30 Years of Modal Transition Systems: Survey of Extensions and Analysis

Jan Křetínský(✉)

Technical University of Munich, Munich, Germany
jan.kretinsky@in.tum.de

Abstract. We survey the specification formalism of modal transition systems (MTS). We discuss various extensions of MTS, their relationships and modelling capabilities. The extensions include more involved modalities, quantitative aspects, or infinite state spaces. Further, we discuss problems arising in verification and analysis of these systems. We cover refinement checking, model checking and synthesis, standard logical and structural operations as used in specification theories as well as the respective tool support.

1 Introduction

Correctness of complex systems can be ensured in various ways. The key idea of verification is to first *specify* a property the system under development should satisfy and then to *verify* that this is indeed the case. An alternative to verification is *refinement* of the original specification into an implementation, which is guaranteed to satisfy the specification, for the refinement is designed to preserve the properties of interest. The refinement can be done either in one step, where the implementation is *synthesized* from the specification, or in more steps in a process of *stepwise refinement*. The latter is particularly useful when some details of the requirements are not known at the beginning of the design process, or synthesis of the whole system is infeasible, or in the component-based design where other systems can be reused as parts of the new system.

The difference between verifying and refining systems is reflected in two fundamentally different approaches to specifications. Firstly, the *logical* approach, relying on model checking algorithms, makes use of specifications given as formulae of temporal or modal logics. Secondly, the *behavioural* approach, relying on refinement, requires specifications to be given in the same formalism as implementations, e.g. a kind of a machine with an operational interpretation. We focus on the latter.

Example 1. Consider the scenario of developing a piece of software illustrated in Fig. 1. We start with a viewpoint V_1 on the system, e.g. the client's view on the service functionality. This gets iteratively refined into a more concrete description V_m. Further, assume there is also another viewpoint W_1, e.g. a description of the service from the server point of view, which is refined in a similar fashion

L. Aceto et al. (Eds.): Larsen Festschrift, LNCS 10460, pp. 36–74, 2017.
DOI: 10.1007/978-3-319-63121-9_3

resulting in W_n. After these viewpoints are precise enough (although still very underspecified), we merge them into one, say S, using an operation of *conjunction*. The complete description is now modelled by S, which is to be implemented. Suppose we have components C and D at our disposal, which perform subroutines needed in S. We put C and D together into a component T using an operation of *parallel composition*. What remains to be designed is a component X that we can compose with T in parallel so that the result conforms to the specification S. The most general such X is called the *quotient* of S by T. Once we have X we can further refine the underspecified behaviour in any desired way resulting in a specification Y. The final step is to automatically *synthesize* an implementation Z that, for instance, satisfies additional temporal *logic constraints* φ and/or is the *cheapest* implementation with respect to specified costs $\mathfrak{C}$. Specification theories [Lar90, BDH+12] are mathematical formalisms allowing for such development in a rigorous way. △

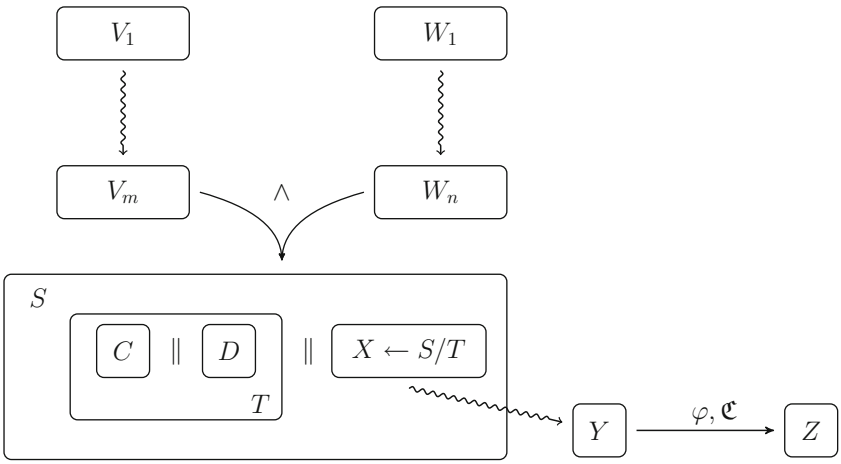

Fig. 1. An example of a component-based step-wise design scheme

A good specification theory should (i) allow for all the operations mentioned in the example and efficient algorithms to compute them. Moreover, it should (ii) be expressive enough to allow for convenient modelling. The behavioural formalism of *modal transition systems* (MTS) [LT88] provides a convenient basis for such a theory and has attracted a lot of attention. Unfortunately, it does not satisfy either of the two stipulations above completely. In this paper, we survey *extensions* of MTS that meet all these demands and efficient algorithms for their *analysis* such as the mentioned operations, refinements, verification and synthesis. Further, we discuss a link between the MTS extensions and logics, thus building a bridge between the behavioural and the logical world, allowing us to combine them, enjoying the best of both worlds.

1.1 History of Modal Transition Systems

Modal transition systems (MTS) were introduced by Larsen and Thomsen [LT88] three decades ago. The goal was to obtain an expressive specification formalism with operational interpretation, allowing for refinement. The main advantage of MTS is that they are a simple extension of labelled transition systems, which have proved appropriate for behavioural description of systems as well as their compositions.

MTS consist of a set of states and two transition relations. The *must* transitions prescribe what behaviour has to be present in every refinement of the system; the *may* transitions describe the behaviour that is allowed, but need not be realized in the refinements. This allows us to underspecify non-critical behaviour in the early stages of design, focusing on the main properties, verifying them and sorting out the details of the yet unimplemented non-critical behaviour later.

Example 2. An MTS specification of a coffee machine is displayed in Fig. 2 on the left. May transitions are depicted using dashed arrows, must transitions using solid arrows. In the left state, the machine can either start to clean or accept a coin. It may not always be possible to take the coin action, but if we do so the machine must offer coffee and possibly supplement the choice with tea. An implementation of this specification is displayed on the right. Here the clean is scheduled regularly after every two beverages. In addition, tea can always be chosen instead of coffee. △

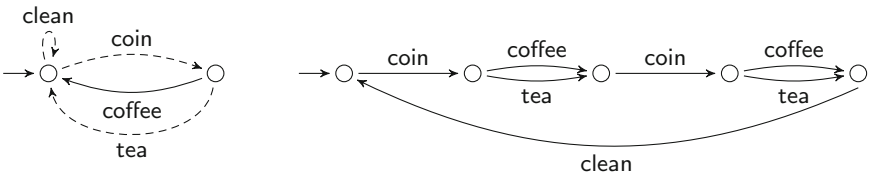

Fig. 2. An MTS specification and its implementation

The formalism of MTS has proven to be useful, most importantly in compositional reasoning and component-based design. Industrial applications are as old as [Bru97] where MTS have found use for an air-traffic system at Heathrow airport. Besides, MTS are advocated as an appropriate base for interface theories in [AHL+08a, RBB+09b, RBB+09a, RBB+11] and for product line theories in [LNW07a, Nym08, tBDG+15, tBFGM16, DSMB16]. Further, MTS-based software engineering methodology for design via merging partial descriptions of behaviour has been established in [UC04, BCU06, UBC07] and using residuation in [Rac07, Rac08, Ben08]. The theory found its applications also in testing [BDH+15, LML15]. MTS are used for program analysis using abstraction

[GHJ01, DGG97, Nam03, DN04, dAGJ04, NNN08, CGLT09, GNRT10]. MTS specification formalisms are supported by several tools [DFFU07, BML11, KS13a, VR14].

Over the years, many extensions of MTS have been proposed. While MTS can only specify whether or not a particular transition is required, some extensions equip MTS with more general abilities to describe what *combinations* of transitions are possible. For instance, disjunctive MTS (DMTS) [LX90] can specify that at least one of a given set of transitions is present. One-selecting MTS [FS08] specify that exactly one of them is present. Acceptance automata [Rac07] can even express any Boolean combination of transitions, but only for deterministic systems. In all the mentioned cases, every must transition is also automatically a may transition, modelling that whatever is required is also allowed. Sometimes this assumption is dropped and the two transition relations are given independently, giving rise to mixed transition systems (MixTS) [DGG97, AHL+08b].

These formalisms have also been studied under other names in different contexts. To some extent equivalent variations of MTS have been adapted for model-checking: Kripke modal transition systems (KMTS) [HJS01], partial Kripke structures [BG00], and 3-valued Kripke structures [CDEG03]. In the same manner MixTS correspond to Belnap transition systems [GWC06a]. Further, DMTS correspond to generalized KMTS [SG04] or abstract transition systems [dAGJ04]. While the variants of MTS and MixTS have been used in practical symbolic model-checkers (e.g. [CDEG03, GC06, GWC06b]), the "hypermust" transitions in DMTS are hard to encode efficiently into BDDs. A comparison of usability of these systems for symbolic model checking can be found in [WGC09]. Acceptance automata were also studied as acceptance trees [Hen85].

1.2 Outline of the Paper

Section 2 introduces modal transition systems formally, recalls several logics used later, and explains the stipulations on good specification formalisms. Section 3 discusses extensions of MTS with respect to specifying the combinations of present transitions. Section 4 discusses extensions of MTS with respect to the underlying graph structure of the MTS, focusing on weighted graphs and infinite graphs. In Sect. 5, results on refinements, operations, implementation synthesis, and the available tools are surveyed. Section 7 concludes and mentions several possible future directions.

1.3 Further Sources

This survey is an updated adaptation of the author's thesis [Kře14]. An excellent overview, however older, is provided in [AHL+08a]. Particular topics are explained in depth in several theses, e.g. applications to interface and product-line theories [Nym08], or extensions of MTS such as acceptance automata [Rac07], disjunctive MTS [Ben12], MTS under different semantics [Fis12], in quantitative settings [Juh13], with data [Bau12], or parameters and synchronization [Møl13].

2 Preliminaries

2.1 Modal Transition Systems

The original modal transition systems were introduced in [LT88] as follows, where Σ is an *action* alphabet.

Definition 1 (Modal transition system). *A* modal transition system (MTS) *is a triple* $(P, \dashrightarrow, \longrightarrow)$*, where* P *is a set of* processes *and* $\longrightarrow \subseteq \dashrightarrow \subseteq P \times \Sigma \times P$ *are* must *and* may *transition relations, respectively.*

The must and may transitions capture the required and allowed behaviour, as discussed in the introduction. The most fundamental notion of the theory of modal transition systems is the modal refinement. Intuitively, a process s refines a process t if s concretizes t (or in other words, t is more abstract than s). Since processes are meant to serve as specifications, this is defined by (i) only allowing in s what is already allowed in t and (ii) requiring in s what is already required in t.

Definition 2 (Modal refinement). *Let* $(P_1, \dashrightarrow_1, \longrightarrow_1)$, $(P_2, \dashrightarrow_2, \longrightarrow_2)$ *be MTS and* $s \in P_1, t \in P_2$ *be processes. We say that* s modally refines t*, written* $s \leq_m t$*, if there is a* refinement relation $R \subseteq P_1 \times P_2$ *satisfying* $(s,t) \in R$ *and for every* $(p,q) \in R$ *and every* $a \in \Sigma$*:*

1. *if* $p \overset{a}{\dashrightarrow}_1 p'$ *then there is a transition* $q \overset{a}{\dashrightarrow}_2 q'$ *such that* $(p',q') \in R$*, and*
2. *if* $q \overset{a}{\longrightarrow}_2 q'$ *then there is a transition* $p \overset{a}{\longrightarrow}_1 p'$ *such that* $(p',q') \in R$*.*

Example 3. In the course of the refinement process, must transitions are preserved, may transitions can turn into must transitions or disappear, and no new transitions are added. Note that refinement is a more complex notion than that of subgraph. Indeed, the same transition can be refined in different ways in different places as illustrated in Fig. 3. △

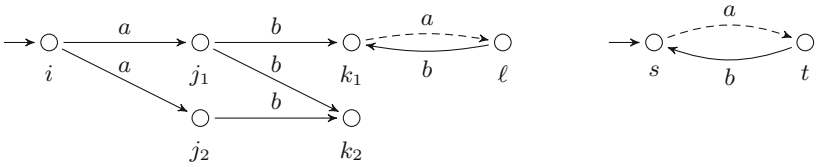

Fig. 3. The refinement $i \leq_m s$ is witnessed by the refinement relation $\{(i,s),(j_1,t),(j_2,t),(k_1,s),(k_2,s),(\ell,t)\}$. Note that whenever there is a must transition in an MTS, we do not depict its underlying may transitions. Moreover, when a designated process of an MTS is considered initial, it is depicted with an incoming arrow.

Whenever $s \leq_m t$, we call s a refinement of t and t an abstraction of s. We often consider MTS with a designated initial process; in such a case we say that an MTS refines another one if this is true of their initial processes.

One may refine MTS in a stepwise manner until $\dashrightarrow = \longrightarrow$ is obtained and no further refinement is possible. MTS with $\dashrightarrow = \longrightarrow$ are called *implementations* and can be considered as the standard labelled transition systems (LTS). Given a process s we denote by $[\![s]\!] = \{i \mid i \text{ is an implementation and } i \leq_m s\}$ the set of all implementations of s.[1] In the previous example, j_1 is not an implementation, while j_2 is considered an implementation since all reachable transitions satisfy the requirement. Further notice that $k_2 \in [\![s]\!]$.

Note that on implementations the refinement coincides with the strong bisimilarity, and on modal transition systems without any must transitions it corresponds to the simulation preorder. Further, the refinement has a respective game characterization [BKLS09b] similar to (bi)simulation games.

2.2 Logics

A set of implementations can be specified not only by a behavioural specification such as an MTS, but also by a formula of a logic. Here we briefly recall two logics: μ-calculus [Koz83] and LTL [Pnu77]. Let Ap be a set of atomic propositions.

μ-calculus is given by the syntax

$$\varphi ::= \mathbf{tt} \mid \mathbf{ff} \mid p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid [a]\varphi \mid \langle a \rangle \varphi \mid \mu X.\varphi \mid \nu X.\varphi$$

where p ranges over Ap, a over Σ, and X over a set Var of variables. We call μ the least fixpoint and ν the greatest fixpoint.

Linear temporal logic (LTL) is given by the syntax

$$\varphi ::= \mathbf{tt} \mid \mathbf{ff} \mid p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathbf{X}\varphi \mid \mathbf{X}_a\varphi \mid \mathbf{F}\varphi \mid \mathbf{G}\varphi \mid \varphi \mathbf{U} \varphi$$

where p ranges over Ap and a over Σ. Given an implementation $(P, \longrightarrow)$ and a valuation $\nu : P \to 2^{Ap}$ over its state space, any run (maximal path in the directed graph of the LTS) induces a sequence from $(2^{Ap} \times \Sigma)^{\mathbb{N}}$ capturing the labelling of the visited processes and the actions taken there. The semantics here is a mixture of state-based and action-based properties:[2] given a sequence $\alpha_0 a_0 w$ we define $\alpha_0 a_0 w \models \mathbf{X}\varphi$ iff $w \models \varphi$; besides, $\alpha_0 a_0 w \models \mathbf{X}_a\varphi$ iff $w \models \varphi$ and $a = a_0$. The semantics of other operators is standard. An LTS satisfies φ if all runs from its initial process satisfy φ.

Example 4. Consider the LTS and its valuation depicted in Fig. 4. While it satisfies $\mathbf{G}p$ and $\nu X.p \wedge [a]X$, it does satisfy neither $\mathbf{F}q$ nor $\mu X.q \vee [a]X$ due to the run looping in s. △

[1] The notation introduced in [BKLS09b] is adopted from semantics.

[2] In the context of MTS, [Ben12] elaborates on the differences of the two.

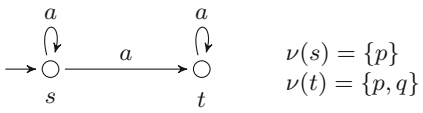

Fig. 4. An LTS with a valuation ν

2.3 Specification Theories

In order to support component based development, many specification theories have been designed. One usually requires existence and effective computability of several operations subject to various axioms. In the following, let s and t be processes, arguments of the operations.

Some operations are structural, stemming from the nature of behavioural descriptions, such as the operations of parallel composition and quotient. The *parallel composition* $\parallel$ should satisfy

(parallel) for any processes x and $y, x \parallel y \leq_m s \parallel t$ if $x \leq_m s$ and $y \leq_m t$,

called *independent implementability*. The *quotient* is an adjoint to parallel composition, hence the quotient s/t of s by t should satisfy

(quotient) for any process $x, x \leq_m s/t$ if and only if $t \parallel x \leq_m s$.

Given a specification s of the whole system and t of its component, the quotient s/t is thus a compact description of all systems that can be put in parallel with t to get a system complying with s.

Other operations are inherited from the logical view, such as Boolean operations. A *conjunction* of two systems is the most general refinement of the two systems. As the greatest lower bound with respect to $\leq_m$ it must satisfy

(conjunction) for any process $x, x \leq_m s \wedge t$ if and only if $x \leq_m s$ and $x \leq_m t$.

A bit weaker notion is that of *consistency* relation: a set of systems is consistent if they have a common implementation, i.e. if their conjunction has a non-empty set of implementations. Dually, one can define *disjunction* by requiring

(disjunction) for any process $x, s \vee t \leq_m x$ if and only if $s \leq_m x$ and $t \leq_m x$.

The remaining Boolean operation is that of *complement*:

(complement) for any process $x, x \leq_m \bar{s}$ if and only if $x \not\leq_m s$.

For the related notion of difference, see e.g. [SCU11].

It is often not possible to satisfy all axioms in this strong form. For instance, automata-based specification formalisms are sometimes too weak to express the complement, which is the case also for MTS. Besides, the "complete specification theories" of [BDH+12] only require **(parallel)** in the above-mentioned "if" form. The other desired direction cannot in general be achieved in MTS

[HL89,BKLS09b], see Fig. 5. Further, according to [BDH+12], existence of quotients and conjunctions is required if they have non-empty set of implementations. Here we presented a simpler version of the operator requirements, which is equivalent when MTS are enriched with the "inconsistent" specification with no implementations.

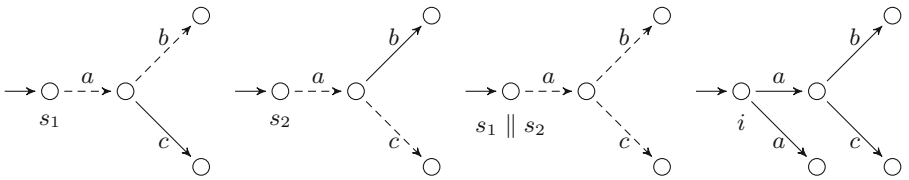

Fig. 5. $i \leq_m s_1 \parallel s_2$, but i cannot be written as $i_1 \parallel i_2$ for any $i_1 \leq_m s_1, i_2 \leq_m s_2$

3 Extensions of Modalities

Since the modelling capabilities of basic MTS are quite limited, many extensions have appeared in the literature. In this section, we focus on extensions of the may and must transition relations. Standard MTS have two transition relations $\longrightarrow, \dashrightarrow \subseteq P \times \Sigma \times P$ satisfying $\longrightarrow \subseteq \dashrightarrow$, which is called the syntactic consistency requirement. If this requirement is not imposed we obtain *mixed transition systems* as introduced in [DGG97].

Definition 3 (Mixed transition system). *A* mixed transition system (MixTS) *over an action alphabet* Σ *is a triple* $(P, \dashrightarrow, \longrightarrow)$*, where* P *is a set of* processes *and* $\longrightarrow, \dashrightarrow \subseteq P \times \Sigma \times P$ *are* must *and* may *transition relations, respectively.*

This extension allows us not only to have inconsistent specifications, but also a certain form of enforced non-deterministic choice:

Example 5. The specification of Fig. 6 requires an a transition followed by either only b's or only c's. Indeed, the must transition under a enforces a transition, but does not automatically allow it; only the two may transitions under a are allowed. △

Nevertheless, even this feature is often insufficient to specify which *combinations* of transitions can be implemented.

Example 6. Figure 7 on the left depicts an MTS that specifies the following. A request from a client may arrive. Then we can process it directly on the server or make a query to a database where we are guaranteed an answer. In both cases we send a response.

An MTS can be refined in two ways: a may transition is either implemented (and becomes a must transition) or omitted (and disappears as a transition).

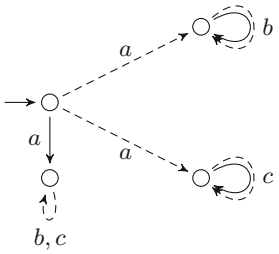

Fig. 6. A mixed transition system. Since must transitions are not necessarily also may transitions in MixTS, we depict may transitions explicitly for mixed systems here, even when there is a corresponding must transition.

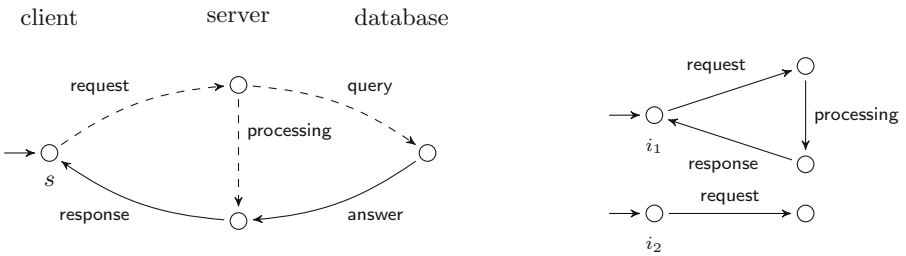

Fig. 7. A potentially deadlocking s and two of its implementations i_1, i_2

On the right of Fig. 7 there is an implementation i_1 of the system, where the processing branch is implemented and the database query branch is omitted. Similarly, there is also an implementation omitting the process branch and implementing only the query. However, there is also an undesirable implementation i_2 that does not implement any option and deadlocks as seen on the right of Fig. 7. △

To avoid deadlocking, we want to specify that *either* processing *or* query will be implemented. This is possible in *disjunctive modal transition systems* [LX90]. They were actually introduced as natural means for solutions to process equations since they can express both conjunctions and disjunctions of properties.

Definition 4 (Disjunctive modal transition system). *A* disjunctive modal transition system *(DMTS) is a triple* $(P, \dashrightarrow, \longrightarrow)$*, where* P *is a set of* processes *and* $\dashrightarrow \subseteq P \times \Sigma \times P$ *is the* may *and* $\longrightarrow \subseteq P \times 2^{\Sigma \times P}$ *the* must *(or* hyper-must*) transition relation.*

Example 7. Intuitively, in DMTS we can enforce a choice between arbitrary transitions, not just with the same action as in Example 5. Instead of forcing a particular transition, a must transition in DMTS specifies a whole set of transitions at least one of which must be present in any refinement. In our example, it would be the set consisting of processing and query transitions, see Fig. 8. △

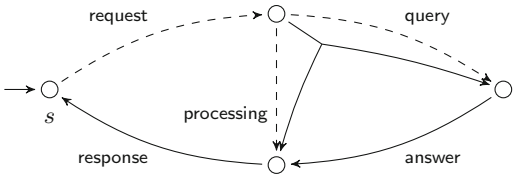

Fig. 8. A disjunctive modal transition system

Note that DMTS are capable of forcing any positive Boolean combination of transitions, simply by turning it into the conjunctive normal form. If the choice is supposed to be exclusive, we can use *one-selecting MTS (1MTS)* introduced in [FS08] with the property that *exactly* one transition from the set must be present. In 1MTS and also in *underspecified transition systems (UTS)* [FS05], both (hyper)must and (hyper)may transition relations are subsets of $P \times 2^{\Sigma \times P}$. For UTS, the syntactic consistency is required, i.e. the hyper-may is larger than the hyper-must.

Finally, explicit listing of all allowed combinations of outgoing transitions is used in *acceptance automata* [Rac08]. However, the language-theoretic definition is limited to deterministic systems.

Definition 5 (Acceptance automaton). *An* acceptance automaton *(AA) is a pair* $(P, PossibleTranSets)$*, where* P *is a prefix-closed language over* Σ *and* $PossibleTranSets : P \to 2^{2^{\Sigma}} \setminus \emptyset$ *satisfies the consistency condition:* $wa \in P$ *if and only if* $a \in TranSet \in PossibleTranSets(w)$ *for some* $TranSet$*.*

Nevertheless, as the following example shows, convenient modelling requires even more features such as conditional or persistent choices.

Example 8. Consider a simple specification of a traffic light controller for several national variants for vehicles as well as for pedestrians, displayed on the right of Fig. 9. At any moment it is in one of the four states *red*, *green*, *yellow* or *yellowRed*. The intuitive requirements are: if *green* is on then the traffic light may either change to *red* or *yellow*, and if it turned *yellow* (as for vehicles) it must go to *red* afterwards; if *red* is on then it may either turn to *green* (as for pedestrians and also for vehicles in some countries) or *yellowRed*, and if it turns *yellowRed* it must go to *green* afterwards.

However, these requirements (expressible as MTS) allow for three different undesirable implementations: (i) the light is constantly *green*, (ii) the lights switch non-deterministically, (iii) the lights switch deterministically, but *yellow* is only displayed sometimes (e.g. every other time). While the first problem can be avoided using the choice in DMTS, the latter two cannot. To eliminate the second implementation, one needs an exclusive choice, as in 1MTS; for the third implementation to be removed, one needs a persistent choice. These can be modelled in *parametric MTS* [BKL+11, BKL+15] where a parameter describes whether and when the yellow light is used, making the choices permanent in the

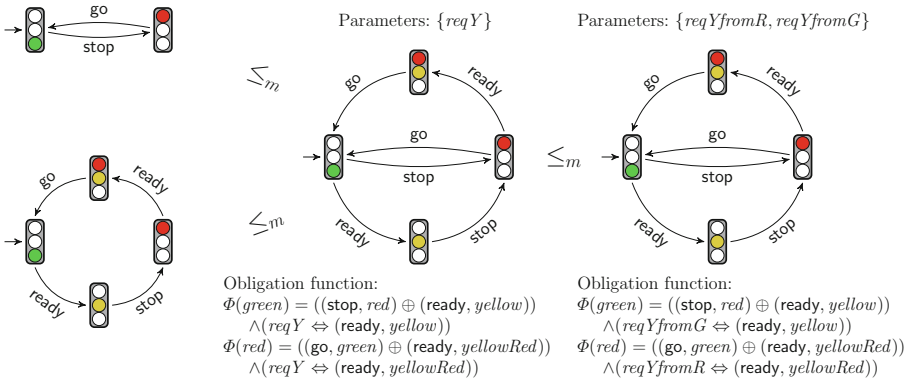

Fig. 9. Examples of PMTS and their modal refinement (Color figure online)

whole implementation. Additionally, the dependence on the parameter allows for modelling a conditional choice. Indeed, as illustrated in the middle of Fig. 9, depending on the value of another parameter, the yellow light can be consistently used or skipped in both phases. △

Definition 6 (Parametric modal transition system). *A* parametric MTS (PMTS) *is a tuple* $(P, \dashrightarrow, Par, \Phi)$ *where* P *is a set of* processes, $\dashrightarrow \subseteq P \times \Sigma \times P$ *is a* transition relation, Par *is a finite set of* parameters, *and* $\Phi : P \to \mathcal{B}ool\mathcal{E}xp((\Sigma \times P) \cup Par)$ *is an* obligation function *assigning to each process a Boolean expression over outgoing transitions and parameters.*

These systems are "mixed"; a syntactic consistency $\forall s \in P : \forall (a,t) \in \Phi(s) : s \overset{a}{\dashrightarrow} t$ may be additionally required, making them "pure". Intuitively, a set S of transitions from s is allowed if $\Phi(s)$ is true under the valuation induced by S and the fixed parameters; for an example see Fig. 9. A PMTS is

- *Boolean MTS (BMTS)* [BKL+11] if it is *parameter-free*, i.e. if $Par = \emptyset$,
- *transition system with obligation (OTS)* [BK10] if it is BMTS and only parameters can be negated,
- *DMTS* is an OTS with $\Phi(s)$ in the conjunctive normal form for all $s \in P$, DMTS is considered both mixed [LX90] and pure [BČK11],
- *MixTS* is a DMTS with $\Phi(s)$ being a conjunction of positive literals (transitions) for all $s \in P$ (and the syntactic consistency not required),
- *MTS* is a MixTS with the and the syntactic consistency required,
- *LTS* is an MTS with $\Phi(s) = \bigwedge T(s)$ for all $s \in P$, where $T(s) = \{(a,t) \mid s \overset{a}{\dashrightarrow} t\}$ is the set of all outgoing transitions of s.

The modal refinement over BMTS is an expected extension of that for MTS. Technically, let $\mathrm{Tran}(s) = \{E \subseteq T(s) \mid E \models \Phi(s)\}$ be the set of all admissible sets of transitions from s and the refinement relation satisfies for every $(p,q) \in R$:

$$\forall M \in \mathrm{Tran}(p) : \exists N \in \mathrm{Tran}(q) : \quad \forall (a, p') \in M : \exists (a, q') \in N : (p', q') \in R \quad \wedge$$
$$\forall (a, q') \in N : \exists (a, p') \in M : (p', q') \in R.$$

For PMTS, intuitively, whatever parameters of the refining system we pick, the abstract system can emulate the same behaviour (by BMTS refinement) for some choice of its parameters. The original definition [BKL+11] requires a single refinement relation for all parameter choices. Later it was superseded by a more natural definition [BKL+15] where different relations are allowed for different parameter valuations; it is closer to the semantically defined notion of thorough refinement, see Definition 10, and keeps the same complexity.

Example 9. Consider the rightmost PMTS in Fig. 9. It has two parameters *reqYfromG* and *reqYfromR* whose values can be set independently and it can be refined by the system in the middle of the figure having only one parameter *reqY*. This single parameter binds the two original parameters to the same value. The PMTS in the middle can be further refined into the implementations where either *yellow* is always used in both cases, or never at all. △

Expressive Power

Most of the formalisms have the same expressive power, as summarized in Fig. 10. However, they differ significantly in succinctness. In [KS13b], PMTS are transformed into exponentially larger BMTS and BMTS into exponentially larger DMTSm, see Fig. 10. Here Cm denotes a class C where systems are considered with more (but only finitely many) initial processes.

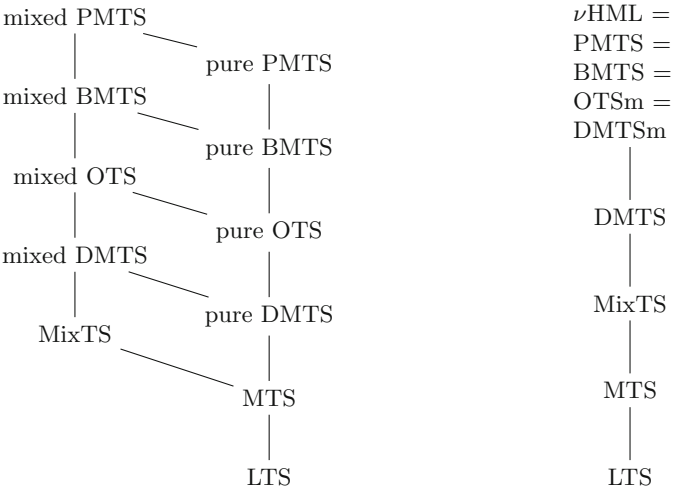

Fig. 10. The syntactic hierarchy of MTS extensions (on the left) and the semantic one, not considering empty specifications (on the right)

Except for the already discussed extra power of MixTS over MTS, mixed variants of systems can be transformed into pure, again at an exponential cost [BK10], up to the inconsistent specification, i.e. specifications with no implementations. Since this difference is not very important, we shall only deal with pure systems unless stated otherwise.

Each of the formalisms presented so far in this section was an automata-based behavioural formalism. These are often preferred as they are easier to read than, for instance, formulae of modal logics. The choice between logical and behavioural specifications is not only a question of preference. Automata-based specifications [Lar89, BG99] have a focus on compositional and incremental design in which logical specifications are somewhat lacking, with the trade-off of generally being less expressive than logics. Logical specification formalisms put a powerful logical language at the disposal of the user, and the logical approach to model checking [QS82, CE81] has seen a lot of success and tool implementations. Therefore, one would like to establish connections between behavioural and logical formalisms to exploit advantages of both at once. The relationship of MTS to logic was studied in [BL92, FP07]. It is established that MTS are equivalent to a fragment of μ-calculus where formulae are (1) consistent, (2) "prime", meaning the disjunction is allowed only in very special cases, and (3) do not contain the least fixpoint. Further, [BDF+13] proves that DMTSm (and thus BMTS and PMTS) are equivalent to ν-calculus (or Hennessy-Milner logic with greatest fixpoints, abbreviated νHML), which is a fragment of μ-calculus without the least fixpoint μ. Finally, the refinement corresponds to implication [FLT14], similarly to the refinement calculus for HML with recursion of [Hol89]. Moreover, both formalisms can be equipped with the desired operations coming from the other formalism, see Fig. 11, as further discussed in Sect. 5, bridging the gap between the two approaches.

logic		MTS
model	$\sim$	implementation
implication/entailment	$\sim$	refinement
conjunction $\wedge$	$\sim$	?
disjunction $\vee$	$\sim$	?
?	$\sim$	parallel composition $\|$
?	$\sim$	quotient /

Fig. 11. Correspondences between the logical and the behavioural world

Example 10. Consider the following property: "at all time points after executing request, no idle nor further requests but only work is allowed until grant is executed". The property can be written in e.g. CTL [CE81] as

$$\mathrm{AG}(\mathsf{request} \Rightarrow \mathrm{AX}(\mathsf{work}\ \mathrm{AW}\ \mathsf{grant}))$$

Figure 12 shows an example of an equivalent νHML formula and a DMTS corresponding to this property. △

$X = [\mathsf{grant}, \mathsf{idle}, \mathsf{work}]X \wedge [\mathsf{request}]Y$
$Y = (\langle \mathsf{work} \rangle Y \vee \langle \mathsf{grant} \rangle X) \wedge [\mathsf{idle}, \mathsf{request}]\mathbf{ff}$

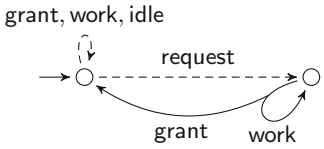

Fig. 12. Example of a νHML formula and an equivalent DMTS

Apart from the logical characterization, one can also describe processes using a process algebra and obtain the discussed subclasses mixed DMTS, pure DMTS, MixTS, MTS, LTS as syntactic subclasses [BK10].

4 Extensions of Transition Systems

The extensions discussed in the previous section focus on what *combinations* of transitions are possible. In this section, we discuss mainly extensions concerned with quantitative features or infinite memory/communication. Besides, to provide a better basis for interface theories, MTS have been also combined with I/O automata [GSSL94] and interface automata [dAH01] into *modal I/O transition systems* [LNW07a, RBB+09a, BHJ10, BMSH10, BHW10, BHB10, KM12, KDMU14] with input, output and internal actions, and its subset *modal interface automata* [LV13, LVF15, BFLV16, SH15]. Other MTS extensions feature specifically modified semantics, e.g., [BV15, BSV15, DSMB16].

4.1 Quantities

Here we discuss lifting the underlying transition systems to *quantitative settings* [LL12], with clear applications in the embedded systems design. This includes probabilistic specifications (see below) and various weighted specifications, where weights stand for various quantitative aspects (e.g. time, power or memory), which are highly relevant in the area of embedded systems. As far as the particular case of *timed* systems is concerned, the quantity of time can be refined in various ways. In the early work [CGL93, LSW95], the precise quantities are almost disregarded. More recently [JLS12, BPR09, BLPR09, DLL+10], the possible times are usually specified as time intervals, which can be narrowed down and thus made more specific. A more general option is to permit label refinement to anything smaller with respect to some *abstract ordering* of labels; [BJL+12a] provides the following conservative extension of MTS modal refinement along these lines:

Definition 7 (Modal refinement of MTS with structured labels). *Let the alphabet Σ be equipped with an ordering $\sqsubseteq$. Let $(P_1, \dashrightarrow_1, \longrightarrow_1), (P_2, \dashrightarrow_2, \longrightarrow_2)$ be MTS over Σ and $s \in P_1, t \in P_2$ be processes. We say that s* modally refines *t, written $s \leq_m t$, if there is a refinement relation $\mathcal{R} \subseteq P_1 \times P_2$ such that $(s,t) \in \mathcal{R}$ and for every $(p,q) \in R$ and every $a \in \Sigma$:*

1. *if $p \overset{a}{\dashrightarrow}_1 p'$ then $q \overset{\bar{a}}{\dashrightarrow}_2 q'$ for some $a \sqsubseteq \bar{a}$ and $(p',q') \in R$, and*
2. *if $q \overset{\bar{a}}{\longrightarrow}_2 q'$ then $p \overset{a}{\longrightarrow}_1 p'$ for some $a \sqsubseteq \bar{a}$ and $(p',q') \in R$.*

Example 11. Consider $\Sigma = L \times \mathfrak{I}$ where L is a finite set ordered by identity and $\mathfrak{I}$ is the set of intervals ordered by inclusion and Σ is ordered point-wise, standing for the action and the time required to perform it. A transition labelled by $(\ell, [a, b])$ can thus be implemented by a transition (ℓ, c) for any $c \in [a, b]$. △

This definition generalizes also previously studied MTS with more weights at once [BJL+12b]. Moreover, one can also consider MTS with timed-automata clocks [BLPR12, FL12]. In all the quantitative settings, it is also natural to extend the qualitative notion of refinement into a quantitative notion of distance of systems [BFJ+11, BFLT12].

Another previously studied instantiation is the *modal transition systems with durations (MTSD)* [BKL+12]. It models time durations of transitions as controllable or uncontrollable intervals. Controllable intervals can be further refined into narrower intervals, whereas uncontrollable are considered under the control of an unpredictable environment and cannot be further narrowed down. Additionally, the actions are assigned running cost (or rewards) per time unit.

MTS have also been lifted to the *probabilistic* setting. In the classical setting, LTS is underspecified in that the presence of a certain transition is not specified. For Markov chains, one can underspecify the probability distributions on the outgoing transitions. *Interval Markov chains* [JL91] describe them with intervals of possible values. Additionally, we can consider 3-valued valuations of atomic propositions in processes (similarly to [HJS01, BG00, CDEG03], useful for abstractions), yielding *abstract Markov chains* [FLW06]. This approach is extensible also to continuous-time Markov chains [KKLW07, KKLW12]. Besides, *constraint Markov chains* [CDL+10] use richer constraints than intervals and usual operations on them have also been studied [DLL14]. Finally, *abstract probabilistic automata* [DKL+11a] combine this with the MTS may-must modality on transitions, allowing for abstractions of Markov decision processes. They have been studied with respect to the supported operations [DKL+11b, DFLL14], state space reduction [SK14], hidden actions (stutter steps) [DLL14], and there is a support by the tool APAC [DLL+11].

Moreover, probabilistic and timed-automata extensions are combined in *abstract probabilistic timed automata* [HKKG13]. Finally, *modal continuous-time automata* [HKK13] extend MTS with continuous time constraints on stochastic waiting times, allowing for specification of systems with stochastic continuous time.

Specification theories have been lifted to the quantitative settings and equipped with the notion of distance between systems [BFLT12, FL14, FKLT14, FLT14].

4.2 Infinite State Space

In this section, we consider *infinite-state* extensions of MTS. Several extensions have been proposed, such as systems with asynchronous communication based on FIFO [BHJ10] or Petri nets [EBHH10, HHM13]. Other extensions focus on input/output extensions of MTS with data constraints [BHB10, BHW10] or explicit representation of data [BLL+14].

A systematic exploration of infinite-state MTS is also possible. A convenient unifying framework for (non-modal) infinite-state systems is provided by process rewrite systems (PRS) [May00]. A PRS Δ is a finite set of rewriting rules, which model sequential and parallel computation. Depending on the syntactic restrictions imposed on the rules, we obtain many standard models such as pushdown automata (PDA) or Petri nets (PN), see Fig. 13. A finite PRS Δ thus induces possibly infinite LTS $\mathcal{LTS}(\Delta)$.

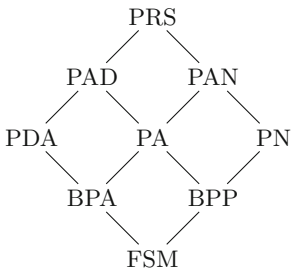

Fig. 13. PRS hierarchy

Example 12. A transition t of a Petri net with input places p, q and output places r, s can be described by the rule $p \parallel q \xrightarrow{t} r \parallel s$. A transition of a pushdown automaton in a state s with a top stack symbol X reading a letter a resulting in changing the state to r and pushing Y onto the stack can be written as $sX \xrightarrow{a} rYX$.

One can naturally lift PRS to the modal world [BK12] by having two sets of rules: may and must rules. The finite set of rules then generates a generally infinite MTS.

Definition 8 (Modal process rewrite system). *A* modal process rewrite system (mPRS) *is a tuple* $\Delta = (\Delta_{\text{may}}, \Delta_{\text{must}})$ *where* $\Delta_{\text{must}} \subseteq \Delta_{\text{may}}$ *are two PRS. The mPRS* Δ *induces an MTS* $\mathcal{MTS}(\Delta) = (\mathcal{E}, \dashrightarrow, \longrightarrow)$ *defined by* $(\mathcal{E}, \dashrightarrow) = \mathcal{LTS}(\Delta_{\text{may}})$ *and* $(\mathcal{E}, \longrightarrow) = \mathcal{LTS}(\Delta_{\text{must}})$.

Each subclass $\mathcal{C}$ of PRS has a corresponding modal extension m$\mathcal{C}$ containing all mPRS $(\Delta_{\text{may}}, \Delta_{\text{must}})$ with both Δ_{may} and Δ_{must} in $\mathcal{C}$. For instance, mFSM correspond to the standard finite MTS and mPN are modal Petri nets as introduced in [EBHH10].

Definition 9 (Modal refinement). *Given mPRS* $\Delta_1 \in m\mathcal{C}_1, \Delta_2 \in m\mathcal{C}_2$ *and process terms* δ_1, δ_2, *we say* δ_1 *refines* δ_2, *written* $\delta_1 \leq_m \delta_2$, *if* $\delta_1 \leq_m \delta_2$ *as processes of* $MTS(\Delta_1)$ *and* $MTS(\Delta_2)$, *respectively.*

What is the use of infinite MTS? Firstly, potentially infinite-state systems such as Petri nets are very popular for modelling whenever communication and/or synchronization between processes occurs. This is true even in cases where they are actually bounded and thus with a finite state space.

Example 13. Consider the following may rule (we use dashed arrows to denote may rules) generating a small Petri net.

$$\mathsf{resource} \overset{\mathsf{produce}}{\dashrightarrow} \mathsf{money} \parallel \mathsf{trash}$$

If this is the only rule with trash on the right side a safety property is guaranteed for all implementations of this system, namely that trash can only arise if there is at least one resource. On the other hand, it is not guaranteed that money can indeed be produced in such a situation. This is very useful as during the design process new requirements can arise, such as necessity of adding more participants to perform this transition. For instance,

$$\mathsf{resource} \parallel \mathsf{permit} \overset{\mathsf{produce}}{\dashrightarrow} \mathsf{money} \parallel \mathsf{trash}$$

expresses an auxiliary condition required to produce trash, namely that a permit is available. Replacing the old rule with the new one is equivalent to adding an input place permit to the modal Petri net, see Fig. 14 in yellow. In the modal transition system view, the new system *refines* the old one. Indeed, the new system is only more specific about the allowed behaviour than the old one and does not permit any previously forbidden behaviour.

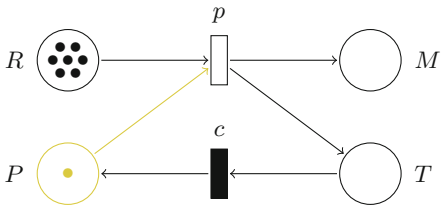

Fig. 14. A modal Petri net given by rules $\mathsf{Resource} \parallel \mathsf{Permit} \overset{\mathsf{produce}}{\dashrightarrow} \mathsf{Money} \parallel \mathsf{Trash}$ and $\mathsf{Trash} \overset{\mathsf{clean}}{\longrightarrow} \mathsf{Permit}$ with may transitions drawn as empty boxes and must transitions as full boxes (Color figure online)

One can further refine the system into the one given by

$$\mathsf{resource} \parallel \mathsf{permit} \parallel \mathsf{bribe} \overset{\mathsf{produce}}{\longrightarrow} \mathsf{money} \parallel \mathsf{trash}$$

where additional condition is imposed and now the money-producing transition has to be available (denoted by an unbroken arrow) whenever the left hand side condition is satisfied. △

Further, infinitely many states are useful to capture unbounded memory. For instance, consider a specification where the total amount of permits is not explicitly limited. In an implementation, the number of permits might need to be remembered in the state of the system.

Example 14. Consider a basic process algebra (BPA) given by rules $X \xrightarrow{(} XX$ and $X \xrightarrow{)} \varepsilon$ for correctly parenthesized expressions with $X \xrightarrow{a} X$ for all other symbols a, i.e. with no restriction on the syntax of expressions. One can easily refine this system into a PDA that accepts correct arithmetic expressions by remembering in a control state whether the last symbol read was an operand or an operator. △

5 Analysis

In this section, we survey algorithms for and complexities of the most important problems on MTS and their extensions.

5.1 Refinements

Modal refinement is a syntactically defined notion extending both bisimulation and simulation. Similarly to bisimulation having a semantic counterpart in trace equivalence, here the semantic counterpart of modal refinement is the *thorough refinement*. As opposed to the syntactic definition using local notions, the semantic definition relates (by inclusion) the sets of implementations of the specifications. The definition is universal for all extensions of MTS as it only depends on the notion of implementation and not on syntax of the particular extension.

Definition 10 (Thorough refinement). *Given processes s and t, we say that s* thoroughly refines *t, written $s \leq_t t$, if $[\![s]\!] \subseteq [\![t]\!]$.*

Note that the two refinements are in general different as we illustrate in the following example due to [BKLS09b], simplifying [HL89]:

Example 15. Consider processes s and t of Fig. 15. On the one hand, the sets of implementations of s and t are the same, namely those that can perform either no action or one a or two a's or combine the latter two options. On the other hand, s does not modally refine t. Indeed, whenever $s \leq_m t$ then either $s' \leq_m t_1$ or $s' \leq_m t_2$. However, neither is true, as s' allows a transition while t_1 does not, and s' does not require any transition while t_2 does.

Although the two refinements differ, modal refinement is a sound underapproximation of the thorough refinement. Indeed, whenever we have $s \leq_m t$ and $i \in [\![s]\!]$ we also have $i \leq_m s$ and by transitivity of the modal refinement we obtain $i \leq_m t$.

Proposition 1. *Let s, t be processes. If $s \leq_m t$ then $s \leq_t t$.*

Moreover, [BKLS09b] shows the other direction holds whenever the refined system is deterministic. A process is *deterministic* if, for each process s of its underlying MTS and for each $a \in \Sigma$, there is at most one s' such that $s \xrightarrow{a} s'$.

Fig. 15. $s \leq_t t$, but $s \not\leq_m t$

Proposition 2. *Let s, t be processes and t deterministic. If $s \leq_t t$ then $s \leq_m t$.*

In Table 1 we give an overview of the results related to deciding modal and thorough refinements for different combinations of processes on the left- and right-hand side (here D stands for deterministic processes and N for non-deterministic processes). Note that the co-inductive refinement relations are easy to compute using a fixed-point computation, although other methods are also possible, e.g. logical programming [AKRU11] or QBF solving [KS13b,BKL+15].

Table 1. MTS refinement complexity for various cases of (non)determinism

	Modal refinement $\leq_m$	Thorough refinement $\leq_t$
D≤D	NL-complete [BKLS09b]	NL-complete [BKLS09b]
N≤D	NL-complete [BKLS09b]	NL-complete [BKLS09b]
D≤N	$\in$ P [KS90,PT87]	$\in$ EXP [AHL+08b]
	P-hard [BKLS09b]	EXP-hard [BKLS12]
N≤N	$\in$ P [KS90,PT87]	$\in$ EXP [AHL+08b,BKLS09a]
	P-hard [BGS92]	EXP-hard [BKLS09a]

Since the thorough refinement is EXP-hard, it is much harder than the modal refinement. Therefore, we also investigate how the thorough refinement can be approximated by the modal refinement. While under-approximation is easy, as modal refinement implies thorough refinement, over-approximation is more difficult. Here one can use the method of the *deterministic hull* for MTS [BKLS09b]. The deterministic hull $\mathcal{D}$ is a generalization of the powerset construction on finite automata and it is the *smallest* (w.r.t. modal refinement) deterministic system refined by the original system.

Proposition 3. *Let s be an arbitrary MTS process. Then $\mathcal{D}(s)$ is a deterministic MTS process such that $s \leq_m \mathcal{D}(s)$ and, for every deterministic MTS process t, if $s \leq_t t$ then $\mathcal{D}(s) \leq_m t$.*

Corollary 1. *For any processes s, t, if $s \not\leq_m \mathcal{D}(t)$ then $s \not\leq_t t$.*

There are also other notions of refinements of systems close to MTS, such as alternating refinements [AHKV98,AFdFE+11], branching refinement [FBU09], refinement preserving the termination possibility [CR12], or refinement for product lines [DSMB16].

Extensions. As to extensions of MTS with more complex modalities, the local conditions in modal refinement are more complex. Although various extensions have the same expressive power (see Fig. 10), the transformations are exponential and thus the extensions differ in succinctness. Therefore, the respective refinement problems are harder for the more succinct extensions. All the cases depending on the type of the left-hand and right-hand sides are discussed in [BKL+15]. In most cases without parameters, the refinement can be decided in P or NP, which is feasible using SAT solvers. For systems with parameters, the complexity is significantly higher, reaching up to Π_4^{p}. Since all the complexities are included in PSPACE, QBF solvers have been applied to this problem, improving scalability to systems with hundreds of states [KS13c, BKL+15]. The QBF approach basically eliminates the complexity threat of parameters, but is quite sensitive to the level of non-determinism.

Furthermore, the decision algorithm for thorough refinement checking over MTS [BKLS12, BKLS12] has been extended to the setting of DMTS [BČK10] and of BMTS and PMTS [KS13b], see Table 2. [KS13b] also generalizes the notion of the deterministic hull.

Table 2. Complexity of the thorough refinement and the relationship to the modal refinement

	MTS	DMTS	BMTS	PMTS
$\mathrm{N}\leq_t\mathrm{N} \in$	EXP	EXP	NEXP	2-EXP
for N≤D	$\leq_m = \leq_t$	$\leq_m = \leq_t$	$\leq_m = \leq_t$	$\leq_m \neq \leq_t$

We now turn our attention to the refinement problems on other kinds of extensions of MTS. Assuming polynomial procedures for operations on structured labels, the complexity of the modal refinement stays the same as for the underlying MTS. As for infinite state systems, [BK12] shows that refinement between finite MTS and modal pushdown automaton and between modal visibly pushdown automata is decidable and EXP-complete, whereas between basic process algebras it is undecidable. When parallelism is involved, undecidability occurs very soon, already for finite MTS and basic parallel process. However, it is decidable for Petri nets when a weak form of determinism is imposed [EHH12, HHM13]. Finally, in the spirit of [AHKV98], a symmetric version of refinement resulting into a bisimulation notion over MTS is considered and shown decidable between a finite MTS and any modal process rewrite system, using the results of [KŘS05]. This allows us to check whether we can replace an infinite MTS with a particular finite one, which in turn may allow for checking further refinements.

5.2 Operations

Specification theories require the specification formalism to be closed under certain operations, as described in Sect. 2. However, not all classes of modal systems

support all the operations. As an automata-based formalism, MTS automatically allow to compose systems structurally, whereas logical operations are either difficult to compute or cannot be expressed in the formalism at all. Therefore, most of the focus has been directed to the simple deterministic case, where some operations can be defined using local syntactic rules, even for the quantitative extensions [BFJ+13].

$\parallel$ The *parallel composition* can often be lifted to the modal setting simply by applying the same SOS rules, e.g. for synchronous message passing, to both may and must transition functions. This holds for a wide class of operators as described in [BKLS09b] for MTS. Parallel composition can be extended to DMTS and other classes [BČK10, BDF+13]. Unfortunately, they inherit the incompleteness with respect to modal refinement from MTS, see [HL89, BKLS09b]. Therefore, the axiom (parallel) is only satisfied in one direction (the independent implementability), but not every implementation of the composition can actually be decomposed into a pair of implementations, see Fig. 5, and in general we have $[\![s]\!] \parallel [\![t]\!] \subset [\![s \parallel t]\!]$.

/ The *quotient* for deterministic MTS can be defined syntactically, using a few SOS rules [Rac07, Rac08]. For non-deterministic MTS, the problem is considerably more complex and the question was open for a long time. A construction for BMTS and DMTSm and an exponentially smaller one for MTS was given in [BDF+13]. Further related questions such as decomposition of a system into several components put in parallel [SUBK12] or quotient under reachability constraints [VR15] have also been investigated, but again only for deterministic systems.

$\wedge$ The situation is similar with *conjunction*. For deterministic MTS, we can again define it syntactically. For non-deterministic systems, there were several attempts. Unfortunately, the resulting MTS is not minimal (with respect to modal refinement) [UC04], or not finite even when claimed to be finite [FU08]: the "clone" operation may not terminate even in cases when it is supposed to, for example, for processes s_1, s_2 of Fig. 16 where the self-loops are redirected back to the initial processes. Actually, MTS are not closed under conjunction, see Fig. 16. However, a conjunction of two MTS has a unique greatest DMTS solution.

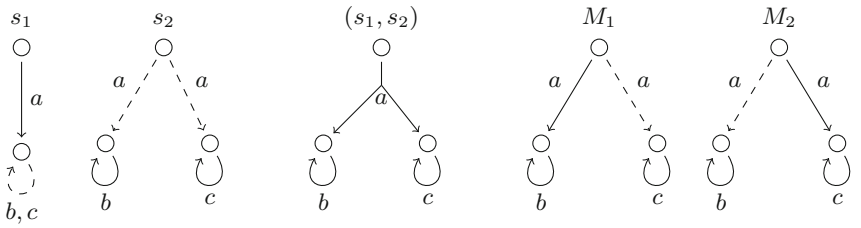

Fig. 16. MTS processes s_1, s_2, their greatest lower bound (s_1, s_2), and their two maximal MTS lower bounds M_1, M_2

Moreover, DMTS with one or more initial processes, and thus also BMTS and PMTS are closed under conjunction [BČK11]. The result of the construction is based on the synchronous product. Thus it is a system over tuples of processes where the length of the tuple is the number of input systems. This means that the conjunction (and thus also a common implementation) can be constructed in polynomial time, if n is fixed; and in exponential time, if n is a part of the input. Further, if deterministic MTS are input, the algorithm produces a deterministic MTS. Moreover, the conjunction is also the greatest lower bound with respect to the thorough refinement: $[\![s_1 \wedge s_2]\!] = [\![s_1]\!] \cap [\![s_2]\!]$ which is not achievable for the parallel composition. The conjunction construction was later extended to systems with different alphabets [BDCU13] and invisible actions [BCU16].

$\vee$ *Disjunction* is easy to obtain for DMTSm, BMTS, and PMTS, again in the stronger form $[\![s_1 \vee s_2]\!] = [\![s_1]\!] \cup [\![s_2]\!]$. However, for MTS (deterministic or not) and DMTS with a single initial process this is not possible. Indeed, consider the MTS specifications in Fig. 17. While the disjunction can be described simply as a BMTS with obligation $\Omega(s_1 \vee s_2) = ((a, \bullet) \wedge (b, \bullet)) \vee (\neg(a, \bullet) \wedge \neg(b, \bullet))$, no DMTS can express this.

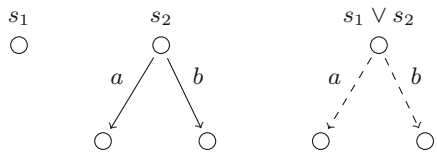

Fig. 17. MTS s_1 and s_2, and their MTS and BMTS least upper bounds $s_1 \vee s_2$

$\neg$ While MTS are not closed under *complement* (not even deterministic ones), there have been attempts at characterizing symmetric difference [SCU11].

The results are summed up in the following statements and Table 3. With operations $\wedge$ and $\vee$, the set of BMTS (or DMTSm) processes forms a bounded distributive lattice up to $(\leq_m \cap \geq_m)$-equivalence. Moreover, with operations $\wedge, \vee, \|$ and $/$, the set of BMTS (or DMTSm) forms a commutative residuated lattice up to $(\leq_m \cap \geq_m)$-equivalence [BDF+13].

We are also interested in questions closely related to the discussed conjunction. The *common implementation* decision problem (CI) contains tuples of systems, such that there is an implementation refining each system of the tuple. For tuples of size two this is equivalent to non-emptiness of the conjunction, for one system (for instance a MixTS) this is equivalent to semantic consistency (or non-emptiness) [LNW07b], i.e. existence of implementation. Note that despite the lack of results on conjunction of non-deterministic systems the complexity was known long ago. The complexity improves when the input processes are deterministic (CI_D problem). Finally, rather surprisingly, the problem whether there is a deterministic common implementation (dCI) is hard. We display the

Table 3. Closure properties

	$\wedge$	$\vee$	$\neg$	$\|\|$	/
Deterministic MTS	✓	×	×	✓	✓
MTS	×	×	×	✓	?
MixTS	✓	×	×	✓	?
DMTS	✓	×	×	✓	?
DMTSm/BMTS/PMTS	✓	✓	×	✓	✓

Table 4. Complexity of the common implementation problems

	Single MTS	Single MixTS	Fixed # of systems	Arbitrary # of systems
CI	Trivial	EXP-c. [AHL+09]	P-c. [BGS92, HH08]	EXP-c. [AHL+09]
CI_D	Trivial	Trivial	NL-c. [BKLS09b]	PSPACE-c. [BKLS09b]
dCI	EXP-c. [BKLS09b]	EXP-c. [BKLS09b]	EXP-c. [BKLS09b]	EXP-c. [BKLS09b]

known results in Table 4 for several cases depending on whether the number of input processes is fixed or a part of the input. The results again indicate that several problems become more tractable if the given specifications are deterministic.

5.3 Model Checking and Synthesis

Given a valuation $\nu : P \to 2^{Ap}$ assigning to each process a set of atomic propositions valid in the process, one can check whether an MTS satisfies a CTL, LTL or μ-calculus formula φ over Ap. Since an MTS stands for a class of implementations, the question of satisfaction can be posed in two flavours:

($\models_\forall$-problem) Do all implementations satisfy φ?
($\models_\exists$-problem) Is there an implementation satisfying φ?

The problem of generalized model checking is to decide which of the three possible cases holds: whether all, only some, or no implementations satisfy φ. Further, if there exists a satisfying implementation it should also be automatically synthesized.

Generalized model checking of MTS was investigated with respect to a variant of safety [DDM10] as well as computation tree logic (CTL) [AHL+08a, GAW13], establishing it EXP-complete and providing a polynomial over- and under-approximation, similarly for μ-calculus. The EXP lower bound follows from the hardness of satisfiability of CTL and μ-calculus; the upper bound can be obtained through alternating tree automata [BG00].

In the rest, we focus on LTL. In [GP09] the generalized model checking of LTL over partial Kripke structures (PKS) is shown to be 2-EXP-hard. Further, [GJ03] describes a reduction from generalized model checking of μ-calculus over PKS to μ-calculus over MTS [Hut02, Hut99, GHJ01]. However, the hardness for

LTL does not follow since the encoding of an LTL formula into μ-calculus is exponential. There is thus no straightforward way to use the result of [GJ03] to establish the complexity for LTL.

On the one hand, answering the $\models_\forall$-problem is easy. Indeed, it is sufficient to perform standard model checking on the "greatest" implementation, i.e. such where all mays are turned into musts and thus all possible runs are present. On the other hand, the $\models_\exists$-problem is trickier. Similarly to the for $\models_\forall$-problem, we can take the minimal implementation of the MTS. However, whenever a deadlock occurs, the corresponding finite runs are ignored since LTL is usually interpreted over infinite words only. However, an undesirable consequence of this problem definition (call it ω, standing for infinite runs) is that all formulae are satisfied whenever there is an implementation without infinite runs, i.e. without a lasso of must transitions. There are several ways to avoid this vacuous satisfaction. Firstly, we can define LTL also on finite words [BČK11], which we denote by ∞ (for both finite and infinite runs). Secondly, we can consider only implementations without deadlocks, which we denote *df*. The deadlock-free approach has been studied in [UBC09] and the proposed solution was implemented in the tool MTSA [DFCU08]. It attempts to find a *deadlock-free* implementation of a given MTS that satisfies a given formula. However, the solution given in [UBC09] is incorrect in that the existence of a deadlock-free implementation satisfying a given formula is claimed even in some cases where no such implementation exists.

Example 16. The flaw can be seen on an example given in Fig. 18 [BČK11]. Clearly, s has no deadlock-free implementation with action a only, i.e. satisfying $\mathbf{GX}_a\mathbf{tt}$. Yet the method of [UBC09] as well as the tool [DFCU08] claim that such an implementation exists. △

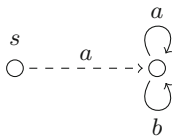

Fig. 18. No deadlock-free implementation of s satisfies $\mathbf{GX}_a\mathbf{tt}$

While the solution attempt of [UBC09] yields a PSPACE algorithm, the *df*-problem is actually 2-EXP-complete[BČK11]. Note that in this setting, there are no minimal implementations; non-trivial decisions have to be made which transitions to implement. For example, an MTS with only one may a-successor and one may b-successor cannot avoid deadlock in a unique way. Moreover, even if deadlocks are allowed, not implementing any choice may result in not satisfying $\mathbf{Xtt}$.

A solution to both *df* and ∞ as well as DMTS is provided in [BČK11]. It reduces the problem to a game where one player decides which transitions to

implement in each step and another player chooses which of the implemented transitions is taken. Decisions of the players determine a run. The objective of the first player is to satisfy the formula on the run. He can always succeed irrespective of what the other player does if and only if there is an implementation satisfying the formula. Such LTL games are in general 2-EXP-complete [PR89]. The consequences are summarized in Table 5. Note that the winning strategy in the game yields a satisfying implementation, thus also solving the synthesis problem. This approach of reduction to an LTL game was also used to solve a similar problem of deciding whether all/some implementation can be pruned to satisfy a given LTL formula [DBPU12].

Table 5. Complexities of generalized LTL model checking (ω denoting finite runs are ignored, *df* denoting deadlock-free implementations are ignored, ∞ denoting no restriction)

	$\models_\forall$	$\models_\exists$
ω-MTS	PSPACE-complete	PSPACE-complete
df-MTS, ∞-MTS, DMTS	PSPACE-complete	2-EXP-complete

The best known time complexity bounds with respect to the size of system $|S|$ and the size of LTL formula $|\varphi|$ are the following. In all PSPACE-complete cases the time complexity is $\mathcal{O}(|S| \cdot 2^{|\varphi|})$; in all 2-EXP-complete cases the time complexity is $|S|^{2^{\mathcal{O}(|\varphi|)}} \cdot 2^{2^{\mathcal{O}(|\varphi|)}}$. The latter upper bound is achieved by translating the LTL formula into a deterministic (possibly generalized) Rabin automaton of size $2^{2^{\mathcal{O}(|\varphi|)}}$ with $2^{\mathcal{O}(|\varphi|)}$ accepting pairs, thus changing the LTL game into a Rabin game. For an efficient translation see e.g. [EK14, KK14]; for an algorithm solving Rabin games see [PP06, CGK13].

Another synthesis problem is the cheapest implementation, considered for (P)MTS with durations [BKL+12]. Intuitively, the constraint on the implementation here is to maximize the average reward per time unit while conforming to the specification and a budget allowing only for some combinations of actions implemented. The problem is NP-complete. Further, the problem of synthesizing a satisfying implementation in the form of a bounded Petri net was considered and shown undecidable [Sch16]. Finally, MTS themselves can be synthesized from constraints given as e.g. scenarios [SBUK13].

LTL model checking has also shed a better light on the problem of incompleteness of the parallel composition. Recall that there is a composition $s_1 \parallel s_2$ with an implementation i that does *not* arise as a composition $i_1 \parallel i_2$ of any two implementations $i_1 \leq_m s_1, i_2 \leq_m s_2$. Completeness can be achieved only under some restrictive conditions [BKLS09b]. [BČK11] shows that composition is sound and complete with respect to every logic of linear time: For DMTS and both ω and ∞,

$s_1 \parallel s_2 \models_\forall \varphi$ iff $i_1 \parallel i_2 \models \varphi$ for all implementations $i_1 \leq_m s_1, i_2 \leq_m s_2$

$s_1 \parallel s_2 \models_\exists \varphi$ iff $i_1 \parallel i_2 \models \varphi$ for some implementations $i_1 \leq_m s_1, i_2 \leq_m s_2$

Thus $\|$ is "LTL complete", i.e. preserves and reflects all LTL properties. Therefore, the only spurious implementations are sums of legal implementations.

6 Tools

The tool support is quite extensive; we focus our attention to the support for the operations required for complete specification theories [BDH+12] and several further problems. This includes modal refinement checking, parallel composition, quotient, conjunction (merge) and the related consistency checking and maximal implementation generation, deterministic hull and generalized LTL model checking. The comparison of the functionality of the available tools is depicted in Table 6. Apart from no longer maintained `TAV` [BLS95], the currently available tools are the following:

MTSA (Modal Transition System Analyzer) [DFFU07]

- is a tool for MTS,
- supports modal refinement, parallel composition and consistency using the cloning operation, which may not terminate; it also offers a model checking procedure, which is, unfortunately, flawed as discussed in Example 16.

MIO (Modal Input Output Workbench) [BML11, BMSH10]

- is a tool for modal I/O automata (MIOA) [LNW07a, RBB+11], which combine MTS and interface automata based on I/O automata; although MIOA have three types of may and must transitions (input, output, and internal), if we restrict to say only input transitions, the refinement works the same as for MTS, and some other operations, too,
- supports modal refinement, the MIOA parallel composition, conjunction for deterministic systems, and quotient for deterministic systems.

$\overrightarrow{\longrightarrow\Longrightarrow\dashrightarrow}$
MoTraS (Modal Transition Systems) [KS13a]

- is a tool for MTS and DMTS, with partial support for BMTS and PMTS,
- supports full functionality for MTS as well as more general DMTS and in all cases also for *non-deterministic* systems; in particular, the algorithms for conjunction and quotient are considerably more complex than for the deterministic case; further, it features QBF-based algorithms for BMTS and PMTS refinement; finally, it also provides the deterministic hull, which enables us to both over- and under-approximate the very hard thorough refinement using the fast modal refinement.

MAccS (Marked Acceptance Specifications) [VR14]

- is a tool for acceptance automata (deterministic BMTS) with accepting states,
- features all the operations for acceptance automata, hence also for *deterministic* MTS.

Table 6. Functionality of the available tools. Here "det." denotes a functionality limited to deterministic systems.

Operation	MTS			DMTS	B/PMTS	det.AA
Parallel composition	MTSA	MIO	MoTraS	MoTraS		MAccS
Consistency	MTSA(of 2 systems)	MIO(det.)	MoTraS	MoTraS		MAccS
Conjunction		MIO(det.)	MoTraS	MoTraS		MAccS
Quotient (det.)		MIO	MoTraS			MAccS
Generalized LTL	MTSA(incorrect)		MoTraS	MoTraS		
Det. hull			MoTraS	MoTraS	MoTraS	
Refinement	MTSA	MIO	MoTraS	MoTraS	MoTraS	MAccS

Note that both MTSA and MIO can only handle modal systems, not their disjunctive extension. MoTraS supports DMTS, which have more expressive power. In contrast to (non-deterministic) MTS, DMTS are rich enough to express solutions to process equations [LX90] (hence a specification of a missing component in a system can be computed) and are closed under all operations, particularly conjunction. MAccS is similar in that AA are equally expressive and it supports all the operations, however, only for deterministic systems.

In order to make the tools easily extensible, a file format *xmts* was designed [MTS], which facilitates textual representation of different extensions of modal transition systems.

Besides, there are the following tools for related formalisms:
ECDAR (Environment for Compositional Design and Analysis of Real Time Systems) [DLL+10]

- is a tool for timed I/O automata (with no modalities);
- supports refinement, conjunction, composition and quotient, but all for only deterministic systems, as can be expected in the timed setting.

APAC (Abstract Probabilistic Automata Checker) [DLL+11]

- is a tool for abstract probabilistic automata;
- supports refinement, abstraction, conjunction, and composition.

7 Conclusion and Some Directions for Future Work

Firstly, we have surveyed MTS and its many extensions, including more involved modalities (combined, exclusive, persistent or conditional choices), quantitative models, or infinite-state systems. The comparison of various classes leads us to identifying a robust class of DMTS with more initial states, equivalent to several other formalisms, including the modal ν-calculus. This unifies the behavioural and logical approach to specification and verification and enables us to mix the two.

Secondly, we have surveyed solutions to problems arising in system design via MTS, such as logical and structural operations, refinement (modal, thorough,

approximations using the deterministic hull) and synthesis of implementations based on temporal or reward constraints. We have also discussed the tool support for these problems.

As for future work, we mention several open issues. Firstly, although the *complexity* of many problems has been established, there are still several complexity gaps left open, for instance, the complexity of thorough refinement for BMTS and PMTS, the quotient construction (we conjecture the exponential blow-up is in general unavoidable), whether MTS, MixTS and DMTS are closed under quotient (we conjecture the opposite), or conditions on decidability of refinement over infinite systems, e.g. determinism as in [BKLS09b,EBHH10,HHM13].

Secondly, one may also extend the *model checking and synthesis* algorithms to more complex settings such as the cheapest implementation with an additional requirement that the partial sums stay within given bounds as done in [BFL+08], or cheapest implementation satisfying a temporal property as suggested in [CdAHS03,CD10], model checking metric temporal logic (LTL with time durations) [Koy90], model checking infinite-state MTS similarly to PDA in [Wal96], or cheapest implementation of mPDA using methods like [CV12].

Thirdly, on the *practical* side, all the tools only offer a limited support. In particular, the quotient of non-deterministic systems is very important for practical design and has not yet been implemented. Refinement algorithms do not scale too well on MTS extensions. Apart from multi-threading for all algorithms, one could use a combined modal refinement checker, which uses the standard modal refinement checker to prune the initial relation before the QBF-based checker is called. Altogether, the topic is still lively and subject to further practical developments, e.g. the currently prepared update of $\overrightarrow{\text{MoTraS}}$ features faster model checking due to integrating a better LTL-to-automata translator `Rabinizer 3` [KK14] and the cheapest implementation synthesizer [BKL+12,Man13].

Finally, the practical usability of MTS could be greatly improved by providing a higher-level *language*, possibly tailored to particular domains, which has MTS semantics, but a friendlier appearance to the domain-specific engineering practice.

Acknowledgement. I would like to thank Kim G. Larsen for introducing me to the topic of MTS and research in general, the pleasant collaboration during my Erasmus stay as a Master student at AAU a decade ago and ever since then. Arriving to Aalborg right after the paper *20 Years of Modal and Mixed Specifications* [AHL+08a] was published, my first paper with Kim [BKLS09b] was on MTS as a part of the Festschrift to Mogens Nielsen's 60th birthday [CSV09]. It is a pleasure and an honour to contribute today to Kim's Festschrift.

References

[AFdFE+11] Aceto, L., Fábregas, I., Frutos Escrig, D., Ingólfsdóttir, A., Palomino, M.: Relating modal refinements, covariant-contravariant simulations and partial bisimulations. In: Arbab, F., Sirjani, M. (eds.) FSEN 2011. LNCS, vol. 7141, pp. 268–283. Springer, Heidelberg (2012). doi:10.1007/978-3-642-29320-7_18

[AHKV98] Alur, R., Henzinger, T.A., Kupferman, O., Vardi, M.Y.: Alternating refinement relations. In: Sangiorgi, D., Simone, R. (eds.) CONCUR 1998. LNCS, vol. 1466, pp. 163–178. Springer, Heidelberg (1998). doi:10.1007/BFb0055622

[AHL+08a] Antonik, A., Huth, M., Larsen, K.G., Nyman, U., Wasowski, A.: 20 years of modal and mixed specifications. Bull. EATCS **95**, 94–129 (2008)

[AHL+08b] Antonik, A., Huth, M., Larsen, K.G., Nyman, U., Wąsowski, A.: Complexity of decision problems for mixed and modal specifications. In: Amadio, R. (ed.) FoSSaCS 2008. LNCS, vol. 4962, pp. 112–126. Springer, Heidelberg (2008). doi:10.1007/978-3-540-78499-9_9

[AHL+09] Antonik, A., Huth, M., Larsen, K.G., Nyman, U., Wasowski, A.: EXPTIME-complete decision problems for modal and mixed specifications. Electron. Notes Theor. Comput. Sci. **242**(1), 19–33 (2009)

[AKRU11] Alrajeh, D., Kramer, J., Russo, A., Uchitel, S.: An inductive approach for modal transition system refinement. In: Gallagher, J.P., Gelfond, M. (eds) ICLP (Technical Communications). LIPIcs, vol. 11, pp. 106–116. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2011)

[Bau12] Bauer, S.S:. Modal specification theories for component-based design. Ph.D. thesis, Ludwig Maximilians University Munich (2012)

[BČK10] Beneš, N., Černá, I., Křetínský, J.: Disjunctive modal transition systems and generalized LTL model checking. Technical report FIMU-RS-2010-12, Faculty of Informatics, Masaryk University, Brno (2010)

[BČK11] Beneš, N., Černá, I., Křetínský, J.: Modal transition systems: composition and LTL model checking. In: Bultan, T., Hsiung, P.-A. (eds.) ATVA 2011. LNCS, vol. 6996, pp. 228–242. Springer, Heidelberg (2011). doi:10.1007/978-3-642-24372-1_17

[BCU06] Brunet, G., Chechik, M., Uchitel, S.: Properties of behavioural model merging. In: Misra, J., Nipkow, T., Sekerinski, E. (eds.) FM 2006. LNCS, vol. 4085, pp. 98–114. Springer, Heidelberg (2006). doi:10.1007/11813040_8

[BCU16] Ben-David, S., Chechik, M., Uchitel, S.: Observational refinement and merge for disjunctive MTSs. In: Artho, C., Legay, A., Peled, D. (eds.) ATVA 2016. LNCS, vol. 9938, pp. 287–303. Springer, Cham (2016). doi:10.1007/978-3-319-46520-3_19

[BDCU13] Ben-David, S., Chechik, M., Uchitel, S.: Merging partial behaviour models with different vocabularies. In: D'Argenio, P.R., Melgratti, H. (eds.) CONCUR 2013. LNCS, vol. 8052, pp. 91–105. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40184-8_8

[BDF+13] Beneš, N., Delahaye, B., Fahrenberg, U., Křetínský, J., Legay, A.: Hennessy-milner logic with greatest fixed points as a complete behavioural specification theory. In: D'Argenio, P.R., Melgratti, H. (eds.) CONCUR 2013. LNCS, vol. 8052, pp. 76–90. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40184-8_7

[BDH+12] Bauer, S.S., David, A., Hennicker, R., Guldstrand Larsen, K., Legay, A., Nyman, U., Wąsowski, A.: Moving from specifications to contracts in component-based design. In: Lara, J., Zisman, A. (eds.) FASE 2012. LNCS, vol. 7212, pp. 43–58. Springer, Heidelberg (2012). doi:10.1007/978-3-642-28872-2_3

[BDH+15] Beneš, N., Daca, P., Henzinger, T.A., Křetínský, J., Nickovic, D.: Complete composition operators for IOCO-testing theory. In: Kruchten, P., Becker, S., Schneider, J.-G. (eds.) Proceedings of the 18th International ACM SIGSOFT Symposium on Component-Based Software Engineering, CBSE 2015, Montreal, QC, Canada, 4–8 May 2015, pp. 101–110. ACM (2015)

[Ben08] Benveniste, A.: Multiple viewpoint contracts and residuation. In: 2nd International Workshop on Foundations of Interface Technologies (FIT) (2008)

[Ben12] Beneš, N.: Disjunctive modal transition systems. Ph.D. thesis, Masaryk University (2012)

[BFJ+11] Bauer, S.S., Fahrenberg, U., Juhl, L., Larsen, K.G., Legay, A., Thrane, C.: Quantitative refinement for weighted modal transition systems. In: Murlak, F., Sankowski, P. (eds.) MFCS 2011. LNCS, vol. 6907, pp. 60–71. Springer, Heidelberg (2011). doi:10.1007/978-3-642-22993-0_9

[BFJ+13] Bauer, S.S., Fahrenberg, U., Juhl, L., Larsen, K.G., Legay, A., Thrane, C.R.: Weighted modal transition systems. Formal Methods Syst. Des. **42**(2), 193–220 (2013)

[BFL+08] Bouyer, P., Fahrenberg, U., Larsen, K.G., Markey, N., Srba, J.: Infinite runs in weighted timed automata with energy constraints. In: Cassez, F., Jard, C. (eds.) FORMATS 2008. LNCS, vol. 5215, pp. 33–47. Springer, Heidelberg (2008). doi:10.1007/978-3-540-85778-5_4

[BFLT12] Bauer, S.S., Fahrenberg, U., Legay, A., Thrane, C.: General quantitative specification theories with modalities. In: Hirsch, E.A., Karhumäki, J., Lepistö, A., Prilutskii, M. (eds.) CSR 2012. LNCS, vol. 7353, pp. 18–30. Springer, Heidelberg (2012). doi:10.1007/978-3-642-30642-6_3

[BFLV16] Bujtor, F., Fendrich, S., Lüttgen, G., Vogler, W.: Nondeterministic modal interfaces. Theor. Comput. Sci. **642**, 24–53 (2016)

[BG99] Bruns, G., Godefroid, P.: Model checking partial state spaces with 3-valued temporal logics. In: Halbwachs, N., Peled, D. (eds.) CAV 1999. LNCS, vol. 1633, pp. 274–287. Springer, Heidelberg (1999). doi:10.1007/3-540-48683-6_25

[BG00] Bruns, G., Godefroid, P.: Generalized model checking: reasoning about partial state spaces. In: Palamidessi, C. (ed.) CONCUR 2000. LNCS, vol. 1877, pp. 168–182. Springer, Heidelberg (2000). doi:10.1007/3-540-44618-4_14

[BGS92] Balcázar, J.L., Gabarró, J., Santha, M.: Deciding bisimilarity is p-complete. Formal Asp. Comput. **4**(6A), 638–648 (1992)

[BH11] Bultan, T., Hsiung, P.-A. (eds.): ATVA 2011. LNCS, vol. 6996. Springer, Heidelberg (2011). doi:10.1007/978-3-642-24372-1

[BHB10] Bauer, S.S., Hennicker, R., Bidoit, M.: A modal interface theory with data constraints. In: Davies, J., Silva, L., Simao, A. (eds.) SBMF 2010. LNCS, vol. 6527, pp. 80–95. Springer, Heidelberg (2011). doi:10.1007/978-3-642-19829-8_6

[BHJ10] Bauer, B.B., Hennicker, R., Janisch, S.: Interface theories for (a)synchronously communicating modal I/O-transition systems. In: Legay, A., Caillaud, B. (eds.) FIT. EPTCS, vol. 46, pp. 1–8 (2010)

[BHW10] Bauer, S.S., Hennicker, R., Wirsing, M.: Building a modal interface theory for concurrency and data. In: Mossakowski, T., Kreowski, H.-J. (eds.) WADT 2010. LNCS, vol. 7137, pp. 1–12. Springer, Heidelberg (2012). doi:10.1007/978-3-642-28412-0_1

[BJL+12a] Bauer, S.S., Juhl, L., Larsen, K.G., Legay, A., Srba, J.: Extending modal transition systems with structured labels. Math. Struct. Comput. Sci. **22**(4), 581–617 (2012)

[BJL+12b] Bauer, S.S., Juhl, L., Larsen, K.G., Srba, J., Legay, A.: A logic for accumulated-weight reasoning on multiweighted modal automata. In: Margaria, T., Qiu, Z., Yang, H. (eds.) TASE, pp. 77–84. IEEE (2012)

[BK10] Beneš, N., Křetínský, J.: Process algebra for modal transition systemses. In: Matyska, L., Kozubek, M., Vojnar, T., Zemcik, P., Antos, D. (eds.) MEMICS. OASICS, vol. 16, pp. 9–18. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany (2010)

[BK12] Beneš, N., Křetínský, J.: Modal process rewrite systems. In: Roychoudhury, A., D'Souza, M. (eds.) [RD12], pp. 120–135

[BKL+11] Beneš, N., Křetínský, J., Larsen, K.G., Møller, M.H., Srba, J.: Parametric modal transition systems. In: Bultan, T., Hsiung, P.-A. (eds.) ATVA 2011. LNCS, vol. 6996, pp. 275–289. Springer, Heidelberg (2011). doi:10.1007/978-3-642-24372-1_20

[BKL+12] Beneš, N., Křetínský, J., Guldstrand Larsen, K., Møller, M.H., Srba, J.: Dual-priced modal transition systems with time durations. In: Bjørner, N., Voronkov, A. (eds.) LPAR 2012. LNCS, vol. 7180, pp. 122–137. Springer, Heidelberg (2012). doi:10.1007/978-3-642-28717-6_12

[BKL+15] Beneš, N., Křetínský, J., Larsen, K.G., Møller, M.H., Sickert, S., Srba, J.: Refinement checking on parametric modal transition systems. Acta Inf. **52**(2–3), 269–297 (2015)

[BKLS09a] Beneš, N., Křetínský, J., Larsen, K.G., Srba, J.: Checking thorough refinement on modal transition systems is EXPTIME-complete. In: Leucker, M., Morgan, C. (eds.) [LM09], pp. 112–126

[BKLS09b] Beneš, N., Křetínský, J., Larsen, K.G., Srba, J.: On determinism in modal transition systems. Theor. Comput. Sci. **410**(41), 4026–4043 (2009)

[BKLS12] Beneš, N., Křetínský, J., Larsen, K.G., Srba, J.: EXPTIME-completeness of thorough refinement on modal transition systems. Inf. Comput. **218**, 54–68 (2012)

[BL92] Boudol, G., Larsen, K.G.: Graphical versus logical specifications. Theor. Comput. Sci. **106**(1), 3–20 (1992)

[BLL+14] Bauer, S.S., Larsen, K.G., Legay, A., Nyman, U., Wasowski, A.: A modal specification theory for components with data. Sci. Comput. Program. **83**, 106–128 (2014)

[BLPR09] Bertrand, N., Legay, A., Pinchinat, S., Raclet, J.-B.: A compositional approach on modal specifications for timed systems. In: Breitman, K., Cavalcanti, A. (eds.) ICFEM 2009. LNCS, vol. 5885, pp. 679–697. Springer, Heidelberg (2009). doi:10.1007/978-3-642-10373-5_35

[BLPR12] Bertrand, N., Legay, A., Pinchinat, S., Raclet, J.-B.: Modal event-clock specifications for timed component-based design. Sci. Comput. Program. **77**(12), 1212–1234 (2012)

[BLS95] Børjesson, A., Larsen, K.G., Skou, A.: Generality in design and compositional verification using TAV. Formal Methods Syst. Des. **6**(3), 239–258 (1995)

[BML11] Bauer, S.S., Mayer, P., Legay, A.: MIO workbench: a tool for compositional design with modal input/output interfaces. In: Bultan, T., Hsiung, P.-A. (eds.) ATVA 2011. LNCS, vol. 6996, pp. 418–421. Springer, Heidelberg (2011). doi:10.1007/978-3-642-24372-1_30

[BMSH10] Bauer, S.S., Mayer, P., Schroeder, A., Hennicker, R.: On weak modal compatibility, refinement, and the MIO workbench. In: Esparza, J., Majumdar, R. (eds.) TACAS 2010. LNCS, vol. 6015, pp. 175–189. Springer, Heidelberg (2010). doi:10.1007/978-3-642-12002-2_15

[BPR09] Bertrand, N., Pinchinat, S., Raclet, J.-B.: Refinement and consistency of timed modal specifications. In: Dediu, A.H., Ionescu, A.M., Martín-Vide, C. (eds.) LATA 2009. LNCS, vol. 5457, pp. 152–163. Springer, Heidelberg (2009). doi:10.1007/978-3-642-00982-2_13

[Bru97] Bruns, G.: An industrial application of modal process logic. Sci. Comput. Program. **29**(1–2), 3–22 (1997)

[BSV15] Bujtor, F., Sorokin, L., Vogler, W.: Testing preorders for DMTS: deadlock- and the new deadlock/divergence-testing. In: 15th International Conference on Application of Concurrency to System Design, ACSD 2015, Brussels, Belgium, 21–26 June 2015, pp. 60–69. IEEE Computer Society (2015)

[BV15] Bujtor, F., Vogler, W.: Failure semantics for modal transition systems. ACM Trans. Embed. Comput. Syst. **14**(4), 67:1–67:30 (2015)

[CD10] Chatterjee, K., Doyen, L.: Energy parity games. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010. LNCS, vol. 6199, pp. 599–610. Springer, Heidelberg (2010). doi:10.1007/978-3-642-14162-1_50

[CdAHS03] Chakrabarti, A., Alfaro, L., Henzinger, T.A., Stoelinga, M.: Resource interfaces. In: Alur, R., Lee, I. (eds.) EMSOFT 2003. LNCS, vol. 2855, pp. 117–133. Springer, Heidelberg (2003). doi:10.1007/978-3-540-45212-6_9

[CDEG03] Chechik, M., Devereux, B., Easterbrook, S.M., Gurfinkel, A.: Multi-valued symbolic model-checking. ACM Trans. Softw. Eng. Methodol. **12**(4), 371–408 (2003)

[CDL+10] Caillaud, B., Delahaye, B., Larsen, K.G., Legay, A., Pedersen, M.L., Wasowski, A.: Compositional design methodology with constraint Markov chains. In: QEST, pp. 123–132. IEEE Computer Society (2010)

[CE81] Clarke, E.M., Emerson, E.A.: Design and synthesis of synchronization skeletons using branching time temporal logic. In: Kozen, D. (ed.) Logic of Programs 1981. LNCS, vol. 131, pp. 52–71. Springer, Heidelberg (1982). doi:10.1007/BFb0025774

[CGK13] Chatterjee, K., Gaiser, A., Křetínský, J.: Automata with generalized rabin pairs for probabilistic model checking and LTL synthesis. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 559–575. Springer, Heidelberg (2013). doi:10.1007/978-3-642-39799-8_37

[CGL93] Čerāns, K., Godskesen, J.C., Larsen, K.G.: Timed modal specification—theory and tools. In: Courcoubetis, C. (ed.) CAV 1993. LNCS, vol. 697, pp. 253–267. Springer, Heidelberg (1993). doi:10.1007/3-540-56922-7_21

[CGLT09] Campetelli, A., Gruler, A., Leucker, M., Thoma, D.: *Don't know* for multi-valued systems. In: Liu, Z., Ravn, A.P. (eds.) ATVA 2009. LNCS, vol. 5799, pp. 289–305. Springer, Heidelberg (2009). doi:10.1007/978-3-642-04761-9_22

[CR12] Caillaud, B., Raclet, J.-B.: Ensuring reachability by design. In: Roychoudhury, A., D'Souza, M. (eds.) [RD12], pp. 213–227

[CSV09] Carbone, M., Sobocinski, P., Valencia, F.D.: Foreword: Festschrift for mogens nielsen's 60th birthday. Theor. Comput. Sci. **410**(41), 4001–4005 (2009)

[CV12] Chatterjee, K., Velner, Y.: Mean-payoff pushdown games. In: LICS, pp. 195–204. IEEE (2012)
[dAGJ04] de Alfaro, L., Godefroid, P., Jagadeesan, R.: Three-valued abstractions of games: uncertainty, but with precision. In: LICS04 [LIC04], pp. 170–179
[dAH01] de Alfaro, L., Henzinger, T.A.: Interface automata. In: ESEC/SIGSOFT FSE, pp. 109–120. ACM (2001)
[DBPU12] D'Ippolito, N., Braberman, V.A., Piterman, N., Uchitel, S.: The modal transition system control problem. In: Giannakopoulou, D., Méry, D. (eds.) [GM12], pp. 155–170
[DDM10] Darondeau, P., Dubreil, J., Marchand, H.: Supervisory control for modal specifications of services. In: WODES, pp. 428–435 (2010)
[DFCU08] D'Ippolito, N., Fischbein, D., Chechik, M., Uchitel, S.: MTSA: the modal transition system analyser. In: ASE, pp. 475–476. IEEE (2008)
[DFFU07] D'Ippolito, N., Fischbein, D., Foster, H., Uchitel, S.: MTSA: eclipse support for modal transition systems construction, analysis and elaboration. In: Cheng, L.-T., Orso, A., Robillard, M.P. (eds.) ETX, pp. 6–10. ACM (2007)
[DFLL14] Delahaye, B., Fahrenberg, U., Larsen, K.G., Legay, A.: Refinement and difference for probabilistic automata. Log. Methods Comput. Sci. **10**(3) (2014)
[DGG97] Dams, D., Gerth, R., Grumberg, O.: Abstract interpretation of reactive systems. ACM Trans. Program. Lang. Syst. **19**(2), 253–291 (1997)
[DKL+11a] Delahaye, B., Katoen, J.-P., Larsen, K.G., Legay, A., Pedersen, M.L., Sher, F., Wąsowski, A.: Abstract probabilistic automata. In: Jhala, R., Schmidt, D. (eds.) VMCAI 2011. LNCS, vol. 6538, pp. 324–339. Springer, Heidelberg (2011). doi:10.1007/978-3-642-18275-4_23
[DKL+11b] Delahaye, B., Katoen, J.-P., Larsen, K.G., Legay, A., Pedersen, M.L., Sher, F., Wasowski, A.: New results on abstract probabilistic automata. In: Caillaud, B., Carmona, J., Hiraishi, K. (eds.) 11th International Conference on Application of Concurrency to System Design, ACSD 2011, Newcastle Upon Tyne, UK, 20–24 June 2011, pp. 118–127. IEEE Computer Society (2011)
[DLL+10] David, A., Larsen, K.G., Legay, A., Nyman, U., Wąsowski, A.: ECDAR: an environment for compositional design and analysis of real time systems. In: Bouajjani, A., Chin, W.-N. (eds.) ATVA 2010. LNCS, vol. 6252, pp. 365–370. Springer, Heidelberg (2010). doi:10.1007/978-3-642-15643-4_29
[DLL+11] Delahaye, B., Larsen, K.G., Legay, A., Pedersen, M.L., Wasowski, A.: APAC: a tool for reasoning about abstract probabilistic automata. In: Eighth International Conference on Quantitative Evaluation of Systems, QEST 2011, Aachen, Germany, 5–8 September 2011, pp. 151–152. IEEE Computer Society (2011)
[DLL14] Delahaye, B., Larsen, K.G., Legay, A.: Stuttering for abstract probabilistic automata. J. Log. Algebr. Program. **83**(1), 1–19 (2014)
[DM13] D'Argenio, P.R., Melgratti, H. (eds.): CONCUR 2013. LNCS, vol. 8052. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40184-8
[DN04] Dams, D., Namjoshi, K.S.: The existence of finite abstractions for branching time model checking. In: LICS04 [LIC04], pp. 335–344
[DSMB16] Diskin, Z., Safilian, A., Maibaum, T., Ben-David, S.: Faithful modeling of product lines with kripke structures and modal logic. Sci. Ann. Comput. Sci. **26**(1), 69–122 (2016)

[EBHH10] Elhog-Benzina, D., Haddad, S., Hennicker, R.: Process refinement and asynchronous composition with modalities. In: Donatelli, S., Kleijn, J., Machado, R.J., Fernandes, J.M. (eds.) ACSD/Petri Nets Workshops. CEUR Workshop Proceedings, vol. 827, pp. 385–401. CEUR-WS.org (2010)

[EHH12] Elhog-Benzina, D., Haddad, S., Hennicker, R.: Refinement and asynchronous composition of modal petri nets. Trans. Petri Nets Other Models Concurr. **5**, 96–120 (2012)

[EK14] Esparza, J., Křetínský, J.: From LTL to deterministic automata: a safraless compositional approach. In: Biere, A., Bloem, R. (eds.) CAV 2014. LNCS, vol. 8559, pp. 192–208. Springer, Cham (2014). doi:10.1007/978-3-319-08867-9_13

[FBU09] Fischbein, D., Braberman, V.A., Uchitel, S.: A sound observational semantics for modal transition systems. In: Leucker, M., Morgan, C. (eds.) [LM09], pp. 215–230

[Fis12] Fischbein, D.: Foundations for behavioural model elaboration using modal transition systems. Ph.D. thesis, Imperial College London, UK, (2012)

[FKLT14] Fahrenberg, U., Křetínský, J., Legay, A., Traonouez, L.-M.: Compositionality for quantitative specifications. In: Lanese, I., Madelaine, E. (eds.) [LM15], pp. 306–324

[FL12] Fahrenberg, U., Legay, A.: A robust specification theory for modal event-clock automata. In: Bauer, S.S., Raclet, J.-B. (eds) FIT. EPTCS, vol. 87, pp. 5–16 (2012)

[FL14] Fahrenberg, U., Legay, A.: General quantitative specification theories with modal transition systems. Acta Inf. **51**(5), 261–295 (2014)

[FLT14] Fahrenberg, U., Legay, A., Traonouez, L.-M.: Structural refinement for the modal nu-Calculus. In: Ciobanu, G., Méry, D. (eds.) ICTAC 2014. LNCS, vol. 8687, pp. 169–187. Springer, Cham (2014). doi:10.1007/978-3-319-10882-7_11

[FLW06] Fecher, H., Leucker, M., Wolf, V.: *Don't know* in probabilistic systems. In: Valmari, A. (ed.) SPIN 2006. LNCS, vol. 3925, pp. 71–88. Springer, Heidelberg (2006). doi:10.1007/11691617_5

[FP07] Feuillade, G., Pinchinat, S.: Modal specifications for the control theory of discrete event systems. Discret. Event Dyn. Syst. **17**(2), 211–232 (2007)

[FS05] Fecher, H., Steffen, M.: Characteristic μ-calculus formulas for underspecified transition systems. Electr. Notes Theor. Comput. Sci. **128**(2), 103–116 (2005)

[FS08] Fecher, H., Schmidt, H.: Comparing disjunctive modal transition systems with an one-selecting variant. J. Log. Algebr. Program. **77**(1–2), 20–39 (2008)

[FU08] Fischbein, D., Uchitel, S.: On correct and complete strong merging of partial behaviour models. In: Harrold, M.J., Murphy, G.C. (eds.) SIGSOFT FSE, pp. 297–307. ACM (2008)

[GAW13] Guerra, P.T., Andrade, A., Wassermann, R.: Toward the revision of CTL models through Kripke modal transition systems. In: Iyoda, J., Moura, L. (eds.) SBMF 2013. LNCS, vol. 8195, pp. 115–130. Springer, Heidelberg (2013). doi:10.1007/978-3-642-41071-0_9

[GC06] Gurfinkel, A., Chechik, M.: Why waste a perfectly good abstraction? In: Hermanns, H., Palsberg, J. (eds.) TACAS 2006. LNCS, vol. 3920, pp. 212–226. Springer, Heidelberg (2006). doi:10.1007/11691372_14

[GHJ01] Godefroid, P., Huth, M., Jagadeesan, R.: Abstraction-based model checking using modal transition systems. In: Larsen, K.G., Nielsen, M. (eds.) CONCUR 2001. LNCS, vol. 2154, pp. 426–440. Springer, Heidelberg (2001). doi:10.1007/3-540-44685-0_29

[GJ03] Godefroid, P., Jagadeesan, R.: On the expressiveness of 3-valued models. In: Zuck, L.D., Attie, P.C., Cortesi, A., Mukhopadhyay, S. (eds.) VMCAI 2003. LNCS, vol. 2575, pp. 206–222. Springer, Heidelberg (2003). doi:10.1007/3-540-36384-X_18

[GM12] Giannakopoulou, D., Méry, D. (eds.): FM 2012. LNCS, vol. 7436. Springer, Heidelberg (2012). doi:10.1007/978-3-642-32759-9

[GNRT10] Godefroid, P., Nori, A.V., Rajamani, S.K., Tetali, S.: Compositional may-must program analysis: unleashing the power of alternation. In: Hermenegildo, M.V., Palsberg, J. (eds.) POPL, pp. 43–56. ACM (2010)

[GP09] Godefroid, P., Piterman, N.: LTL generalized model checking revisited. In: Jones, N.D., Müller-Olm, M. (eds.) [JMO09], pp. 89–104

[GSSL94] Gawlick, R., Segala, R., Søgaard-Andersen, J., Lynch, N.: Liveness in timed and untimed systems. In: Abiteboul, S., Shamir, E. (eds.) ICALP 1994. LNCS, vol. 820, pp. 166–177. Springer, Heidelberg (1994). doi:10.1007/3-540-58201-0_66

[GWC06a] Gurfinkel, A., Wei, O., Chechik, M.: Systematic construction of abstractions for model-checking. In: Emerson, E.A., Namjoshi, K.S. (eds.) VMCAI 2006. LNCS, vol. 3855, pp. 381–397. Springer, Heidelberg (2005). doi:10.1007/11609773_25

[GWC06b] Gurfinkel, A., Wei, O., Chechik, M.: YASM: a software model-checker for verification and refutation. In: Ball, T., Jones, R.B. (eds.) CAV 2006. LNCS, vol. 4144, pp. 170–174. Springer, Heidelberg (2006). doi:10.1007/11817963_18

[Hen85] Hennessy, M.: Acceptance trees. J. ACM **32**(4), 896–928 (1985)

[HH08] Hussain, A., Huth, M.: On model checking multiple hybrid views. Theor. Comput. Sci. **404**(3), 186–201 (2008)

[HHM13] Haddad, S., Hennicker, R., Møller, M.H.: Specification of asynchronous component systems with modal I/O-petri nets. In: Abadi, M., Lluch Lafuente, A. (eds.) TGC 2013. LNCS, vol. 8358, pp. 219–234. Springer, Cham (2014). doi:10.1007/978-3-319-05119-2_13

[HJS01] Huth, M., Jagadeesan, R., Schmidt, D.: Modal transition systems: a foundation for three-valued program analysis. In: Sands, D. (ed.) ESOP 2001. LNCS, vol. 2028, pp. 155–169. Springer, Heidelberg (2001). doi:10.1007/3-540-45309-1_11

[HKK13] Hermanns, H., Krčál, J., Křetínský, J.: Compositional verification and optimization of interactive Markov chains. In: D'Argenio, P.R., Melgratti, H.C. (eds.) [DM13], pp. 364–379

[HKKG13] Han, T., Krause, C., Kwiatkowska, M.Z., Giese, H.: Modal specifications for probabilistic timed systems. In: Bortolussi, L., Wiklicky, H. (eds.) QAPL. EPTCS, vol. 117, pp. 66–80 (2013)

[HL89] Hüttel, H., Larsen, K.G.: The use of static constructs in a model process logic. In: Meyer, A.R., Taitslin, M.A. (eds.) Logic at Botik 1989. LNCS, vol. 363, pp. 163–180. Springer, Heidelberg (1989). doi:10.1007/3-540-51237-3_14

[Hol89] Holmström, S.: A refinement calculus for specifications in Hennessy-Milner logic with recursion. Formal Asp. Comput. **1**(3), 242–272 (1989)

[Hut99] Huth, M.: A unifying framework for model checking labeled kripke structures, modal transition systems, and interval transition systems. In: Rangan, C.P., Raman, V., Ramanujam, R. (eds.) FSTTCS 1999. LNCS, vol. 1738, pp. 369–380. Springer, Heidelberg (1999). doi:10.1007/3-540-46691-6_30

[Hut02] Huth, M.: Model checking modal transition systems using Kripke structures. In: Cortesi, A. (ed.) VMCAI 2002. LNCS, vol. 2294, pp. 302–316. Springer, Heidelberg (2002). doi:10.1007/3-540-47813-2_21

[JL91] Jonsson, B., Larsen, K.G.: Specification and refinement of probabilistic processes. In: LICS, pp. 266–277. IEEE Computer Society (1991)

[JLS12] Juhl, L., Larsen, K.G., Srba, J.: Modal transition systems with weight intervals. J. Log. Algebr. Program. **81**(4), 408–421 (2012)

[JMO09] Jones, N.D., Müller-Olm, M. (eds.): VMCAI 2009. LNCS, vol. 5403. Springer, Heidelberg (2009). doi:10.1007/978-3-540-93900-9

[Juh13] Juhl, L.: Quantities in games and modal transition systems. Ph.D. thesis, Department of Computer Science, Aalborg University (2013)

[KDMU14] Krka, I., D'Ippolito, N., Medvidović, N., Uchitel, S.: Revisiting compatibility of input-output modal transition systems. In: Jones, C., Pihlajasaari, P., Sun, J. (eds.) FM 2014. LNCS, vol. 8442, pp. 367–381. Springer, Cham (2014). doi:10.1007/978-3-319-06410-9_26

[KK14] Komárková, Z., Křetínský, J.: Rabinizer 3: safraless translation of LTL to small deterministic automata. In: Cassez, F., Raskin, J.-F. (eds.) ATVA 2014. LNCS, vol. 8837, pp. 235–241. Springer, Cham (2014). doi:10.1007/978-3-319-11936-6_17

[KKLW07] Katoen, J.-P., Klink, D., Leucker, M., Wolf, V.: Three-valued abstraction for continuous-time Markov chains. In: Damm, W., Hermanns, H. (eds.) CAV 2007. LNCS, vol. 4590, pp. 311–324. Springer, Heidelberg (2007). doi:10.1007/978-3-540-73368-3_37

[KKLW12] Katoen, J.-P., Klink, D., Leucker, M., Wolf, V.: Three-valued abstraction for probabilistic systems. J. Log. Algebr. Program. **81**(4), 356–389 (2012)

[KM12] Krka I., Medvidovic, N.: Revisiting modal interface automata. In: Gnesi, S., Gruner, S., Plat, N., Rumpe, B. (eds.) Proceedings of the First International Workshop on Formal Methods in Software Engineering - Rigorous and Agile Approaches, FormSERA 2012, Zurich, Switzerland, 2 June 2012, pp. 30–36. IEEE (2012)

[Koy90] Koymans, R.: Specifying real-time properties with metric temporal logic. Real-Time Syst. **2**(4), 255–299 (1990)

[Koz83] Kozen, D.: Results on the propositional μ-calculus. Theor. Comput. Sci. **27**, 333–354 (1983)

[Kře14] Křetínský, J.: Modal transition systems: extensions and analysis. Ph.D. thesis, Masaryk University, Brno, Department of Computer Science (2014)

[KŘS05] Křetínský, M., Řehák, V., Strejček, J.: Reachability of Hennessy-Milner properties for weakly extended PRS. In: Sarukkai, S., Sen, S. (eds.) FSTTCS 2005. LNCS, vol. 3821, pp. 213–224. Springer, Heidelberg (2005). doi:10.1007/11590156_17

[KS90] Kanellakis, P.C., Smolka, S.A.: CCS expressions, finite state processes, and three problems of equivalence. Inf. Comput. **86**(1), 43–68 (1990)

[KS13a] Křetínský, J., Sickert, S.: MoTraS: a tool for modal transition systems and their extensions. In: Hung, D., Ogawa, M. (eds.) ATVA 2013. LNCS, vol. 8172, pp. 487–491. Springer, Cham (2013). doi:10.1007/978-3-319-02444-8_41

[KS13b] Křetínský, J., Sickert, S.: On refinements of Boolean and parametric modal transition systems. In: Liu, Z., Woodcock, J., Zhu, H. (eds.) ICTAC 2013. LNCS, vol. 8049, pp. 213–230. Springer, Heidelberg (2013). doi:10.1007/978-3-642-39718-9_13

[KS13c] Křetínský, J., Sickert, S.: On refinements of Boolean and parametric modal transition systems. Technical report abs/1304.5278, arXiv.org (2013)

[Lar89] Larsen, K.G.: Modal specifications. In: Sifakis, J. (ed.) CAV 1989. LNCS, vol. 407, pp. 232–246. Springer, Heidelberg (1990). doi:10.1007/3-540-52148-8_19

[Lar90] Guldstrand Larsen, K.: Ideal specification formalism = expressivity + compositionality + decidability + testability +. In: Baeten, J.C.M., Klop, J.W. (eds.) CONCUR 1990. LNCS, vol. 458, pp. 33–56. Springer, Heidelberg (1990). doi:10.1007/BFb0039050

[LIC04] 19th IEEE Symposium on Logic in Computer Science (LICS 2004), 14–17 July 2004, Turku, Finland, Proceedings. IEEE Computer Society (2004)

[LL12] Larsen, K.G., Legay, A.: Quantitative modal transition systems. In: Martí-Oliet, N., Palomino, M. (eds.) WADT 2012. LNCS, vol. 7841, pp. 50–58. Springer, Heidelberg (2013). doi:10.1007/978-3-642-37635-1_3

[LM09] Leucker, M., Morgan, C. (eds.): ICTAC 2009. LNCS, vol. 5684. Springer, Heidelberg (2009). doi:10.1007/978-3-642-03466-4

[LM15] Lanese, I., Madelaine, E. (eds.): FACS 2014. LNCS, vol. 8997. Springer, Cham (2015). doi:10.1007/978-3-319-15317-9

[LML15] Luthmann, L., Mennicke, S., Lochau, M.: Towards an I/O conformance testing theory for software product lines based on modal interface automata. In: Atlee, J.M., Gnesi, S. (eds.) Proceedings 6th Workshop on Formal Methods and Analysis in SPL Engineering, FMSPLE@ETAPS 2015, London, UK, 11 April 2015. EPTCS, vol. 182, pp. 1–13 (2015)

[LNW07a] Larsen, K.G., Nyman, U., Wąsowski, A.: Modal I/O automata for interface and product line theories. In: Nicola, R. (ed.) ESOP 2007. LNCS, vol. 4421, pp. 64–79. Springer, Heidelberg (2007). doi:10.1007/978-3-540-71316-6_6

[LNW07b] Larsen, K.G., Nyman, U., Wąsowski, A.: On modal refinement and consistency. In: Caires, L., Vasconcelos, V.T. (eds.) CONCUR 2007. LNCS, vol. 4703, pp. 105–119. Springer, Heidelberg (2007). doi:10.1007/978-3-540-74407-8_8

[LSW95] Larsen, K.G., Steffen, B., Weise, C.: Fischer's protocol revisited: a simple proof using modal constraints. In: Alur, R., Henzinger, T.A., Sontag, E.D. (eds.) HS 1995. LNCS, vol. 1066, pp. 604–615. Springer, Heidelberg (1996). doi:10.1007/BFb0020979

[LT88] Larsen, K.G., Thomsen, B.: A modal process logic. In: LICS, pp. 203–210. IEEE Computer Society (1988)

[LV13] Lüttgen, G., Vogler, W.: Modal interface automata. Log. Methods Comput. Sci. **9**(3) (2013)

[LVF15] Lüttgen, G., Vogler, W., Fendrich, S.: Richer interface automata with optimistic and pessimistic compatibility. Acta Inf. **52**(4–5), 305–336 (2015)

[LX90] Larsen, K.G., Xinxin, L.: Equation solving using modal transition systems. In: LICS, pp. 108–117. IEEE Computer Society (1990)

[Man13] Manta, A.: Implementation of algorithms for modal transition systems with durations. Bachelor's thesis, Technische Universität München (2013)

[May00] Mayr, R.: Process rewrite systems. Inf. Comput. **156**(1–2), 264–286 (2000)

[Møl13] Møller, M.H.: Modal and component-based system specifications. Ph.D. thesis, Department of Computer Science, Aalborg University (2013)

[MTS] Motras. http://www7.in.tum.de/kretinsk/motras.html

[Nam03] Namjoshi, K.S.: Abstraction for branching time properties. In: Hunt, W.A., Somenzi, F. (eds.) CAV 2003. LNCS, vol. 2725, pp. 288–300. Springer, Heidelberg (2003). doi:10.1007/978-3-540-45069-6_29

[NNN08] Nanz, S., Nielson, F., Riis Nielson, H.: Modal abstractions of concurrent behaviour. In: Alpuente, M., Vidal, G. (eds.) SAS 2008. LNCS, vol. 5079, pp. 159–173. Springer, Heidelberg (2008). doi:10.1007/978-3-540-69166-2_11

[Nym08] Nyman, U.: Modal transition systems as the basis for interface theories and product lines. Ph.D. thesis, Aalborg Universitet (2008)

[Pnu77] Pnueli, A. The temporal logic of programs. In: FOCS, pp. 46–57. IEEE Computer Society (1977)

[PP06] Piterman, N., Pnueli, A.: Faster solutions of Rabin and Streett games. In: LICS, pp. 275–284. IEEE Computer Society (2006)

[PR89] Pnueli, A., Rosner, R.: On the synthesis of a reactive module. In: POPL, pp. 179–190. ACM Press (1989)

[PT87] Paige, R., Tarjan, R.E.: Three partition refinement algorithms. SIAM J. Comput. **16**(6), 973–989 (1987)

[QS82] Queille, J.P., Sifakis, J.: Specification and verification of concurrent systems in CESAR. In: Dezani-Ciancaglini, M., Montanari, U. (eds.) Programming 1982. LNCS, vol. 137, pp. 337–351. Springer, Heidelberg (1982). doi:10.1007/3-540-11494-7_22

[Rac07] Raclet, J.-B.: Quotient de spécifications pour la réutilisation de composants. Ph.D. thesis, Université de Rennes I (2007). (In French)

[Rac08] Raclet, J.-B.: Residual for component specifications. Electr. Notes Theor. Comput. Sci. **215**, 93–110 (2008)

[RBB+09a] Raclet, J.-B., Badouel, E., Benveniste, A., Caillaud, B., Legay, A., Passerone, R.: Modal interfaces: unifying interface automata and modal specifications. In:Chakraborty, S., Halbwachs, N., (eds.) EMSOFT, pp. 87–96. ACM (2009)

[RBB+09b] Raclet, J.-B., Badouel, E., Benveniste, A., Caillaud, B., Passerone, R.: Why are modalities good for interface theories? In: ACSD, pp. 119–127. IEEE Computer Society (2009)

[RBB+11] Raclet, J.-B., Badouel, E., Benveniste, A., Caillaud, B., Legay, A., Passerone, R.: A modal interface theory for component-based design. Fundam. Inform. **108**(1–2), 119–149 (2011)

[RD12] Roychoudhury, A., D'Souza, M. (eds.): ICTAC 2012. LNCS, vol. 7521. Springer, Heidelberg (2012). doi:10.1007/978-3-642-32943-2

[SBUK13] Sibay, G.E., Braberman, V.A., Uchitel, S., Kramer, J.: Synthesizing modal transition systems from triggered scenarios. IEEE Trans. Softw. Eng. **39**(7), 975–1001 (2013)

[Sch16] Schlachter, U.: Bounded petri net synthesis from modal transition systems is undecidable. In: Desharnais, J., Jagadeesan, R. (eds.), 27th International Conference on Concurrency Theory, CONCUR 2016, Québec City, Canada, 23–26 August 2016. LIPIcs, vol. 59, pp. 15:1–15:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2016)

[SCU11] Sassolas, M., Chechik, M., Uchitel, S.: Exploring inconsistencies between modal transition systems. Softw. Syst. Model. **10**(1), 117–142 (2011)

[SG04] Shoham, S., Grumberg, O.: Monotonic abstraction-refinement for CTL. In: Jensen, K., Podelski, A. (eds.) TACAS 2004. LNCS, vol. 2988, pp. 546–560. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24730-2_40

[SH15] Siirtola, A., Heljanko, K.: Parametrised modal interface automata. ACM Trans. Embed. Comput. Syst. **14**(4), 65:1–65:25 (2015)

[SK14] Sharma, A., Katoen, J.-P.: Layered reduction for abstract probabilistic automata. In: 14th International Conference on Application of Concurrency to System Design, ACSD 2014, Tunis La Marsa, Tunisia, 23–27 June 2014, pp. 21–31. IEEE Computer Society (2014)

[SUBK12] Sibay, G.E., Uchitel, S., Braberman, V.A., Kramer, J.: Distribution of modal transition systems. In: Giannakopoulou, D., Méry, D. (eds.) [GM12], pp. 403–417

[tBDG+15] Beek, M.H., Damiani, F., Gnesi, S., Mazzanti, F., Paolini, L.: From featured transition systems to modal transition systems with variability constraints. In: Calinescu, R., Rumpe, B. (eds.) SEFM 2015. LNCS, vol. 9276, pp. 344–359. Springer, Cham (2015). doi:10.1007/978-3-319-22969-0_24

[tBFGM16] ter Beek, M.H., Fantechi, A., Gnesi, S., Mazzanti, F.: Modelling and analysing variability in product families: model checking of modal transition systems with variability constraints. J. Log. Algebr. Meth. Program. **85**(2), 287–315 (2016)

[UBC07] Uchitel, S., Brunet, G., Chechik, M.: Behaviour model synthesis from properties and scenarios. In: ICSE, pp. 34–43. IEEE Computer Society (2007)

[UBC09] Uchitel, S., Brunet, G., Chechik, M.: Synthesis of partial behavior models from properties and scenarios. IEEE Trans. Softw. Eng. **35**(3), 384–406 (2009)

[UC04] Uchitel, S., Chechik, M.: Merging partial behavioural models. In: Taylor, R.N., Dwyer, M.B. (eds.) SIGSOFT FSE, pp. 43–52. ACM (2004)

[VR14] Verdier, G., Raclet, J.-B.: Maccs: a tool for reachability by design. In: Lanese, I., Madelaine, E. (eds.) [LM15], pp. 191–197

[VR15] Verdier, G., Raclet, J.-B.: Quotient of acceptance specifications under reachability constraints. In: Dediu, A.-H., Formenti, E., Martín-Vide, C., Truthe, B. (eds.) LATA 2015. LNCS, vol. 8977, pp. 299–311. Springer, Cham (2015). doi:10.1007/978-3-319-15579-1_23

[Wal96] Walukiewicz, I.: Pushdown processes: games and model checking. In: Alur, R., Henzinger, T.A. (eds.) CAV 1996. LNCS, vol. 1102, pp. 62–74. Springer, Heidelberg (1996). doi:10.1007/3-540-61474-5_58

[WGC09] Wei, O., Gurfinkel, A., Chechik, M.: Mixed transition systems revisited. In: Jones, N.D., Müller-Olm, M. (eds.) [JMO09], pp. 349–365