]

$P_{>0.90}$ [G !guiding & !demo]

MDP policy

- The notion of policy π refers to the way a Decision Maker (DM) solves nondeterminism of a MDP
- Deterministic policy[1,2]
  - π: S —> A, prescribes the action to take given a state
  - DM objective: choose π which maximizes the expected cumulated reward over an infinite horizon
    - This is called best policy π*
- Definition of the best deterministic policy
  - Given R(s, a), i.e., the one-step expected reward
  - We can compute the value function V(s) for each state

$$R(s,a) = \sum_{s' \in S} p_{s,a,s'} \; r_{s,a,s'}$$

Bellman's equation

$$V(s) = \max_{a \in A} \{R(s,a) + \gamma \sum_{s' \in S} p_{s,a,s'} V(s')\}$$

Best policy

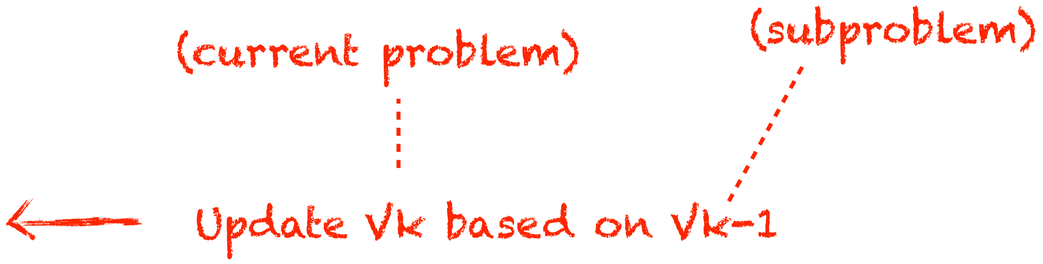
$$\pi^*(s) = \arg\max_{a \in A} \{R(s,a) + \gamma \sum_{s' \in S} p_{s,a,s'} V(s')\}$$

1. Given a deterministic policy, the MDP reduces to a Discrete Time Markov Chain (DTMC).
2. Martin L. Puterman. 1994. Markov Decision Processes: Discrete Stochastic Dynamic Programming (1st. ed.). John Wiley & Sons, Inc., USA

MDP policy — Value iteration

1: **Procedure ValueIteration**(S,A,P,R,θ)
2: **Inputs**
3: S set of all states
4: A set of all actions
5: P transition function P(s,a,s')
6: R reward function R(s,a,s')
7: θ a threshold, θ>0
8: **Output**
9: π[S] optimal policy
10: V[S] value function
11: **Local**
12: real array V_k[S] is a sequence of value functions
13: action array π[S]
14: assign V_0[S] arbitrarily
15: k ←0
16: **repeat**
17: k ←k+1
18: **for each** state s **do**
19: $V_k[s] = \max_a \sum_{s'} P(s,a,s') (R(s,a,s') + \gamma V_{k-1}[s'])$
20: π[S] = a
21: **until** ∀s $|V_k[s] - V_{k-1}[s]| < \theta$
22: **return** π,V_k

- The ValueIteration procedure uses dynamic programming
  - Memoization + recursion (or iteration)
- This procedure converges no matter what is the initial value function V_0



Model-based testing of probabilistic systems

- Offline vs online approaches
- Conformance relation
- Probabilistic alternating simulation and refinement

Model-based testing

- Model-based testing vs Formal verification
  - MBT is testing! provides runtime evidence that the SUT behaves as defined in the (verified) specification
    - Limitation: testing is not complete (i.e., *“testing can only show the presence of errors, not their absence”* [D. Parnas])
  - Formal verification — prove that the model (i.e., formal specification) satisfies requirements
- Basic idea
  - A model that specifies the behavior of the System Under Test (SUT) is used as baseline of
    - test case generation
    - Construction of the oracle
  - Test suites are automatically extracted from models and then executed

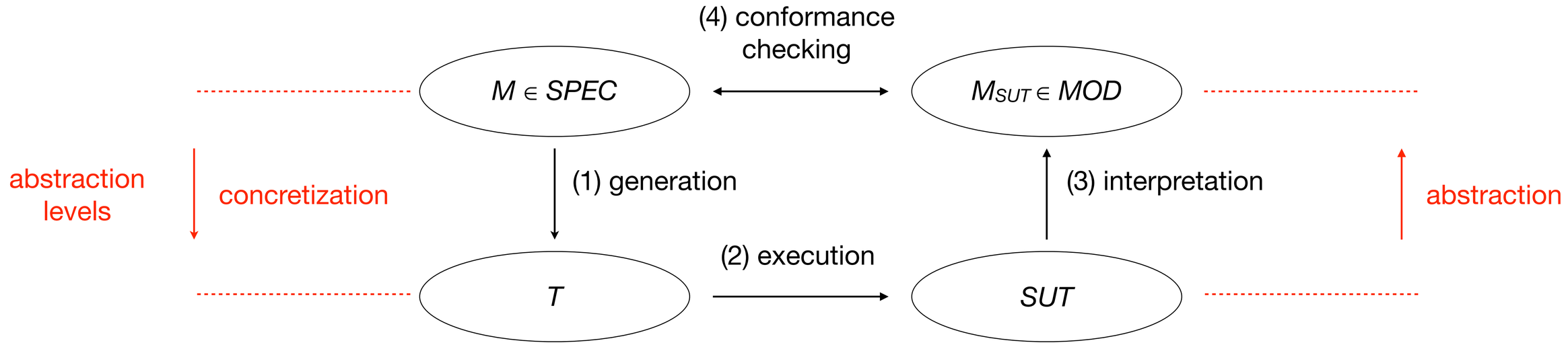
MBT — terminology

- System (or implementation) Under Test
  - Piece of hardware/software, a software system, an embedded system, a CPS, etc.
  - The SUT is viewed as a black-box (secret internal structure)
  - The tester controls and observes the SUT via its interfaces (e.g., sensors, actuators, APIs)
- Specification
  - Describes what the SUT should do using a formal notation (or language)
  - *SPEC* — set of all valid models in a formal notation
    - A specification is $M \in SPEC$
- Conformance
  - Formalizes the notion of correct behavior of a SUT w.r.t. $M \in SPEC$
  - Problem: we'd like to define a relation between elements of different domains
    - M (formal entity) <—> SUT (not a formal entity)

MBT — terminology (2)

- Conformance Problem: *M* (formal domain) <—> SUT (not formal domain)
  - Trick — test assumption
    - Interpretation of the SUT behavior using the same level of abstraction of *M*
    - SUT behavior is a model $M_{SUT} \in MOD \subseteq SPEC$
      - MOD — universe of implementation models
      - M_{SUT} not a-priori known
- Conformance (under the test assumption)
  - expressed as a binary relation between MOD and $SPEC$ elements
  - $conf \subseteq MOD \times SPEC$
  - M_{SUT} is correct w.r.t. M if $M_{SUT}\ conf\ M$
- Conformance checking
  - Assess by testing whether $M_{SUT}\ conf\ M$
  - *Create* T *(test suite) s.t.* $M_{SUT}\ conf\ M \implies M_{SUT}$ *passes* T

MBT process

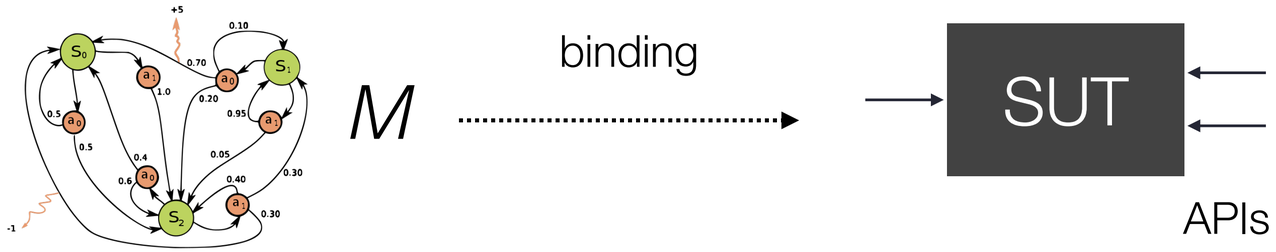


- Offline vs online [1]
  - Offline — steps 1-4 are separated
  - Online (or on-the-fly) — steps 1-4 are merged into a one-iteration step
    - Test cases are created dynamically and take advantage of the knowledge gained by exploring *M*
    - Iterative approach —> 2-players game: controller + observer

1. Utting, Mark, and Bruno Legeard. Practical model-based testing: a tools approach. Elsevier, 2010

Online MBT with MDPs

- Binding (concretization)
  - Defines a mapping between the MDP model spec and the SUT behavior



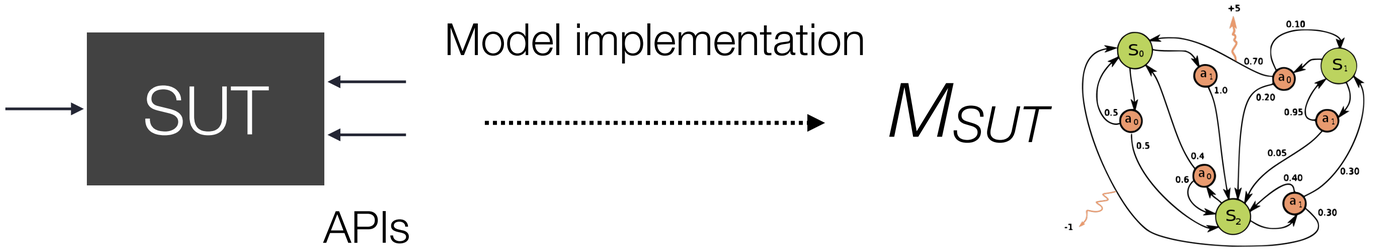
- Formal definition
  - Given a MDP $M = (S, s_0, A, P)$ and a SUT, i.e., a set of exported operations H (having signature and arguments), a binding is a tuple of partial functions $(h, i, post)$ s.t.
    - $h(s, a),\ a \in A(s)$ identifies an operation in H
    - $i(s, a),\ a \in A(s)$ identifies a vector v_{in} for $h(s, a)$
    - $post(s, a, s'),\ a \in A(s)$ maps to a post-condition that must hold for v_{out} outcome of $h(s, a)$ on input v_{in}

Controllable SUT components

Observable SUT behavior

Online MBT with MDPs (2)

- Model program (abstraction)
  - Defines the abstract interpretation of the SUT behavior in terms of MDP model



- Formal definition
  - Given a MDP $M = (S, s_0, A, P)$ and a binding $(h, i, post)$ the model program is a MDP $M_{SUT} = (S', s'_0, A', P')$ s.t.
    - S' ⊆ S
    - A ⊆ A'
    - $P'(s, a, s') > 0$ iff $post(s, a, s')$ holds for v_{out} with $v_{out} = h(s, a)(v_{in})$

All observable SUT states exist in M

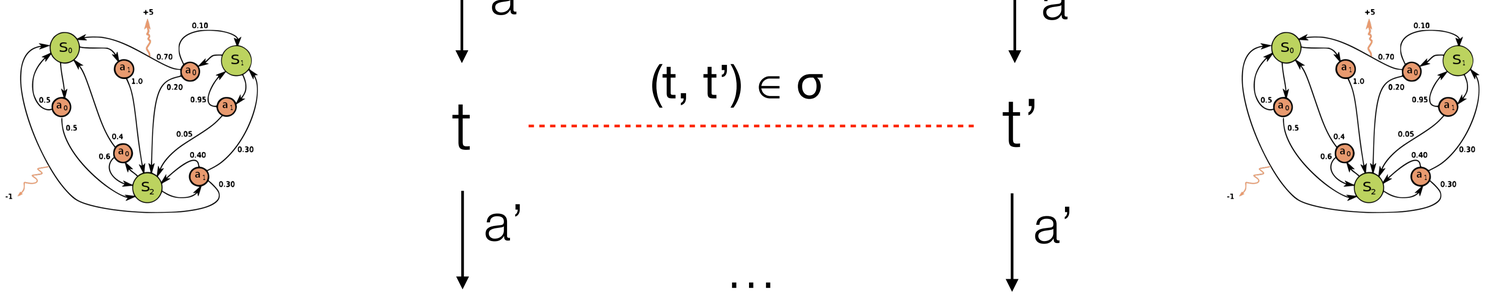
All controllable actions in M are feasible in SUT

M_{SUT} transitions are defined in terms of SUT behavior

Online MBT with MDPs (3)

- Conformance checking
  - Defined in terms of probabilistic alternating simulation + refinement

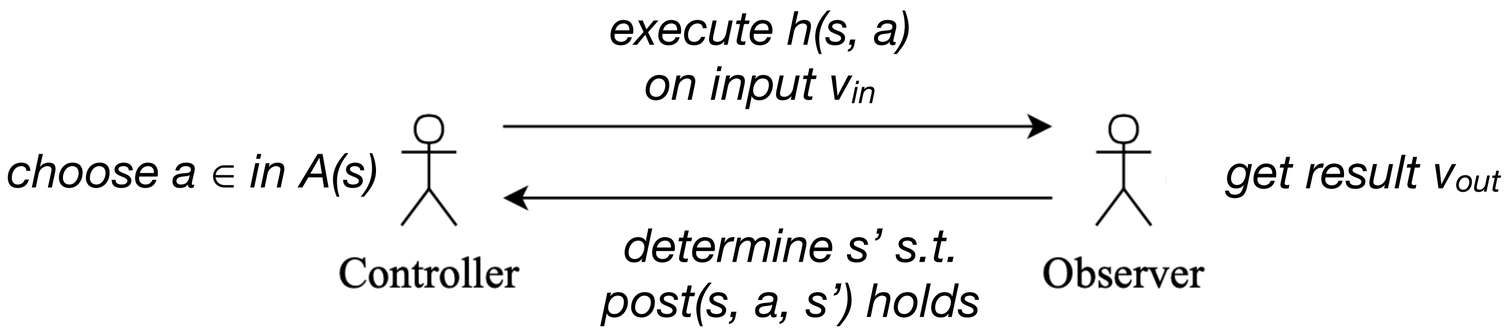
- Probabilistic alternating simulation
  - between M and M_{SUT} is a binary relation $\sigma \subseteq S \times S'$, s.t. for all $(s, s') \in \sigma$
    - $A(s) \subseteq A'(s')$
    - For each $t \in S : P(s, a, t) > 0$, there exists $t' \in S' : P(s', a, t') > 0$ and $(t, t') \in \sigma$



Online MBT with MDPs (4)

- Refinement
  - M_{SUT} refines M iff there exists a probabilistic alternating simulation σ s.t. $(s_0, s'_0) \in \sigma$

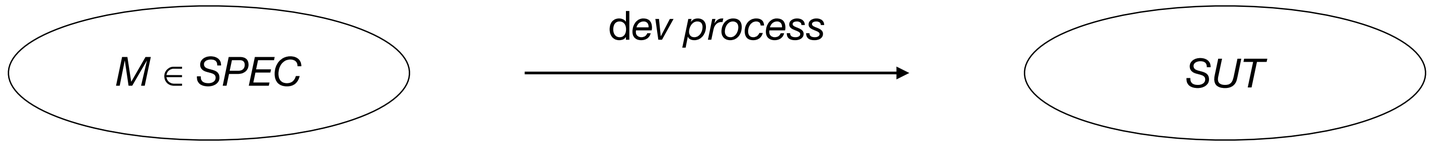
- Conformance game
  - The notion of refinement is verified in practice by means of a conformance game between
    - Controller —> chooses actions based on a given test case generation strategy
    - Observer —> verifies the result out of a test execution



Online MBT under Uncertainty

- Problem statement
- Uncertain model paramenters
- Bayesian inference
- Framework and test case generation strategies

Problem statement



Is it complete?
or is it based on partial knowledge?

How to perform MBT if part of M is uncertain?

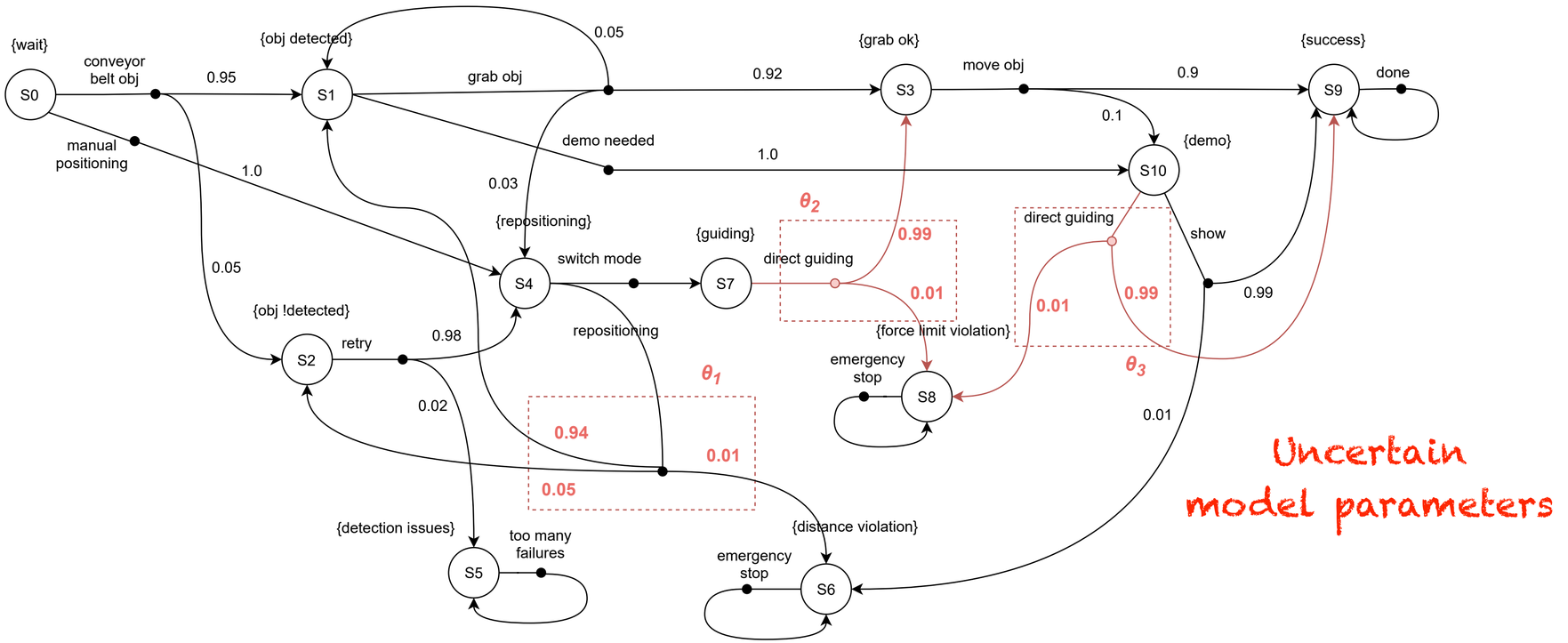
- Problem
  - Design-time models are imperfect and include assumptions
  - Assumptions are affected by sources of uncertainty
  - Uncertain aspects in the human-robot interaction scenario:
    - Outcome of the visual perception component (depending on environment conditions)
    - Safety guarantees in close-collaboration settings (human misbehavior, emergency brake latency)

The very idea

- Objective
  - Refine partial knowledge of design-time assumptions through runtime evidence
- Assumption
  - Sources of uncertainty affect model parameters (i.e., uncertain transition probability values in the MDP)
- How
  - Calibration of uncertain model parameters during system testing by combining
    - Online Model-based Testing
    - Bayesian inference

Uncertain model parameters

- Uncertainty in the human-robot collaboration
  - The robotic arm could violate the protective distance or the power and force limit during close interaction with the operator
  - This depends on the capability of the visual perception component as well as the ability to promptly actuate the right movements
  - Model parameters (probability values) governing such a behavior are uncertain



Uncertain regions

- Uncertain region
  - Set of uncertain transition probability values θ_i identified by a <state, action> pair
  - θ_i is formally described through a Categorical distribution $\theta_i \sim Cat(p_1, \ldots, p_k)$

region	state-action	target states	distribution
θ_1	$(s_4, repositioning)$	s_1, s_2, s_6	$Cat(0.94,0.05,0.01)$
θ_2	$(s_7, directGuiding)$	s_3, s_8	$Cat(0.99,0.01)$
θ_3	$(s_{10}, directGuiding)$	s_8, s_9	$Cat(0.99,0.01)$

Uncertain parameters (hypothesis)

- Intuition
  - Mitigate the uncertainty of θ regions by observing (multiple times) the SUT
  - Run-time observations provide evidence to update the uncertain regions

Bayesian inference

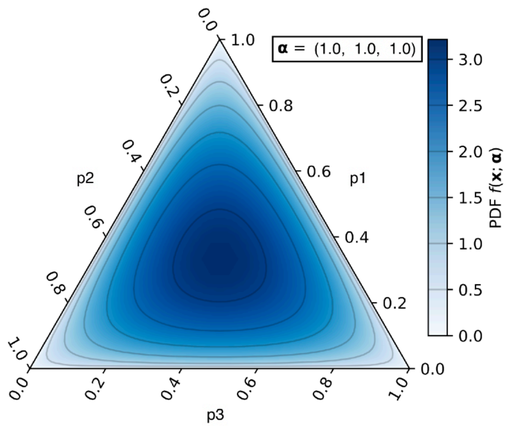
- Statistical machinery used to update a probability distribution that describes θ based on the evidence
- Formulation [1]
  - To learn θ we collect a sample $y = (y_1, \ldots, y_n)$
  - $Posterior \propto Likelihood \cdot Prior$
    - $Posterior\ f(\theta \mid y)$ — *updated knowledge given the evidence*
    - $Prior\ f(\theta)$ — *hypothesis*
    - $Likelihood\ f(y \mid \theta)$ — *compatibility of the evidence given the hypothesis*
- In our context
  - The natural conjugate Prior of the Categorical distribution is the Dirichlet distribution
  - $f(\theta_i) \sim Dir(\alpha_1, \ldots, \alpha_k),\ \alpha_i > 0$

1. Robert, Christian. The Bayesian choice: from decision-theoretic foundations to computational implementation. Springer Science & Business Media, 2007

Bayesian inference (2)

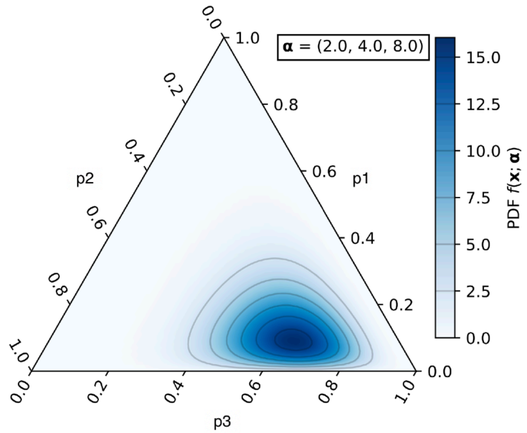
- Updating rule
  - $f(\theta) \sim Dir(\alpha_1, \ldots, \alpha_k) \rightarrow f(\theta \mid y) \sim Dir(\alpha_1 + n_1, \ldots, \alpha_k + n_k)$

$f(\theta_1) \sim Dir(1.0,1.0,1.0)$



Observations:
1xs1, 4xs2, 7xs6

$f(\theta_1 \mid y) \sim Dir(1.0 + 1.0,1.0 + 4.0,1.0 + 7.0)$



Bayesian inference (2)

- Summarization
  - Prior/Posterior knowledge can be summarized by using
    - Punctual summarization Mean: $p_i = \alpha_i / \sum_{j=1}^{k} \alpha_j$
    - Interval summarization Highest Density Region $HDR : P(\theta \in HDR) \geq 0.95$

$f(\theta_1) \sim Dir(1.0,1.0,1.0)$ → $f(\theta_1 \mid y) \sim Dir(2.0,4.0,8.0)$

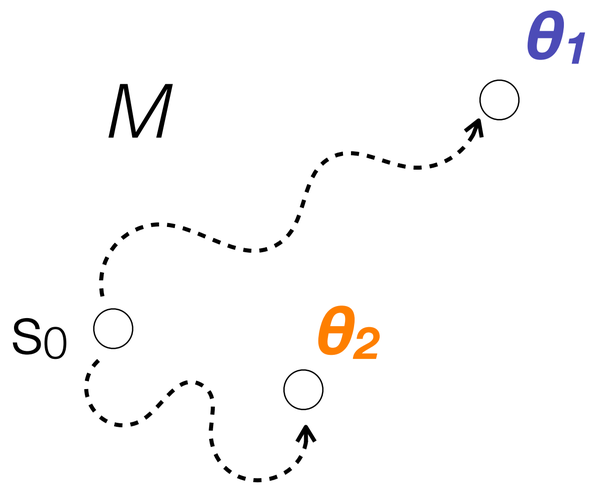
Mean = { 0.33, 0.33, 0.33 }
HDR = { [3.6E-6, 7.8E-1], [2.7E-6, 7.7E-1], [1.4E-5, 7.8E-1] }
HDR magnitude = 0.78

Mean = { 0.14, 0.29, 0.57 }
HDR = { [0.0, 0.31], [0.07, 0.51], [0.32, 0.81] }
HDR magnitude = 0.41

Lower value -> higher accuracy

Online MBT + Bayesian inference

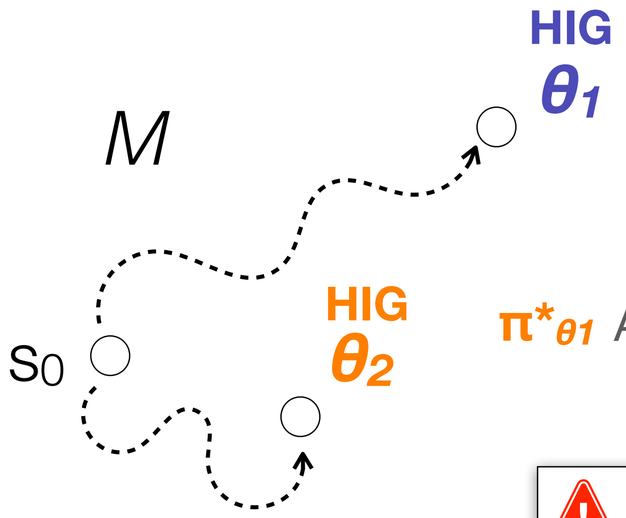
- Objective (reminder)
  - Collect runtime evidence and refine uncertain assumptions
- How
  - Perform a controlled exploration using online MBT to stress the uncertain model regions
  - Gather evidence and run bayesian inference to reduce the uncertainty



- Uncertainty-aware MBT strategy
  - Explore by maximize the probability of reaching θ regions
  - Reduces to an optimization problem:
    - Find out the actions a decision maker should take to maximize the exploration of θ regions

Uncertainty-aware strategy

- Computation of the best policies
  - For each θ_i
    - construct a reward structure that assigns HIG reward to θ_i transitions, LOW elsewhere
    - Compute the best policy $\pi^*_{\theta i}$ (value iteration)
      - For each state, it selects the action that maximizes the probability to reach θ_i



$\pi^*_{\theta 1}$ Actions maximizing the probability to reach θ_1

$\pi^*_{\theta 1}$ Actions maximizing the probability to reach θ_1

Back to our mental note

We'll see how to leverage rewards in a "unconventional" way to drive testing.

Uncertainty-aware strategy (2)

- How to combine the best policies $\pi^*_{\theta i}$?
  - Simple scenario —> there exists just a single θ region
  - Otherwise —> different exploration strategies may be constructed/adopted
    - Strategies represent decision makers (i.e., testers) that use a probabilistic function

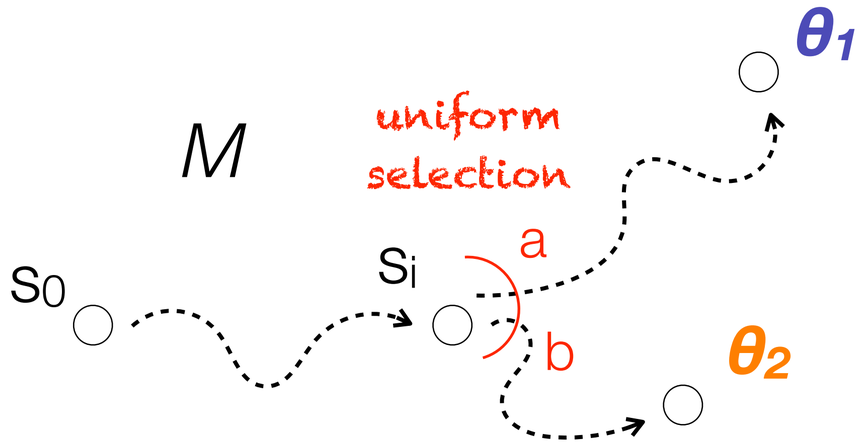
$$\mathcal{P}(s,a) = \begin{cases} 0 & \omega(s,a) = 0 \\ \omega(s,a) / \sum_{a' \in A(s)} \omega(s,a') & otherwise \end{cases}$$

    - The ω weight selectively increase/decrease the probability of choosing a specific action *a* from state *s*

Uncertainty-aware strategy (3)

- Flat strategy
  - Actions selected by different policies $\pi^*_{\theta i}$ have equal probability
  - Uniform random sampling of the available policies

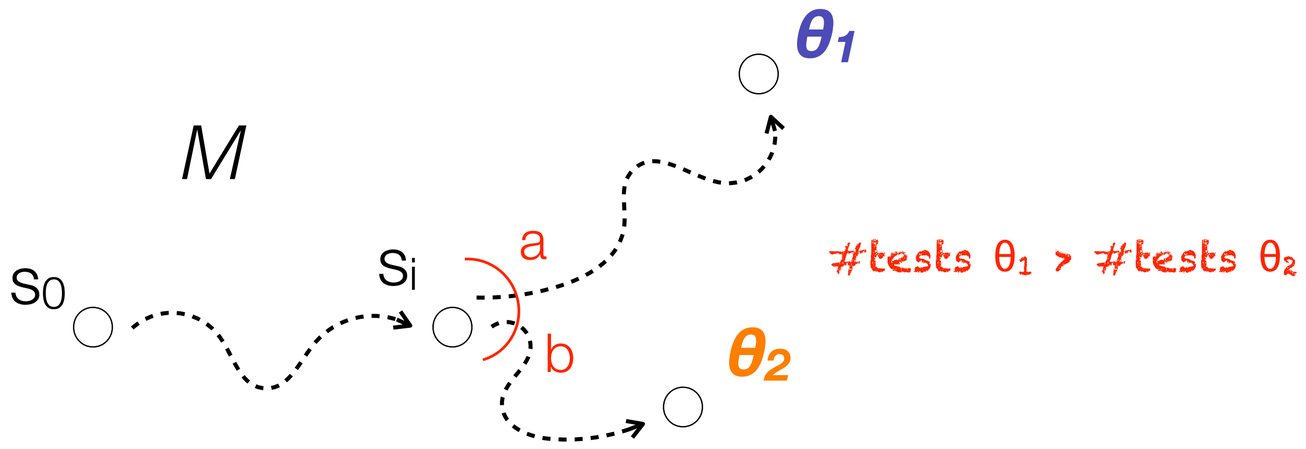
$$\omega^{RT}(s,a) = \begin{cases} 1 & \exists i : \pi_i^*(s) = a \\ 0 & otherwise \end{cases}$$



Uncertainty-aware strategy (4)

- History-based strategy
  - Tries to keep balanced the number of times θ regions are tested
  - We leverage decrementing weights inversely proportional to #selections of state-action pairs

$$\omega^{HT}(s,a) = \begin{cases} 1/\#(s,a) & \exists i : \pi_i^*(s) = a \\ 0 & otherwise \end{cases}$$

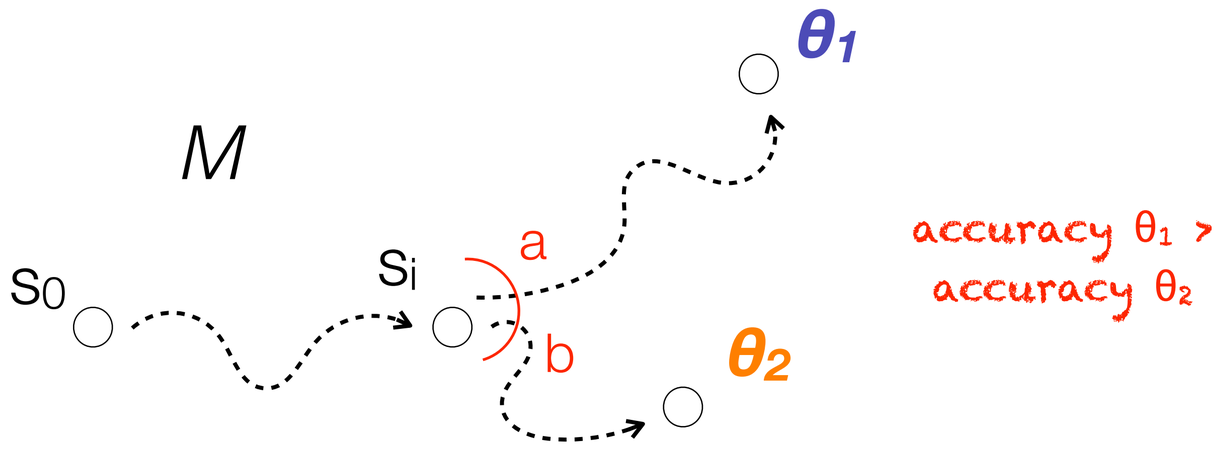


the higher the #selections of a,
the lower the likelihood of selecting it again

Uncertainty-aware strategy (5)

- Distance strategy
  - Tries to deliver uniform degree of accuracy for all θ
  - The weight is proportional to the HDR magnitude of θ_i

$$\omega^{DT}(s,a) = \begin{cases} |\mathrm{HPD}_{\theta_i}| & \exists i : \pi_i^*(s) = a \\ 0 & otherwise \end{cases}$$



the larger the |HDR| of the target θ,
the higher the likelihood of selecting the corresponding action

Termination condition

- Limit on the effort
  - Traditional termination condition based on #tests limit
- Bayes factor
  - Tries to recognize when the inference process converges

$$\mathcal{F} = \frac{f(y|\theta)}{f(y|\theta')}$$

Likelihood that data y are produced under different assumptions θ and θ′

$$\mathcal{F} \in [10^0, 10^{1/2}]$$

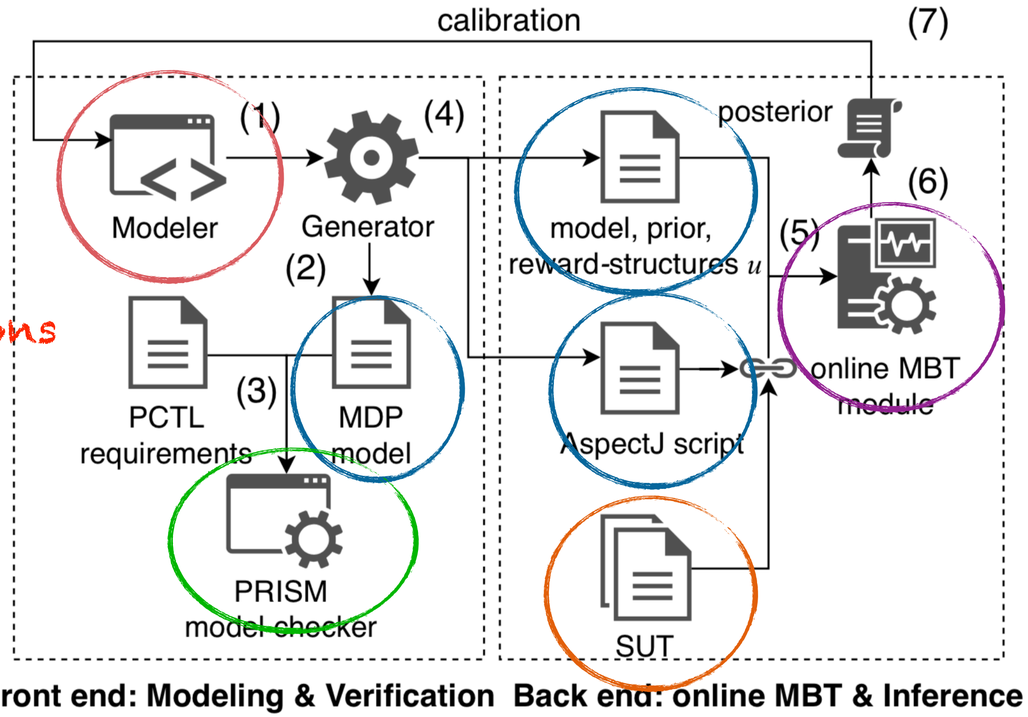
Difference between assumptions θ and θ′ is not substantial

Current toolchain implementation

MDP definition + uncertain transition probabilities θ

binding to the SUT:
actions -> inputs
acs -> routine postconditions

automatic generation



online MBT and inference/ calibration

model checking of PCTL requirements

Java program

Current toolchain implementation (2)

MBT module

https://github.com/SELab-unimi/mbt-module

- Current stage
  - Uncertainty-aware strategies have been implemented
  - Systematic evaluation of their cost-effectiveness —> currently inedited

Current toolchain implementation (3)

- Evaluation summary
  - We assessed statistical difference (Mann-Whitney U test [1])
  - We evaluated practical value (Vargha & Delaney's A^{12} measure [1])
    - Assuming same effort (i.e., #tests), the probability that target strategy yields smaller HPD width values than flat one (i.e., baseline)

Table 3: Vargha and Delaney's $\hat{A}_{12}$ measure

	%uncertainty			#actions		
balanced	20	50	80	5	10	20
hist	1.000	0.716	0.531	0.617	0.704	0.926
dist	1.000	0.790	0.679	0.741	0.802	0.951
unbalanced	20	50	80	5	10	20
hist	0.963	0.716	0.556	0.531	0.642	0.901
dist	0.988	0.951	0.691	0.741	0.741	0.975

1. Andrea Arcuri and Lionel Briand, A practical guide for using statistical tests to assess randomized algorithms in software engineering, ICSE'11, New York, NY, USA

Summary

- We discussed MBT and the problem of testing with uncertain model regions
- Depending on the Prior knowledge (hypothesis) and information that can be collected during testing, we derived different uncertainty-aware exploration strategies and evaluated their cost-effectiveness
  - Flat —> uniform selection
  - History —> balanced exploration
  - Distance —> balanced delivered confidence
- Next
  - Hands on session with the MBT module
  - Design/develop an additional exploration strategy

One more thing: collaboration opportunities

- Relationship among uncertain regions
  - Identify the uncertain regions depending on other uncertain regions along execution paths
  - Compute Posterior probabilities following the discovered relationships
- Refactoring of MDP models
  - Identify critical portion of the models w.r.t. requirements
  - Model-based refactoring and evaluation
- Automatic construction of MDP model from the implementation
  - Static analysis or symbolic execution
- Testing using Reinforcement Learning (based on MDP theory)
  - Discover the location of the uncertain regions
- PCTL falsification
  - Combination of metaheuristic optimizing search and MBT to find environment conditions that leads to violate PCTL requirements

References

- Camilli M., Gargantini A., Scandurra P., Bellettini C. (2017) Towards Inverse Uncertainty Quantification in Software Development (Short Paper). In: Cimatti A., Sirjani M. (eds) Software Engineering and Formal Methods. SEFM 2017. LNCS, vol 10469. Springer, Cham.
- M. Camilli, C. Bellettini, A. Gargantini and P. Scandurra, (2018) Online Model-Based Testing under Uncertainty, IEEE 29th International Symposium on Software Reliability Engineering (ISSRE), Memphis, TN, 2018, pp. 36-46.
- M. Camilli, A. Gargantini, R. Madaudo and P. Scandurra, (2019) HYPpOTesT: Hypothesis Testing Toolkit for Uncertain Service-based Web Applications. In proceeding of the 15th International conference on integrated Formal Methods. iFM2019. LNCS, vol 11918. Springer, Cham.
- M. Camilli, A. Gargantini, and P. Scandurra, (2020) Model-based Hypothesis Testing of Uncertain Software Systems, Software Testing Verification and Reliability, John Wiley & Sons, Ltd. https://doi.org/10.1002/stvr.1730
- M. Camilli, and B. Russo, (2020) Model-Based Testing Under Parametric Variability of Uncertain Beliefs. International Conference on Software Engineering and Formal Methods. Springer, Cham.
- M. Camilli, A. Gargantini, P. Scandurra, and C. Trubiani (2021) Uncertainty-aware Exploration in Model-based Testing, IEEE International Conference on Software Testing, Verification and Validation 2021 (ICST), To appear.

License of these slides