Domains of view: a foundation for specification and analysis

Michael Huth

Department of Computing & Information Sciences, Kansas State University,
Manhattan, KS 66506-2302, huth@cis.ksu.edu,
WWW home page: http://www.cis.ksu.edu/~huth

Abstract. We propose a platform for the specification and analysis of systems. This platform contain models, their refinement and abstraction, and a temporal logic semantics; rendering a sound framework for property validation and refutation. The platform is parametric in a *domain of view*, an abstraction of a construction based on the Plotkin power domain. For each domain of view `E`, the resulting platform `P[E]`[1] contains *partial, incomplete* systems and complete systems — the actual implementations. Complete systems correspond to the platform that has as parameter a domain `D` that is, as a set, isomorphic to the maximal elements of `E`. If one restricts `P[E]` to implementations, but retains the temporal logic semantics, refinement, and abstraction relations, one recovers the platform `P[D]`. This foundation recasts existing work on modal transition systems, presents fuzzy systems, and ponders on the nature of probabilistic platforms. For domains of view `E` that are determined by a linearly ordered, complete lattice, we present a category of "relations" as a step toward a view-based semantics of predicate logic.

Keywords. modal transition systems, refinement, abstract interpretation, partial systems, property verification, property refutation, fuzzy systems, linear t-norms, Markov chains, Plotkin power domain.

1 Introduction

The specification and analysis of programs and software or hardware designs is an increasingly important and complex task that more and more working professionals are being confronted with. This paper is proposing a domain-theoretic foundation for frameworks within which systems that are possibly *incomplete* can be specified and analyzed.

Domain theory [2], as developed within the project of *denotational semantics* [42, 41, 36], had the original intent of providing a mathematical theory and framework for a formal programming language semantics. This project relied on domain theory's conceptual contribution of viewing complete information (often some infinite structure) as something that can be approximated, in a continuous

[1] This notation is only used within the abstract to simplify the discussion.

fashion, by partial information (a finite, computable structure). Since the degree of completeness is encoded in order-theoretic terms [39], complete information tokens are represented as maximal elements of a domain.[2]

The aim of this paper is to demonstrate that domain theory and its way of modeling complete and partial information can be successfully applied to the specification and analysis of systems. We identify implementations with complete specifications and propose refinement notions that render implementations as maximal elements. This allows for the consideration of partial or underspecified systems, where the degree and nature of the underspecification is being controlled by a *domain of view*. We show that a host of existing work on the specification and analysis of systems can be unified and extended within such a conceptual framework.

As a case study, we consider modal transition systems, as developed by K. G. Larsen and B. Thomsen [28] in Section 2. Based on a domain of view, we redefine these systems, their abstraction and refinement notions, and give them a semantics for a temporal logic. We then prove that this semantics provides a sound methodology for the validation and refutation of properties expressed in that logic. Domains of view are defined formally in that Section. *Concrete*, completely specified, modal transition systems, their refinement, and their semantics render the framework of labeled transition systems, bisimulation, and the standard semantics of temporal logics [30].[3] A domain of view, based on the interval domain [31, 39], is being applied to formulate a fuzzy version of modal transition systems in Section 3. It then becomes apparent that domains of view determine the notions of models, their refinement and abstraction, and their semantics of properties. Proofs of theorems within these frameworks are *generic* in such domains of views.

The same domain of view that is the basis for fuzzy modal transition systems is being used for a probabilistic framework, sketched in Section 4. Therein, we develop modal Markov chains and a corresponding notion of modal probability measures. The concrete notions turn out to be the established ones of Markov chains and probability measures, respectively. Section 5 explores the categorical foundations of such frameworks. It provides proof that, for a large class of domains of view, the corresponding notions of modal relations form a category, providing an initial step toward a modal semantics of predicate logic with respect to a domain of view. Finally, Section 6 points out past and ongoing, related work.

This paper only focuses on the foundational side of how to formulate and use specification and analysis frameworks in a uniform way, based on domains of view. However, we want to emphasize that such frameworks have algorithmic support for deciding refinement and abstraction instances and for computing temporal logic semantics. These aspects of view-based specification and analysis will be addressed in subsequent work.

[2] Alas, this identification does not preserve important type constructors — such as function spaces.

[3] This was already noted in the work of K. G. Larsen and B. Thomsen.

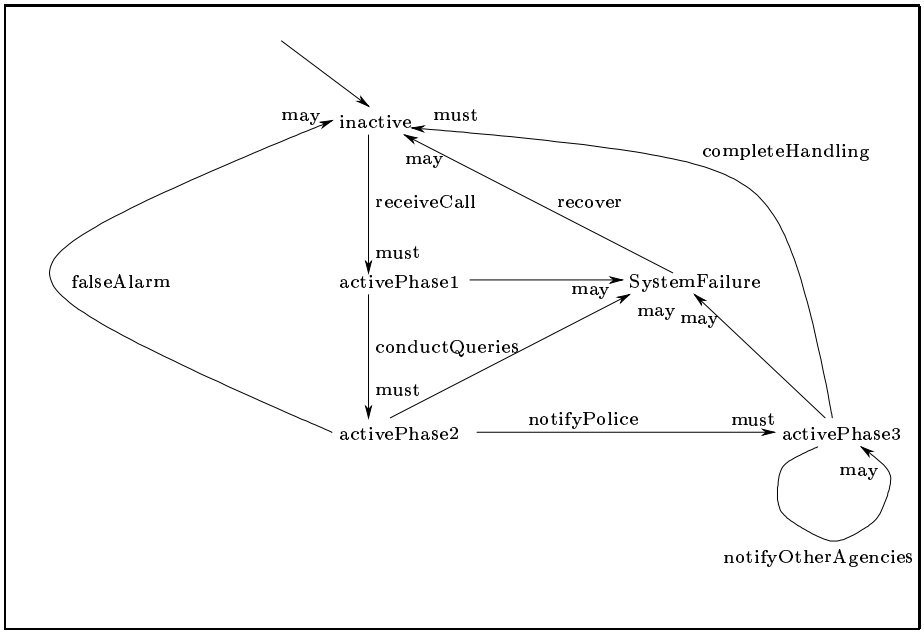

Fig. 1. Specification of a 911 calling center

2 Kripke modal transition systems

2.1 Specifying a 911 calling center

Imagine a 911 calling center: the facilities and processes that ensure that emergency calls will be received and acted upon accordingly and in a timely fashion. Figure 1 depicts a specification of such a center. In its initial state, `inactive`, the center must allow the reception of incoming calls. The specification documents this by a *must*-transition from `inactive` to the state `activePhase1`. In that state, an emergency call is active and the center *must* have the capacity to ask the "who, what, when, how many" kinds of questions, documented by a *must*-transition from `activePhase1` to `activePhase2`.

However, the center and its operating system may break down, e.g. due to a power outage. This is modeled by a *may*-transition from `activePhase1` to `SystemFailure`. For similar reasons, there are two more incoming *may*-transitions to `SystemFailure` from other active states. In state `activePhase2`, the operator has received all the information necessary to react to the call; we see a *must*-transition to state `activePhase3`, notifying the police; and a *may*-transition back to the initial state, caused by a potential false alarm. From state `activePhase3`, the center has to be able to complete the particular call, requiring a *must*-transition back to the initial state. For example, critical pre-arrival instructions may have to be given to the caller on how to perform the Heimlich Maneuver before emergency personnel could take further action.

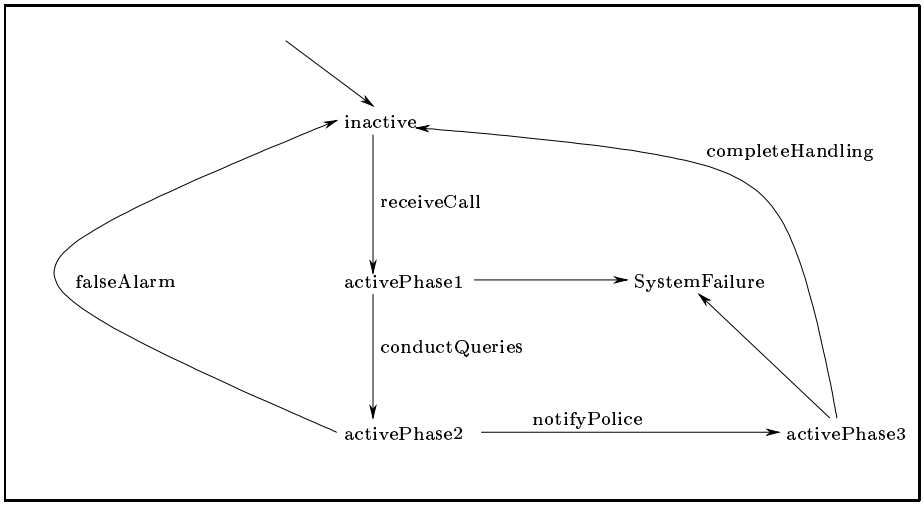

Fig. 2. A 911 calling center

A calling center may want to notify additional authorities, e.g. the fire fighters, an ambulance, or the State Department; this is modeled by a *may*-transition from `activePhase3` back to itself. Finally, a system failure — as devastating as it would be — may allow for some kind of recovery, documented by a *may*-transition from `SystemFailure` to `inactive`.[4]

In [28, 26], the intuition is that all *must*-transitions have to be implemented. The *may*-transitions are allowed, but not required, in an implementation. Additionally, only *may*- and *must*-transitions may be implemented. The presence of *may*-transitions allows for greater flexibility in implementation work. One realization could omit the *may*-transition from `activePhase1` to `inactive`, because false alarms could routinely be notified to the authorities. (It may be a legal offense to call 911 without proper cause.) Similarly, the *may*-transition from `activePhase3` back to itself may be dropped if the additional notifications are handled and determined by the police. Finally, an overconfident designer may omit some or all of the *may*-transitions into `SystemFailure`. In the extreme case, `SystemFailure` and all its incoming and outgoing transitions may not be implemented: the dream of any supervisor responsible for such centers.

Figures 2 and 3 show two valid implementations of the specification in Figure 1. Notice that both implementations possess all the *must*-transitions of Figure 1 as capacities, as required. They have different takes, though, on implementing or ignoring the specification's *may*-transitions.

[4] Some designer may prefer a *must*-transition from `SystemFailure` to `inactive`, arguing that, if such a failure indeed occurs, then the center desperately needs the capacity to regain the initial state, even if that would mean the loss of all currently active calls.

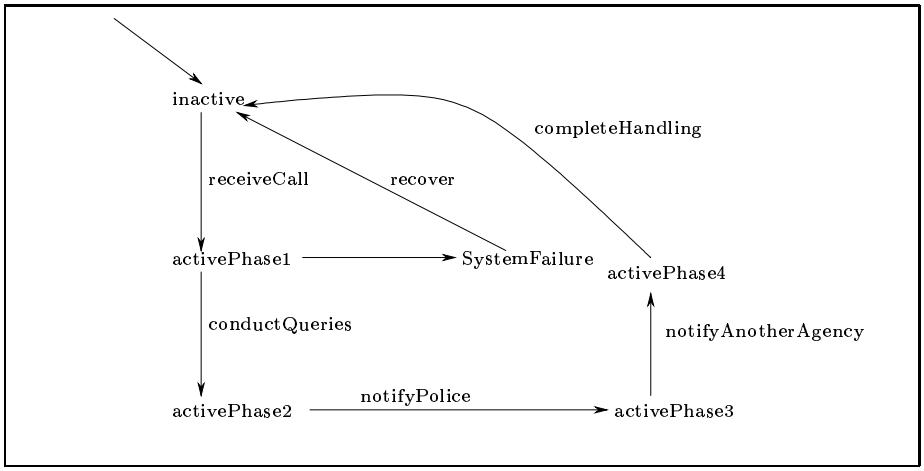

Fig. 3. Another 911 calling center

The first center opts out of the possibility of notifying agencies other than the police. It also guarantees that, once the queries are successfully completed, no system failure can occur before the police has been notified. The bad news is that this center cannot recover from such failure!

The second center drops the option of handling false alarms, rules out system failure after the successful completion of queries, allows for recovery from such failure, and always notifies one additional agency other than the police.

It should be clear that these two centers are quite different as far as their mode of operation is concerned. The specification and the two centers are all examples of modal transition systems (MTSs) [28, 26]. The two centers are *conrete* MTSs: all *may*-transitions have either been removed or implemented as *must*-transitions, making it unnecessary to use the *must*-annotations in the Figures. Thus such structures are labeled transition systems (LTSs) [30].

2.2 Kripke MTSs, refinement, and abstraction

We widen the scope of this paper to Kripke MTSs, generalizing the definition of MTSs and their refinements in [28]:

Definition 1 (Kripke MTS).

1. *A* Kripke modal transition system (Kripke MTS) *is a tuple,*

$$\mathcal{K} = \langle \Sigma_K, \mathtt{Act}, \mathtt{AP}, \longrightarrow_\Box, \longrightarrow_\Diamond, L_\Box, L_\Diamond \rangle, \tag{1}$$

where Σ_K is a set of states, Act *is a set of actions,* AP *is a set of atomic propositions, and*

- $\longrightarrow_\Box \subseteq \Sigma_K \times \mathtt{Act} \times \Sigma_K$ *is a set of transitions;*

- $\longrightarrow_\Diamond \subseteq \Sigma_K \times \texttt{Act} \times \Sigma_K$ *is a set of transitions such that for all* $s \in \Sigma_K$, $\{s' \in \Sigma_K \mid a \in \texttt{Act},\ s \longrightarrow^a_\Diamond s'\}$ *is finite;*[5]
- $L_\Box$ *and* $L_\Diamond$ *are functions of type* $\Sigma_K \to \mathcal{P}_{\text{fin}}(\texttt{AP})$;[6]
- $\longrightarrow_\Box \subseteq \longrightarrow_\Diamond$.
- *and* $L_\Box(s) \subseteq L_\Diamond(s)$ *for all* $s \in \Sigma_K$.

2. *A Kripke MTS is* pointed *if it has a distinguished start state.*
3. *A (pointed) MTS is defined similarly, by omitting* $\texttt{AP}, L_\Box$, *and* $L_\Diamond$ *in (1).*
4. *A Kripke MTS where* $\longrightarrow_\Box = \longrightarrow_\Diamond$ *and* $L_\Box = L_\Diamond$ *is* concrete.

Remark 1. In (1), $\longrightarrow_\Box$ represents *must*-transitions, whereas $\longrightarrow_\Diamond$ denotes *must*- or *may*-transitions. Notice that $\longrightarrow_\Box \subseteq \longrightarrow_\Box$ entails that the sets $\{s' \in \Sigma_K \mid a \in \texttt{Act},\ s \longrightarrow^a_\Box s'\}$ are finite as well.

Concrete MTSs are labeled transition systems (LTSs) [30]; concrete Kripke MTSs are doubly labeled transition systems (L^2TSs) [11] — the concrete MTSs only list the transition relation twice. In an ideal development world, specifications are refined to their implementations, and this refinement relationship is formal and can be validated. In most development worlds, implementations may lack specifications, or their format — e.g. `Java` source code — makes it difficult and error-prone to relate source code back to original specifications — e.g. given in `Z` [40]. An *abstraction* of software — e.g. a finite automata — allows validation of program behavior. However, since existing abstraction techniques add computation traces to abstractions [10, 9], *only safety properties ("something bad never happens") or invariants can be validated.* MTSs can be used as a framework for refinement, abstraction, and the validation of properties such that the latter can *combine* safety and liveness ("something good will happen") features.

Definition 2 (Refinement and abstraction). *[19, 37] A* refinement relation *between Kripke MTSs* $\mathcal{C}$ *and* $\mathcal{A}$, *and an* abstraction relation *between* $\mathcal{A}$ *and* $\mathcal{C}$, *is a relation* $\mathcal{R} \subseteq \Sigma_C \times \Sigma_A$ *such that, if* $s\mathcal{R}t$, *then*

1. *if* $t \longrightarrow^a_\Box t'$, *then there exists some* $s' \in \Sigma_C$ *such that* $s \longrightarrow^a_\Box s'$ *and* $s'\mathcal{R}t'$;
2. *if* $s \longrightarrow^a_\Diamond s'$, *then there exists some* $t' \in \Sigma_A$ *such that* $t \longrightarrow^a_\Diamond t'$ *and* $s'\mathcal{R}t'$;
3. $L_\Box(t) \subseteq L_\Box(s)$; *and*
4. $L_\Diamond(s) \subseteq L_\Diamond(t)$.

If $\mathcal{A}$ *and* $\mathcal{C}$ *are pointed with initial states* s_a *and* s_c, *respectively, we also insist that* $s_a\mathcal{R}s_c$ *hold.*

For general reasons [30, 28], each Kripke MTS $\mathcal{K}$ has a greatest refinement relation, denoted by $\prec_K$. One of the conceptual beauties of (Kripke) MTSs is that abstraction is simply the dual notion of refinement. The same techniques and tools may be used for validation of such relationships.

Example 1. Both 911 calling centers are easily shown to be refinements of the specification in Figure 1; at the same time, that specification may be seen as an abstraction of those centers.

[5] We write $s \longrightarrow^a_\Diamond s'$ and $s \longrightarrow^a_\Box s'$ for $(s, a, s') \in \longrightarrow_\Diamond$ and $(s, a, s') \in \longrightarrow_\Box$, respectively.

[6] We write $\mathcal{P}_{\text{fin}}(X)$ for the set of *finite* subsets of X.

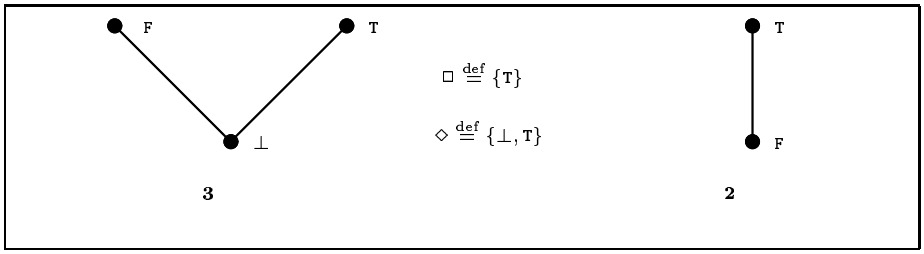

Fig. 4. The domain of view for Kripke MTSs

2.3 Domains of view

Remark 2. Consider the domains **3** and **2**, depicted in Figure 4.

1. (Kripke) MTSs can be seen as structures $(\Sigma_K, \mathtt{Act}, \mathtt{AP}, \longrightarrow, L)$, where $\longrightarrow\colon \Sigma_K \times \mathtt{Act} \times \Sigma_K \to \mathbf{3}$ and $L\colon \Sigma_K \times \mathtt{AP} \to \mathbf{3}$ are *functions* [19]. We may recover the original presentation of (1) by defining $\longrightarrow_\Box \stackrel{\text{def}}{=} \longrightarrow^{-1} \{\mathtt{T}\}$, $\longrightarrow_\Diamond \stackrel{\text{def}}{=} \longrightarrow^{-1} \{\bot, \mathtt{T}\}$, $L_\Box(s) \stackrel{\text{def}}{=} L(s,\cdot)^{-1}\{\mathtt{T}\}$, and $L_\Diamond(s) \stackrel{\text{def}}{=} L(s,\cdot)^{-1}\{\bot, \mathtt{T}\}$.
2. If the images of the functions $\longrightarrow$ and L are contained in **2**, such structures correspond to (doubly) labeled transition systems.
3. The set $\{\bot, \mathtt{T}\}$ is a lower set in **3**; the set $\{\mathtt{T}\}$ is an upper set in **3**; and the latter is strictly contained in the former.

This remark suggests that **3** is the underlying domain for (Kripke) MTSs, whereas **2** is the base for (doubly) labeled transition systems. Moreover, $\{\bot, \mathtt{T}\}$ models $\Diamond$ and $\{\mathtt{T}\}$ models $\Box$. We seek to capture the essence of this situation, so that it can be applied to other modeling scenarios, e.g. real-time, probabilistic, or fuzzy ones. Some of these scenarios will be addressed in subsequent Sections.

Definition 3 (Domain of view). *A* domain of view *for partial system specification and analysis is a structure* $(P, T, \Box^P, \Diamond^P)$ *such that*

- P *and* T *are domains;*
- $\Box^P$ *is an non-empty upper set in* P *and* $\Diamond^P$ *is a non-empty lower set in* P *such that* $\Box^P$ *is a proper subset of* $\Diamond^P$*;*
- T*, as a set, is isomorphic to the set of maximal elements of* P*; and*
- $\Box^P \cap T = \Diamond^P \cap T$*, if we identify* T *with its isomorphic image in* P*.*

The proviso that $\Diamond^P$ be non-empty is immediate; otherwise, no "view" is present. The proviso that $\Box^P$ be a *strict* subset of $\Diamond^P$ is that domains of view ought to allow *non-concrete* descriptions as well. Domains of view are closed under finite products and sums in an obvious, categorical, way.

Example 2. The domain of view for Kripke MTS specification and analysis is

$$(\mathbf{3}, \mathbf{2}, \{\mathtt{T}\}, \{\bot, \mathtt{T}\}) \tag{2}$$

where $\Box^{\mathbf{3}} \stackrel{\text{def}}{=} \{\mathtt{T}\}$ and $\Diamond^{\mathbf{3}} \stackrel{\text{def}}{=} \{\bot, \mathtt{T}\}$. Clearly, $\mathbf{2}$ is isomorphic to the set of maximal elements of $\mathbf{3}$, $\Box^{\mathbf{3}}$ is an non-empty upper and $\Diamond^{\mathbf{3}}$ a non-empty lower set in $\mathbf{3}$, strictly containing $\Box^{\mathbf{3}}$. Both $\Box^{\mathbf{3}} \cap \mathbf{2}$ and $\Diamond^{\mathbf{3}} \cap \mathbf{2}$ equal $\{\mathtt{T}\}$.

Each Kripke MTS $\mathcal{K}$ has a $\Box$-component, $(\Sigma_K, \mathtt{Act}, \longrightarrow_{\Box}, L_{\Box})$, and a $\Diamond$-component, $(\Sigma_K, \mathtt{Act}, \longrightarrow_{\Diamond}, L^{\mathtt{poss}})$; these are doubly labeled transition systems that result from retaining only the $\Box^{\mathbf{3}}$ and $\Diamond^{\mathbf{3}}$ information of $\mathcal{K}$, respectively.

2.4 Temporal logic for Kripke MTSs

In [26], K. Larsen proposed a temporal logic and gave it a semantics[7] that he used to characterize refinement. We extend this logic to ActCTL, a CTL-variant for MTSs, as well as to Kripke MTSs. Formulas of ActCTL are generated by

$$\phi ::= \top \mid p \mid \neg\phi \mid \phi \wedge \phi \mid \mathtt{AX}_K\, \phi \mid \mathtt{EX}_K\, \phi \mid \mathtt{AG}_K\, \phi \mid \mathtt{EG}_K\, \phi \mid \mathtt{A}_K[\phi\, \mathtt{U}\, \phi] \mid \mathtt{E}_K[\phi\, \mathtt{U}\, \phi] \quad (3)$$

where K ranges over all subsets of actions Act and p over the set AP of atomic propositions. We will omit the subscript K whenever it equals Act.

As done for the original logic CTL [8], we read A as "for all paths," E as "there exists a path," X as "next transition," G as "globally true along all transitions," and U as "until there is a transition." ($\top$ denotes "true.") The subscript, K, lists the actions that can appear along the paths we study. Given the MTS in Figure 1, we write

1. $\mathtt{AX}_{\{\mathtt{receiveCall}\}} \top$ to assert, "the next execution step is a receiveCall"
2. $\mathtt{A}[\top\, \mathtt{U}\, (\mathtt{EX}_{\{\mathtt{conductQueries}\}} \top)]$ to assert, "all paths include a chance to conduct queries"
3. $\mathtt{AG}[\mathtt{SystemFailure} \rightarrow \mathtt{EX}_{\{\mathtt{recover}\}}]$ to say, "whenever the system fails, the next execution step can recover". (We write $\phi \rightarrow \psi$ as an abbreviation of $\neg(\phi \wedge \neg\psi)$.)

Formula 1 intuitively holds in state inactive: the only possible execution step receives a call. Formula 2 holds for similar reasons. Formula 3 states that, at all reachable states where the system fails, there is a next execution step that recovers from this failure. (Note that we used Systemfailure as an atomic proposition.) This formula is possibly true, if the *may*/recover transition is being implemented. On the other hand, the formula is not necessarily true, for an implementor may choose not to realize that very transition. Formula 3 reveals that there are two principal ways of validating properties for MTSs.

Definition 4 (Semantics). *[19, 37] Let s be the start state of an MTS. We write*

1. $s \models^{\mathtt{nec}} \phi$ *("ϕ necessarily holds true at s"), meaning that ϕ holds for* all *refinements of s, including s itself;*

[7] Essentially, the semantics $\models^{\mathtt{nec}}$ below for a logic *without* negation or implication. In loc. cit., the notion $\models^{\mathtt{poss}}$ is not defined.

2. $s \models^{\mathtt{poss}} \phi$ (*"ϕ possibly holds true at s"), denoting that ϕ holds for* some *refinement of s.*

In stating the semantics of the two interpretations, we write $E_1 \Rightarrow E_2$ to denote that the result of evaluating E_1 is the result of evaluating E_2 in the complete lattice **2**. *The relations above hold if they evaluate to* `T`*; otherwise, they evaluate to* `F`*, so they won't hold. We write $\neg$, $\wedge$, $\bigvee$, and $\bigwedge$ for the respective logical operations in* **2**.

1. $(s \models^{M} \top) \Rightarrow \mathtt{T}$, *for* $M \in \{\mathtt{nec}, \mathtt{poss}\}$
2. $(s \models^{\mathtt{nec}} p) \Rightarrow (p \in L_{\Box}(s))$; $(s \models^{\mathtt{poss}} p) \Rightarrow (p \in L_{\Diamond}(s))$
3. $(s \models^{M} \phi_1 \wedge \phi_2) \Rightarrow (s \models^{M} \phi_1) \wedge (s \models^{M} \phi_2)$, *for* $M \in \{\mathtt{nec}, \mathtt{poss}\}$
4. $(s \models^{\mathtt{nec}} \neg\phi) \Rightarrow \neg(s \models^{\mathtt{poss}} \phi)$; $(s \models^{\mathtt{poss}} \neg\phi) \Rightarrow \neg(s \models^{\mathtt{nec}} \phi)$
5. $(s \models^{\mathtt{nec}} \mathtt{AX}_K \phi) \Rightarrow \bigwedge\{s' \models^{\mathtt{nec}} \phi \mid s \longrightarrow^{a}_{\Diamond} s',\ a \in K\}$;
   $(s \models^{\mathtt{poss}} \mathtt{AX}_K \phi) \Rightarrow \bigwedge\{s' \models^{\mathtt{poss}} \phi \mid s \longrightarrow^{a}_{\Box} s',\ a \in K\}$
6. $(s \models^{\mathtt{nec}} \mathtt{EX}_K \phi) \Rightarrow \bigvee\{s' \models^{\mathtt{nec}} \phi \mid s \rightarrow^{a}_{\Box} s',\ a \in K\}$;
   $(s \models^{\mathtt{poss}} \mathtt{EX}_K \phi) \Rightarrow$ *as above: replace* `nec` *by* `poss` *and* $\Box$ *by* $\Diamond$
7. $(s \models^{\mathtt{nec}} \mathtt{AG}_K \phi) \Rightarrow \bigwedge_{\pi}\{s_i \models^{\mathtt{nec}} \phi \mid \pi$ *equals* $s = s_0 \rightarrow^{a_1}_{\Diamond} s_1 \rightarrow^{a_2}_{\Diamond} s_2 \rightarrow \cdots \rightarrow^{a_i}_{\Diamond} s_i \ldots,$ *all* $a_i \in K\}$;
   $(s \models^{\mathtt{poss}} \mathtt{AG}_K \phi) \Rightarrow$ *as above: replace* `nec` *by* `poss` *and* $\Diamond$ *by* $\Box$
8. $(s \models^{\mathtt{nec}} \mathtt{EG}_K \phi) \Rightarrow \bigvee_{\pi}\{s_i \models^{\mathtt{nec}} \phi \mid \pi$ *equals* $s = s_0 \rightarrow^{a_1}_{\Box} s_1 \rightarrow^{a_2}_{\Box} s_2 \rightarrow \cdots \rightarrow^{a_i}_{\Box} s_i \ldots,$ *all* $a_i \in K\}$;
   $(s \models^{\mathtt{poss}} \mathtt{EG}_K \phi) \Rightarrow$ *as above: replace* `nec` *by* `poss` *and* $\Box$ *by* $\Diamond$
9. $(s \models^{\mathtt{nec}} \mathtt{A}_K[\phi_1 \,\mathtt{U}\, \phi_2]) \Rightarrow \bigwedge_{\pi} \bigvee_{j \geq 0}\{(s_j \models^{\mathtt{nec}} \phi_2) \wedge (s_0 \models^{\mathtt{nec}} \phi_1) \wedge (s_1 \models^{\mathtt{nec}} \phi_1) \wedge \cdots \wedge (s_{j-1} \models^{\mathtt{nec}} \phi_1) \mid \pi$ *equals* $s = s_0 \rightarrow^{a_1}_{\Diamond} s_1 \rightarrow^{a_2}_{\Diamond} s_2 \rightarrow \ldots,$ *all* $a_i \in K\}$;
   $(s \models^{\mathtt{poss}} \mathtt{A}_K[\phi_1 \,\mathtt{U}\, \phi_2]) \Rightarrow$ *as above: replace* `nec` *by* `poss` *and* $\Diamond$ *by* $\Box$
10. $(s \models^{\mathtt{nec}} \mathtt{E}_K[\phi_1 \,\mathtt{U}\, \phi_2]) \Rightarrow$ *and* $(s \models^{\mathtt{poss}} \mathtt{E}_K[\phi_1 \,\mathtt{U}\, \phi_2]) \Rightarrow$ *: similar to above.*

Admittedly, the style of this definition is somewhat unconventional. But it emphasizes the operational nature of these relations. For example, to evaluate $s \models^{\mathtt{nec}} \neg\phi$, evaluate $s \models^{\mathtt{poss}} \phi$ and negate that result. The semantics for negation is due to P. Kelb [23]. The other clauses just encode the usual CTL semantics for $\models^{\mathtt{nec}}$ and $\models^{\mathtt{poss}}$ respectively, only that the quantifiers `A` and `E` focus on the $\Box$- or $\Diamond$-components of $\mathcal{K}$. For example, $s \models^{\mathtt{nec}} \mathtt{A}[\phi_1 \mathtt{U} \phi_2]$ holds iff for all $\Diamond$-paths that begin in s and take only actions from K, there is a state s_j on that path that satisfies $s_j \models^{\mathtt{nec}} \phi_2$ and all previous states, s', on that path satisfy $s' \models^{\mathtt{nec}} \phi_1$. Another reason for choosing this definitional style is that, in subsequent Sections, we will make good use of it when re-interpreting such properties over Kripke MTSs that have numerical information attached to propositions and transitions.

Notice that, in evaluating $\models^{\mathtt{nec}}$, the `A` connective always inspects $\Diamond$-transitions and $\Diamond$-propositions, whereas the `E` connective always inspect $\Box$-transitions and $\Box$-propositions; the dual observation holds for $\models^{\mathtt{poss}}$.

For concrete Kripke structures, we have $L_{\Box} = L_{\Diamond}$ and $\longrightarrow_{\Box} = \longrightarrow_{\Diamond}$. In that case, Definition 4 specializes to the familiar semantics of `ActCTL` over LTSs (see, e.g., [5]).

Remark 3. 1. Let $\mathcal{K}$ be a Kripke structure such that $L_\Box = L_\Diamond$ and $\longrightarrow_\Box = \longrightarrow_\Diamond$. Then $\models^{\mathtt{nec}}$ equals $\models^{\mathtt{poss}}$ and corresponds to the usual semantics of ActCTL over labeled transition systems.

2. Let $\mathcal{K}$ be a MTS such that $\longrightarrow_\Box = \longrightarrow_\Diamond$. Then the notions of refinement and abstraction on $\mathcal{K}$ coincide with that of bisimulation [29, 32].

Thus our framework for partial systems — Kripke MTSs, their ActCTL semantics, and refinement — subsumes LTSs, their ActCTL semantics, and bisimulation as the corresponding notions of concrete MTSs.[8] This theme re-occurs when we consider fuzzy MTSs, modal Markov chains, and modal relations in Sections 3, 4, and 5, respectively.

Example 3. Let us evaluate

$$\mathtt{inactive} \stackrel{?}{\models}^{\mathtt{nec}} \mathtt{AG}[\mathtt{SystemFailure} \rightarrow \mathtt{EX}_{\{\mathtt{recover}\}}], \tag{4}$$

saying that "whenever the system fails, it is necessarily the case that the next execution step can recover". (Note that $\phi \rightarrow \psi$ is an abbreviation of $\neg(\phi \wedge \neg\psi)$). Inspecting Definition 4, we need to establish that $s \models^{\mathtt{nec}} \mathtt{EX}_{\{\mathtt{recover}\}}$ holds for all states s that are reachable from inactive by means of a $\Diamond$-path and that satisfy $s \models^{\mathtt{poss}} \mathtt{SystemFailure}$. Note the use of $\models^{\mathtt{poss}}$, caused by negating the antecedent of the implication. We discover that (4) does *not* hold: there is a $\Diamond$-path to a state where SystemFailure must be true (and therefore *may* be true), such that no next step necessarily recovers: the only recovering transition is a *may*-transition that is *not* a *must*-transition. This result is to be expected, for the 911 calling center in Figure 2 does not validate that check either.

The center in Figure 3, however, does satisfy that property, so

$$\mathtt{inactive} \stackrel{?}{\models}^{\mathtt{poss}} \mathtt{AG}[\mathtt{SystemFailure} \rightarrow \mathtt{EX}_{\{\mathtt{recover}\}}] \tag{5}$$

should hold. Indeed, all $\Box$-paths that reach a state where SystemFailure must hold — there is only one such state — validate $\mathtt{SystemFailure} \models^{\mathtt{poss}} \mathtt{EX}_{\{\mathtt{recover}\}}$ due to the recovering *may*-transition.

This example illustrates the sound use of filtering techniques [12], based on conditionals: verify ψ under the filtering assumption ϕ — which typically encodes properties of an environment within which the system is being placed — by verifying $\phi \rightarrow \psi$. Because of the negative polarity of ϕ in $\phi \rightarrow \psi$, the soundness of such filtering is problematic, if carried out in conventional abstraction frameworks [10, 9].

The logic ActCTL is very expressive and many properties that combine safety and liveness aspects, such as Formula 3, have straightforward codings within this logic. Thus the semantics of ActCTL for MTSs renders a sound framework for property validation and refutation:

[8] This was essentially realized in [26] already.

Theorem 1 (Sound validation and refutation). *[19, 37] For all ϕ of* ActCTL *and states s, s', t, t' of MTSs,*

1. *if $s \models^{\mathtt{nec}} \phi$, then $t \models^{\mathtt{nec}} \phi$ for all refinements t of s; dually,*
2. *if $t' \models^{\mathtt{poss}} \phi$, then $s' \models^{\mathtt{poss}} \phi$ for all abstractions s' of t'.*

This Theorem can be shown by standard methods (see e.g. [30]); one merely has to note the novel treatment of negation and adhere to Definition 4. To illustrate this treatment of negation, we prove part of the Theorem below; it secures the consistency of our framework for validation and refutation.

Theorem 2 (Consistency of validation and refutation). *For a MTS, we have $\models^{\mathtt{nec}} \subseteq \models^{\mathtt{poss}}$, $s \not\models^{\mathtt{nec}} \phi \wedge \neg\phi$, and $s \models^{\mathtt{poss}} \phi \vee \neg\phi$, for all states s and all $\phi \in$ ActCTL.*

Proof. 1. The inclusion $\models^{\mathtt{nec}} \subseteq \models^{\mathtt{poss}}$ is shown by a standard inductive argument.
2. We have $s \models^{\mathtt{nec}} \phi \wedge \neg\phi$ iff we have $s \models^{\mathtt{nec}} \phi$ and $s \models^{\mathtt{nec}} \neg\phi$; the former implies $s \models^{\mathtt{poss}} \phi$ by the first item, the latter means $s \not\models^{\mathtt{poss}} \phi$, contradicting that $\models^{\mathtt{poss}}$ is a binary relation.
3. If $s \not\models^{\mathtt{poss}} \phi$, then $s \models^{\mathtt{nec}} \neg\phi$ by definition. But then $s \models^{\mathtt{poss}} \neg\phi$ follows from the first item. Thus $s \models^{\mathtt{poss}} \phi \vee \neg\phi$ holds.

Sound validation of ϕ for a pointed MTS with initial state s means a positive check of $s \models^{\mathtt{nec}} \phi$. Sound refutation of ϕ in that MTS means a positive check of $s \models^{\mathtt{nec}} \neg\phi$, i.e. a negative check of $s \models^{\mathtt{poss}} \phi$.

3 Fuzzy Kripke modal transition systems

3.1 Specifying a fuzzy 911 calling center

The specification of a 911 calling center and its two implementations in Section 2 only capture the *qualitative* nature of such centers; they do not prescribe, nor guarantee, any *quantitative* assertions, such as the probability, the cost, or other performance measures of individual transitions. If such additional information is being supplied, we anticipate the latter to be consistent with the center's qualitative specification. In this Section, we present a *fuzzy* view of specifications and analyzes of systems. In Figure 5, we amended the specification of Figure 1 with numerical information. Each transition is annotated with an interval $[x, y]$ such that x and y are the minimal and maximal "likelihood", respectively, of this transition to be executed. We will re-formulate Kripke MTSs, their refinement/abstraction, and their ActCTL semantics under this fuzzy view. In doing so, we will take care in stressing the uniformity of such frameworks across these domains of view.

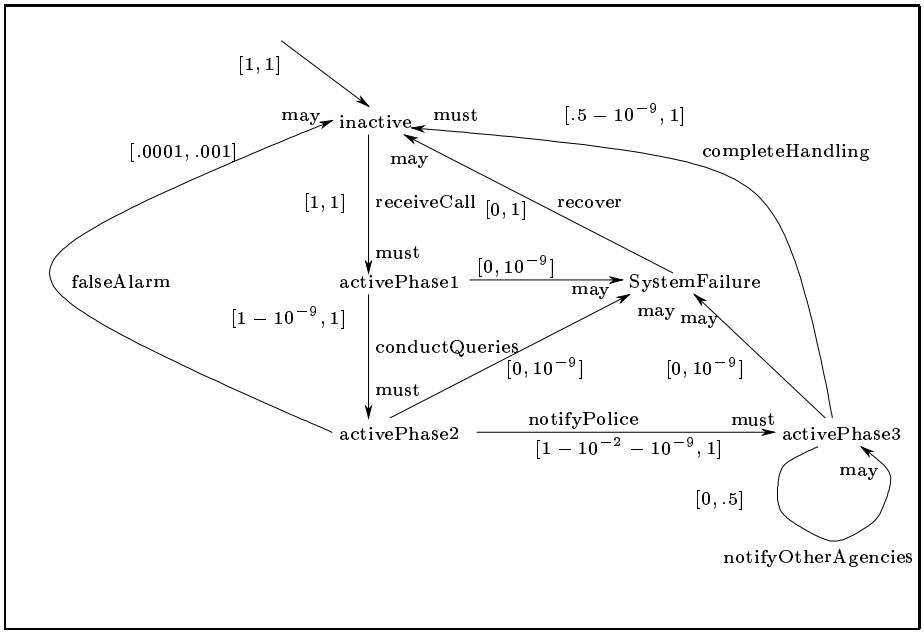

Fig. 5. Fuzzy specification of a 911 calling center

3.2 A fuzzy domain of view

First, we identify the underlying domain of view for the fuzzy framework, seen in Figure 6. In Section 4, we see that this is also the domain of view for modal Markov chains.

Definition 5 (Fuzzy domain of view). *A* domain of view for fuzzy or probabilistic system specification and analysis *is the structure* $(\mathbf{I}, [\mathbf{0}, \mathbf{1}], \Box^{\mathbf{I}}, \Diamond^{\mathbf{I}})$, *where*

1. $\mathbf{I}$ *is the interval domain [31, 39]: the set of all intervals* $[x, y]$ *with* $0 \leq x \leq y \leq 1$, *ordered by*

$$[x, y] \leq [u, v] \text{ iff } x \leq u,\ v \leq y; \tag{6}$$

   $[\mathbf{0}, \mathbf{1}]$ *is the unit interval, a complete lattice with* 0 *as bottom and* 1 *as top, in the usual order;*
2. $\Diamond^{\mathbf{I}} \stackrel{\text{def}}{=} \{[x, y] \in \mathbf{I} \mid y > 0\}$;
3. $\Box^{\mathbf{I}} \stackrel{\text{def}}{=} \{[x, y] \in \mathbf{I} \mid x > 0\}$.

Lemma 1. *The structure of Definition 5 is a domain of view.*

Proof. – $\mathbf{I}$ and $[\mathbf{0}, \mathbf{1}]$ are domains;
– $\Box^{\mathbf{I}}$ is an non-empty upper set in $\mathbf{I}$: e.g. $[.5, .7] \in \Box^{\mathbf{I}}$, and if $x > 0$ in (6), then $u > 0$ follows; $\Diamond^{\mathbf{I}}$ is a non-empty lower set in $\mathbf{I}$: e.g. $[0, .2] \in \Diamond^{\mathbf{I}}$, and if

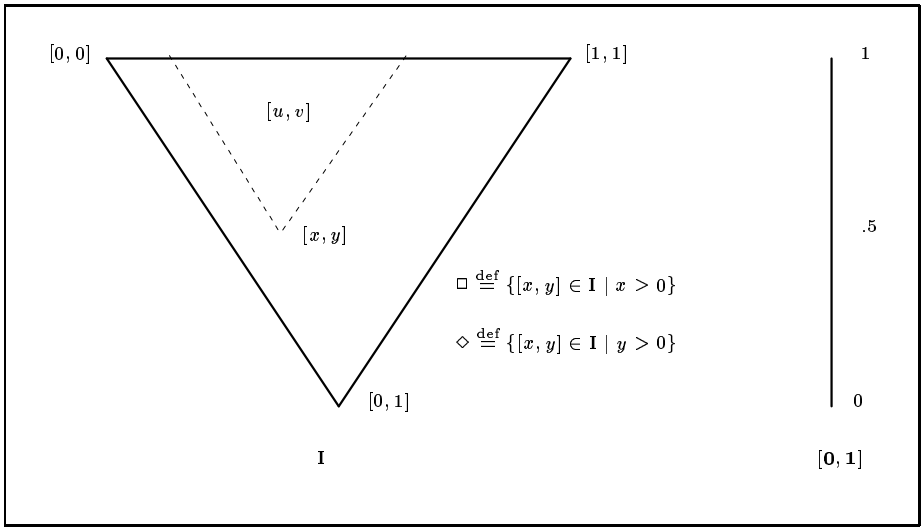

Fig. 6. The domain of view for fuzzy Kripke MTSs

$v > 0$ in (6), then $y > 0$ follows; and $\Box^{\mathbf{I}} \subseteq \Diamond^{\mathbf{I}}$: if $x > 0$ in (6), then $y > 0$ follows — this inclusion is strict since, for example, $[0, .2]$ is contained in the latter, but not in the former set;
- $[\mathbf{0}, \mathbf{1}]$, as a set, is isomorphic to the set of maximal elements of P, which is $\{[x, x] \mid x \in [\mathbf{0}, \mathbf{1}]\}$; and
- $\Box^P \cap T = \Diamond^P \cap T$, if we identify T with its isomorphic image in P: this is immediate since such elements are of the form $[x, x]$.

The domains of view from Figures 4 and 6 are special cases of a general construction that generates domains of view with the aid of the Plotkin power domain [33].

Proposition 1 (Constructing domains of view). *Let T be a domain and U a non-empty, proper, upper set in T. Then $(\mathsf{P}[T], T, \Box^{\mathsf{P}[T]}, \Diamond^{\mathsf{P}[T]})$ is a domain of view, where*

- $\mathsf{P}[T]$ *is the Plotkin power domain of T, the set of Lawson-compact, order-convex, non-empty subsets of T, ordered by* reverse *inclusion;*
- $\Box^{\mathsf{P}[T]} \stackrel{\text{def}}{=} \{C \in \mathsf{P}[T] \mid C \subseteq U\}$;
- $\Diamond^{\mathsf{P}[T]} \stackrel{\text{def}}{=} \{C \in \mathsf{P}[T] \mid C \cap U \neq \emptyset\}$.

Proof. – $\mathsf{P}[T]$ is a domain if T is one [33].
- $\Box^{\mathsf{P}[T]}$ is an non-empty upper set in $\mathsf{P}[T]$: since U is non-empty, each $\{c\}$ with $c \in U$ is in $\Box^{\mathsf{P}[T]}$, and if $C \subseteq U$ and C' is above C is $\mathsf{P}[T]$ (i.e. $C' \subseteq C$), then $C' \subseteq U$ follows; $\Diamond^{\mathsf{P}[T]}$ is a non-empty lower set in $\mathsf{P}[T]$: non-emptiness follows from the non-emptiness of U, and if $C \cap U \neq \emptyset$ and C'' is below C in $\mathsf{P}[T]$ (i.e. $C \subseteq C''$), then $C'' \cap U \neq \emptyset$ follows; $\Box^{\mathsf{P}[T]} \subseteq \Diamond^{\mathsf{P}[T]}$: if $C \subseteq U$, then

$C \cap U \neq \emptyset$ follows, for U is non-empty; to see the strictness of the inclusion, we have some $c \in U$ and $\{t \in T \mid t \leq c\}$ is Lawson-closed, non-empty, order-convex, and contains 0. Moreover, it intersects U, but it is not contained in U, for $0 \in U$ would imply $U = T$;
- T, as a set, is isomorphic to the set of maximal elements of $\mathsf{P}[T]$, which is $\{\{x\} \mid x \in T\}$; and
- $\Box^{\mathsf{P}[T]} \cap T = \Diamond^{\mathsf{P}[T]} \cap T$, if we identify T with its isomorphic image in $\mathsf{P}[T]$: this is immediate since such elements are of the form $\{x\}$.

The proviso that U be non-empty guarantees that $\Diamond^{\mathsf{P}[T]}$ is non-empty. The proviso that U is a strict subset of T ensures that $\Box^{\mathsf{P}[T]}$ is a strict subset of $\Diamond^{\mathsf{P}[T]}$.

Remark 4. The domains of view from Example 2 and Lemma 1 are examples of this construction. For the former, the non-empty, proper, upper set in **2** is $\{\mathtt{T}\}$ and **3** is isomorphic to the Plotkin power domain of **2**. For the latter, the non-empty, proper, upper set in $[\mathbf{0}, \mathbf{1}]$ is $(0, 1]$, the set of strictly positive numbers in $[\mathbf{0}, \mathbf{1}]$. The Plotkin power domain of $[\mathbf{0}, \mathbf{1}]$ is isomorphic to $\mathbf{I}$. The reader will easily verify that the definitions of $\Box^P$ and $\Diamond^P$ from Example 2 and Lemma 1 match the ones given in the construction above.

3.3 Fuzzy Kripke MTSs, refinement, and abstraction

Replacing the domain of view from Example 2 with the one from Lemma 1, we arrive at the framework of fuzzy Kripke MTSs.

Definition 6. *[19]*

1. *A* fuzzy Kripke modal transition system (fuzzy Kripke MTS) *is a tuple,*

$$\mathcal{K} = \langle \Sigma_K, \mathtt{Act}, \mathtt{AP}, \longrightarrow, L \rangle, \tag{7}$$

   where Σ_K is a set of states, $\mathtt{Act}$ *is a set of actions,* $\mathtt{AP}$ *is a set of atomic propositions, and*
   - $\longrightarrow : \Sigma_K \times \mathtt{Act} \times \Sigma_K \to \mathbf{I}$ *is a function such that for all $s \in \Sigma_K$ and all $a \in \mathtt{Act}$, $\{s' \in \Sigma_K \mid s \longrightarrow^a_\Diamond s'\}$ is finite,*[9]
   - $L \colon \Sigma_K \times \mathtt{AP} \to \mathbf{I}$ *is a function such that, for all $s \in \Sigma_K$, the set $\{p \in \mathtt{AP} \mid \Diamond L(s,p)\}$ is finite;*[10]
2. *A fuzzy Kripke MTS is* pointed *if it has a distinguished start state.*
3. *A (pointed) fuzzy MTS is defined similarly, by omitting* $\mathtt{AP}$ *and L in (7).*
4. *A fuzzy Kripke MTS where $\longrightarrow$ and L map into the image of $[\mathbf{0}, \mathbf{1}]$ in $\mathbf{I}$ is* concrete.

Notice that $\Diamond^{\mathbf{I}} \subseteq \Box^{\mathbf{I}}$ entails that the sets $\{s' \in \Sigma_K \mid s \longrightarrow^a_\Box s'\}$ and $\{p \in \mathtt{AP} \mid \Box L(s,p)\}$ are finite as well. The specification in Figure 5 is a fuzzy MTS. Its two implementations in Figures 7 and 8 are concrete fuzzy MTSs. We formalize refinement and abstraction relations for such systems. With these concepts, one easily validates that the implementations of Figures 7 and 8 are refinements of the specification in Figure 5.

[9] We write $s \longrightarrow^a_\Diamond s'$ for $s \longrightarrow^a s' \in \Diamond^{\mathbf{I}}$, and $s \longrightarrow^a_\Box s'$ for $s \longrightarrow^a s' \in \Box^{\mathbf{I}}$.

[10] We write $\Diamond$ instead of $\Diamond^{\mathbf{I}}$.

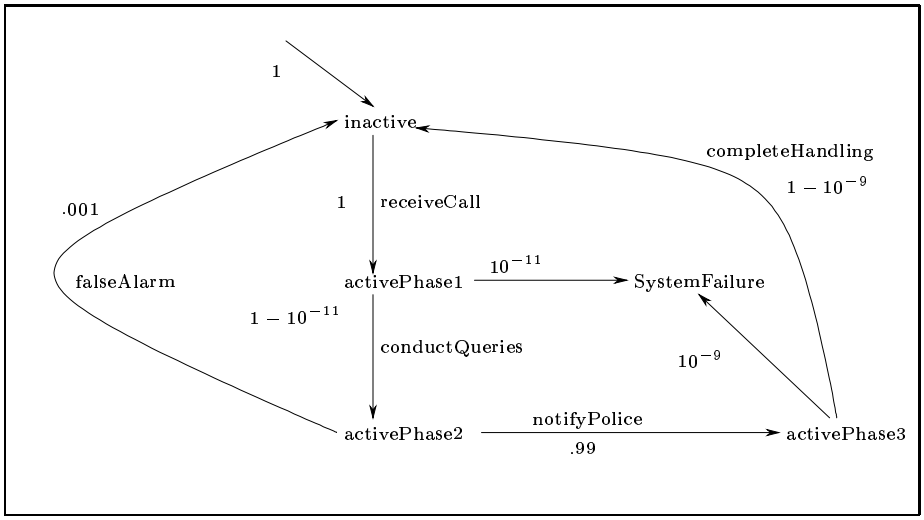

Fig. 7. A fuzzy 911 calling center

Definition 7 (Refinement and abstraction). *[19] A* refinement relation *between fuzzy Kripke MTSs $\mathcal{C}$ and $\mathcal{A}$, and an* abstraction relation *between $\mathcal{A}$ and $\mathcal{C}$, is a relation $\mathcal{R} \subseteq \Sigma_C \times \Sigma_A$ such that, if $s\mathcal{R}t$, then*

1. *if $t \longrightarrow^a_\Box t'$, then there exists some $s' \in \Sigma_C$ such that $s \longrightarrow^a_\Box s'$, $(t \longrightarrow^a t') \leq (s \longrightarrow^a s')$, and $s'\mathcal{R}t'$;*
2. *if $s \longrightarrow^a_\Diamond s'$, then there exists some $t' \in \Sigma_A$ such that $t \longrightarrow^a_\Diamond t'$, $(t \longrightarrow^a t') \leq (s \longrightarrow^a s')$, and $s'\mathcal{R}t'$;*
3. *for all $p \in \mathtt{AP}$, $\Box L(t,p)$ implies $\Box L(s,p)$ and $L(t,p) \leq L(s,p)$; and*
4. *for all $p \in \mathtt{AP}$, $\Diamond L(s,p)$ implies $\Diamond L(t,p)$ and $L(t,p) \leq L(s,p)$.*

If $\mathcal{A}$ and $\mathcal{C}$ are pointed with initial states s_a and s_c, respectively, we also insist that $s_a \mathcal{R} s_c$ hold.

For general reasons [30, 28, 22], each fuzzy Kripke MTS $\mathcal{K}$ has a greatest refinement relation, denoted by $\prec_K$. Notice the inclusion of the inequality

$$(t \longrightarrow^a t') \leq (s \longrightarrow^a s') \tag{8}$$

in the co-inductive definition of $\prec_K$. This ensures that, say, a *must*-transition with interval $[.2, .6]$ is not only refined with a *must*-transition, but with one whose interval in contained in $[.2, .6]$. This proviso is *absent* from the corresponding refinement definition for Kripke MTSs and also absent in [28, 26]. In Definition 2 we could have included it, insisting on a *may*-transition which is not a *must*-transition to be abstracted by a transition that is also not a *must*-transition. Theorem 3 could then also be proved, giving us soundness of validation and refutation as before [19], but we would not obtain the logical characterization of [26]. However, both refinement notions coincide if the refining system happens

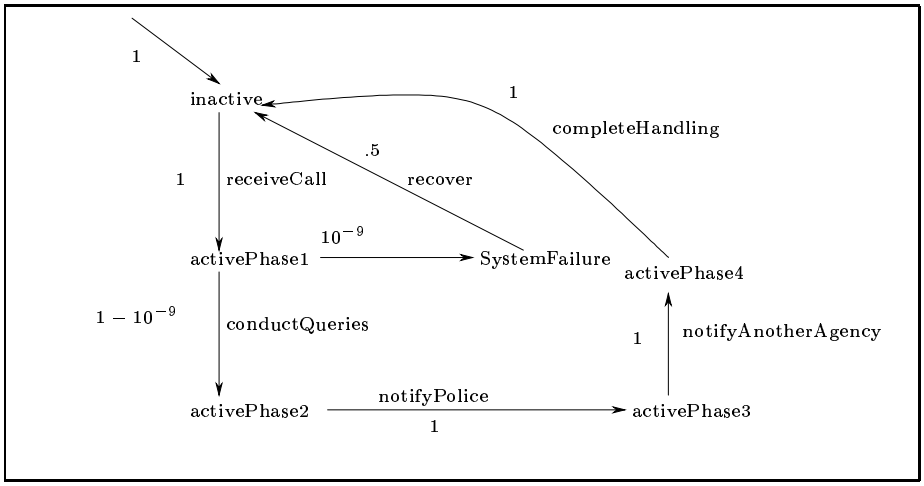

Fig. 8. Another fuzzy 911 calling center

to be concrete. Thus, the difference is largely irrelevant from a practical point of view.[11] In any event, the inequalities (8) have to be enforced in the quantitative view.

3.4 Temporal logic for fuzzy Kripke MTSs

The temporal logic ActCTL can be interpreted over fuzzy Kripke MTSs. For the sake of brevity, we will specify this semantics for ActMu, a version of the modal mu-calculus [24].[12] The logic ActCTL can be seen as a fragment of that logic; for details see, e.g., [5]. The grammar for ActMu is given by

$$\phi ::= \top \mid Z \mid p \mid \neg\phi \mid \phi \wedge \phi \mid \mathtt{AX}_K\, \phi \mid \mu Z.\phi \tag{9}$$

where K ranges over all subsets of actions Act, p over the set AP of atomic propositions, and Z over a set of variables. We will omit the subscript K whenever it equals Act. This free grammar is constrained by allowing only recursive bodies ϕ in $\mu Z.\phi$ such that any free[13] occurrence of Z in ϕ is in the scope of an even number of $\neg$ symbols. For example, $\mu Z.\neg Z$ is not allowed, whereas $\mu Z.p \wedge \neg(q \vee \neg Z)$ is. We can recover all formulas of the logic ActCTL. For example, $\mathtt{EX}_K\, \phi$ is expressed as $\neg\mathtt{AX}_K\, \neg\phi$; and $\mathtt{AG}_K\, \phi$ is coded as $\neg\mu Z.\neg(\phi \wedge \mathtt{AX}_K\, \neg Z)$ [5].

The semantics we choose again involves two relations, $\models^{\mathtt{nec}}$ and $\models^{\mathtt{poss}}$. But this time, expressions $(s \models^M \phi)$ denote elements of $[\mathbf{0}, \mathbf{1}]$, for $M \in \{\mathtt{nec}, \mathtt{poss}\}$. Intuitively, $(s \models^{\mathtt{nec}} \phi) = .99$ means that there is a 99% guarantee that ϕ holds at

[11] This is similar to the situation of branching bisimulation and weak bisimulation in [43], where both notions coincide if one system has no tau moves.

[12] Our results from Section 2 naturally extend to the more expressive logic ActMu.

[13] We think of μZ as a binding construct [5], similar to quantifiers in logic.

s and all of its refinements. Dually, $(s \models^{\mathtt{poss}} \phi) = 1$ means that there is a 100% possibility that ϕ holds at s for some refinement. The extreme values 0 and 1 can be shown to reflect the corresponding truth values F and T for the Krikpe MTSs obtained from a fuzzy Kripke MTS by "forgetting the numbers, but retaining $\Box$ and $\Diamond$". We will not discuss such cross-consistency issues further, but note that such consistencies between these two semantics exist. A fuzzy semantics needs a compositional way of dealing with conjunction.

Definition 8. *A* linear t-norm *over a complete lattice T is a function $N\colon T \times T \to T$ that preserves all least upper bounds in each coordinate separately such that $(T, N, 1)$ is a commutative monoid.*

If we weaken the requirement of preserving all least upper bounds to being monotone, we arrive at the notion of t-norms [38]. Clearly, every linear t-norm is a t-norm. We need to secure a property of t-norms that is part of the folklore of fuzzy logic.

Lemma 2. *Let $N\colon T \times T \to T$ be a t-norm. Then $N(x, y) = 0$ implies $x = 0$ or $y = 0$. If N is linear, the converse holds as well.*

Proof. If N is linear, then the converse holds as N preserves 0, the least upper bound of $\emptyset$, in each coordinate. Given a t-norm N, we compute $N(x, y) \leq N(x, 1) = x$ since $(T, N, 1)$ is a monoid and N is monotone. But $N(x, y) = N(y, x)$ since the monoid $(T, N, 1)$ is commutative. So we get $N(x, y) = N(y, x) \leq N(y, 1) = y$ as before. Thus $N(x, y)$ is a lower bound of $\{x, y\}$, so $N(x, y) \leq x \wedge y$. But if $x = 0$ or $y = 0$, we have $x \wedge y = 0$, so $N(x, y) \leq x \wedge y$ implies $N(x, y) = 0$.

Example 4. Linear t-norms over $[\mathbf{0}, \mathbf{1}]$ are $(a, b) \mapsto \min(a, b)$ and $(a, b) \mapsto a \cdot b$. Not every t-norm over $[\mathbf{0}, \mathbf{1}]$ is linear. For example, the t-norm $(a, b) \mapsto \max(a + b - 1, 0)$ is not linear: given a and b to be .5, that norm computes 0.

Definition 9 (Semantics). *Let s be the start state of an MTS and N a linear t-norm. We write*

1. $(s \models^{\mathtt{nec}} \phi) = x \in [\mathbf{0}, \mathbf{1}]$, *meaning that "$\phi$ necessarily has a x% guarantee that it holds at all refinements of s, including s itself";*[14]
2. $(s \models^{\mathtt{poss}} \phi) = y \in [\mathbf{0}, \mathbf{1}]$, *meaning that "$\phi$ has a y% possibility that it holds for* some *refinement of s".*

In stating the semantics of the two interpretations, we write $E_1 \Rightarrow E_2$ to denote that the result of evaluating E_1 is the result of evaluating E_2 in the complete lattice $[\mathbf{0}, \mathbf{1}]$. We write $\wedge$, $\bigvee$, and $\bigwedge$ for the respective logical/lattice-theoretical operations in $[\mathbf{0}, \mathbf{1}]$. We write

$$\mathrm{pr}_{\Box}, \mathrm{pr}_{\Diamond} : \mathbf{I} \to [\mathbf{0}, \mathbf{1}] \tag{10}$$

[14] These notions implicitly depend upon the t-norm N.

for the projections into the first and second coordinate, respectively.[15] *As standard, we define these relations with respect to environments ρ, functions that map variables Z to elements in* $\mathbf{I}$.

1. $(s \models^{M}_{\rho} \top) \Rightarrow 1$, *for* $M \in \{\mathtt{nec}, \mathtt{poss}\}$
2. $(s \models^{\mathtt{nec}}_{\rho} Z) \Rightarrow \mathrm{pr}_{\Box}\rho(Z)$;[16] $(s \models^{\mathtt{poss}}_{\rho} Z) \Rightarrow \mathrm{pr}_{\Diamond}\rho(Z)$
3. $(s \models^{\mathtt{nec}}_{\rho} p) \Rightarrow \mathrm{pr}_{\Box}L(s,p)$; $(s \models^{\mathtt{poss}}_{\rho} p) \Rightarrow \mathrm{pr}_{\Diamond}L(s,p)$
4. $(s \models^{\mathtt{nec}}_{\rho} \neg\phi) \Rightarrow 1 - (s \models^{\mathtt{poss}}_{\rho} \phi)$; $(s \models^{\mathtt{poss}}_{\rho} \neg\phi) \Rightarrow 1 - (s \models^{\mathtt{nec}}_{\rho} \phi)$
5. $(s \models^{M}_{\rho} \phi_1 \wedge \phi_2) \Rightarrow N(s \models^{M}_{\rho} \phi_1, s \models^{M}_{\rho} \phi_2)$, *for* $M \in \{\mathtt{nec}, \mathtt{poss}\}$
6. $(s \models^{\mathtt{nec}}_{\rho} \mathtt{AX}_K\phi) \Rightarrow \bigwedge\{N(\mathrm{pr}_{\Box}s \longrightarrow^{a} s', s' \models^{\mathtt{nec}}_{\rho} \phi) \mid s \longrightarrow^{a}_{\Diamond} s',\ a \in K\}$;[17]
   $(s \models^{\mathtt{poss}}_{\rho} \mathtt{AX}_K\phi) \Rightarrow \bigwedge\{N(\mathrm{pr}_{\Diamond}s \longrightarrow^{a} s', s' \models^{\mathtt{poss}}_{\rho} \phi) \mid s \longrightarrow^{a}_{\Box} s',\ a \in K\}$
7. *For the least fixed point* $\mu Z.\phi$, *we define two functions*

$$F_{\Box} \colon (\Sigma_K \to [\mathbf{0},\mathbf{1}]) \to (\Sigma_K \to [\mathbf{0},\mathbf{1}]),\ (F_{\Box}f)\,s \stackrel{\mathrm{def}}{=} (s \models^{\mathtt{nec}}_{\rho[Z \mapsto f(s)]} \phi) \quad (11)$$

$$F_{\Diamond} \colon (\Sigma_K \to [\mathbf{0},\mathbf{1}]) \to (\Sigma_K \to [\mathbf{0},\mathbf{1}]),\ (F_{\Diamond}f)\,s \stackrel{\mathrm{def}}{=} (s \models^{\mathtt{poss}}_{\rho[Z \mapsto f(s)]} \phi), \quad (12)$$

where $\rho[Z \mapsto f(s)]$ *is the environment that behaves like* ρ, *except for* Z, *which it maps to* $f(s)$. *Both functions* $F_{\Box}$ *and* $F_{\Diamond}$ *are monotone, so they possess least and greatest fixed points over the complete lattice* $\Sigma_K \to [\mathbf{0},\mathbf{1}]$, *ordered pointwise. We define* $s \models^{\mathtt{nec}}_{\rho} \mu Z.\phi$ *as the least fixed point of* $F_{\Box}$ *and* $s \models^{\mathtt{poss}}_{\rho} \mu Z.\phi$ *as the greatest fixed point of* $F_{\Diamond}$.

Note how the meaning of $\mathtt{AX}_K\phi$ is being determined. For $\models^{\mathtt{nec}}$, we compute the worst-case scenario: a quantification over all $\Diamond/K$-transitions of $\models^{\mathtt{nec}}$ meanings of ϕ in such successor states; these meanings are weighted, using the linear t-norm N, with the necessity of these transitions to occur: $\mathrm{pr}_{\Box}s \longrightarrow^{a} s'$. The situation for $\models^{\mathtt{poss}}$ is dual. We state some relevant facts about this semantics.

Theorem 3 (Soundness of validation and refutation). *Let* $\mathcal{K}$ *be a fuzzy Kripke MTS as in (7).*

1. *Then* $(s \models^{\mathtt{nec}}_{\rho} \phi) \leq (s \models^{\mathtt{poss}}_{\rho} \phi)$ *for all states* $s \in \Sigma_K$, *all environments* ρ, *and all* $\phi \in \mathtt{ActMu}$.
2. *If* s *refines* t, *then*

$$[t \models^{\mathtt{nec}}_{\rho} \phi, t \models^{\mathtt{poss}}_{\rho} \phi] \leq [s \models^{\mathtt{nec}}_{\rho} \phi, s \models^{\mathtt{poss}}_{\rho} \phi]. \quad (13)$$

Proof. – The first claim follows from a standard inductive argument. For fixed points, one obtains that $F_{\Box} \leq F_{\Diamond}$, but then the least fixed point of the former is below the greatest fixed point of the latter.

[15] That is, $\mathrm{pr}_{\Box}[x,y] \stackrel{\mathrm{def}}{=} x$ and $\mathrm{pr}_{\Diamond}[x,y] \stackrel{\mathrm{def}}{=} y$.

[16] As standard, in computing $s \models^{M}_{\rho} \phi$, we need to assume that ρ is defined for all free variables of ϕ.

[17] One may replace $\bigwedge$ in this clause with an $n$ary application of N.

– We prove this part in full detail as it reveals the mechanics of the refinement notion. We do an inductive argument.
  1. Let ϕ be $\mathtt{T}$. Then (13) reads as $[1,1] \leq [1,1]$ which is valid.
  2. Let ϕ be Z. Then (13) holds since both intervals are determined by Z and ρ and independent of any states.
  3. Let ϕ be p. Recalling the definition of $\models^M$ for p, it is immediate that the inequality (13) is simply claiming that $L(t,p) \leq L(s,p)$. Since s refines t, this follows readily whenever $\Box L(t,p)$ or $\Diamond L(s,p)$. Otherwise, (13) reads as $[0,\cdot] \leq [0,0]$, which is always valid.
  4. Let ϕ be $\neg\psi$. The claim follows readily from induction on ψ and the fact that $[x,y] \mapsto [1-y, 1-x] \colon \mathbf{I} \to \mathbf{I}$ is monotone.
  5. Let ϕ be $\phi_1 \wedge \phi_2$. Then this follows from induction on ϕ_1 and ϕ_2, noting that $N \colon [\mathbf{0},\mathbf{1}] \times [\mathbf{0},\mathbf{1}] \to [\mathbf{0},\mathbf{1}]$ is a monotone map since it preserves all suprema in each coordinate separately.
  6. Let ϕ be $\mathtt{AX}_K\psi$ and assume that (13) holds for ψ and all pairs in $\prec_K$.
     (a) We show $(t \models^{\mathtt{nec}}_\rho \mathtt{AX}_K\psi) \leq (s \models^{\mathtt{nec}}_\rho \mathtt{AX}_K\psi)$; the latter is $\bigwedge\{N(\mathrm{pr}_\Box s \longrightarrow^a s', s' \models^{\mathtt{nec}}_\rho \psi) \mid s \longrightarrow^a_\Diamond s',\ a \in K\}$. But for any $s \longrightarrow^a_\Diamond s'$ with $a \in K$, there exists some t' such that $s' \prec_K t'$, $t \longrightarrow^a_\Diamond t'$, and (8) hold. By induction, $(t' \models^{\mathtt{nec}}_\rho \psi) \leq (s' \models^{\mathtt{nec}}_\rho \psi)$ holds. By (8), $\mathrm{pr}_\Box t \longrightarrow^a t' \leq \mathrm{pr}_\Box s \longrightarrow^a s'$. Since N is monotone, we get

$$N(\mathrm{pr}_\Box t \longrightarrow^a t', t' \models^{\mathtt{nec}}_\rho \psi) \leq N(\mathrm{pr}_\Box s \longrightarrow^a s', s' \models^{\mathtt{nec}}_\rho \psi). \tag{14}$$

     But $t \models^{\mathtt{nec}}_\rho \mathtt{AX}_K\psi$ is the greatest lower bound of all the left hand sides of (14) for which $t \longrightarrow^a_\Diamond t'$ and $a \in K$. Therefore, $(t \models^{\mathtt{nec}}_\rho \mathtt{AX}_K\psi) \leq (s \models^{\mathtt{nec}}_\rho \mathtt{AX}_K\psi)$ follows.
     (b) We show $(s \models^{\mathtt{poss}}_\rho \mathtt{AX}_K\psi) \leq (t \models^{\mathtt{poss}}_\rho \mathtt{AX}_K\psi)$ in a dual way; the latter is $\bigwedge\{N(\mathrm{pr}_\Diamond t \longrightarrow^a t', t' \models^{\mathtt{nec}}_\rho \psi) \mid t \longrightarrow^a_\Box t',\ a \in K\}$. But for any $t \longrightarrow^a_\Box t'$ with $a \in K$, there exists some s' such that $s' \prec_K t'$, $s \longrightarrow^a_\Box s'$, and (8) hold. By induction, $(s' \models^{\mathtt{poss}}_\rho \psi) \leq (t' \models^{\mathtt{poss}}_\rho \psi)$ holds. By (8), $\mathrm{pr}_\Diamond s \longrightarrow^a s' \leq \mathrm{pr}_\Diamond t \longrightarrow^a t'$. Since N is monotone, we get

$$N(\mathrm{pr}_\Diamond s \longrightarrow^a s', s' \models^{\mathtt{nec}}_\rho \psi) \leq N(\mathrm{pr}_\Diamond t \longrightarrow^a t', t' \models^{\mathtt{nec}}_\rho \psi). \tag{15}$$

     But $s \models^{\mathtt{poss}}_\rho \mathtt{AX}_K\psi$ is the greatest lower bound of all the right hand sides in (15) for which $s \longrightarrow^a_\Box s'$ and $a \in K$. Therefore, $(s \models^{\mathtt{poss}}_\rho \mathtt{AX}_K\psi) \leq (t \models^{\mathtt{poss}}_\rho \mathtt{AX}_K\psi)$ follows.
  7. Let ϕ be $\mu Z.\psi$. By induction, we infer that (13) holds for ψ and all updated environments of the form $\rho[Z \mapsto f(s)]$. It is instructive to think of the fixed-point computations for $F_\Diamond$ and $F_\Box$ as a *common, least fixed-point* computation on $\mathbf{I}$ for $F_\Box \times F_\Diamond$. The first iteration yields $[0,1]$ for s and t, so (13) is secured. By induction, we see that each further iteration preserves this inequality. But in $\mathbf{I}$, directed suprema are computed as

$$\bigvee_{i \in I} [x_i, y_i] = [\bigvee_{i \in I} x_i, \bigwedge_{i \in I} y_i], \tag{16}$$

which ensures the validity of (13) in the limit.

Notice how (13) encodes two inequalities, as in (6): the first one says that the refinement cannot have less guarantees, the second one expresses that the abstraction cannot have less possibilities. This matches the corresponding fact about Kripke MTSs in Theorem 1. The proof above, although given for the semantics and refinement of fuzzy MTSs, is *generic in nature* and one could formulate a meta-result, stating Theorem 3 more abstractly for predomains of view as discussed in Section 5. For brevity, we omit this.

Example 5. We illustrate these results by computing the meanings of the ActCTL formula $\mathtt{AX}(\mathtt{EX}_{\mathtt{falseAlarm}}\top)$ at state activePhase1. In ActMu, this formula is written as $\mathtt{AX}(\neg\mathtt{AX}_{\mathtt{falseAlarm}}\neg\top)$. We base our example on the linear t-norm $N(a,b) = a \cdot b$. We compute

$$\begin{aligned}(\mathtt{activePhase1} \models^{\mathtt{nec}} \mathtt{AX}(\neg\mathtt{AX}_{\{\mathtt{falseAlarm}\}}\neg\top)) &= \\ \bigwedge\{0 \cdot (\mathtt{SystemFailure} \models^{\mathtt{nec}} \neg\mathtt{AX}_{\{\mathtt{falseAlarm}\}}\neg\top), (1-10^{-9}) \cdot (\mathtt{activePhase2} \models^{\mathtt{nec}} \neg\mathtt{AX}_{\{\mathtt{falseAlarm}\}}\neg\top)\} &= \\ 0 \quad .\end{aligned}$$

Similarly, we compute

$$\begin{aligned}(\mathtt{activePhase1} \models^{\mathtt{poss}} \mathtt{AX}(\neg\mathtt{AX}_{\{\mathtt{falseAlarm}\}}\neg\top)) &= \\ \bigwedge\{1 \cdot (\mathtt{activePhase2} \models^{\mathtt{poss}} \neg\mathtt{AX}_{\{\mathtt{falseAlarm}\}}\neg\top)\} &= \\ 1 - (\mathtt{activePhase2} \models^{\mathtt{nec}} \mathtt{AX}_{\{\mathtt{falseAlarm}\}}\neg\top)\} &= \\ 1 - \bigwedge\{.0001 \cdot (\mathtt{inactive} \models^{\mathtt{nec}} \neg\top)\} &= \\ 1 - .0001 \cdot (1 - (\mathtt{inactive} \models^{\mathtt{poss}} \top)) &= \\ 1 - .0001 \cdot (1-1) &= \\ 1 \quad .\end{aligned}$$

These results confirm our intuition. There is no guarantee whatsoever for this property to hold, but there are also no limits on its possibility. Of course, in examples that include atomic propositions and their fuzzy labelings, such results can turn out to be any number in $[\mathbf{0}, \mathbf{1}]$.

The semantics of Definition 9 is truly fuzzy as it does not enjoy the properties of Theorem 2. In general, $0 < (s \models^{\mathtt{nec}} \phi \wedge \neg\phi)$ and $(s \models^{\mathtt{poss}} \phi \vee \neg\phi) < 1$.

4 Modal Markov chains

4.1 Modal probability measures

In Section 3, we studied fuzzy Kripke MTSs by means of the domain of view based on $([\mathbf{0}, \mathbf{1}], (0, 1])$ — in the sense of Proposition 1 — and we gave such

models a fuzzy semantics of `ActMu`, the modal mu-calculus. In this Section, we take a *probabilistic* point of view. For that, we retain the domain of view for fuzzy MTSs, define modal Markov chains — a special class of fuzzy MTSs — , don't change the notions of abstraction and refinement as defined for fuzzy MTSs, but sketch a probabilistic semantics of properties.[18]

Regarding the choice of intervals on their transitions, the refinements of fuzzy Kripke MTSs are constrained only by the ordering on $\mathbf{I}$. E.g., both fuzzy 911 calling centers from Figures 7 and 8 are refinements of the specification in Figure 5. This freedom is being restricted, though, if we intend concrete models to be Markov chains [21]. The center of Figure 7 is an example of a Markov chain. However, the second implementation in Figure 8 is not. The sum of all "probabilities" on outgoing transitions from state `SystemFailure` does not add up to 1. The original specification, though, reflects a certain connection and consistency between the given lower and upper bounds of actual transition probabilities. This connection can be formalized.

Definition 10. *Let $\Sigma(X)$ be a sigma-algebra [17], a non-empty set of subsets of a set X that is closed under the formation of complements and countable unions.*

1. *A* $\Box$-probability measure *is a subprobability measure, a function* $\mu\colon \Sigma(X) \to [\mathbf{0},\mathbf{1}]$ *such that*

$$\mu(\emptyset) = 0 \tag{17}$$

$$\mu(A \cup B) = \mu(A) + \mu(B) \qquad (A \text{ and } B \text{ disjoint}) \tag{18}$$

$$\mu(X) \leq 1. \tag{19}$$

2. *Given a* $\Box$*-probability measure* $\mu\colon \Sigma(X) \to [\mathbf{0},\mathbf{1}]$, *where X is finite and* $\Sigma(X) = \mathcal{P}_{\mathrm{fin}}(X)$, *we define the corresponding* $\Diamond$-probability measure *by*

$$\mu_\Diamond\colon \Sigma(X) \to [0,\infty) \tag{20}$$

$$\mu_\Diamond(\{x\}) \stackrel{\mathrm{def}}{=} 1 - \sum_{y \in X \setminus \{x\}} \mu(\{y\}). \tag{21}$$

3. *We write* $\mathcal{S}_{\Sigma(X)}$ *for the domain of all* $\Box$*-probability measures, ordered by*

$$\mu \leq \mu' \text{ iff for all } x \in X,\ \mu(\{x\}) \leq \mu'(\{x\}). \tag{22}$$

Proposition 2. *1. The structure* $\mathcal{S}_{\Sigma(X)}$ *is a domain with bottom. Its maximal elements are the probability measures over* $\Sigma(X)$.

2. If $\Sigma(X) = \mathcal{P}_{\mathrm{fin}}(X)$ *and X is finite, then* $\mu_\Diamond$ *in (21) satisfies*

$$\mu_\Diamond(\emptyset) = 0 \tag{23}$$

$$\mu_\Diamond(A \cup B) = \mu_\Diamond(A) + \mu_\Diamond(B) \qquad (A \text{ and } B \text{ disjoint}) \tag{24}$$

$$\mu_\Diamond(X) \geq 1. \tag{25}$$

[18] This Section can be reworked for Kripke models, but we chose to present a more compact notion of model.

Moreover, $\mu \leq \mu'$ in $\mathcal{S}_{\Sigma(X)}$ is equivalent to $\mu'_\diamond \leq \mu_\diamond$ in the pointwise ordering of $[0, \infty)$.

3. *A $\Box$-probability measure μ is a probability measure iff $\mu = \mu_\diamond$.*

Proof. 1. If $\mu(X) = 1$, then $\mu \leq \mu'$ in $\mathcal{S}_{\Sigma(X)}$ implies $\mu = \mu'$, for $\mu(\{x\}) < \mu'(\{x\}$ would result in an inconsistency as $\mu(X) = 1$ and $\mu \leq \mu'$ in $\mathcal{S}_{\Sigma(X)}$. Thus all probability measures are maximal elements. Conversely, if $\mu(X) < 1$, then we can distribute $1 - \mu(X)$ among the values $\mu(\{x\})$ $(x \in X)$, resulting in some $\mu' \in \mathcal{S}_{\Sigma(X)}$ with $\mu < \mu'$. Thus only probability measures are maximal elements. The bottom is the function $A \mapsto 0 \colon \Sigma(X) \to [\mathbf{0}, \mathbf{1}]$. Directed suprema are computed componentwise in (22), preserving the continuous conditions in (17)–(19).

2. The definition of $\mu_\diamond$ was assuming and enforcing (23) and (24). As for $\mu_\diamond(X) \geq 1$, this follows from (21) by summing both sides over all $x \in X$:

$$\mu_\diamond(X) = \sum_{x \in X} \left(1 - \sum_{y \in X,\, y \neq x} \mu(\{y\}) \right). \tag{26}$$

   But then (25) follows from $\mu(X) \leq 1$ and (26).

3. If $\mu = \mu_\diamond$, then μ is a probability measure, based on (19) and (25). Conversely, if μ is a probability measure, then (21) is ensuring $\mu_\diamond(\{x\}) = \mu(\{x\})$, for probability measures satisfy $\mu(A) = 1 - \mu(X \setminus A)$ for all $A \in \Sigma(X)$ [17].

4.2 Modal Markov chains

Definition 11 (Modal Markov chains). *A* modal Markov chain *is a tuple,*

$$\mathcal{K} = \langle \Sigma_K, \mathtt{Act}, \longrightarrow \rangle, \tag{27}$$

where Σ_K is a set of states, $\mathtt{Act}$ is a set of actions, and

- $\longrightarrow : \Sigma_K \times \mathtt{Act} \times \Sigma_K \to \mathbf{I}$ *is a function such that for all $s \in \Sigma_K$, the sets*

$$X_s \stackrel{\text{def}}{=} \{s' \in \Sigma_K \mid a \in \mathtt{Act},\ s \longrightarrow^a_\diamond s'\} \tag{28}$$

  are finite;
- *and $\mu \in \mathcal{S}_{\mathcal{P}_{\text{fin}}(X_s)}$, defined by $\mu(\{s'\}) \stackrel{\text{def}}{=} \sum_{a \in \mathtt{Act}} \text{pr}_\Box s \longrightarrow^a s'$, is a subprobability measure such that $\mu_\diamond(\{s'\}) = \sum_{a \in \mathtt{Act}} \text{pr}_\diamond s \longrightarrow^a s'$, for all $s' \in X_s$.*[19]

Remark 5. A modal Markov chain is a Markov chain iff for all its states s, their $\Box$-probability measure on successor states is a probability measure, i.e. a maximal elements in $\mathcal{S}_{\mathcal{P}_{\text{fin}}(X_s)}$.

Example 6. The calling centers of Figures 5 and 7 are modal Markov chains. The calling center of Figure 8, however, isn't: at state `SystemFailure` and for action `recover`, we compute $\mu_\diamond(\{\mathtt{inactive}\}) = 1 - \sum_\emptyset \cdot = 1$, which is different from .5.

[19] As done for LTSs [30], we assume that $s \longrightarrow^a_\diamond s'$ and $s \longrightarrow^a_\diamond s''$ imply $s' = s''$.

We may retain the refinement notion for fuzzy MTSs in analyzing modal Markov chains. This is not to say that we preclude the use of a rich literature on probabilistic refinement (see, e.g., [27, 22, 3, 4]). Yet, a designer of a qualitative system, in changing her view to a probabilistic analysis, may be hesitant to consider complicated probabilistic branching structures that are not present in the original design.

We may rework the fuzzy semantics of Definition 9 for modal Markov chains. As usual, the path quantifiers A and E make little sense for probabilistic meanings of branching structures, so this view identities them. We only sketch the essence of such a semantics for the inductive case of $\mathtt{AX}_{\{a\}}\phi$:

$$(s \models^{\mathtt{nec}} \mathtt{AX}_{\{a\}}\phi) \Rightarrow \sum_{s' \in X_s} (\mathrm{pr}_{\Box} s \longrightarrow^{a} s') \cdot (s' \models^{\mathtt{nec}} \phi) \tag{29}$$

$$(s \models^{\mathtt{poss}} \mathtt{AX}_{\{a\}}\phi) \Rightarrow 1 \wedge \sum_{s' \in X_s} (\mathrm{pr}_{\Diamond} s \longrightarrow^{a} s') \cdot (s' \models^{\mathtt{poss}} \phi). \tag{30}$$

One can now reformulate and reprove Theorem 3 for modal Markov chains in a suitable fragment of ActMu — typically linear time temporal logic, LTL [34, 35]. For concrete modal Markov chains, this recovers established probabilistic semantics [44].

5 Modal relations

The semantics of ActCTL and ActMu for LTSs can be written out symbolically, using a relational calculus [6, 7]. In that way, systems and their intended behavioral properties are represented within one formal syntax and property verification can be achieved by analyzing, or evaluating, such syntactic expressions; whence the name "symbolic model checking" [6, 7]. For LTSs, we may write $\mathtt{EX}_{\{a\}}\phi$ symbolically as the set of states that satisfy this formula in the usual semantics [5]:

$$\{s \in \Sigma_K \mid \exists s' \in \Sigma_K \, (\mathsf{R}(s, a, s') \wedge \overline{\phi})\}, \tag{31}$$

where R and $\overline{\phi}$ are symbolic encodings of a concrete transition relation, $\longrightarrow$, and the set of states that satisfy ϕ, respectively [6, 7]. In this Section, we show that a certain class of domains of view determines categories of modal relations, opening up the possiblity of symbolic encodings of modal systems. Two important instances will be the views associated with MTSs and fuzzy MTSs, respectively. These categories are domain-enriched [13]: their hom-sets are domains; moreover, the maximal sets of the hom-sets model concrete modal relations and form a category with its structure given by the ambient category. For MTSs, such maximal elements recover the ordinary category of binary relations. For fuzzy MTSs, we regain fuzzy relations [45].

Definition 12. *A* predomain of view *is a tuple (T, N), where T is a complete, linearly ordered,*[20] *lattice and N a linear t-norm over T. We write 0 and 1 for the least and greatest element of T, respectively.*

Proposition 3. *There is only one predomain of view for the complete, linearly ordered, lattice* $\mathbf{2}$.

Proof. The truth table semantics of conjunction, $\wedge\colon \mathbf{2} \to \mathbf{2}$ is a linear t-norm over $\mathbf{2}$. If N is any linear t-norm over $\mathbf{2}$, then $N(a,b) = a \wedge b$ whenever $a = \mathtt{F}$ or $b = \mathtt{F}$, for N preserves all suprema, including $\bigvee \emptyset$, which is $\mathtt{F}$. Otherwise, $a = \mathtt{T}$ and $b = \mathtt{T}$ imply $a \wedge b = \mathtt{T}$; but then the assumption $N(a,b) \neq \mathtt{T}$, i.e. $N(a,b) = \mathtt{F}$, results in $a = \mathtt{F}$ or $b = \mathtt{F}$ by Lemma 2, as $\mathtt{F}$ is the bottom of $\mathbf{2}$ and N a linear t-norm.

Lemma 3. *Given a linearly ordered, complete lattice T, the Plotkin power domain, $\mathsf{P}[T]$, of T is isomorphic to the set of all intervals $[x,y]$ such that $0 \leq x \leq y \leq 1$; ordered as in (6).*

Proof. Elements of $\mathsf{P}[T]$ are non-empty, Lawson-compact, order-convex sets C. Since T is linearly ordered, C is closed under finite suprema and infima. But then the infimum and supremum of C can be written as a filtered and directed one, respectively. Since C is Lawson-closed, it therefore contains $\bigwedge C$ and $\bigvee C$. But then C, being order-convex, equals $[\bigwedge C, \bigvee C]$. The ordering on $\mathsf{P}[T]$ is reverse inclusion.

Definition 13. *For a predomain of view (T, N), we consider the domain of view constructed from $(T, T \setminus \{0\})$ as prescribed in Proposition 1, thereby defining $\Box$ and $\Diamond$. Based on that, we define a structure $\mathbf{C}$, consisting of*

1. *a class of objects, ranging over all sets;*
2. *for each pair of objects X and Y, a hom-set $\mathbf{C}(X,Y)$, defined as the set of all functions from $X \times Y$ into the Plotkin power domain of T ($\mathsf{P}[T]$, represented in the form of Lemma 3);*
3. *for each set X, an identity morphism $\mathrm{id}\colon X \times X \to \mathsf{P}[T]$, mapping (x,y) to $[0,0]$ if $x \neq y$; otherwise, it returns $[1,1]$.*
4. *for each pair of morphisms $\mu \in \mathbf{C}(X,Y)$ and $\nu \in \mathbf{C}(Y,Z)$, a composition $\mu;\nu$, defined by*

$$\mathrm{pr}_{\Box}\mu;\nu(x,z) \stackrel{\mathrm{def}}{=} \bigvee_y \{N(\mathrm{pr}_{\Box}\mu(x,y), \mathrm{pr}_{\Box}\nu(y,z)) \mid \Box\mu(x,y), \Box\nu(y,z)\} \quad (32)$$

$$\mathrm{pr}_{\Diamond}\mu;\nu(x,z) \stackrel{\mathrm{def}}{=} \bigvee_y \{N(\mathrm{pr}_{\Diamond}\mu(x,y), \mathrm{pr}_{\Diamond}\nu(y,z)) \mid \Diamond\mu(x,y), \Diamond\nu(y,z)\}. \quad (33)$$

[20] That is, all elements of T are comparable.

Theorem 4. *For all predomains of view (T, N), the resulting structure* $\mathbf{C}$ *defined above is a category, the* category of modal (T, N)-relations*, whose hom-sets are domains. Moreover, the identity morphisms are maximal elements of their respective domain-enriched hom-sets, and the set of maximal morphisms is closed under composition, rendering a subcategory of* $\mathbf{C}$.

Proof. 1. The map $\mu;\nu\colon X \times Z \to \mathsf{P}[T]$ is well defined: since $\Box \subseteq \Diamond$, we see that (32) $\leq$ (33).

2. The morphisms id are two-sided identities with respect to composition. For sake of illustration, let $\mathrm{id} \in \mathbf{C}(Y, Y)$ and $\mu \in \mathbf{C}(X, Y)$. Then

$$\mathrm{pr}_\Box \mu; \mathrm{id}(x, y) \stackrel{\mathrm{def}}{=} \bigvee_{y'} \{N(\mathrm{pr}_\Box \mu(x, y'), \mathrm{pr}_\Box \mathrm{id}(y', y)) \mid \Box\mu(x, y'), \Box\mathrm{id}(y', y)\}. \tag{34}$$

Note that $\Box\mathrm{id}(y', y)$ is equivalent to $y' = y$. So the expression in (34) evaluates to $N(\mathrm{pr}_\Box \mu(x, y), 1)$, which equals $\mathrm{pr}_\Box \mu(x, y)$ since $(T, N, 1)$ is a (commutative) monoid. (The reasoning for $\mathrm{pr}_\Diamond \mu(x, y)$ is dual and omitted.)

3. We prove that composition is associative. Let $\mu \in \mathbf{C}(X, Y)$, $\nu \in \mathbf{C}(Y, Z)$, and $\eta \in \mathbf{C}(Z, W)$. Let a be the $\mathrm{pr}_\Box$ component of $(\mu;\nu);\eta(x, w)$. We need to show that a equals b, the $\mathrm{pr}_\Box$ component of $\mu;(\nu;\eta)(x, w)$. (The proof for the $\mathrm{pr}_\Diamond$ components is dual and omitted.) We compute

$$a = \bigvee \{N(\mathrm{pr}_\Box \mu; \nu(x, z), \mathrm{pr}_\Box \eta(z, w)) \mid \Box\mu; \nu(x, z), \Box\eta(z, w)\} \tag{35}$$

$$b = \bigvee \{N(\mathrm{pr}_\Box \mu(x, z), \mathrm{pr}_\Box \nu; \eta(z, w)) \mid \Box\mu(x, z), \Box\nu; \eta(z, w)\}. \tag{36}$$

But $\Box\mu;\nu(x, z)$ holds iff there is some y, such that $\Box\mu(x, y)$ and $\Box\nu(y, z)$; for the upper set U that determines $\Box$ equals $T \setminus \{0\}$, and $N(\alpha, \beta) = 0$ implies $\alpha = 0$ or $\beta = 0$ by Lemma 2. Since this remark applies to all compositions, we can easily convert a into b, using the fact that N is associative and preserves $\bigvee$ in each coordinate.

4. The claim about the maximal elements is immediate, for id maps into the set of maximal elements of $\mathsf{P}[T]$, and $\Box$ and $\Diamond$ agree on that set.

We emphasize that this proof only succeded since U was assumed to be $T \setminus \{0\}$, and that most properties of linear t-norms were used.

Example 7. For the predomain of view $(\mathbf{2}, \{\mathtt{T}\})$, the category $\mathbf{C}$ is the one of "modal relations"; the subcategory of conrete modal relations is the ordinary category of binary relations [25] in disguise.

For the predomain of view $([\mathbf{0}, \mathbf{1}], (0, 1])$, the category $\mathbf{C}$ is the one of "modal fuzzy relations"; the subcategory of conrete modal relations has Zadeh's fuzzy relations [45] as morphisms.

Proposition 4. *Given a predomain of view (T, N) and its corresponding category of modal (T, N)-relations, we have two endofunctors, $G_\Box, G_\Diamond\colon \mathbf{C} \to \mathbf{C}$,*

that leave objects invariant and map modal relations to their corresponding $\Box$- *and* $\Diamond$*-components:*

$$G_{\Box}(\mu) \stackrel{\text{def}}{=} (x, y) \mapsto [\mathrm{pr}_{\Box}\mu(x, y), \mathrm{pr}_{\Box}\mu(x, y)] \tag{37}$$

$$G_{\Diamond}(\mu) \stackrel{\text{def}}{=} (x, y) \mapsto [\mathrm{pr}_{\Diamond}\mu(x, y), \mathrm{pr}_{\Diamond}\mu(x, y)]. \tag{38}$$

In both cases, the image is the category of concrete modal (T, N)*-relations.*

6 Related work

The programme of this paper has already been put forward in [19]. In loc. cit., one also finds a brief discussion of a generic specification language, a process algebra whose prefix operation is annotated with elements of the domain of view. Additionally, that paper discusses transformations of models *across* domains of views and how these transformations are reflected by the resulting changes of property validation. Our paper, though, greatly improved on the latter in that we arrived at a more compact definition of domains of view, gave a generic construction for such domains, proved results on the categories of modal relations for this general construction, made the paper more readable by presenting the semantics via two notions $\models^{\mathtt{nec}}$ and $\models^{\mathtt{poss}}$ which compute the end points of the intervals of [19], and added more motivational material on the specification and analysis of systems.

Quite a few presentational choices — e.g. the use of $\models^{\mathtt{nec}}$ and $\models^{\mathtt{poss}}$ instead of a $\mathsf{P}[T]$-valued meaning function — were inspired from [37]. We recommend reading [37] as a more authorative and much broader discussion of using LTSs and MTSs in the specification and analysis of sytems. For example, that paper discusses homomophisms and Galois connections and their applications in abstraction frameworks, a topic that we chose not to include in this paper.

Many results of this paper are scattered throughout the technical literature. Our main contribution is to show that a host of seemingly unrelated existing work can be cast in a uniform framework that is founded on first principles of domain theory. Our restriction to finitely-branching structures can be lifted: Martín Escardó pointed out that our semantics remains computable if the sets of $\Box/a$-successors and $\Diamond/a$-successors of states are compact in a Hausdorff topology. Independently from that, Radha Jagadeesan has pointed out that the mixed power domain [16, 18] is an ideal denotational universe for formulating modal refinement, abstraction, validation and refutation semantics.[21] Current work [20] achieved a conceptual breakthrough by capturing a precise mathematical model of the class of (countable) MTSs via a domain equation that is similar to the one Samson Abramsky proposed for LTSs [1]; using the mixed power domain instead of the (pointed) Plotkin power domain [20].

[21] In that domain, the topological assumptions cited by Martín Escardó are met with respect to the Lawson topology [14, 15].

Acknowledgments

A number of people have made valuable suggestions. Among them were Martín Escardó, Stephen Gilmore, Jane Hillston, Achim Jung, Marta Kwiatkowska, Annabelle McIver, Carroll Morgan, Gordon Plotkin, and Jeff Sanders. I would like to thank the organizers of the ISDT'99 conference, Ying-Ming Liu, Yi-Xiang Chen, Klaus Keimel, and Guo-Qiang Zhang, for their kind invitation to present part of this material during a stimulating and very hospitable conference. David Schmidt and Radha Jagadeesan have profoundly influenced my appreciation of modal transition systems as a tool for specifying and analyzing programs and software designs. David Schmidt's slot-machine example [37] inspired the 911 calling center and its discussion. Last, but definitely not least, I mean to thank for the criticism and most useful suggestions made by the anonymous referees.

References

1. S. Abramsky. A domain equation for bisimulation. *Information and Computation*, 92:161–218, 1991.
2. 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. Clarendon Press, 1994.
3. C. Baier. Polynomial Time Algorithms for Testing Probabilistic Bisimulation and Simulation. In *Proceedings of CAV'96*, number 1102 in Lecture Notes in Computer Science, pages 38–49. Springer Verlag, 1996.
4. C. Baier and H. Hermanns. Weak bisimulation for fully probabilistic processes. In *Proc. 9th International Conference on Computer Aided Verification (CAV'97)*, volume 1254 of *Lecture Notes in Computer Science*, pages 119–130, 1997.
5. J. C. Bradfield. *Verifying Temporal Properties Of Systems.* Birkhaeuser, Boston, Mass., 1991.
6. 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.
7. J. R. Burch, E. M. Clarke, D. L. Dill K. L. McMillan, and J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–170, 1992.
8. E. M. Clarke and E. M. Emerson. Synthesis of synchronization skeletons for branching time temporal logic. In D. Kozen, editor, *Proc. Logic of Programs*, volume 131 of *LNCS*. Springer Verlag, 1981.
9. E. M. Clarke, O. Grumberg, and D. E. Long. Model Checking and Abstraction. In *19th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 343–354. ACM Press, 1992.
10. 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.
11. R. de Nicola and F. Vaandrager. Three Logics for Branching Bisimulation. *Journal of the Association of Computing Machinery*, 42(2):458–487, March 1995.
12. 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.

13. S. Eilenberg and G. M. Kelly. Closed categories. In S. Eilenberg, D. K. Harrison, S. MacLane, and H. Röhrl, editors, *Proceedings of the Conference on Categorical Algebra, La Jolla 1965*, pages 421–562. Springer Verlag, 1966.
14. J. M. G. Fell. A hausdorff topology for the closed subsets of a locally compact non-hausdorff space. *Proc. Amer. Math. Soc.*, 13:472–476, 1962.
15. G. Gierz, K. H. Hofmann, K. Keimel, J. D. Lawson, M. Mislove, and D. S. Scott. *A Compendium of Continuous Lattices*. Springer Verlag, 1980.
16. C. Gunter. The mixed power domain. *Theoretical Computer Science*, 103:311–334, 1992.
17. P. R. Halmos. *Measure Theory*. D. van Norstrand Company, 1950.
18. R. Heckmann. Power domains and second order predicates. *Theoretical Computer Science*, 111:59–88, 1993.
19. M. Huth. A Unifying Framework for Model Checking Labeled Kripke Structures, Modal Transition Systems, and Interval Transition Systems. In *19th International Conference on the Foundations of Software Technology & Theoretical Computer Science*, volume 1738 of *Lecture Notes in Computer Science*, pages 369–380. Springer Verlag, 1999.
20. M. Huth, R. Jagadeesan, and D. Schmidt. Modal transition systems: new foundations and new applications. To appear as a KSU-CIS Techreport, August 2000.
21. D. L. Isaacson and R. W. Madsen. *Markov Chains Theory and Applications*. Probability and Mathematical Statistics. John Wiley & Sons, 1976.
22. B. Jonsson and K. G. Larsen. Specification and Refinement of Probabilistic Processes. In *Proceedings of the International Symposium on Logic in Computer Science*, pages 266–277. IEEE Computer Society, IEEE Computer Society Press, July 1991.
23. P. Kelb. Model checking and abstraction: a framework preserving both truth and failure information. Technical Report Technical report, OFFIS, University of Oldenburg, Germany, 1994.
24. D. Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27:333–354, 1983.
25. S. Mac Lane. *Categories for the Working Mathematician*. Springer Verlag, 1971.
26. K. G. Larsen. Modal Specifications. In J. Sifakis, editor, *Automatic Verification Methods for Finite State Systems*, number 407 in Lecture Notes in Computer Science, pages 232–246. Springer Verlag, June 12–14, 1989 1989. International Workshop, Grenoble, France.
27. K. G. Larsen and A. Skou. Bisimulation through Probabilistic Testing. *Information and Computation*, 94(1):1–28, September 1991.
28. 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.
29. R. Milner. A modal characterisation of observable machine behaviours. In G. Astesiano and C. Böhm, editors, *CAAP '81*, volume 112 of *Lecture Notes in Computer Science*, pages 25–34. Springer Verlag, 1981.
30. R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.
31. R. E. Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, 1966.
32. D. M. Park. Concurrency on automata and infinite sequences. In P. Deussen, editor, *Conference on Theoretical Computer Science*, volume 104 of *Lecture Notes in Computer Science*. Springer Verlag, 1981.
33. G. D. Plotkin. A powerdomain construction. *SIAM Journal on Computing*, 5:452–487, 1976.

34. A. Pnueli. The temporal logic of programs. In *Proceedings of the 19th Annual Symposium on the Foundations of Computer Science*. IEEE Computer Society Press, 1977.
35. A. Pnueli. Applications of temporal logic to the specification and verification of reactive systems: a survey of current trends. In J.W. de Bakker, editor, *Current Trends in Concurrency*, volume 224 of *Lecture Notes in Computer Science*, pages 510–584. Springer-Verlag, 1985.
36. D. A. Schmidt. *Denotational Semantics*. Allyn and Bacon, 1986.
37. D.A. Schmidt. Binary relations for abstraction and refinement. *Elsevier Electronic Notes in Computer Science*, November 1999. Workshop on Refinement and Abstraction, Osaka, Japan. To appear.
38. B. Schweizer and A. Sklar. Associative functions and abstract semigroups. *Publ. Math. Debrecen*, 10:69–81, 1963.
39. D. S. Scott. Continuous lattices. In F. Lawvere, editor, *Toposes, Algebraic Geometry and Logic*, volume 274 of *Lecture Notes in Mathematics*, pages 97–136. Springer Verlag, 1972.
40. J. M. Spivey. *The Z Notation: A Reference Manual*. Prentice Hall, 1992. Second edition.
41. J. E. Stoy. *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory*. The MIT Press, 1977.
42. C. Strachey. Towards a formal semantics. In T. B. Steel, editor, *Formal Language Description Languages for Computer Programming*, pages 198–220, Amsterdam, 1966. North-Holland.
43. 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.
44. M. Vardi. Automatic Verification of Probabilistic Concurrent Finite-State Programs. In *Proc. FOCS'85*, pages 327–338. IEEE, 1985.
45. L. A. Zadeh. Fuzzy Sets. *Information and Control*, 8:338–353, 1965.