An abstraction framework for mixed non-deterministic and probabilistic systems

Michael Huth

Department of Computing, Imperial College London, 180 Queen's Gate, London, SW7 2BZ, United Kingdom, huth@doc.ic.ac.uk

Abstract. We study abstraction techniques for model checking systems that combine non-deterministic with probabilistic behavior, emphasizing the discrete case. Existing work on abstraction offers a host of isolated techniques which we discuss uniformly through the formulation of abstracted model-checking problems (MCPs). Although this conceptualization is a useful focal point for surveying the literature on abstraction-based model checking even beyond such combined systems, it also opens up new research opportunities and challenges for abstract model checking of mixed systems. In particular, we sketch how quantitative domain theory may be used to specify the precision of answers obtained from abstract model checks.

1 Motivation and objectives

This survey deals with the abstraction of dynamical systems whose computations are guided by a mix of non-deterministic and probabilistic (or quantitative) behavior. For sake of brevity, this paper uses the term "mixed models" as a general reference to models of such systems. E.g. a Markov decision process is a mixed model, since its dynamics strictly alternates between non-deterministic and probabilistic state transitions. Concurrent or under-specified randomized algorithms are another example of mixed models and we will meet more kinds of mixed models in this survey. Although the proper design of models is a central concern in engineering at large, this paper will focus on the analysis of mixed models through abstraction. Ideally, analysis activities lead to an incremental and validated design of a computing artifact. Not surprisingly, formal analysis is mandatory for safety-critical systems and becomes increasingly important in business-critical information systems as well. The objectives of this survey are to

- motivate the use of mixed models for the design and analysis of computational systems;
- motivate the need for abstraction techniques in the design and analysis of mixed models;
- present the design and analysis of mixed models as model-checking problems (MCPs);
- propose a unified theory of abstraction for MCPs;

- study main abstraction techniques for mixed models as instances of this unified theory; and
- formulate research challenges for abstracting mixed models and their MCPs.

Outline of paper. In Section 2, we develop a general and quite flexible framework for specifying model-checking problems as the formal topic of our discourse and present an important model-checking problem for Markov decision processes in Section 3. The need for having abstract versions of model-checking problems is discussed in Section 4. In Section 5, we survey the various aspects of model-checking problems that can be subjected to abstractions. Model-checking problems give rise to query equivalences which leads to a discussion of bisimulations in Section 6. In Section 7, we define the approximation problem for abstract model-checking problems and point out connections to work in quantitative domain theory. Section 8 reflects on the need of diagnostics as supporting evidence for answers to model-checking instances and Section 9 lists the caveats for effective use of abstract model-checking problems. Although soundness is at the heart of abstraction-based model checking, Section 10 states the main reasons and circumstances in which one wants to abandon such soundness. In Section 11, we present a subjective list of research challenges in the area of abstraction-based model checking of mixed models and Section 12 concludes.

2 Model-checking problems

The design of a computational system requires a model of this system. Such a model is necessarily an abstraction of the real system. E.g. a hardware design may start out as a mathematical model of a boolean circuit, but its realization needs to take into account cost and physical constraints. Although models abstract from real systems, their abstractions tend to leave out "irrelevant" details only. Models, unlike real systems, can be represented in a computer and may therefore be subjected to formal and automatic analysis. In this paper, we focus on the latter advantage of models over real systems.

Similar systems will have similar kinds of models and queries about those models. We may axiomatize such a "kind" as

- a class $\mathcal{M}$ of models ($M \in \mathcal{M}$);
- a class Φ of queries about such models ($\phi \in \Phi$);
- a partial order $\mathtt{Ans}$ of possible answers to such queries ($a \in \mathtt{Ans}$) — thus also allowing answer domains used for performance modeling (e.g. [42, 44]); and
- a specification $\models : \mathcal{M} \times \Phi \to \mathtt{Ans}$ of how to compute answers to queries over models.

It is customary to use infix notation for applications of $\models$:

$$(M \models \phi) \in \mathtt{Ans}. \tag{1}$$

In this paper, the syntax $M \models \phi$ denotes a *model-checking instance* whose evaluation (1) is referred to as a *model check*. With a set of data as above, we call the tuple

$$\mathcal{P} = (\mathcal{M}, \Phi, \mathtt{Ans}, \models) \tag{2}$$

a *model-checking problem* (MCP). An example of a MCP is SMV [57], where $\mathcal{M}$ is determined by the class of SMV programs, Φ is computation-tree logic (CTL) [17], $\mathtt{Ans} = \{\mathtt{ff} < \mathtt{tt}\}$, and $\models$ is the standard semantics of CTL over Kripke structures [17].

Queries need not have logical structure. For example, models may be computer programs, queries may be questions such as "which portions of the code will never be reached" (dead code detection, e.g. [13]), and $\models$ may be the specification of a family of program analyses [81].

For an example of a MCP with mixed models, consider $\mathcal{M}$ as the class of Markov decision processes and Φ as linear-time temporal logic (LTL) [17]. A suitable partial order of answers is the interval domain [93]

$$\mathcal{I} = \{[r, s] \mid 0 \leq r \leq s \leq 1\} \tag{3}$$

ordered by $[r, s] \leq [r', s']$ iff $r' \leq r$ and $s \leq s'$. The answer $[r, s]$ to a model check $M \models \phi$ is specified to represent

- r: the worst-case probability of M satisfying the LTL formula ϕ;
- s: the best-case probability of M satisfying the LTL formula ϕ.

We denote this MCP by MDP subsequently.

3 Markov decision processes

MDP is a natural choice for modeling probabilistic algorithms. E.g. for the Java method in Figure 1, based on [75], we may want to know the best-case/worst-case probabilities of satisfying the post-condition, given the pre-condition. If we are interested in computing the interval $(M \models (0 \leq x \leq 2) \rightarrow \mathsf{G}(x < 3))$ — where G denotes "**g**lobally" or "at all future points" — , then a natural model M of this method is depicted in Figure 2. Non-determinism results from the under-specification of the parametric method input x. Probabilistic behavior is caused by the method's coin flip. The model in Figure 2 is an abstraction, since it already assumes the pre-condition in its non-deterministic choice and abstracts $x \geq 3$ into a single state.

The models M of a MCP $(\mathcal{M}, \Phi, \mathtt{Ans}, \models)$ are typically represented symbolically in a programming language (e.g. `Java`), a specification language (e.g. process algebras [45, 69, 2, 41]) or through data structures that implement model checks (e.g. binary decision diagrams [9], their multiple-valued variants MTBBDs [16], and ADDs [91]). A structural operational semantics [89] then maps such syntactic structures to full mathematical models. Abstraction and soundness of

```
int aRandomUpdate(x : int) {   // pre-condition: 0 <= x <= 2
  for (i = 0; i < 5; i++) {
    // coin_flip() returns 0 or 1 with probability 0.5
    x = x + coin_flip();
  }
  return x;
}   // post-condition: x < 3
```

Fig. 1. A probabilistic method with return type `int` and qualitative pre-conditions and post-conditions.

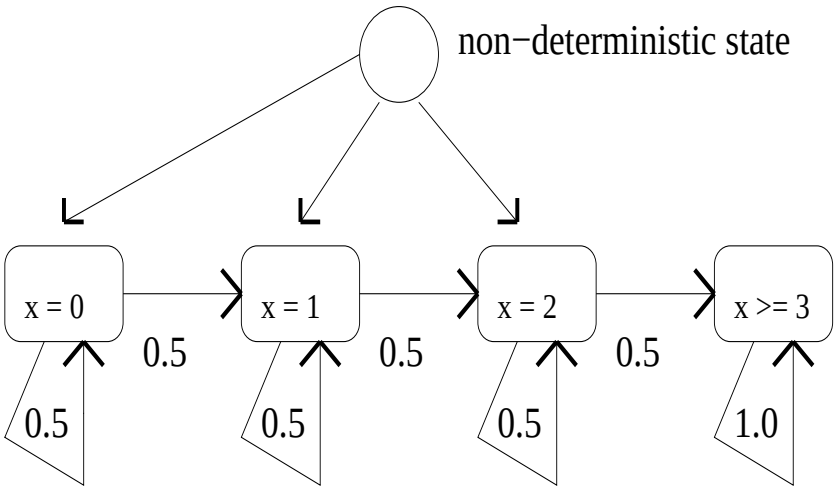

Fig. 2. A mixed model of the method `aRandomUpdate` of Figure 1. Non-deterministic nodes and probabilistic nodes may not alternate strictly. The choice of model was informed by the structure of the program and the query.

abstraction are typically defined and proved for those mathematical models, e.g. the abstract model in Figure 2.

In our example, the interval $(M \models (0 \leq x \leq 2) \rightarrow \mathsf{G}(x < 3))$ equals $[0, 0]$ since any non-deterministic choice results in a state that eventually has to increase the value of `x`. Thus, one needs an abstract interpretation of these checks which restricts the scope of the G modality to the first five states on each path. This is sound since the method's sole for-statement executes exactly five times. We may reduce the computation of this check to the one of $(M \models \mathsf{F}(x >= 3))$ since the precondition is enforced in the model and since F and G are logical duals. Thus, we now evaluate a bounded-until query

$$\mathtt{tt}\,\mathsf{U}^{\leq 5}\,(x >= 3) \tag{4}$$

whose semantics $[r, s]$ can be computed through fixed-point iterations or linear programming techniques. We leave the computation of these values as an exercise. Neither r nor s are conventional probabilities. For r, a fictitious opponent of the verifier controls the initial value of `x` with the aim of minimizing that probability, choosing `x == 0`. Dually, for s the opponent turns into a collaborator who chooses `x == 2` in order to maximize that value. The meaning of $\mathsf{G}(x < 3)$ is then $[1-s, 1-r]$; this negation swaps the roles of opponent and collaborator, the opponent now chooses `x == 2` and the collaborator picks `x == 0`. These different roles of controller and collaborator are also reflected in the process algebra CSP in the form of demonic and angelic choice (respectively) [45].

The two numbers r and s are the extremal outcome points of a set of possible opponent strategies. In that sense, the interval $[r, s]$ is already an abstraction of itself. For example, an opponent may put a uniform probability measure on the pre-condition domain of `x`, resulting in a form of probabilistic testing which turns the model into a fully probabilistic one, a Markov chain. Since each value of `x` determines a likelihood $f(x)$ of the post-condition's being true, we obtain a measurable function f from the value domain of `x` to $[0, 1]$ and the post-condition's expected likelihood is then the integral $\int f \mathrm{d}\mu$ of that function f over the probability measure μ on the values of `x`. In our example, let that measure μ assign $\frac{1}{3}$ to each value 0, 1, and 2. Then the expected likelihood in the worst-case equals the one in the best-case since all non-determinism is resolved by the distribution on the method's input. We compute this likelihood as $\int f \mathrm{d}\mu = \frac{1}{3} \cdot p_0 + \frac{1}{3} \cdot p_1 + \frac{1}{3} \cdot p_2$, where p_i is the probability of the trace property in question at state `x == i`; these are actual probabilities since these states are not and cannot reach non-deterministic states. Soundness of this interpretation means that the expected likelihood be in between the angelic (s) and demonic (r) one.

The origin of these semantic ideas can be traced back to Kozen's work [58, 59] which pioneered the semantics of deterministic programs as transformers of measures or measurable functions for deterministic programs; we highly recommend reading these papers. This line of work has been extended by the Probabilistic Systems Group at Oxford University to abstraction and the presence of non-determinism; see e.g. [79]. In their work, mathematical models treat non-determinism as an abstraction of probabilities such that qualitative non-

determinism can be refined by any probabilities. For example, consider the process-algebra specification

$$(\texttt{heads} \oplus_{.25} \texttt{tails}) \sqcap (\texttt{heads} \oplus_{.75} \texttt{tails}) \tag{5}$$

where $P \oplus_p Q$ is the process that, with probability p behaves like P, and, with probability $1-p$ behaves as Q; and $P \sqcap Q$ models non-deterministic choice between P and Q — controlled by some unspecified environment. If $P \oplus_{[r,s]} Q$ denotes a process that can act as $P \oplus_p Q$ for any p in the interval $[r,s] \subseteq [0,1]$, then their semantics equates $\texttt{heads} \oplus_{[.25,.75]} \texttt{tails}$ with the specification in (5). Such "convex closure" semantics occur frequently in models that mix probabilities and non-deterministic choice — be it explicitly in the modeling formalism or explicitly by forming the convex closure of models or analysis results.

Pre/post-condition validation of `aRandomUpdate` can be reduced to a *probabilistic reachability problem* on the Markov decision process in Figure 2: "What is the likelihood of reaching state `x >= 3` from the initial state?" Probabilistic LTL checking [99] is tailored for answering such questions. Our example also suggests that *pre/post-conditions* (as well as object invariants and control-flow constraints of the program) *guide us in determining how to abstract a model at the source level.*

Bounded model checking [15] abstracts $\models$ by truncating the computation of least fixed points in model checks for a branching-time temporal logic and qualitative models (Kripke structures). This idea applies to LTL and our probabilistic models as well: approximate reachability probabilities by putting bounds on the length of computation traces. In our example, the for-statement in Figure 2 supplies us with such a bound that is even guaranteed not to lose any precision.

E.g. for ϕ being $\mathsf{F}\, p$, we may abstract ϕ with ϕ' iff ϕ' equals p or $p \vee \mathsf{X}\, p$ or $p \vee \mathsf{X}\,(p \vee \mathsf{X}\, p)$ or ... Without knowing the implementation of a model checker, it is difficult to say whether this really abstracts queries or just models. Queries $\mathsf{G}\, p$ ("Globally, p holds.") may be rewritten as $p \wedge \mathsf{X}\, \mathsf{G}\, p$, but all finite approximations $p \wedge \mathsf{X}\,(p \wedge \mathsf{X}(p \wedge \ldots))$ are unsound in general since paths can be infinite. The semantic equivalence of $\mathsf{G}\, p$ and $\neg \mathsf{F}\, \neg p$, therefore illustrates the delicate role of negation in abstraction-based model checking.

Cousot & Cousot [21] developed abstract interpretation as a theory for systematically designing such sound program abstractions; this theory was originally developed for qualitative computation and has been extended to Markov decision processes by Monniaux in his doctoral thesis [76] and subsequent papers [71–75]. These papers are *mandatory* reading on abstraction of Markov decision processes. Implementing `coin_flip` is another, potentially unsound, source of abstraction.

4 Why abstraction?

Since the specification and computation of abstractions incur costs, we need to justify the need for constructing such abstractions for the purpose of analysis.

4.1 Decidability

The specification of $\models$ may not be computable, e.g. if we were to extend the MCP of SMV to infinite-state Kripke structures, we could encode the Halting problem as a model check. Since Markov decision processes faithfully subsume such qualitative systems, the MCP of infinite-state Markov decision processes and CTL is undecidable as well. Note that models of under-specified or concurrent probabilistic programs are potentially infinite state.

The answers to individual model-checking instances $M \models \phi$ may be very hard to determine and such a determination may be closely tied to finding a magic abstraction. E.g. the non-probabilistic method `Collatz` in Figure 3 determines an infinite-state deterministic Kripke structure M in x and c. The answer to the CTL check $(M \models (1 \leq x) \rightarrow \mathsf{AF}(c = 1))$ is unknown at the time of writing. One could conceivably prove that its answer is `tt` by (1) specifying a well-founded order and a variant; and (2) showing that each execution of the while-statement decreases the variant in that well-founded order. It is plausible to believe that such a feat is tantamount to specifying a suitable abstraction of M that does not possess infinite computation paths. Since the MCP of infinite-state Kripke structures with LTL "embeds" into MDP, our example of Figure 3 is relevant to this discussion.

```
int Collatz(x : int) {   // pre-condition: 1 <= x
  int c = x;
  while (c != 1) {
    if (c%2 == 0) { c = c/2; } // %2 is ``modulo 2''
    else { c = 3*c + 1; }
  }
  return c;
}   // post-condition: c = 1.
```

Fig. 3. A non-probabilistic method which determines an infinite-state Kripke structure M in x and c. The answer to the CTL check $(M \models (1 \leq x) \rightarrow \mathsf{AF}(c = 1))$ is unknown.

4.2 Complexity

For MCPs with decidable specification $\models : \mathcal{M} \times \Phi \rightarrow \mathtt{Ans}$, the complexity of computing its instances may be too high. Such assessments depend on the type of complexity (e.g. worst-case, average case or randomized [83]) as well as on its type of bound (lower or upper bounds). For example, the decision problem STOCHASTIC SCHEDULING [82], a scheduling problem of tasks with completion time determined by a Poisson distribution, is PSPACE-complete; and so is the problem of deciding whether the synchronous composition of processes can reach

a deadlocked state. For mixed models in MDP, the worst-case upper bound time complexity of $M \models \phi$ is doubly exponential in the size of ϕ and polynomial in the size of M if M is finite [20]. The complexity bottleneck resides in the semantics of conjunction $\phi_1 \wedge \phi_2$ which may contain conditional probabilities, a non-compositional notion. Baier et al. [5] therefore propose to approximate this semantics, an interval $[r, s]$, in a compositional manner. The approximation is computed via the function $c\colon [0,1] \times [0,1] \to [0,1]$ given by

$$c([r_1, s_1], [r_2, s_2]) = [\max\{0, r_1 + r_2 - 1\}, \min\{s_1, s_2\}]. \tag{6}$$

For $[r', s'] = c([r_1, s_1], [r_2, s_2])$ soundness of this approach is guaranteed since $r' \leq r_1, r_2$ and $s_1, s_2 \leq s'$ follow from the modularity axiom of measures [36]. This new interpretation of conjunction modifies $\models$ to $\models'$ and makes the time complexity single exponential in the size of the formula and has a linear space complexity [5]. The function c is built from t-norms which have been used for similar abstractions for Φ being a probabilistic mu-calculus [49]. The first component of c in (6) is well known and applied in artificial intelligence as it computes the best lower bound on the joint probability of two events, without knowing anything about the independence of these events; that t-norm forms and integral part of a characterization of a modal algebra for quantitative temporal logics [78].

If M is a finite-state Kripke structure and ϕ a CTL formula, then checks are linear in the sizes of M and ϕ respectively [10]. Although the latter complexity is as good as one could hope for, model checks cannot be conducted in practice whenever M is too large. Alas, this is often observed due to the state-space explosion problem: the composition of n components often results in a model whose size is of the order 2^n. Of course, what applies to the linear bound for SMV becomes even more so true for the polynomial bound of MDP. There is a reasonable consensus in the formal methods community that, in light of the known complexity of model checks, scalability is only achievable through the use of aggressive abstraction techniques.

It is worth noting that the computation of $\models$ may be approximated efficiently with randomized algorithms (see e.g. [80]) if the average complexity of model checks is feasible. In that case, one obtains a modification of $\models$ to some $\models'$ which may compromise soundness. Essentially, such approaches replace formal model checks with a form of probabilistic testing [75, 88, 64]. A similar phenomenon can be observed in the complexity of decision problems. For example, trace equivalence in labeled transition systems is PSPACE complete [56], but trace equivalence in fully probabilistic systems is time polynomial [51]; thus, randomization can improve a problem's complexity.

4.3 Environments

If models M are to interact with the world in which they are embedded, they need to be *open*. The method of Figure 1 has an open model, since it interacts with its environment through its formal parameter `x`. Open models under-specify their

environment. In our example, we only know the range for the value of `x`. In open models, we may also abstract from the interaction with such an environment. The code for method `aRandomUpdate` does not reveal the semantics of parameter binding, e.g. call-by-value, call-by-name, etc. Our model, however, assumes call-by-value.

The formal analysis of models typically requires that models be *closed* in the sense that they are fully specified. In our example, non-determinism in the choice of value for `x` closes the model. There are many techniques for closing open models. For example, one may encode the interaction with an environment as an LTL formula over a set of communication actions and then synthesize source code from such a formula for the abstraction of a driver [85]. One may also refine such a closure by replacing a check $M \models \phi$ with a check $M \models \psi \rightarrow \phi$ where ψ encodes an additional assumption about the interaction with an environment, provided that Φ supports a sound abstract interpretation of implications — as is the case for LTL [30]. We mention that these techniques seem similar to the ones being applied when systems are enriched with features [33].

One may also randomize the interaction with an environment, e.g. by replacing the non-deterministic choice for parameter values in methods with a suitable probability distribution. In this manner, the relational semantics of qualitative programs turns into probabilistic predicate transformers [58, 79].

More generally, the probabilities of sets of traces are a function of what constitutes a legal trace in a model, i.e. a legal strategy of resolving non-determinism. For example, in deciding her next non-deterministic move an adversary may have access to [76]

- the current state only; this Markov condition of history-free strategy is used in computing answers to queries for models in MDP;
- the execution trace from the initial to the present state (history-dependent strategy); or
- may see all the system's random choices during its unfolding, e.g. this is discussed in [19]; Monte Carlo methods such as [75] are based on this.

For a more detailed discussion of the role of adversaries, we refer to chapter 5 of Segala's thesis [94]. Clearly, designers of model-checking frameworks and their users need to be aware of the expressive power of strategies for the generation of trace sets. For example, it is well known that the probabilistic semantics of safety properties over Markov decision processes does not change if one replaces history-free with history-dependent strategies; this is no longer true for liveness properties (see e.g. page 105 in [76]).

4.4 Composition

Connecting interfaces of components makes models M more abstract through the hiding of internal communication, as can be seen in the process algebras CCS [69], CSP [45], ACP [2], the π-calculus [70], and probabilistic process algebras [41] where components connect via typed channels. Such hiding of internal

computation results in weak bisimulation as a natural model equivalence [69]. Since the complexity of checks is also a significant function of the size of the model, it may therefore be desirable to replace a model with a weakly bisimilar one with small or minimal state set.

4.5 Specifications

We argued that open models allow for the under-specification of an interaction environment. Under-specification in the form of under-determination or uncertainty is also useful in the description of closed models. Models M then abstract *sets* of implementations. Consequently, the verification of checks on such models applies to the entire set of its implementations. This allows flexibility for an implementation without compromising formal analysis. Examples of under-specified models and their checks abound in the literature. We only mention the modal transition systems of Larsen & Thomsen [63, 61, 12], Bruns & Godefroid's model checker for partial systems [7, 8], Sagiv et al.'s heap shape analysis of `C` programs [92], CSP expressions and refinement checking [31], and Morgan & McIver's quantitative mu-calculus [67]. The latter works over systems that mix non-determinism and probabilistic choice.

There are also good reasons for under-specification in the abstraction-based model checking of *completely* specified mixed models, whenever one requires the soundness of such checks for queries with unrestricted use of negation [48].

4.6 Occam's razor

For MCPs, Occam's razor specializes to

> Whatever aspects of M, $\models$, and ϕ are not needed in computing $(M \models \phi)$ should be discarded.

As an example, consider software slicing, e.g. [40], which abstracts M — the model of a program p — to M' and then checks $M' \models \phi$. The model M' is computed by

- defining the program points C of interest, typically determined by the query ϕ;
- computing all program points C' on which points in C depend; and
- by using C' to compress the model M to some M'.

The soundness of this abstraction rests on the soundness of determining C and C'. For realistic fragments of programming languages, e.g. a concurrent subset of Java — where models are mixed due to probabilistic assignments and compositional reasoning at method boundaries — such soundness proofs can be a monumental task [40]. Our model in Figure 2 was obtained by slicing the infinite-state model of method `aRandomUpdate` along its pre-condition and post-condition. We don't know of any formal work on slicing probabilistic programs.

4.7 Viewpoints

System design and analysis needs to consider multiple points of view. A telecommunications software may be conceived with the utmost formal rigor, its functional correctness may be proven or tested to complete satisfaction, but millions of Canadian dollars may have been wasted because the system's performance turned out to be too slow by a factor of 100 upon deployment [97]. In Figure 4, we see an adaptation of a figure from [55] that illustrates this point in a trivial system. Qualitative, probabilistic, and cost aspects are all important in the design and analysis of this system. Changing such viewpoints constitutes an abstraction. E.g. in making a transition from the probabilistic to the qualitative view, we may want to identify non-zero probabilities with `tt` and 0 with `ff`. We mention some formal work on the soundness of such viewpoint changes [55, 46] and point out that no abstraction framework for such changes of viewpoints exists to date.

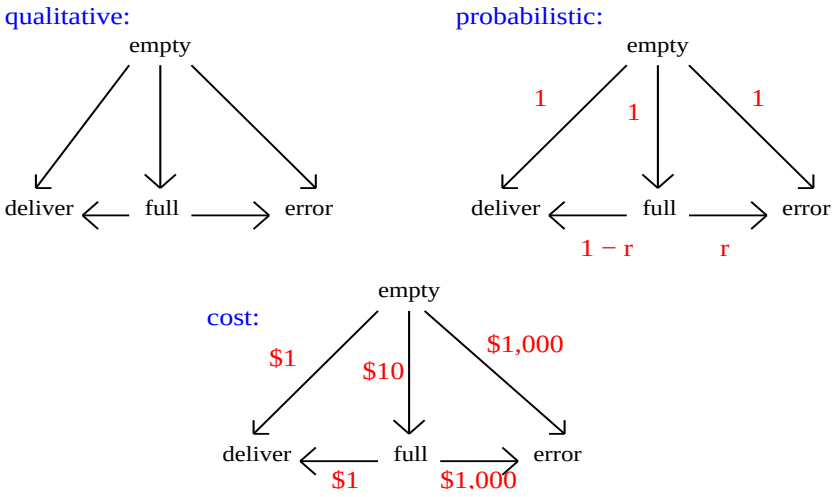

Fig. 4. A system, based on [55], described from three points of view: qualities, probabilities, and costs.

5 What to abstract?

Given a MCP as in (2), we may want to abstract

– models M to M', e.g. through abstract interpretation [21, 76], probabilistic bisimulation [62], the abstraction of an SMV program that describes M [18], the reduction of probabilistic timed automata to timed automata [60], etc.

- queries ϕ to some ϕ', e.g. the approximations of ATL* queries with ATL queries whose model-checking complexity is much more feasible [39]; and
- the answer domain $\mathtt{Ans}$ to an abstract answer domain $\mathtt{Ans}'$, e.g. by reducing "with probability 1" MCPs over Markov decision processes to model checking over Kripke structures; or by replacing a probabilistic measure of information flow with a possibilistic answer [14].

This suggests to define a signature for abstract MCPs of (2).

Definition 1 (Abstract MCP). *A MCP* $\mathcal{P}' = (\mathcal{M}', \Phi', \mathtt{Ans}', \models')$ *abstracts the MCP* $\mathcal{P} = (\mathcal{M}, \Phi, \mathtt{Ans}, \models)$ *with witness* (α, β, γ) *iff*

1. $\alpha \subseteq \mathcal{M} \times \mathcal{M}'$ *is a relation where* $(M, M') \in \alpha$ *means that model* M' *abstracts model* M*;*
2. $\beta \subseteq \Phi \times \Phi'$ *is a relation where* $(\phi, \phi') \in \beta$ *means that the query* ϕ' *abstracts the query* ϕ*; and*
3. $\gamma\colon \mathtt{Ans}' \to \mathtt{Ans}$ *is a total function such that* $\gamma(a')$ *is the concrete interpretation of the abstract answer* $a' \in \mathtt{Ans}'$*.*[1]

Naturally, abstract answers computed on abstract models or queries should yield meaningful concrete answers for the un-abstracted models and queries. Since $\gamma(M' \models' \phi')$ is the concrete interpretation of an abstract check, we need to understand how $\gamma(M' \models' \phi')$ and $(M \models \phi)$ should relate in case that $(M, M') \in \alpha$ and $(\phi, \phi') \in \beta$. Standard MCPs and their soundness of abstraction-based model checking are a reliable guide for determining this relationship.

Sound verification certificates. Suppose that $\mathcal{P}'$ abstracts $\mathcal{P}$ with $\mathtt{Ans} = \{\mathtt{ff} < \mathtt{tt}\}$ and witness $(\alpha, \{(\phi, \phi) \mid \phi \in \Phi\}, \gamma = \lambda a.a)$, i.e. only models are being abstracted. E.g. this is the case in the thesis [76] and the papers [73–75, 72, 71]. Given $(M, M') \in \alpha$, we may want that

$$(M' \models \phi) = \mathtt{tt} \quad \Rightarrow \quad (M \models \phi) = \mathtt{tt}, \tag{7}$$

i.e. certified verifications on abstractions apply to abstracted models as well. Using the order $\mathtt{ff} < \mathtt{tt}$ in $\mathtt{Ans}$, we may express this as $\gamma(M' \models \phi) \leq (M \models \phi)$. A concrete example of a MCP with sound verification certificates are Markov decision processes with reachability properties as queries and probabilistic simulations [23] as instances of α.

Sound refutation certificates. Under the same assumptions on $\mathcal{P}$ as above, we may want that certified refutations on abstractions are refutations of the un-abstracted model:

$$(M, M') \in \alpha \;\&\; (M' \models \phi) = \mathtt{ff} \quad \Rightarrow \quad (M \models \phi) = \mathtt{ff}. \tag{8}$$

[1] Alternatively, one could define $\gamma \subseteq \mathtt{Ans}' \times \mathtt{Ans}$ as a right-total relation: every abstract answer a' has at least one concrete answer a. However, powerdomains [1] over answer domains are able to express such relations as total functions.

Again, $\gamma(M' \models \phi) \leq (M \models \phi)$ guarantees such a sound transfer of results, provided that now $\texttt{tt} < \texttt{ff}$. Thus, a combination of sound refutation and verification seems impossible at first sight. Indeed, for probabilistic automata and various forms of (weak) simulations only very limited fragments of PCTL observe such a sound transfer [95]. However, if $\texttt{tt}$ and $\texttt{ff}$ are maximal elements in $\texttt{Ans}$ such a combination is indeed possible [7, 92], as demonstrated for Markov decision processes and a branching-time temporal mu-calculus in [48].

Sound [worst,best]-case probabilities. Consider MDP as an abstraction of itself, where only α is non-trivial and left unspecified here. Then $\gamma(M' \models \phi) \leq (M \models \phi)$ reads as $[r', s'] \leq [r, s]$ which means $r' \leq r$ and $s \leq s'$. Note that the order of approximation in $\mathcal{I}$ naturally captures the soundness of abstract model checks according to $\gamma(M' \models \phi) \leq (M \models \phi)$: the abstract worst-case probability r' is a conservative abstraction of the concrete one r for the worst-case; similarly, the abstract best-case probability s' is a conservative abstraction of the concrete one s for the best-case.

The general formulation of sound abstract MCPs is now clear.

Definition 2 (Sound abstract MCP). *Let $\mathcal{P}' = (\mathcal{M}', \Phi', \texttt{Ans}', \models')$ and $\mathcal{P} = (\mathcal{M}, \Phi, \texttt{Ans}, \models)$ be MCPs such that $\mathcal{P}'$ abstracts $\mathcal{P}$ with witness (α, β, γ). Then $\mathcal{P}'$ is* sound *for $\mathcal{P}$ iff for all $M \in \mathcal{M}$, $M' \in \mathcal{M}'$, $\phi \in \Phi$, and $\phi' \in \Phi'$ we have*

$$(M, M') \in \alpha \;\&\; (\phi, \phi') \in \beta \;\Rightarrow\; \gamma(M' \models' \phi') \leq (M \models \phi). \tag{9}$$

We hasten to point out that the utility of this notion depends on mild assumptions about the witnesses α and β. The former is typically right-total and left-total, the latter should at least be right-total — i.e. every $\phi \in \Phi$ has at least one $\phi' \in \Phi'$ with $(\phi, \phi') \in \beta$ — and will often be a function. The function γ need not enjoy special properties, but its monotonicity ensures that the composition of sound abstract MCPs remains sound. In that case, if (α, β, γ) witnesses that $\mathcal{P}'$ abstracts $\mathcal{P}$ and if $(\alpha', \beta', \gamma')$ witnesses that $\mathcal{P}''$ abstracts $\mathcal{P}'$, then $(\alpha; \alpha', \beta; \beta', \gamma'; \gamma)$ witnesses that $\mathcal{P}''$ abstracts $\mathcal{P}$.

Example 1 (Abstract MCPs). We remark that the MCPs for sound refutation certificates, sound verification certificates, and sound [worst,best]-case probabilities presented above are all instances of sound abstract MCPs.

It turns out that there are many kinds of models, queries, and answers if we mix non-deterministic and probabilistic behavior. Models may be Markov decision processes (e.g. [90]), probabilistic automata [95], etc; queries could be formulas of a temporal logic [99, 66], random variables [58], etc; and the legitimate answer types could be truth values [17], real numbers [58], real-valued intervals [50], etc. The query ϕ could even represent another, less specified, model and $(M \models \phi) = \texttt{tt}$ may mean "M implements ϕ." This is the basis of the model checker FDR whose mathematical foundation has been extended to probabilistic choice by Morgan et al. [77, 79, 96] and Jifeng et al. [52]. In [77], the process algebra CSP is enriched with a probabilistic choice that distributed

through all other operators, non-deterministic choice in particular. A process has a denotational semantics using the probabilistic power domain [54, 53] and the failure/divergence semantics [45] provides a sound notion of process refinement.

Given this abundance of different approaches and interpretations, it comes as no surprise that we make (2) a syntactic leitmotif of this survey since it helps to present and organize the vast literature on this topic in a rather uniform and more coherent manner.

Papers on abstraction in model checking typically focus on one aspect of abstraction at a time:

- simulations [68] and abstract interpretation [21] specify a sound instance of α, where queries support only a limited form of negation such as "for all computation paths, ..."; in that case, $\mathtt{Ans} = \{\mathtt{ff} < \mathtt{tt}\}$, and $\gamma = \lambda a.a$;
- bisimulations [84, 69] specify another sound instance of α, where queries have an unrestricted form of negation, $\gamma = \lambda a.a$, and $\mathtt{Ans} = \{\mathtt{ff}, \mathtt{tt}\}$ in the *discrete* ordering;
- the bounded unfolding of least-fixed point queries specifies a sound instance of β;
- bounded model-checking techniques [15] specify an abstract version $\models'$ of $\models$ and
- possibilistic abstractions, e.g. [14] and [55], specify an abstract answer domain.

In practice, several of these abstractions may be employed to abstract a check and such a combination is then encoded in the witness (α, β, γ). In any event, (9) specifies the meaning of soundness for such abstract checks. Although we argue that Definition 2 unifies most abstraction techniques encountered in practice, a single MCP cannot embrace every form of abstraction there is. For example, the annotation of LTL or CTL* path modalities (next, until, etc) with probability thresholds [37, 38] does not give rise to a sound β unless β is a function of the model M under analysis [50], forcing $\mathcal{M}$ to be a singleton.

Example 2 (Abstracting answers and queries). Consider a federal officer whose job is to certify the reliability of safety-critical programs. She will primarily be interested in knowing, whether the probability of meeting the safety requirement is above a legally prescribed threshold. Her query set Φ, with atomic observables o, is given in Figure 5; note that her queries ϕ are elements $\phi' \in \Phi' =$ LTL annotated with threshold values at X and U connectives. The semantics $\models_{\geq}$ of these thresholds is that the worst-case probability of the underlying trace set be $\geq p$.

In [50], a connection between these two semantics is established: the worst-case and best-case probabilities determine two maps $(\cdot)^\flat, (\cdot)^\sharp \colon \Phi' \to \Phi$ (respectively) such that a positive worst-case probabilities for ϕ' makes $(\phi')^\flat$ true and a best-case probability for ϕ' below 1 makes $(\phi')^\sharp$ false; this is shown for t-norms, such as the one used in [5], as a semantics of conjunction ($\wedge$).

A worst-case abstraction $\alpha^\forall \colon [0,1] \to \{\mathtt{ff} < \mathtt{tt}\}$ identifies $\mathtt{ff}$ with $\{0\}$ and $\mathtt{tt}$ with $(0,1]$; dually, the best-case abstraction $\alpha^\exists \colon [0,1] \to \{\mathtt{ff} < \mathtt{tt}\}$ maps 1 to

$$\phi ::= \mathtt{tt} \mid o \mid \neg\phi \mid \phi \wedge \phi \mid [\mathsf{X}\,\phi]_{\geq p} \mid [\phi\,\mathsf{U}\,\phi]_{\geq p}$$

Fig. 5. Syntax of $\mathrm{LTL}_{\geq}$ with thresholds $p \in [0,1]$.

$\mathtt{tt}$ and $[0,1)$ to $\mathtt{ff}$. These abstractions have a common concretization $\xi\colon \{\mathtt{ff} < \mathtt{tt}\} \to [0,1]$ which maps $\mathtt{ff}$ to 0 and $\mathtt{tt}$ to 1. The pairs $(\xi, \alpha^{\forall})$ and $(\xi, \alpha^{\exists})$ constitute Galois embeddings [21, 32] where $\alpha^{\forall}$ and $\alpha^{\exists}$ are the lower and upper adjoint of ξ (respectively). This mirrors the situation of Galois adjunctions used in [22] for sound abstraction of a temporal logic with unrestricted use of negation, where the concretization function has a universal and existential abstraction function as adjoints.

MDP abstracts the MCP $(\mathcal{M}', \Phi, \mathtt{Ans}, \models)$, where $\models$ equals $\models_{\geq} \times \models_{\geq}$, Φ equals $\mathrm{LTL}_{\geq} \times \mathrm{LTL}_{\geq}$, and $\mathtt{Ans}$ equals $\{\mathtt{ff} < \mathtt{tt}\} \times \{\mathtt{tt} < \mathtt{ff}\}$. The abstraction relation on models, α, is the identity; the abstraction relation on queries is defined as those pairs $((\psi, \eta), \phi)$ with $\psi = \phi^{\flat}$ and $\eta = \phi^{\sharp}$; and $\gamma\colon \mathtt{Ans}' \to \mathtt{Ans}$ maps intervals $[r,s]$ to $(\alpha^{\forall}(r), \alpha^{\exists}(s)) \in \mathtt{Ans}$. Soundness (9) follows from the result in [50] stated above.

Many other people have also realized such connections between answer domains. For example, Jonsson & Larsen [55] relate systems whose transitions are annotated with probability intervals $[r,s]$ to modal transition systems [63, 61] — abstracting the interval domain $\mathcal{I}$ to the flat booleans $\mathbb{B}_{\perp}$; guaranteed $[r,s]$-transitions are those with $0 < r$; possible $[r,s]$-transitions are those for which $0 < s$. This matches our definitions of $\alpha^{\forall}$ and $\alpha^{\exists}$: if $0 < r$, then every $p \in [r,s]$ is mapped to $\mathtt{tt}$ via $\alpha^{\forall}$; if $s < 1$, then $\alpha^{\exists}(p) = \mathtt{ff}$ for all $p \in [r,s]$. We refer to [46, 47] for a more detailed study of such abstractions.

6 Query equivalence

Occam's razor also applies whenever models cannot be distinguished by checks. Given a MCP $\mathcal{P}$ as in (2), we define for $M, M' \in \mathcal{M}$ the relation

$$M \sim M' \qquad \text{iff} \qquad \lambda\phi\colon \Phi.(M \models \phi) = \lambda\phi\colon \Phi.(M' \models \phi). \tag{10}$$

The query equivalence $\sim$ partitions $\mathcal{M}$ and induces an abstract MCP $\mathcal{P}'$ for $\mathcal{P}$, where α is $\sim$, $\beta = \{(\phi, \phi) \mid \phi \in \Phi\}$, and $\mathtt{Ans}' = \mathtt{Ans}$ with $\gamma(a') = a'$ for all $a' \in \mathtt{Ans}'$. As an example, consider the MCP of labeled transition systems with a propositional modal mu-calculus as query language [6]. Then all query-equivalent models are also trace equivalent but not the other way around [69]. Occam's razor suggests that M in $M \models \phi$ be replaced with some M' such that $M \sim M'$ and the computation of $(M' \models \phi)$ have minimal cost — e.g. M' may have minimal state space among all models that are query equivalent to M. It should be apparent that such a reduction is less aggressive than the ones that are driven by a particular query ϕ, where the equality of the functions in (10) is

being replaced with the equality of their evaluation at ϕ. We already encountered software slicing as such an example and note that property-driven abstraction is not adequately explored in the study of mixed models.

This approach of model simplification is not always guaranteed to be fruitful. For example, there are continuous-space labeled Markov chains that are not equivalent to any finite-state model for a simple probabilistic modal logic Φ [27].

Bisimulations [84, 69] are a device for establishing instances of the query equivalence $\sim$ for branching-time query languages such as CTL and the modal mu-calculus. Bisimulations relate models in an intuitive operational manner, lead to co-inductive reasoning principles, and have typically efficient decision algorithms [3, 4, 86, 11]. Bisimulations are sound for a query language if bisimilar models are also query equivalent. The converse relationship is called completeness. Bisimulation of labeled transition systems is sound and complete for CTL and the propositional modal mu-calculus.

If internal actions are hidden from the top layer of an operational semantics, then state transitions may be pre- or post-composed with a finite number of internal, silent actions. This results in a new state-transition relation whose bisimulation is called "weak" when applied to the original model [69]. With respect to the original model, such a saturation of state transitions may not be sound (e.g. [35]) or may be sound and complete, as it is the case for Markov decision processes and pCTL* [29]. It is worth noting that branching bisimulation [98] avoids the unsoundness encountered in [35]. A better understanding of the connections between probabilistic bisimulations and branching bisimulations is desired; see [43].

7 Precise abstract MCPs

Model checking is referred to as a property verification technique: given a model M, we mean to check one property ϕ at a time. In the context of an abstract MCP $\mathcal{P}' = (\mathcal{M}', \Phi', \mathtt{Ans}', \models')$ for a MCP $\mathcal{P} = (\mathcal{M}, \Phi, \mathtt{Ans}, \models)$ with witness (α, β, γ), the possible sound abstractions of the check $M \models \phi$ that are offered by $\mathcal{P}'$ are all $\gamma(M' \models' \phi')$ with $(M, M') \in \alpha$ and $(\phi, \phi') \in \beta$. Thus, the set

$$\mathrm{A}_\phi^M = \{\gamma(M' \models \phi') \mid (M, M') \in \alpha,\ (\phi, \phi') \in \beta\} \tag{11}$$

collects all the information about $(M \models \phi)$ that we could ever infer from performing abstract and sound checks according to $\mathcal{P}'$.

Definition 3 (Precise abstract MCPs). *Let $\mathcal{P}' = (\mathcal{M}', \Phi', \mathtt{Ans}', \models')$ and $\mathcal{P} = (\mathcal{M}, \Phi, \mathtt{Ans}, \models)$ be MCPs such that $\mathcal{P}'$ abstracts $\mathcal{P}$ with witness (α, β, γ). For $\phi \in \Phi$, $\mathcal{P}'$ is* ϕ-precise *for $\mathcal{P}$ iff for all $M \in \mathcal{M}$, the set A_ϕ^M has $(M \models \phi)$ as a least upper bound. We call $\mathcal{P}'$* precise *for $\mathcal{P}$ iff it is ϕ-precise for $\mathcal{P}$ for all $\phi \in \Phi$.*

Since $\mathtt{Ans}$ is a partial order, the statement about A_ϕ^M above includes the assumption that the least upper bound of A_ϕ^M exists. Evidently, it makes sense to stipulate that $\mathtt{Ans}$ be a continuous domain [1], a partial order with a rich and formal

notion of approximation. This stipulation holds for all answer domains presented in this survey. Although the set A_ϕ^M may not be directed, we expect that its information content leads to directed approximations. E.g. consider MDP. If $[r', s']$ and $[r'', s'']$ are in A_ϕ^M for $\mathtt{Ans} = \mathcal{I}$, then $[\max(r', r''), \min(s', s'')]$ is a rational re-construction of an upper bound that still approximates $(M \models \phi)$ — despite the fact that $[\max(r', r''), \min(s', s'')]$ may not be an element of A_ϕ^M.

Consider $\mathcal{P}$ as the MCP of Kripke structures and CTL, $\mathcal{P}'$ as the same MCP restricted to finite models M, and $\mathtt{Ans} = \mathtt{Ans}' = \{\mathtt{ff} < \mathtt{tt}\}$. We leave α unspecified, whereas $\beta = \{(\phi, \phi) \mid \phi \in \Phi\}$, and $\gamma = \lambda a.a$. Fix a CTL formula ϕ. Then $\mathcal{P}'$ is ϕ-precise iff for each model M with $(M \models \phi) = \mathtt{tt}$ there is some finite model M' with $(M, M') \in \alpha$ such that $(M' \models \phi) = \mathtt{tt}$ as well. Thus, this amounts to saying that all verifications of ϕ have certified verifications in some finite abstraction. Therefore, finite-model properties for model checking are instances of precise abstract MCPs.

If the query language is a logic with the finite model property, then the abstract MCP may indeed be precise for a suitable abstraction relation. As an example, Desharnais et al. [28] showed that each continuous-state labeled Markov process has a sequence of finite acyclic labeled Markov processes as abstractions such that this sequence is precise for a probabilistic modal logic.

Even if one has proven that $\mathcal{P}'$ is precise for $\mathcal{P}$, practical use of abstract model checking requires feasible strategies or heuristics for finding "precise" abstractions. In the case of $\mathtt{ff} < \mathtt{tt}$, at least, one knows what to look for: a positive check of ϕ on an abstraction does the trick. In the [worst,best]-case probability situation, however, single checks cannot be expected to recover the actual concrete value of $(M \models \phi)$: all members of A_ϕ^M may be strictly below it. In essence, practitioners need to specify a desired degree of accuracy and the result $\gamma(M' \models' \phi')$ is then compared to $(M \models \phi)$. Fortunately, most continuous domains have a natural way of measuring such a distance [65, 101]. For example, the distance between the concrete and abstract checks — $[r, s]$ and $[r', s']$ (respectively) — may be given as

$$|r - r'| + |s - s'|. \tag{12}$$

Quantitative domain theory [65, 101] has the potential of supplying such distance notions for a rich class of answer domains.

Definition 4 (The MCP Approximation Problem). *Let us assume that there is some suitable*[2] *distance function* d: $\mathtt{Ans} \times \mathtt{Ans} \to [0, +\infty)$. *Let* $\mathcal{P}' = (\mathcal{M}', \Phi', \mathtt{Ans}', \models')$ *and* $\mathcal{P} = (\mathcal{M}, \Phi, \mathtt{Ans}, \models)$ *be MCPs such that* $\mathcal{P}'$ *abstracts* $\mathcal{P}$ *with witness* (α, β, γ). *Given* $\epsilon > 0$, $M \in \mathcal{M}$, *and* $\phi \in \Phi$, *are their* $M' \in \mathcal{M}'$ *and* $\phi' \in \Phi'$ *such that* $(M, M') \in \alpha$, $(\phi, \phi') \in \beta$, *and*

$$d(\gamma(M' \models' \phi'), (M \models \phi)) < \epsilon? \tag{13}$$

[2] We require that the distance function be Scott-continuous of type $\mathtt{Ans} \times \mathtt{Ans} \to [0, \infty)^{\mathrm{op}}$ such that $d(a, a') = 0$ implies $a = a'$.

Although this definition is conceptually pleasing, it begs the question of what value $M \models \phi$ computes to. Measurements [65] μ: $\texttt{Ans} \to [0, \infty)$, on the other hand, compute a self-distance which can inform us how far away the analysis is from a complete result. E.g. the canonical measurement for $\mathcal{I}$ is simply $\mu[r, s] = \mid r - s \mid$. The MCP Approximation Problem can then be restated:

> "Given $\epsilon > 0$, $M \in \mathcal{M}$, and $\phi \in \Phi$, are there $M' \in \mathcal{M}'$ and $\phi' \in \Phi'$ such that $(M, M') \in \alpha$, $(\phi, \phi') \in \beta$, and
>
> $$\mu(\gamma(M' \models' \phi')) < \epsilon?" \tag{14}$$

Unfortunately, if the underlying model M is under-specified or partial in some way, then we cannot uniquely identify the source of such partiality — whether it resides in the incompleteness of our abstraction-based analysis or in the partiality of the abstracted model. In Markov decision processes, for example, this partiality is encoded in the non-determinism which gives rise to genuine intervals $[r, s]$ of [worst,best]-case probabilities. Although this partiality and the resulting blur of sources of partiality in answers is absent in Markov chains, the latter tend to have Markov decision processes as abstractions. This blurring of the source of incompleteness seems to be specific to quantitative model checking. Recall that $\texttt{ff} < \texttt{tt}$ for verification certificates; the measure is then $\mu_{\mathrm{v}}(\texttt{ff}) = 1$ and $\mu_{\mathrm{v}}(\texttt{tt}) = 0$. Dually, refutation checks have $\texttt{tt} < \texttt{ff}$ and the measure is $\mu_{\mathrm{r}}(\texttt{tt}) = 1$ and $\mu_{\mathrm{v}}(\texttt{ff}) = 0$. In either case, $\epsilon = 0.5$ will then turn (14) into a check for precision.

8 Diagnostics

The specification of model checks typically stipulates that the answer domain $\texttt{Ans}$ to checks of (2) has an extensional type such as $\{\texttt{ff} < \texttt{tt}\}$ for Kripke structures [17] or the unit interval $[0, 1]$ for a probabilistic semantics of LTL [99]. The practical utility of model checking, however, depends on its ability to attach intensional content to such extensional replies. Little may be gained from knowing that a system can reach deadlock from some initial state, unless information about a possible execution trace from some initial to a deadlocked state be provided. Without a strict regime on managing negation and path quantifiers, this demand applies to refuted (answer is $\texttt{ff}$) and validated properties (answer is $\texttt{tt}$) alike. The utility of abstraction techniques will therefore also depend on their ability to produce such diagnostic information. For systems that mix non-deterministic and probabilistic behavior, this constitutes a major research challenge.

9 Cost effectiveness

We need to remark that all the wonderful work on abstraction in model checking is of little use if

– the cost of specifying or computing executable abstractions outweighs the savings over executing un-abstracted models; or
– if there are no heuristics or other support that guides one in the choice of an abstraction, even in case that the abstract MCP is precise for the un-abstracted one.

The details of assessing this trade-off are too dependent on the particular MCP at hand to be studied in the general framework presented in this survey. However, the reader should keep in mind that the first objection is mute if the MCP under consideration is undecidable; abstraction then functions as a last straw. Also, there is already some work on the establishment of strategies for the refinement of abstractions of probabilistic transition systems with respect to certain reachability properties [23, 24].

10 Abandoning soundness

Formal modeling and analysis presupposes that the underlying methodology be sound. We formulated such soundness for abstract MCPs in (9). Yet there are a number of good reasons and situations in which one wishes to abandon (9).

– A model may only be subjected to a limited exploration of its computation paths: testing of software, e.g. with the tool Verisoft [34], is a viable alternative to full verification through model checking, especially in the presence of tight time-to-market constraints.
– Similarly, complexity results on mixed models may suggest not to explore all computation paths but to truncate them at a certain bound, e.g. [64].
– For mixed models, probabilistic testing can result in Monte Carlo methods for the analysis of such models [75, 88]. Not only will probabilistically sampled paths be verified, but one can then often extend this verification to the entire system by specifying a probability for this extension to fail. Specifying a measurable extension error is generally not possible for qualitative models.
– A MCP may not have a finite model property or may not be subject to symbolic analysis. One may then have to do with the ability to run simulations (generate a computation path/tree) of the model in question.
– Query equivalence and the related notions of bisimulation and weak bisimulation are relations. Thus, two models are either related or not. In some practical situations, notably secure information flow in computer systems [25, 26, 100], one cannot hope to ever achieve such equivalences between a model M (the implementation) and M' (the specification as an abstraction of the implementation). Instead, one may want to stipulate that $\gamma(M' \models' \phi')$ and $(M \models \phi)$ have a distance below ϵ for all queries ϕ. For the distance in (12), such a property won't guarantee nor care for (9). As an example we mention the approach of Di Pierro et al. [87].

11 Research challenges

The challenges that lie ahead for abstraction-based model checking of mixed models are many, tremendous, and possibly very discouraging. However, we mean to formulate some key problems that are common to all such MCPs.

- Establish sound formal techniques for a query-driven (ϕ) abstraction M' of a model M such that $(M \models \phi) = (M' \models \phi)$. Investigate the complexity of computing such an M' that has minimal "cost."
- For a specific class of queries, prove or disprove that abstract MCPs are precise for their concrete counterparts.
- Find heuristics or formal techniques for solving the abstract MCP approximation problem. This includes the problem of revising abstractions if their checks or diagnostics are insufficiently informative.
- For MCPs with unrestricted negation of queries, develop diagnostics that capture the evidence of model checking results. For mixed models, further develop probabilistic games as such diagnostics.
- Push established methods and results from qualitative model checking into the model checking of mixed and, more generally, quantitative MCPs.

12 Conclusions

A unified framework for specifying MCPs and abstract MCPs was presented. A partial survey of the literature on mixed models revealed that most work on abstraction can be represented within this framework. We also pointed out possibly fruitful connections to ongoing work in quantitative domain theory. Finally, research issues and connections to ongoing work in verification of mixed models were identified.

Acknowledgments

We gratefully acknowledge Joost-Pieter Katoen for pointing out omissions, typos, and ambiguities in drafts of this tutorial. Annabelle McIver and Carroll Morgan also made many comments that improved this manuscript; they kindly provided the example in (5).

References

1. S. Abramsky and A. Jung. Domain theory. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 3, pages 1–168. Oxford Univ. Press, 1994.
2. J. C. M. Baeten and W. P. Weijland. *Process Algebra*, volume 18 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1990.
3. C. Baier, B. Engelen, and M. Majster-Cederbaum. Deciding Bisimilarity and Similarity for Probabilistic Processes. *Journal of Computer and System Sciences*, 60:187–231, 2000.

4. C. Baier and M.I.A. Stoelinga. Norm fuctions for probabilistic bisimulations with delays. In J. Tiuryn, editor, *Proceedings of 3rd International Conference on Foundations of Science and Computation Structures (FOSSACS)*, Berlin, Germany, March 2000, volume 1784 of *Lecture Notes in Computer Science*, pages 1–16, 2000.
5. Christel Baier, Marta Kwiatkowska, and Gethin Norman. Computing probability bounds for linear time formulas over concurrent probabilistic systems. In Marta Kwiatkowska Christel Baier, Michael Huth and Mark Ryan, editors, *Electronic Notes in Theoretical Computer Science*, volume 22. Elsevier Science Publishers, 2000.
6. J. C. Bradfield. *Verifying Temporal Properties Of Systems.* Birkhäuser, Boston, Mass., 1991.
7. G. Bruns and P. Godefroid. Model Checking Partial State Spaces with 3-Valued Temporal Logics. In *Proceedings of the 11th Conference on Computer Aided Verification*, volume 1633 of *Lecture Notes in Computer Science*, pages 274–287. Springer Verlag, July 1999.
8. G. Bruns and P. Godefroid. Generalized Model Checking: Reasoning about Partial State Spaces. In *Proceedings of* CONCUR*'2000 (11th International Conference on Concurrency Theory)*, volume 1877 of *Lecture Notes in Computer Science*, pages 168–182. Springer Verlag, August 2000.
9. R. R. Bryant. Symbolic Boolean Manipulation with Ordered Binary-Decision Diagrams. *ACM Computing Surveys*, 24(3):293–318, September 1992.
10. J. R. Burch, E. M. Clarke, D. L. Dill K. L. McMillan, and J. Hwang. Symbolic model checking: 10^{20} states and beyond. Proceedings of the Fifth Annual Symposium on Logic in Computer Science, June 1990.
11. S. Cattani and R. Segala. Decision algorithms for probabilistic bisimulation. In *Proceedings of the 13th International Conference on Concurrency Theory (*CONCUR*'02)*, Lecture Notes in Computer Science, Brno, Czech Republic, August 2002. Springer Verlag. To appear.
12. K. Cerans, J. Chr. Godskesen, and K. G. Larsen. Timed modal specification - theory and tools. In *Computer Aided Verification*, pages 253–267, 1993.
13. Y.-F. Chen, E. R. Gansner, and E. Koutsofios. A C++ data model supporting reachability analysis and dead code detection. In M. Jazayeri and H. Schauer, editors, *Proceedings of the Sixth European Software Engineering Conference (ESEC/FSE 97)*, pages 414–431. Springer–Verlag, 1997.
14. D. Clark, C. Hankin, S. Hunt, and R. Nagarajan. Possibilistic Information Flow is safe for Probabilistic Non-Interference. In *Workshop on Issues in the Theory of Security (WITS '00)*, Geneva, Switzerland, 7-8 July 2000.
15. E. Clarke, A. Biere, R. Raimi, and Y. Zhu. Bounded Model Checking Using Satisfiability Solving. *Formal Methods in System Design*, 19(1):7–34, July 2001.
16. E. M. Clarke, M. Fujita, and X. Zhao. *Representations of discrete functions*, chapter Multi-terminal binary decision diagrams and hybrid decision diagrams, pages 93–108. Kluwer academic publishers, 1996.
17. E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking.* The MIT Press, January 2000.
18. E.M. Clarke, O. Grumberg, and D. E. Long. Model checking and abstraction. *ACM Transactions on Programming Languages and Systems*, 16(5):1512–1542, 1994.
19. R. Cleaveland, S. A. Smolka, and A. E. Zwarico. Testing preorders for probabilistic processes. In Werner Kuich, editor, *Automata, Languages and Programming, 19th*

International Colloqium, volume 623 of *Lecture Notes in Computer Science*, pages 708–719, Vienna Austria, 13–17 July 1992. Springer Verlag.
20. C. Courcoubetis and M. Yannakakis. The Complexity of Probabilistic Verification. *Journal of the Association of Computing Machinery*, 42(4):857–907, July 1995.
21. P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs. In *Proc. 4th ACM Symp. on Principles of Programming Languages*, pages 238–252. ACM Press, 1977.
22. P. Cousot and R. Cousot. Temporal abstract interpretation. In *Conference Record of the 27th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 12–25, Boston, Mass., January 2000. ACM Press, New York, NY.
23. P. R. D'Argenio, B. Jeannet, H. E. Jensen, and K. G. Larsen. Reachability Analysis of Probabilistic Systems by Successive Refinements. In L. de Alfaro and S. Gilmore, editors, *Process Algebra and Probabilistic Methods: Performance Modelling and Verification*, volume 2165 of *Lecture Notes in Computer Science*, pages 39–56, Aachen, Germany, September 12-14 2001. Springer Verlag.
24. P. R. D'Argenio, B. Jeannet, H. E. Jensen, and K. G. Larsen. Reduction and Refinement Strategies for Probabilistic Analysis. In H. Hermanns and R. Segala, editors, *Process Algebra and Probabilistic Methods, Performance Modeling and Verification, Second Joint International Workshop PAPM-PROBMIV 2002*, volume 2399 of *Lecture Notes in Computer Science*, pages 57–76, Copenhagen, Denmark, July 25-26 2002. Springer.
25. D. Denning. A Lattice Model of Secure Information Flow. *Communications of the ACM*, 19(5):236–243, 1976.
26. D. Denning. Certification of Programs for Secure Information Flow. *Communications of the ACM*, 20(7):504–513, 1977.
27. J. Desharnais, A. Edalat, and P. Panangaden. Bisimulation for Labelled Markov Processes. *Journal of Information and Computation*, 2003. To appear; unified version of their LICS papers of 1997 and 1998.
28. J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden. Approximating Labeled Markov Processes. In *15th Annual IEEE Symposium on Logic in Computer Science (LICS'00)*, Santa Barbara, California, 26-29 June 2000. IEEE Computer Society Press.
29. J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden. Weak Bisimulation is Sound and Complete for PCTL*. In *Proc. 13th Int'l Conference* CONCUR *2002 - Concurrency Theory*, volume 2421 of *Lecture Notes in Computer Science*, pages 355–370, Brno, Czech Republic, 20-23 August 2002. Springer Verlag.
30. M. B. Dwyer and D. A. Schmidt. Limiting State Explosion with Filter-Based Refinement. In *Proceedings of the ILPS'97 Workshop on Verification, Model Checking, and Abstraction*, 1997.
31. URL: http://www.formal.demon.co.uk/FDR2.html.
32. G. Gierz, K. H. Hofmann, K. Keimel, J. D. Lawson, M. Mislove, and D. S. Scott. *A Compendium of Continuous Lattices*. Springer Verlag, 1980.
33. S. Gilmore and M. Ryan, editors. *Language Constructs for Describing Features, Proc. of the FIREworks workshop*. Springer Verlag, 2001.
34. P. Godefroid. Model Checking for Programming Languages using VeriSoft. In *Proceedings of the 24th ACM Symposium on Principles of Programming Languages*, pages 174–186, Paris, January 1997.

35. S. Graf and J. Sifakis. Readiness Semantics for Regular Processes with Silent Actions. In *In: Proc. of ICALP'87*, pages 115–125, 1987.
36. P. R. Halmos. *Measure Theory.* Graduate Texts in Mathematics 18. Springer Verlag, 1950.
37. H. Hansson. *Time and Probability in Formal Design of Distributed Systems.* PhD thesis, Department of Computer Science, Uppsala University, Uppsala, Sweden, 1991.
38. H. A. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.
39. A. Harding, M. Ryan, and P.-Y. Schobbens. Approximating ATL* in ATL. In *Third International Workshop on Verification, Model Checking and Abstract Interpretation*, volume 2294 of *Lecture Notes in Computer Science*, pages 289–301, Venice, Italy, January 21-22 2002. Springer Verlag.
40. J. Hatcliff, J. C. Corbett, M. B. Dwyer, S. Sokolowski, and H. Zheng. A formal study of slicing for multi-threaded programs with JVM concurrency primitives. In *Static Analysis Symposium*, pages 1–18, 1999.
41. H. Hermanns, U. Herzog, and J.-P. Katoen. Process algebra for performance evaluation. *Theoretical Computer Science*, 274(1–2):43–87, 2002.
42. H. Hermanns and J.-P. Katoen. Performance Evaluation := (Process Algebra + Model Checking) × Markov Chains. In K. G. Larsen and M. Nielson, editors, *12th Int'l Conference* CONCUR *2001 - Concurrency Theory*, volume 2154 of *Lecture Notes in Computer Science*, pages 59–81, Aalborg, Denmark, 20-25 August 2001. Springer Verlag.
43. Holger Hermanns. *Interactive Markov Chains*, volume 2428 of *Lecture Notes in Computer Science.* Springer Verlag, September 2002.
44. J. Hillston. *A Compositional Approach to Performance Modelling.* Cambridge University Press, 1996.
45. C. A. R. Hoare. *Communicating Sequential Processes.* Prentice-Hall, 1985.
46. M. Huth. A Unifying Framework for Model Checking Labeled Kripke Structures, Modal Transition Systems, and Interval Transition Systems. In *Proceedings of the 19th International Conference on the Foundations of Software Technology & Theoretical Computer Science*, Lecture Notes in Computer Science, pages 369–380, IIT Chennai, India, December 1999. Springer Verlag.
47. M. Huth. *Domains and Processes*, chapter Domains of view: a foundation for specification and analysis, pages 183–218. Kluwer Academic Press, December 2001.
48. M. Huth. Possibilistic and Probabilistic Abstraction-Based Model Checking. In H. Hermanns and R. Segala, editors, *Process Algebra and Probabilistic Methods, Performance Modeling and Verification, Second Joint International Workshop PAPM-PROBMIV 2002*, volume 2399 of *Lecture Notes in Computer Science*, pages 115–134, Copenhagen, Denmark, July 25-26 2002. Springer.
49. M. Huth and M.Kwiatkowska. Quantitative analysis and model checking. In *Proceedings of the 12th Annual IEEE Symposium on Logic in Computer Science*, pages 111–122, Warsaw, Poland, 1997. IEEE Computer Society Press.
50. Michael Huth. The interval domain: A matchmaker for aCTL and aPCTL. In Michael Mislove Rance Cleaveland and Philip Mulry, editors, *Electronic Notes in Theoretical Computer Science*, volume 14. Elsevier Science Publishers, 2000.
51. T. Huynh and L. Tian. On some Equivalence Relations for Probabilistic Processes. *Fundamenta Informaticae*, 17:211–234, 1992.
52. H. Jifeng, K. Seidel, and A. McIver. Probabilistic models for the guarded command language. *Science of Computer Programming*, 28(2-3):171–192, April 1997.

53. C. Jones. *Probabilistic Nondeterminism.* PhD thesis, Laboratory for the Foundations of Computer Science, University of Edinburgh, Edinburgh, Scotland, 1990. Monograph ECS-LFCS-90-105.
54. C. Jones and G. Plotkin. A probabilistic powerdomain of evaluations. In *In: Proceedings of the IEEE 4th Annual Symposium on Logic in Computer Science*, pages 186–195. IEEE Computer Society Press, 1989.
55. B. Jonsson and K. G. Larsen. Specification and Refinement of Probabilistic Processes. In *6th Annual IEEE Symposium on Logic in Computer Science*, pages 266–277, Amsterdam, The Netherlands, 15-18 July 1991. IEEE Computer Society Press.
56. P. Kannelakis and S. Smolka. CCS Expressions, Finite State Processes and Three Problems of Equivalence. *Journal of Information and Computation*, 86:43–68, 1990.
57. K.L. McMillan. *Symbolic Model Checking.* Kluwer Academic Publishers, 1993.
58. D. Kozen. Semantics of Probabilistic Programs. *Computer and System Sciences*, 22:328–350, 1981.
59. D. Kozen. A Probabilistic PDL. *Computer and System Sciences*, 22(2):162–178, 1985.
60. M. Kwiatkowska, G. Norman, and J. Sproston. Probabilistic Model Checking of the IEEE 802.11 Wireless Local Area Network Protocol. In H. Hermanns and R. Segala, editors, *Process Algebra and Probabilistic Methods, Performance Modeling and Verification, Second Joint International Workshop PAPM-PROBMIV 2002*, volume 2399 of *Lecture Notes in Computer Science*, pages 169–187, Copenhagen, Denmark, July 25-26 2002. Springer.
61. K. G. Larsen. Modal Specifications. In J. Sifakis, editor, *Automatic Verification Methods for Finite State Systems*, volume 407 of *Lecture Notes in Computer Science*, pages 232–246. Springer Verlag, June 12–14 1989. International Workshop, Grenoble, France.
62. K. G. Larsen and A. Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94(1):1–28, September 1991.
63. K. G. Larsen and B. Thomsen. A Modal Process Logic. In *Third Annual Symposium on Logic in Computer Science*, pages 203–210. IEEE Computer Society Press, 1988.
64. R. Lassaigne and S. Peyronnet. Approximate Verification of Probabilistic Systems. In H. Hermanns and R. Segala, editors, *Process Algebra and Probabilistic Methods, Performance Modeling and Verification, Second Joint International Workshop PAPM-PROBMIV 2002*, volume 2399 of *Lecture Notes in Computer Science*, pages 213–214, Copenhagen, Denmark, July 25-26 2002. Springer.
65. K. Martin. The measurement process in domain theory. In *In: Proc. of Automata, Languages and Programming (ICALP'00)*, pages 116–126, 2000.
66. A. McIver and C. Morgan. Almost-certain eventualities and abstract probabilities in quantitative temporal logic. In Colin Fidge, editor, *Electronic Notes in Theoretical Computer Science*, volume 42. Elsevier Science Publishers, 2001.
67. A.K. McIver and C.C. Morgan. Games, probability and the quantitative μ-calculus qMμ. In *Proc. 9th Int. Conf. on Logic for Programming, Artificial Intelligence and Reasoning, LPAR 2002*, volume 2514 of *Lecture Notes in Artificial Intelligence*, pages 292–310, Tbilisi, Georgia, 2002. Springer Verlag.
68. R. Milner. An algebraic definition of simulation between programs. In *2nd International Joint Conference on Artificial Intelligence*, pages 481–489, London, United Kingdom, 1971. British Computer Society.

69. R. Milner. *Communication and Concurrency.* Prentice-Hall, 1989.
70. R. Milner. *Communicating and Mobile Systems: the π-Calculus.* Cambridge University Press, 1999.
71. D. Monniaux. Abstract interpretation of programs as Markov decision processes. Technical report, Départment d'Informatique, École Normale Supérieure, 45, rue d'Ulm, 75230 Paris cedex 5, France, 2001.
72. D. Monniaux. Backwards abstract interpretation of probabilistic programs. In *European Symposium on Programming Languages and Systems (ESOP '01)*, number 2028 in Lecture Notes in Computer Science. Springer-Verlag, 2001.
73. David Monniaux. Abstract interpretation of probabilistic semantics. In *Seventh International Static Analysis Symposium (SAS'00)*, number 1824 in Lecture Notes in Computer Science. Springer-Verlag, 2000. Extended version on the author's web site.
74. David Monniaux. An abstract analysis of the probabilistic termination of programs. In *8th International Static Analysis Symposium (SAS'01)*, number 2126 in Lecture Notes in Computer Science. Springer-Verlag, 2001.
75. David Monniaux. An abstract Monte-Carlo method for the analysis of probabilistic programs (extended abstract). In *28th Symposium on Principles of Programming Languages (POPL '01)*, pages 93–101. Association for Computer Machinery, 2001.
76. David Monniaux. *Analyse de programmes probabilistes par interprétation abstraite.* Thèse de doctorat, Université Paris IX Dauphine, 2001. Résumé étendu en fran cais. Contents in English.
77. C. Morgan, A. McIver, K. Seidel, and J. W. Sanders. Refinement-oriented probability for CSP. *Formal Aspects of Computing*, 8(6):617–647, 1996.
78. Carroll Morgan and Annabelle McIver. An expectation-based model for probabilistic temporal logic. *Logic Journal of the IGPL*, 7(6):779–804, 1999.
79. Carroll Morgan, Annabelle McIver, and Karen Seidel. Probabilistic predicate transformers. *ACM Transactions on Programming Languages and Systems*, 18(3):325–353, May 1996.
80. R. Motvani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.
81. F. Nielson, H. R. Nielson, and C. Hankin. *Principles of Program Analysis.* Springer Verlag, 1999.
82. C. H. Papadimitriou. Games against nature. *Journal of Computer and System Sciences*, 31:288–301, 1985.
83. C. H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.
84. D. M. R. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *In Proc. of the 5th GI Conference*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer Verlag, 1989.
85. C. S. Pasareanu. DEOS kernel: Environment modeling using LTL assumptions. Technical Report #NASA-ARC-IC-2000-196, NASA Ames, July 2000.
86. A. Phillipou, I. Lee, and O. Sokolsky. Weak Bisimulation for Probabilistic Systems. In *Proceedings of the 11th International Conference on Concurrency Theory (*CONCUR*'02)*, volume 1877 of *Lecture Notes in Computer Science*, pages 334–349, University Park, Pennsylvania, August 2000. Springer Verlag.
87. A. Di Pierro, C. Hankin, and H. Wiklicky. Approximate Non-interference. In *In: CSFW'02 15th IEEE Computer Security Foundation Workshop*, pages 1–15, June 2002.

88. A. Di Pierro and H. Wicklicky. Probabilistic Abstract Interpretation and Statistical Testing. In H. Hermanns and R. Segala, editors, *Process Algebra and Probabilistic Methods, Performance Modeling and Verification, Second Joint International Workshop PAPM-PROBMIV 2002*, volume 2399 of *Lecture Notes in Computer Science*, pages 211–212, Copenhagen, Denmark, July 25-26 2002. Springer.
89. G. D. Plotkin. A Structural Approach to Operational Semantics. Technical Report FN-19, DAIMI, Computer Science Department, Aarhus University, Ny Munkegade, Building 540, DK-8000 Aarhus, Denmark, September 1981. Reprinted April 1991.
90. M. L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, 1994.
91. R.I. Bahar, E.A. Frohm, C.M. Gaona, G.D. Hachtel, E. Macii, A. Pardo, and F. Somenzi. Algebraic Decision Diagrams and Their Applications. In *IEEE /ACM International Conference on CAD*, pages 188–191, Santa Clara, California, 1993. IEEE Computer Society Press.
92. M. Sagiv, T. Reps, and R. Wilhelm. Parametric Shape Analysis via 3-Valued Logic. In *Proceedings of the 26th ACM SIGPLAN-SIGACT Symposium on Principles of programming languages*, pages 105–118, January 20-22, San Antonio, Texas 1999.
93. D. Scott. Continuous lattices. In F. W. Lawvere, editor, *Toposes, Algebraic Geometry and Logic*, volume 274 of *Lecture Notes in Mathematics*, pages 97–136. Springer Verlag, 1972.
94. R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, Laboratory for Computer Science, Massachusetts Institute of Technology, June 1995. Available as Technical Report MIT/LCS/TR-676.
95. R. Segala and N. Lynch. Probabilistic Simulations for Probabilistic Processes. *Nordic Journal of Computing*, 2(2):250–273, Summer 1995.
96. K. Seidel, C. Morgan, and A. McIver. An introduction to probabilistic predicate transformers. Technical Report PRG-TR-6-96, Programming Research Group, Oxford Computing Laboratory, Wolfson Building, Parks Road, Oxford OX1 3QD, 1996.
97. Bran Selic. Physical programming: Beyond mere logic, April 2001. Invited Talk at ETAPS 2001.
98. R. J. van Glabbeek and W. P. Weijland. Branching Time and Abstraction in Bisimulation Semantics. *Journal of the ACM*, 43(3):555–600, May 1996.
99. M. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *Proc. 26th IEEE Symp. on Foundations of Computer Science*, pages 327–338, Portland, Oregon, October 1985.
100. D. Volpano. Provably secure programming languages for remote evaluation. *ACM Computing Surveys*, 28A(2): electronic, December 1996.
101. P. Waszkiewicz. *Quantitative Continuous Domains*. PhD thesis, School of Computer Science, University of Birmingham, United Kingdom, July 2002.