A Framework for the Analysis of Formal Description Techniques for Timed Systems

Robert Eschbach, Thomas Deiß,
and Martin Kronenburg

{eschbach|deiss|kronburg}@informatik.uni-kl.de

05/1998

Sonderforschungsbereich 501

Universität Kaiserslautern
Erwin-Schrödinger Straße
D-67653 Kaiserslautern

A Framework for the Analysis of Formal Description Techniques for Timed Systems*

Robert Eschbach, Thomas Deiß, and Martin Kronenburg

University of Kaiserslautern, Erwin Schrödinger Straße,
D-67663 Kaiserslautern, Germany
{eschbach|deiss|kronburg}@informatik.uni-kl.de

Abstract. There are a large variety of quite different formal description techniques (FDTs) to describe the ongoing behavior of systems. In this paper a unifying framework for such FDTs is presented, which enables their comparison in a common setting. This framework is a specialization of the institutions of Goguen and Burstall to signatures and structures involving time. It is shown that the temporal logic MTL can be properly instantiated within this framework.

Keywords. formal description techniques, timed systems, reactive systems, formal analysis, MTL

1 Introduction

A large number of *formal description techniques* (*FDTs*) has been developed up to now as an aid in the development of systems. Their main advantages are that they allow more precise descriptions and to put the analysis of the descriptions on a solid mathematical foundation. To decide which FDT to use in a specific project it is necessary to analyze them for their individual strengths and weaknesses.

For reactive systems there are already several case studies deriving empirical data on FDTs, for example see [ABL96,ACJ+96]. From these and other case studies we collected a large set of criteria to investigate FDTs, cf. [DKZ97]. But when actually applying them to several FDTs, we found that these criteria are defined rather informally. This made it difficult to decide whether an FDT satisfies a criterion. To put the investigation of FDTs on a more solid basis we decided to consider it in a formal manner. The following three tasks have to be performed to establish an approach to the formal analysis of FDTs.

Characterization of the Description Techniques to be investigated
It has to be explained, which conditions allow a description technique to be considered a *formal* one. Since an FDT cannot describe arbitrary aspects of arbitrary classes of systems, this general notion of FDT has to be instantiated to the application area one is interested in. This means to specify precisely the general models which can be represented in descriptions written in an FDT. Thus, the result of the first task is a refinement of the general notion of FDT. It defines the class of description techniques that are interesting w.r.t. the specific application domain considered.

* This work was partially supported by the Deutsche Forschungsgemeinschaft, Sonderforschungsbereich 501, “Entwicklung großer Systeme mit generischen Methoden”.

Definition of Criteria
To investigate concrete FDTs belonging to this class of FDTs, criteria examining a certain aspect of them have to be defined precisely. This includes the definition, whether or to which degree a specific FDT satisfies a criterion. Consequently, the result of this second task is a list of criteria. Since there is a very broad range of criteria including aspects like the difficulty of learning an FDT, it cannot be expected, that all of them can be defined formally, even less, that this is done in a uniform way.

Application of Criteria
To apply the criteria to concrete FDTs, it is necessary at first to embed them in the result of the first task, i.e. in the definition of the class of FDTs one is interested in. Next, the criteria have to be applied to these FDTs in the context of the specific project for which they are intended to be used. The relevance of the criteria in this context must be given, such that the attributes of the FDTs can be combined in a useful way. As the result of this step one gets for each considered concrete FDT a list containing the attributes assigned to this FDT w.r.t. the applied criteria.

This paper is concerned with the first task, i.e. a characterization of those FDTs we are interested in. These are FDTs which can be used to express ongoing behavior of systems. The other tasks are beyond the scope of this paper and are currently investigated by us. In section 2 the general framework is presented, leading to a characterization of an *FDT for timed systems* (T-FDTs). In section 3 it is shown that this notion is defined in a suitable way by proving that the temporal logic MTL is actually a T-FDT. We finish this paper by pointing out related work and by making some concluding remarks.

2 FDTs for Timed Systems

We are interested in the *formal analysis* of FDTs for timed systems. Formal analysis is related to a certain domain of objects which has to be defined exactly. We are interested in description techniques which are *formal* as well as *suitable* for describing *timed systems*.

At first we introduce the notion of a *formal* description technique (FDT). In our opinion an FDT is a logical system, which provides a *formal syntax* and a *formal semantics*. The formal syntax is usually given by a formal language providing *sentences* which can be built. Sentences are used to describe systems. Formal semantics consists of *structures* and a relation between sentences and structures. Often first-order structures are used, i.e. collections of carrier sets with appropriate functions and relations defined over them. The relation between sentences and structures determines whether a sentence is valid in a structure or not. This relation is usually called the *satisfaction relation*. Thus a formal description technique can be considered as a description technique where the vocabulary is precisely defined and sentences are assigned a meaning based on structures and the satisfaction relation in a precise way.

These concepts are captured by the notions of *institutions*, cf. [GB92], or *abstract logics*, cf. [Bar85,CK90]. Formally, we consider an FDT to be an institution. An institution provides a category SIG of *signatures* which form the basic vocabulary to describe a system. Formal syntax and semantics are given by functors Sen and Str from the category SIG to sets of sentences and categories of structures over a signature, respectively. This high abstract level is necessary to embed quite different description techniques commonly considered as formal in our framework. Examples of such description techniques are various kinds of temporal logics. The definition of FDT is given in section 2.1. The basic definitions concerning structures are given in section 2.2.

To define the term *suitable* for describing *timed systems* we characterize the classes of structures which serve as models of such systems, cf. section 2.3. This is done by requiring a distinct subsignature to denote timing aspects and by considering only structures over these *timed signatures*. These structures are called *behavior models*. When assuming linear models of time, the behavior models correspond to the more common notion of *runs*. As derived structures we define *state models*, i.e. behavior models at a specific point of time, and *system models* as collections of behavior models over the same subsignature for denoting time. We consider an FDT to be suitable for timed systems if it provides these kinds of models. This will be made more precise in section 2.4.

2.1 FDTs

As already stated, an institution provides a category of signatures, functors specifying the sentences and structures, and satisfaction relations relating sentences and structures. The satisfaction relations are restricted by two conditions. The first one (coincidence) states, that the truth value of a sentence does not depend on the concrete signature. In [GB92] this is summed up by the slogan: *Truth is invariant under change of notation.* This means that when translating a sentence the structures satisfying the translated sentence can themselves be translated back to structures satisfying the original sentence. The second one (isomorphism) states, that truth of a sentence is invariant under isomorphism of the structures. This condition is taken from the definition of *abstract logics* [Bar85,CK90].

In the following the category SET denotes the category of all sets as objects and mappings between sets as morphisms. We assume the standard Zermelo-Fraenkel axioms (ZFC) for set theory. Taking categories themselves as objects and functors as morphisms, the category CAT is obtained. Strictly speaking, this is the category of all *small* categories to avoid paradoxes as in set theory. For a category $\mathbb{C}$ let $\mid \mathbb{C} \mid$ denote its class of objects. For the sake of simplicity we sometimes omit the bars and write $A \in \mathbb{C}$ for an object A in $\mathbb{C}$. Analogously, we write $f : A \to B \in \mathbb{C}$ for a morphism $f : A \to B$ of $\mathbb{C}$. For an introduction to category theory see [Lan71]. The definitions of some of the terms are presented in appendix A.

Definition 1 (FDT). *An* FDT $\mathcal{F}$ *consists of*

1. *a non-empty category* $\text{S}\text{IG}_{\mathcal{F}}$*, whose objects are called* signatures*,*
2. *a functor* $Sen_{\mathcal{F}} : \text{S}\text{IG}_{\mathcal{F}} \to \text{S}\text{ET}$*,*

3. a contravariant functor $Str_{\mathcal{F}} : \mathrm{SIG}_{\mathcal{F}} \to \mathrm{CAT}$[1]*,*
4. a family of relations $\models_{\mathcal{F},\Sigma} \subseteq |Str_{\mathcal{F}}(\Sigma)| \times Sen_{\mathcal{F}}(\Sigma)$[2] *called* satisfaction relations*,*

such that the following conditions hold for all $\mathfrak{A}, \mathfrak{B} \in Str_{\mathcal{F}}(\Sigma)$*,* $\mathfrak{A}' \in Str_{\mathcal{F}}(\Sigma')$*,* $\varphi \in Sen_{\mathcal{F}}(\Sigma)$*, and* $\sigma : \Sigma \to \Sigma' \in \mathrm{SIG}_{\mathcal{F}}$*:*

1. (Coincidence) $\mathfrak{A}' \models_{\mathcal{F},\Sigma'} Sen_{\mathcal{F}}(\sigma)(\varphi)$ *iff* $Str_{\mathcal{F}}(\sigma)(\mathfrak{A}') \models_{\mathcal{F},\Sigma} \varphi$*,*
2. (Isomorphism) $\mathfrak{A} \cong \mathfrak{B}$ *and* $\mathfrak{A} \models_{\mathcal{F},\Sigma} \varphi$ *implies* $\mathfrak{B} \models_{\mathcal{F},\Sigma} \varphi$*.*

Given a signature $\Sigma \in \mathrm{SIG}_{\mathcal{F}}$ the elements of the set $Sen_{\mathcal{F}}(\Sigma)$ are called *sentences* and the objects in $Str_{\mathcal{F}}(\Sigma)$ are called *structures*. Based on the satisfaction relation, the *model class* $Mod_{\mathcal{F}}(\Phi)$ of a set of sentences Φ and the *theory* $Th_{\mathcal{F}}(\mathfrak{K})$ of a class $\mathfrak{K}$ of structures can be defined as for first-order logic. See appendix B for detailed information.

2.2 First-Order Structures

In this work we consider signatures and structures as they are common in many-sorted first-order logic.

A *signature* $\Sigma = (S, F, P, D)$ consists of a finite set S of sorts, sets F and P of function and predicate symbols, and a set D of declarations $f : \overline{s} \to s$ and $p : \overline{s}$, describing the sorts of the arguments and the range of function and predicate symbols. A *signature morphism* $\sigma : \Sigma \to \Sigma'$ consists of three functions $\sigma = (\sigma_S, \sigma_F, \sigma_P)$, mapping sorts, function, and predicate symbols compatible with the declarations of Σ and Σ'. For the rest of this paper we denote by SIG the category of all signatures with signature morphisms as morphisms.

A *structure* $\mathfrak{A}$ over a signature Σ consists of a family $A = (A_s \mid s \in S)$ of non-empty carrier sets and a mapping $\_^{\mathfrak{A}}$ from the function and predicate symbols to functions and predicates on A compatible with the declarations of Σ. A *homomorphism* h between two structures $\mathfrak{A}, \mathfrak{B}$ over Σ is a family $h = (h_s \mid s \in S)$ of mappings $h_s : A_s \to B_s$ that is compatible with the functions and predicates. $\mathrm{STR}(\Sigma)$ denotes the category of all structures over Σ with homomorphisms as morphisms.

For a signature morphism $\sigma : \Sigma \to \Sigma'$ the structures over these signatures are related by the *forgetful functor* $\_|_\sigma : \mathrm{STR}(\Sigma') \to \mathrm{STR}(\Sigma)$. This functor assigns to a structure $\mathfrak{A}' \in \mathrm{STR}(\Sigma')$ a structure $\mathfrak{A}'|_\sigma = \mathfrak{A} \in \mathrm{STR}(\Sigma)$ such that the carrier sets are given by $A_s = A'_{\sigma(s)}$ for $s \in S$ and the functions and relations by $f^{\mathfrak{A}} = \sigma(f)^{\mathfrak{A}'}$ and $p^{\mathfrak{A}} = \sigma(p)^{\mathfrak{A}'}$. A homomorphism $h' : \mathfrak{A}' \to \mathfrak{B}'$ is mapped to a homomorphism $h'|_\sigma : \mathfrak{A}'|_\sigma \to \mathfrak{B}'|_\sigma$ in an obvious way.

In the following we denote by Str the contravariant functor $Str : \mathrm{SIG} \to \mathrm{CAT}$ which maps a signature Σ to $\mathrm{STR}(\Sigma)$ and a signature morphism σ to the forgetful functor $\_|\sigma$.

[1] Strictly speaking CAT needs to include 'large' categories. As in [GB92] we refer to the 'hierarchy of universes' discussed in [Lan71].
[2] Instead of $(\mathfrak{A}, \varphi) \in \models_{\mathcal{F},\Sigma}$ we write $\mathfrak{A} \models_{\mathcal{F},\Sigma} \varphi$.

2.3 Models for Timed Systems

In this section we define the *models* that characterize *timed systems*. We regard a system being a part of the real world, over which one can talk or which can be analyzed only by using models of it. No system can be classified as a timed one or as another kind of system by itself. Each classification of systems results from the models used to talk about them. Therefore, when using the term *timed systems* we mean systems for which we use models involving timing aspects.

Depending on the kind of models one is interested in, different FDTs are more or less suitable, since each FDT provides different means to describe specific models, for example data models, behavior models, or architectural models. In this paper we present a common framework for such FDTs that provide models representing the evolution of systems over time. This means, that timing aspects, as e.g. the point of time when inputs or outputs occur, or the distance in time between the occurrence of inputs or outputs, are of primary importance and should be expressible.

To characterize these kinds of models of a system we impose further restrictions on the structures used as models and on the signatures on which they are based. We require that all signatures have a distinguished subsignature to represent the time model used. At least this subsignature has to introduce a distinguished sort T representing the time domain and a relation symbol $<$ to denote a precedence between points of time. In addition, since all functions and predicates can depend on time, we require, that the declaration of each function and relation symbol contains the sort T.

Signatures satisfying these properties are called *timed signatures*. Together with signature morphisms which respect the distinguished subsignature they form a subcategory of SIG in an obvious way.

Definition 2 (Timed Signature). *A* timed signature $\Sigma = (S, F, P, D)$ *is a signature which satisfies the following conditions:*

1. *There exists a distinguished subsignature* $\Sigma_T \subseteq \Sigma$ *with* $\Sigma_T = (S_T, F_T, P_T, D_T)$ *called* time signature*. The complement* $\Sigma_D = (S_D, F_D, P_D, D_D)$ *with* $\Sigma_D = \Sigma \setminus \Sigma_T$ *is called the* defined enrichment *of* Σ_T*.*
2. *There exist a distinguished sort* $T \in S_T$ *and a distinguished relation symbol* $< \in P_T$ *with declaration* $<: T, T$*.*
3. *Each function symbol* $f \in F_D$ *and each relation symbol* $p \in P_D$ *has a declaration* $f : \overline{s}, T \to s$ *and* $p : \overline{s}, T$*, respectively, where* $\overline{s} \in S_D^*$ *and* $s \in S_D$*.*

Definition 3 (Timed Signature Morphism). *Given two timed signatures* Σ *and* Σ' *a signature morphism* $\sigma : \Sigma \to \Sigma'$ *is called a* timed signature morphism *if it satisfies*

1. $\sigma(T) = T'$, $\sigma(<) = <'$,
2. $\sigma|_{\Sigma_T} : \Sigma_T \to \Sigma'_T$ *is a signature morphism itself.*

The category of all timed signatures with timed signature morphisms as morphisms is denoted by TSIG*.*

Concrete FDTs often provide a set of unary function symbols representing variables evolving over time, so called *flexible variables*. All other symbols are used to denote functions and predicates which actually do not depend on time. They are called *rigid*. Note that it is quite usual that a signature Σ is mapped by the functor $Sen_{\mathcal{F}}$ to a set of sentences in which this time argument does not occur explicitly. As an example see the definition of $Sen_{\mathcal{F}}$ for the temporal logic MTL in section 3.

Behavior Models

As structures over timed signatures we consider those structures which interpret the time sort and the relation symbol on it as a partial ordering. Structures with this minimal requirement on the interpretation of time are called *behavior models*. Whether the carrier set of the time sort has further properties, as e.g. being a dense or discrete set, can be investigated separately by criteria. Note that one behavior model represents only one single evolution of the functions and predicates over time.

Definition 4 (Behavior Model). *Let $\Sigma \in$ TSIG be a timed signature. A* behavior model $\mathfrak{B}$ *over Σ is a structure over Σ where $(B_T, <^{\mathfrak{B}})$ is a partial ordering.*

The set B_T will usually be referred to as the set of all *time points* of $\mathfrak{B}$. $\mathfrak{B}|_{\Sigma_T}$ is called the *time model* of $\mathfrak{B}$. The behavior models over a timed signature Σ induce a full subcategory of the category $Str(\Sigma)$ of structures.

Definition 5 (Behavior Category). *The full subcategory with all behavior models over $\Sigma \in$ TSIG as objects is denoted by* BEH(Σ).

The categories of timed signatures TSIG and behavior models over a timed signature are related by a contravariant functor Beh : TSIG $\rightarrow$ CAT which assigns the category BEH(Σ) to the timed signature $\Sigma \in$ TSIG and the forgetful functor $\_|\sigma : Beh(\Sigma') \rightarrow Beh(\Sigma)$ to the morphism $\sigma : \Sigma \rightarrow \Sigma' \in$ TSIG. Note that due to the restrictions for timed signature morphisms a behavior model $\mathfrak{B}' \in Beh(\Sigma')$ is actually mapped to a behavior model $\mathfrak{B}'|\sigma \in Beh(\Sigma)$.

State Models

In addition to behavior models considering functions and predicates for all time points, we are also interested in their values at a specific point of time. To remove time from signatures and structures we use a so called *state reduct* functor which maps a timed signature Σ to a signature with the same sorts, function and predicate symbols as the defined enrichment of Σ, and an appropriately changed declaration. The resulting signature is called the *state signature* of Σ.

Definition 6 (State Reduct). *The* state reduct $\_{St}$: TSIG $\rightarrow$ SIG *is a functor from the category of timed signatures to the category of signatures. Let $\Sigma \in$ TSIG be a timed signature, then $\Sigma_{St} = (S_{St}, F_{St}, P_{St}, D_{St})$ is defined by*

1. $S_{St} := S_D$, $F_{St} := F_D$, $P_{St} := P_D$, *and*
2. $D_{St} := \{f : \overline{s} \rightarrow s \mid (f : \overline{s}, T \rightarrow s) \in D_D\} \cup \{p : \overline{s} \mid (p : \overline{s}, T) \in D_D\}$,

where $\Sigma_D = \Sigma \setminus \Sigma_T$ *is the* defined enrichment *of* Σ_T.
The signature morphism $\sigma_{St} : \Sigma_{St} \to \Sigma'_{St}$ *of* $\sigma : \Sigma \to \Sigma'$ *is given by the identities* $\sigma_{St}|_{F_{St} \uplus P_{St}} = \sigma|_{F_{St} \uplus P_{St}}$ *and* $\sigma_{St}|_{S_{St}} = \sigma|_{S_{St}}$.

It is easy to see that the class of all state signatures with the corresponding signature morphisms is a category coinciding with the category SIG. To stress the notion of timed signatures we keep on to speak of state signatures and denote them by SSIG. Note that the state signature of a timed signature can again be a timed signature. The structures associated with a state signature are derived from the behavior models by fixing the point of time. These structures are called *state models*.

Definition 7 (State Model). *Let* $\Sigma \in$ TSIG *be a timed signature,* $\mathfrak{B} \in Beh(\Sigma)$ *a behavior model over* Σ, *and* $t \in B_T$ *a time point. The* state model $\mathfrak{B}_{St,t} \in Str(\Sigma_{St})$ *of* $\mathfrak{B}$ *at the time point* t *is defined by*

1. $B_{St,t} := (B_s \mid s \in S_D)$,
2. $f^{\mathfrak{B}_{St,t}}(\overline{b}) := f^{\mathfrak{B}}(\overline{b}, t)$ *for each* $f : \overline{s} \to s \in \Sigma_{St}$ *and all* $\overline{b} \in B_{\overline{s}}$,
3. $p^{\mathfrak{B}_{St,t}}(\overline{b}) := p^{\mathfrak{B}}(\overline{b}, t)$ *for each* $p : \overline{s} \in \Sigma_{St}$ *and all* $\overline{b} \in B_{\overline{s}}$.

Definition 8 (State Category). *The category of all state models over* $\Sigma \in$ SSIG *as objects and homomorphisms as morphisms is denoted by* STATE(Σ).

The categories of state signatures and state models are related by a contravariant functor $State$: SSIG $\to$ CAT which assigns the category STATE(Σ) to the state signature $\Sigma \in$ SSIG and the forgetful functor $\_|\sigma : State(\Sigma') \to State(\Sigma)$ to the morphism $\sigma : \Sigma \to \Sigma'$.

System Models
A behavior model represents only a single evolution of a system over time. But, as a system can react on different sequences of inputs differently or even non-deterministically, the *overall* behavior of it is represented by a set of behavior models. We do not allow an arbitrary collection of behavior models to represent the overall behavior of a system. Instead we require the behavior models to have the same carrier sets and the same time model. Such a set of behavior models is called a *system model.* As for time models, the question whether system models have further properties as e.g. closure under stuttering equivalence, cf. [Lam94], can be investigated separately by criteria. Similarly, it can be examined under which conditions model classes are system models.

Definition 9 (System Model). *Let* $\Sigma \in$ TSIG *be a timed signature and* $\mathfrak{S} \subseteq |Beh(\Sigma)|$ *a non-empty set of behavior models.* $\mathfrak{S}$ *is called a* system model *over* Σ *if the following holds for all* $\mathfrak{B}, \mathfrak{B}' \in \mathfrak{S}$:

1. $B_s = B'_s$ *for all* $s \in S_D$,
2. $\mathfrak{B}|_{\Sigma_T} = \mathfrak{B}'|_{\Sigma_T}$.

The class of all system models over Σ *is denoted by* SYS(Σ).

There exist many interesting relations between system models which can be expressed by an appropriate definition of *morphisms* between system models. With these morphisms it has to be possible to define for example a suitable notion of an *embedding of a system model into another one* or of an *isomorphism between system models*. As a further example consider a morphism between system models $\mathfrak{S}, \mathfrak{S}' \in \textsc{Sys}(\Sigma)$ expressing $\mathfrak{S} \supseteq \mathfrak{S}'$. This corresponds to *decrease of non-determinism* or *property refinement* (without extending the signature), cf. [Bro96].

We propose to identify a system model $\mathfrak{S}$ with a special behavior model $\prod_S \mathfrak{S}$, the so called *system product* of $\mathfrak{S}$. A system product reflects the model-theoretic properties of a set of behavior models representing a system model $\mathfrak{S}$. Thus we identify a system model with its system product. This can be justified by a one-to-one correspondence between system models and their corresponding system products. This identification leads to a very natural notion of morphisms for system models, namely the homomorphisms between the corresponding system products. These homomorphisms are morphisms in the category $\textsc{Beh}(\Sigma)$ of behavior models over Σ.

Within a system model $\mathfrak{S} = \{\mathfrak{B}_i \mid i \in I\}$, where I denotes an index set, all behavior models have the same time model. The system product has the same time model as the elements of $\mathfrak{S}$. From this it follows easily that the system product is a behavior model, too. The carrier sets of all behavior models in $\mathfrak{S}$ are the same, i.e. these behavior models differ only in the evaluation of functions and relations of Σ_D at each point of time. To represent these evaluations in the system product we use an I-indexed direct product of the carrier sets occurring in $\mathfrak{S}$. For example, if there is a function $f : T \to s$ in the defined enrichment Σ_D, then the evaluation of f at the time point t is represented by $\prod(f^{\mathfrak{B}_i}(t) \mid i \in I)$ which is an element of the carrier set $\prod(B_{i,s} \mid i \in I)$. Thus the system product uses *the* time model of $\mathfrak{S}$ and defines functions and relations of Σ_D coordinatewise at each time point.

Definition 10 (System Product). *Let $\Sigma \in \textsc{TSig}$ be a timed signature, $I \in \textsc{Set}$ an index set, $\mathfrak{S} = \{\mathfrak{B}_i \mid i \in I\} \in \textsc{Sys}(\Sigma)$ a system model over Σ. The* system product *$\prod_S(\mathfrak{B}_i \mid i \in I) \in Beh(\Sigma)$ of $\mathfrak{S}$ is the behavior model $\mathfrak{A} \in Beh(\Sigma)$ defined by*

1. *$\mathfrak{A}|_{\Sigma_T} := \mathfrak{B}|_{\Sigma_T}$, where $\mathfrak{B} \in \mathfrak{S}$,*
2. *$A_s := \prod(B_{i,s} \mid i \in I)$ for all $s \in S_D$,*
3. *$f^{\mathfrak{A}}(\overline{\alpha}, t) := \prod(f^{\mathfrak{B}_i}(\overline{\alpha}(i), t) \mid i \in I)$ for all $f : \overline{s}, T \to s \in \Sigma_D$, $\overline{\alpha} \in A_{\overline{s}}$ and $t \in A_T$,*
4. *$p^{\mathfrak{A}}(\overline{\alpha}, t)$:iff $p^{\mathfrak{B}_i}(\overline{\alpha}(i), t)$ for all $i \in I$, for all $p : \overline{s}, T \in \Sigma_D$, $\overline{\alpha} \in A_{\overline{s}}$ and $t \in A_T$,*

where $\alpha \in A_s$ denotes a function $\alpha : I \to \biguplus\{B_{i,s} \mid i \in I\}$ with $\alpha(i) \in B_{i,s}$ for all $i \in I$.

For a set J with $J \cong I$ one can easily verify that $\prod_S(\mathfrak{B}_i \mid i \in I) \cong \prod_S(\mathfrak{B}_j \mid j \in J)$ holds. This implies, that the system product is uniquely determined up to isomorphism. Therefore we simply write $\prod_S \mathfrak{S}$ (or $\prod_S \mathfrak{B}_i$) instead of $\prod_S(\mathfrak{B}_i \mid i \in I)$. By defining appropriate projections it is possible to retrieve the corresponding system model from a given system product. Therefore we identify in the following a system model with its system product.

Definition 11 (System Homomorphism). *Let $\Sigma \in$ TSIG be a timed signature and $\mathfrak{S}, \mathfrak{S}' \in$ SYS(Σ) two system models. A* system homomorphism $h : \mathfrak{S} \to \mathfrak{S}'$ *is a homomorphism $h : \prod_S \mathfrak{S} \to \prod_S \mathfrak{S}'$ between the corresponding system products.*

With system homomorphisms as morphisms the class SYS(Σ) can be extended to a category, also denoted by SYS(Σ).

Definition 12 (System Category). SYS(Σ) *is the category with all system models over a timed signature $\Sigma \in$ TSIG as objects and system homomorphisms as morphisms.*

As for behavior models, the categories of timed signatures and system models are related by a contravariant functor $Sys :$ TSIG $\to$ CAT assigning the category SYS(Σ) to each timed signature $\Sigma \in$ TSIG and the forgetful functor $|\sigma : Sys(\Sigma') \to Sys(\Sigma)$ to each timed signature morphism $\sigma : \Sigma \to \Sigma'$. Lifting the forgetful functor $\_|\sigma$ to a functor $\_|\sigma : Sys(\Sigma') \to Sys(\Sigma)$ is justified by $\prod_S(\mathfrak{B}_i \mid i \in I)|\sigma \cong \prod_S(\mathfrak{B}_i|\sigma \mid i \in I)$.

2.4 FDTs for Timed Systems

The general notion of an FDT and the models presented previously, namely system, behavior, and state models, are combined now to what we call an *FDT for timed systems*, or *T-FDT* for short. This is done by specializing the general definition of an FDT. This specialization is achieved by demanding further conditions of $\mathrm{SIG}_{\mathcal{F}}$, $Str_{\mathcal{F}}$, and $\models_{\mathcal{F},\Sigma}$.

We require that all signatures of a T-FDT $\mathcal{F}$ are timed or state signatures. We do not insist on *all* possible timed signatures because usually a description technique supports only one particular time model and thus only those signatures allowing to express this time model have to be considered. To be able to distinguish between state and timed signatures we take their disjoint union realized by a coproduct of the categories of the respective signatures. Furthermore, for each state signature there has to be a corresponding timed signature from which it is derived. In this way we exclude T-FDTs with an empty category of timed signatures.

Concerning the functor $Str_{\mathcal{F}}$ we require for a T-FDT that $Str_{\mathcal{F}}$ is derived from the functor Str. Hence $Str_{\mathcal{F}}$ has to assign to each timed signature Σ a subcategory of $Beh(\Sigma)$, i.e. a category containing only behavior models with appropriate homomorphisms between them. Analogously, we require a category of state models for a given state signature. For signature morphisms the functor $Str_{\mathcal{F}}$ has to operate as Str, i.e. assigning the forgetful functor $|_\sigma$ to a signature morphism σ.

We do not require any condition on the functor $Sen_{\mathcal{F}}$. By this abstraction we allow the embedding of quite different FDTs in our framework for timed systems.

The satisfaction relation of a T-FDT has to respect the relationship between a system model as a set of behavior models and its system product, which is a single behavior model. In the category SYS(Σ) this set is identified with its system product, cf. section 2.3. Thus, system models can be seen as special behavior models, namely their corresponding system products. Each T-FDT has to respect this identification, i.e. the theory of a system model as a set of behavior models has to be the same as the theory of its corresponding system product. This is captured by the *system product condition.* We consider this a natural condition and expect that it is always possible to define $Str_{\mathcal{F}}$ such that this condition is satisfied.

Definition 13 (FDT for Timed Systems). *An* FDT for timed systems $\mathcal{F}$ *(T-FDT for short) is an FDT fulfilling*

1. (signature condition) $\mathrm{SIG}_{\mathcal{F}} = \mathrm{TSIG}_{\mathcal{F}} \amalg \mathrm{SSIG}_{\mathcal{F}}$, *where* $\mathrm{TSIG}_{\mathcal{F}} \subseteq \mathrm{TSIG}$, $\mathrm{SSIG}_{\mathcal{F}} \subseteq \mathrm{SSIG}$, *and* $\forall \Sigma \in \mathrm{SSIG}_{\mathcal{F}}\ \exists \Sigma' \in \mathrm{TSIG}_{\mathcal{F}}\ \Sigma = \Sigma'_{St}$,
2. (structure condition) $Str_{\mathcal{F}}(\Sigma) \subseteq \begin{cases} Beh(\Sigma) & \text{if } \Sigma \in \mathrm{TSIG}_{\mathcal{F}} \\ State(\Sigma) & \text{if } \Sigma \in \mathrm{SSIG}_{\mathcal{F}} \end{cases}$ *and* $Str_{\mathcal{F}}(\sigma) = Str(\sigma)$ *for all* $\sigma : \Sigma \to \Sigma' \in \mathrm{SIG}_{\mathcal{F}}$,
3. (system product condition) $\forall \Sigma \in \mathrm{TSIG}_{\mathcal{F}}$ *and each system model* $\{\mathfrak{B}_i \mid i \in I\} \subseteq |Str_{\mathcal{F}}(\Sigma)|$ *with* $\prod_S \mathfrak{B}_i \in |Str_{\mathcal{F}}(\Sigma)|$ *it holds that* $Th_{\mathcal{F}}(\{\mathfrak{B}_i \mid i \in I\}) = Th_{\mathcal{F}}(\prod_S \mathfrak{B}_i)$.

The structure condition guarantees that for a T-FDT $\mathcal{F}$ and a signature morphism $\sigma : \Sigma \to \Sigma' \in \mathrm{SIG}_{\mathcal{F}}$ always

$$\mathfrak{A}' \in Str_{\mathcal{F}}(\Sigma') \text{ implies } \mathfrak{A}'|_{\sigma} \in Str_{\mathcal{F}}(\Sigma)$$

holds. Note, since each T-FDT $\mathcal{F}$ is also an FDT the isomorphism and coincidence condition have to be satisfied by $\mathcal{F}$, too.

3 Example of a T-FDT: MTL

In this section we outline the proof that the description technique *Metric Temporal Logic* (MTL), see [Koy92], is actually a T-FDT. A complete and detailed proof can be found in [KDE98]. There we also show that the two temporal logics CTL* [Eme90] and MTL-$\int$ [LH95] are T-FDTs. Here we describe briefly what has to be done to show that an existing description technique is a T-FDT. We illustrate this process by performing and explaining the main steps in the case of MTL.

To show that a description technique $\mathcal{F}$ is a T-FDT it is necessary to instantiate the definition of a T-FDT according to $\mathcal{F}$. This means to provide definitions of the categories of signatures $\mathrm{SSIG}_{\mathcal{F}}$ and $\mathrm{TSIG}_{\mathcal{F}}$, the functors $Sen_{\mathcal{F}}$ and $Str_{\mathcal{F}}$, and the family of satisfaction relations $\models_{\mathcal{F},\Sigma}$. It has to be justified that this characterization as a T-FDT corresponds to the original definition of $\mathcal{F}$. Furthermore, it has to be proven that the conditions of Definition 1 for FDTs in general and the T-FDT specific conditions of Definition 13 are satisfied. All this is done for MTL in the following.

MTL is a propositional temporal logic introduced by Koymans in [Koy92]. After that many authors have used this notion to denote temporal logics using temporal operators with time bounds (see e.g. [AH90]). We follow mainly the original approach of Koymans, but from the large set of possible temporal operators we consider only the future operator $\mathcal{U}$ (*until*). Based on it the other future temporal operators can be defined in the usual way. Therefore this restriction is not a substantial one. Adding the past operator $\mathcal{S}$ (*since*) is conceptually not difficult, but would blow up the proofs.

3.1 Instantiation

We start the instantiation process with the category SIG_{MTL}, next the two functors Str_{MTL} and Sen_{MTL} are defined. Finally, we specify the family of satisfaction relations $\models_{MTL,\Sigma}$. Here we only give some brief explanations of the instantiations. More motivation is given in [KDE98].

Timed Signatures

We have to specify the two categories TSIG_{MTL} and SSIG_{MTL}. Since in MTL no state signatures are needed SSIG_{MTL} is the empty category. Hence we have $\text{SIG}_{MTL} = \text{TSIG}_{MTL}$.

To define the time model of an MTL behavior model the time signature Σ_T of a MTL timed signature Σ has to provide in addition to T and $<$ a further sort Δ and several function symbols. These are needed to construct a specific metric space where metrical aspects of time can be considered.

Since MTL is a *propositional* temporal logic the defined enrichment contains the two function symbols *true* and *false* and a set PL of boolean variables that can change over time.

Furthermore, we need a set F_{Ind} of function symbols that are used to construct the terms that can occur as indices of the until operator $\mathcal{U}$.

Definition 14 (MTL Timed Signatures). *A timed signature $\Sigma = (S, F, P, D)$ is an* MTL timed signature *if it fulfills the following conditions:*

1. *the time signature $\Sigma_T = (S_T, F_T, P_T, D_T)$ is given by*
   - $S_T = \{T, \Delta\}$
   - $F_T = \{d, +, 0\}$
   - $P_T = \{<\}$
   - $D_T = \{d : T, T \to \Delta, + : \Delta, \Delta \to \Delta, 0 :\to \Delta, <: T, T\}$
2. *the defined enrichment $\Sigma_D = (S_D, F_D, P_D, D_D)$ is given by*
   - $S_D = \{B\}$
   - $F_D = \{true, false\} \uplus PL \uplus F_{Ind}$ *with* $PL = \{p_1, \ldots, p_n\}$, $n \in \mathbb{N}$, *and* $F_{Ind} = \{f_1, \ldots, f_m\}, m \in \mathbb{N}$
   - $P_D = \emptyset$
   - $D_D = \{p : T \to B \mid p \in F_D \setminus F_{Ind}\} \uplus D_{Ind}$ *with* $D_{Ind} = \{f : \Delta^k, T \to \Delta \mid f \in F_{Ind}, k \in \mathbb{N}\}$

The category with MTL timed signatures as objects and the corresponding signature morphisms $\sigma : \Sigma \to \Sigma'$ with $\sigma(false) = false$ and $\sigma(true) = true$ as morphisms is the category TSIG_{MTL}.

For the rest of this section Σ and Σ' always denote MTL timed signatures and $\sigma : \Sigma \to \Sigma'$ a morphism of TSIG_{MTL}.

Behavior Models
The functor Str_{MTL} is defined such that $Str_{MTL}(\Sigma)$ contains all the behavior models over Σ having a time model that fulfills the conditions required for MTL. For a motivation of these conditions see [Koy92] or [KDE98].

A function assigned to a function symbol of the defined enrichment is called *rigid* if it is invariant under time, i.e. let $f \in \Sigma_D$ be a function symbol with the declaration $f : S_1, \ldots S_n, T \to S$ and $\mathfrak{A} \in Beh(\Sigma)$ a behavior model over Σ then $f^{\mathfrak{A}}$ is called *rigid* iff $\forall t, t' \in A_T, \forall \bar{a} \in A_{s_1} \times \ldots \times A_{s_n} : f^{\mathfrak{A}}(\bar{a}, t) = f^{\mathfrak{A}}(\bar{a}, t')$.

Definition 15 (MTL Behavior Model). *A behavior model* $\mathfrak{A} = (A, \_^{\mathfrak{A}})$*,* $\mathfrak{A} \in Beh(\Sigma)$*, is called an* MTL behavior model *over* Σ *if it fulfills the following conditions:*

1. $A_B = \{TRUE, FALSE\}$
2. $\forall t \in A_T : (true^{\mathfrak{A}}(t) = TRUE$ *and* $false^{\mathfrak{A}}(t) = FALSE)$
3. $\forall f \in F_{Ind} : f^{\mathfrak{A}}$ *is rigid.*
4. $(A_T, <^{\mathfrak{A}})$ *is a total ordering.*
5. $d^{\mathfrak{A}}$ *is surjective.*
6. $\forall t, t', t'' \in A_T$:
   - $d^{\mathfrak{A}}(t, t') = 0^{\mathfrak{A}}$ *iff* $t = t'$
   - $d^{\mathfrak{A}}(t, t') = d^{\mathfrak{A}}(t', t)$
   - *if* $t <^{\mathfrak{A}} t' <^{\mathfrak{A}} t''$ *then* $d^{\mathfrak{A}}(t, t'') = d^{\mathfrak{A}}(t, t') +^{\mathfrak{A}} d^{\mathfrak{A}}(t', t'')$
7. $\forall \delta, \delta', \delta'' \in A_\Delta$:
   - $+^{\mathfrak{A}}(\delta, \delta') = +^{\mathfrak{A}}(\delta', \delta)$
   - $+^{\mathfrak{A}}(\delta, +^{\mathfrak{A}}(\delta', \delta'')) = +^{\mathfrak{A}}(+^{\mathfrak{A}}(\delta, \delta'), \delta'')$
   - $+^{\mathfrak{A}}(\delta, 0^{\mathfrak{A}}) = \delta$
   - $+^{\mathfrak{A}}(\delta, \delta') = +^{\mathfrak{A}}(\delta, \delta'') \leadsto \delta' = \delta''$
   - $+^{\mathfrak{A}}(\delta, \delta') = 0^{\mathfrak{A}} \leadsto \delta = 0^{\mathfrak{A}}$ *and* $\delta' = 0^{\mathfrak{A}}$
   - $\exists \tilde{\delta} \in A_\Delta : +^{\mathfrak{A}}(\delta', \tilde{\delta}) = \delta$ *or* $+^{\mathfrak{A}}(\delta, \tilde{\delta}) = \delta'$

The category with MTL behavior models over an MTL timed signature Σ *as objects and the corresponding homomorphisms as morphisms is denoted by* $\text{B}\small{\text{EH}}_{MTL}(\Sigma)$.

In the following we use the relations $\leq^{\mathfrak{A}}, >^{\mathfrak{A}}$, and $\geq^{\mathfrak{A}}$ which are derived from $<^{\mathfrak{A}}$ as usual.

Furthermore we use the relations $\preceq, \prec, \succeq \subseteq A_\Delta \times A_\Delta$. $\preceq$ is defined by: $\delta \preceq \delta'$ iff $\exists \delta'' : \delta' = \delta +^{\mathfrak{A}} \delta''$. The relations $\prec$ and $\succeq$ are derived from $\preceq$ as usual.

Definition 16 (Functor Str_{MTL}**).** *The functor* Str_{MTL}: $\text{SIG}_{MTL} \to \text{CAT}$ *is given by* $Str_{MTL}(\Sigma) := \text{BEH}_{MTL}(\Sigma)$ *and* $Str_{MTL}(\sigma) := Str(\sigma)$.

That this is actually a well defined functor is shown by the proof of the structure condition, see page 15. The problem is the assignment $Str_{MTL} := Str(\sigma)$. By this we assign to a signature morphism $\sigma : \Sigma \to \Sigma'$ the forgetful functor $|_\sigma : Str(\Sigma') \to Str(\Sigma)$. To show that this is well defined we have to proof that given a MTL behavior model $\mathfrak{A}'$ over Σ', i.e. $\mathfrak{A}' \in Str_{MTL}(\Sigma') = \text{BEH}_{MTL}(\Sigma')$, then also $\mathfrak{A} := \mathfrak{A}'|_\sigma$ is a MTL behavior model over Σ, i.e. $\mathfrak{A} \in Str_{MTL}(\Sigma) = \text{BEH}_{MTL}(\Sigma')$. Otherwise, we can not assign $|_\sigma$ to σ by the functor Str_{MTL}.

If nothing else is stated, in the following $\mathfrak{A}$ is always an MTL behavior model over Σ.

Sentences
The definition of Sen_{MTL} consists of four steps. At first we introduce a functor Ind_{MTL} specifying so called *index terms*. $Ind_{MTL}(\Sigma)$ denotes the set of terms that can be used as indices of the operator $\mathcal{U}$. These terms are not considered as sentences on their own but are necessary to construct the set of the so called *connector symbols* $\Gamma_{MTL,\Sigma}$ associated with a signature. These connector symbols are needed to construct more complex sentences out of atomic ones. Hence, we define a functor $AtSen_{MTL}$ specifying the set of atomic sentences we associate with an MTL timed signature. Finally, the functor Sen_{MTL} is defined based on $\Gamma_{MTL,\Sigma}$ and $AtSen_{MTL}$.

Since all function symbols $f \in F_{Ind}$ are required to be rigid we can omit the time argument.

Definition 17 (Index Terms). *The functor* $Ind_{MTL} : \mathrm{SIG}_{MTL} \to \mathrm{SET}$ *is given by:* $Ind_{MTL}(\Sigma)$ *is the least set fulfilling the following conditions*

1. $f \in Ind_{MTL}(\Sigma)$ *for all* $f \in F_{Ind}$ *with* $f : T \to \Delta \in D_{Ind}$
2. *if* $t_1, \dots, t_k \in Ind_{MTL}(\Sigma)$, $k \in \mathbb{N}$, *and* $f \in F_{Ind}$ *with* $f : \Delta^k, T \to \Delta \in D_{Ind}$ *then* $f(t_1, \dots, t_k) \in Ind_{MTL}(\Sigma)$

$c \in Ind_{MTL}(\Sigma)$ *is called an* index term *over* Σ.

On the morphisms of SIG_{MTL} *we define* Ind_{MTL} *as follows. For* $t \in Ind_{MTL}(\Sigma)$ *we define* $Ind_{MTL}(\sigma) : Ind_{MTL}(\Sigma) \to Ind_{MTL}(\Sigma')$ *by:*

1. *if* $t \equiv f$ *with* $f : T \to \Delta \in D_{Ind}$ *then* $Ind_{MTL}(\sigma)(f) := \sigma(f)$
2. *if* $t \equiv f(t_1, \dots t_k), k \geq 1$ *then*
   $Ind_{MTL}(\sigma)(f(t_1, \dots, t_k)) := \sigma(f)(Ind_{MTL}(\sigma)(t_1), \dots, Ind_{MTL}(\sigma)(t_k))$

Usually index terms are not very complex, i.e. in most cases they are just constants, e.g. 4 or 11, or 'small' terms like $t_1 + t_2$.

The set of the connector symbols consists of the implication symbol $\to$ for the propositional basis and the temporal operator $\mathcal{U}$ extended with index terms.

Definition 18 (Connector Symbols). *The alphabet* $\Gamma_{MTL,\Sigma}$ *of* connector symbols *over* Σ *is given by* $\Gamma_{MTL,\Sigma} := \{\to\} \cup \{\mathcal{U}_{\sim c} | \sim \in \{\prec, \succ, =\}, c \in Ind_{MTL}(\Sigma)\}$.

For a given MTL timed signature the set of the atomic sentences corresponds with the function symbols of the propositional part of the defined enrichment.

Definition 19 (Atomic Sentences). *The functor* $AtSen_{MTL} : \mathrm{SIG}_{MTL} \to \mathrm{SET}$ *is given by* $AtSen_{MTL}(\Sigma) := F_D \setminus F_{Ind}$.

On the morphisms of SIG_{MTL} *we define* $AtSen_{MTL}$ *as follows. For* $\varphi \in AtSen_{MTL}(\Sigma)$ *we define* $AtSen_{MTL}(\sigma) : AtSen_{MTL}(\Sigma) \to AtSen_{MTL}(\Sigma')$ *by* $AtSen_{MTL}(\sigma)(\varphi) := \sigma(\varphi)$.

The functor Sen_{MTL} is now the natural extension of $AtSen_{MTL}$ using the set of connector symbols.

Definition 20 (Sentences). *The functor* $Sen_{MTL} : \mathrm{SIG}_{MTL} \to \mathrm{SET}$ *is given by:* $Sen_{MTL}(\Sigma)$ *is the least set fulfilling the following conditions*

1. $AtSen_{MTL}(\Sigma) \subseteq Sen_{MTL}(\Sigma)$,
2. *if* $\varphi, \psi \in Sen_{MTL}(\Sigma)$ *and* $\oplus \in \Gamma_{MTL,\Sigma}$ *then* $\varphi \oplus \psi \in Sen_{MTL}(\Sigma)$.

On the morphisms of SIG_{MTL} *we define* Sen_{MTL} *as follows. For* $\varphi \in Sen_{MTL}(\Sigma)$ *we define* $Sen_{MTL}(\sigma) : Sen_{MTL}(\Sigma) \to Sen_{MTL}(\Sigma')$ *by*

1. *if* $\varphi \in AtSen_{MTL}(\Sigma)$ *then* $Sen_{MTL}(\sigma)(\varphi) := AtSen_{MTL}(\sigma)(\varphi)$,
2. *if* $\varphi \equiv \psi_1 \to \psi_2$ *then* $Sen_{MTL}(\sigma)(\psi_1 \to \psi_2) = Sen_{MTL}(\sigma)(\psi_1) \to Sen_{MTL}(\sigma)(\psi_2)$,
3. *if* $\varphi \equiv \psi_1 \mathcal{U}_{\sim c} \psi_2$, $\sim \in \{\prec, \succ, =\}, c \in Ind_{MTL}(\Sigma)$, *then* $Sen_{MTL}(\sigma)(\psi_1 \mathcal{U}_{\sim c} \psi_2) = Sen_{MTL}(\sigma)(\psi_1) \mathcal{U}_{\sim Ind_{MTL}(\sigma)(c)} Sen_{MTL}(\sigma)(\psi_2)$.

Satisfaction Relations
To define the family of satisfaction relations it is at first necessary to specify the meaning of the index terms w.r.t. a given MTL behavior model $\mathfrak{A}$.

Definition 21 (Interpretation $\mathfrak{I}_\mathfrak{A}$). *An MTL behavior model* $\mathfrak{A}$ *over* Σ *induces an* interpretation $\mathfrak{I}_\mathfrak{A} : Ind_{MTL}(\Sigma) \to A_\Delta$ *which is defined as follows. Let* $t \in A_T$ *be an arbitrary time point.*

1. *if* $f \in Ind_{MTL}(\Sigma)$ *with* $f : T \to \Delta \in D_{Ind}$ *then* $\mathfrak{I}_\mathfrak{A}(f) := f^\mathfrak{A}(t)$
2. *if* $f(t_1, \ldots, t_k) \in Ind_{MTL}(\Sigma)$, $k \geq 1$ *then* $\mathfrak{I}_\mathfrak{A}(f(t_1, \ldots, t_k)) := f^\mathfrak{A}(\mathfrak{I}_\mathfrak{A}(t_1), \ldots, \mathfrak{I}_\mathfrak{A}(t_k), t)$

Since $f^\mathfrak{A}$ is a rigid function for all $f \in F_{Ind}$, i.e. $f^\mathfrak{A}$ is independent of the considered time point t, $\mathfrak{I}_\mathfrak{A}$ is well defined.

The following definition of the satisfaction relation $\models_{MTL,\Sigma}$ is based on an additional relation $\Vert\!\models_{MTL,\Sigma}$. This relation specifies the validity of a sentence w.r.t. a specific time point of an MTL behavior model. Note that according to [Koy92] we do not use the anchored version of the semantics, but the floating one, i.e. we require a sentence to be valid at all time points. We use the infix notation, i.e. $(\mathfrak{A}, t, \varphi) \in \Vert\!\models_{MTL,\Sigma}$ iff $(\mathfrak{A}, t) \Vert\!\models_{MTL,\Sigma} \varphi$.

Definition 22 (Satisfaction Relation). *Let* $t \in A_T$ *be an arbitrary time point. The relation* $\Vert\!\models_{MTL,\Sigma} \subseteq |Str_{MTL}(\Sigma)| \times A_T \times Sen_{MTL}(\Sigma)$ *is defined by*

1. $\forall p \in F_D : (\mathfrak{A}, t) \Vert\!\models_{MTL,\Sigma} p$ *iff* $p^\mathfrak{A}(t) = TRUE$
2. $(\mathfrak{A}, t) \Vert\!\models_{MTL,\Sigma} \varphi \to \psi$ *iff* $(\mathfrak{A}, t) \Vert\!\models_{MTL,\Sigma} \psi$ *or* $(\mathfrak{A}, t) \Vert\!\not\models_{MTL,\Sigma} \varphi$
3. $(\mathfrak{A}, t) \Vert\!\models_{MTL,\Sigma} \varphi \mathcal{U}_{\sim c} \psi$ *iff* $\exists t', t' \geq^\mathfrak{A} t, d^\mathfrak{A}(t', t) \sim \mathfrak{I}_\mathfrak{A}(c)$: $(\mathfrak{A}, t') \Vert\!\models_{MTL,\Sigma} \psi$ *and* $\forall t'', t \leq^\mathfrak{A} t'' <^\mathfrak{A} t' : (\mathfrak{A}, t'') \Vert\!\models_{MTL,\Sigma} \varphi$, $\sim \in \{\prec, \succ, =\}$

$\models_{MTL,\Sigma} \subseteq |Str_{MTL}(\Sigma)| \times Sen_{MTL}(\Sigma)$ *is defined as:*

$$\mathfrak{A} \models_{MTL,\Sigma} \varphi \textit{ iff } \forall t \in A_T : (\mathfrak{A}, t) \Vert\!\models_{MTL,\Sigma} \varphi$$

3.2 Proofs of the Conditions

Most of the proofs of the T-FDT specific conditions are straightforward. The signature condition and the structure condition concerning the object part of Str_{MTL} are direct consequences of the definitions of $\textsc{Sig}_{MTL}$ and $Str_{MTL}(\Sigma)$ because in both cases we consider a subcategory of $\textsc{TSig}$ and $Beh(\Sigma)$, respectively. Moreover, the system product condition is trivially fulfilled, since there are no system products in $\textsc{Beh}_{MTL}(\Sigma)$. The proof of the morphism part of the structure condition as well as the proofs of the coincidence and isomorphism condition are the most difficult tasks. In the following we only present the main ideas of the proofs. The detailed proofs are presented in [KDE98].

Structure Condition The structure condition determines several properties of the functor Str_{MTL}. Concerning the object part of Str_{MTL} the condition $Str_{MTL}(\Sigma) \subseteq Beh(\Sigma)$ is a direct consequence of the definition of Str_{MTL}. That Str_{MTL} fulfills additionally the condition for the morphism part, i.e. $\forall \sigma \in \textsc{Sig}_{MTL} : Str_{MTL}(\sigma) = Str(\sigma)$, is however not obvious.

To prove that Str_{MTL} has this property we have to show that given a structure $\mathfrak{A}' \in Str_{MTL}(\Sigma')$ and a signature morphism $\sigma : \Sigma \to \Sigma'$, then the structure $\mathfrak{A} = \mathfrak{A}'|_\sigma$ is also a MTL behavior model over Σ, i.e. $\mathfrak{A} \in Str_{MTL}(\Sigma)$. If this property is fulfilled by Str_{MTL} we say that Str_{MTL} is closed under signature morphisms. If the functor Str_{MTL} is not closed under signature morphisms it cannot behave as Str.

To prove this closure property we need that the time signature and the propositional part of the defined enrichment of a MTL timed signature are invariant under signature morphisms. For each signature morphism $\sigma : \Sigma \to \Sigma' \in \textsc{Sig}_{MTL}$ it holds (we use the primed versions of the symbols of Definition 14 to denote the elements of Σ'):

$$\begin{array}{ll} 1. & \sigma(<) = <' \\ 2. & \sigma(T) = T', \sigma(\Delta) = \Delta', \sigma(B) = B' \\ 3. & \sigma(d) = d', \sigma(0) = 0', \sigma(+) = +' \\ 4. & \sigma(true) = true, \sigma(\mathit{false}) = \mathit{false}, \{\sigma(p) \mid p \in F_D \setminus F_{Ind}\} \subseteq F'_D \setminus F'_{Ind} \end{array} \tag{1}$$

This fact guarantees a kind of syntactical invariance which can easily be lifted to a semantical invariance resulting in the following structure closure

$$\mathfrak{A} \in Str_{MTL}(\Sigma) \text{ iff } \mathfrak{A}' \in Str_{MTL}(\Sigma') \tag{2}$$

Note that this fact guarantees that Str_{MTL} as given in Definition 16 is well defined. From this the structure condition for morphisms follows directly.

Coincidence Condition To prove the coincidence condition for MTL two lemmata are needed. The first lemma states the corresponding coincidence condition w.r.t. the index terms, namely

$$\mathfrak{I}_{\mathfrak{A}}(c) = \mathfrak{I}_{\mathfrak{A}'}(Ind_{MTL}(\sigma)(c)) \text{ for all } c \in Ind_{MTL}(\Sigma) \tag{3}$$

This lemma is proven by induction over the structure of $c \in Ind_{MTL}(\Sigma)$. It is needed to show the second lemma which guarantees the corresponding version of the coincidence condition for the satisfaction relation $\|\!\!=_{MTL,\Sigma}$:

$$\begin{aligned}&(\mathfrak{A}', t) \|\!\!=_{MTL,\Sigma'} Sen_{MTL}(\sigma)(\varphi) \\ &\text{ iff } (Str_{MTL}(\sigma)(\mathfrak{A}'), t) \|\!\!=_{MTL,\Sigma} \varphi \text{ for all } \varphi \in Sen_{MTL}(\Sigma)\end{aligned} \tag{4}$$

The proof is done by induction over the structure of the sentence $\varphi \in Sen_{MTL}(\Sigma)$ and by using the structure condition.

Based on this lemma it is now quite easy to show that the coincidence condition is satisfied by MTL.

Isomorphism Condition The proof of the isomorphism condition is split into four parts. In the following let $\mathfrak{B}$ be a structure over Σ which is isomorphic to $\mathfrak{A}$, i.e. $\mathfrak{A} \cong \mathfrak{B}$ and $h = (h_s | s \in S)$ an isomorphism between $\mathfrak{A}$ and $\mathfrak{B}$.

At first we show similarly to the proof of the structure condition a closure property for isomorphic behavior models: If $\mathfrak{A} \in Str_{MTL}(\Sigma)$ is a MTL behavior model over Σ then $\mathfrak{B}$ is a MTL behavior model, too.

Next we show that the isomorphism condition is valid for the propositional part of a MTL behavior model:

$$\begin{array}{ll} 1. & h_B(FALSE) = FALSE \text{ and } h_B(TRUE) = TRUE \\ 2. & \forall t \in A_T \text{ and } \forall p \in F_P : p^{\mathfrak{A}}(t) = TRUE \text{ iff } p^{\mathfrak{B}}(h_T(t)) = TRUE \end{array} \tag{5}$$

Analogously to the proof of the coincidence condition special versions of the isomorphism condition concerning the index terms and the satisfaction relation $\|\!\!=_{MTL,\Sigma}$ has to be shown. This is expressed by the following two statements:

$$\forall c \in Ind_{MTL}(\Sigma) : \mathfrak{I}_{\mathfrak{B}}(c) = h(\mathfrak{I}_{\mathfrak{A}}(c)) \tag{6}$$

$$\forall t \in A_T \text{ and } \forall \varphi \in Sen_{MTL} : (\mathfrak{A}, t) \|\!\!=_{MTL,\Sigma} \varphi \text{ iff } (\mathfrak{B}, h(t)) \|\!\!=_{MTL,\Sigma} \varphi \tag{7}$$

Both statements are proven by induction over the structure of c and φ, respectively.

The isomorphism condition itself can easily be derived from the last statement.

This concludes the proof that MTL is actually a T-FDT.

4 Related Work

A similar approach to investigate FDTs is presented in [AR97], where a *pattern for analyzing a formal specification activity* is presented. Notions such as e.g. *end product*, *formal model*, and *rationale* are introduced and applied to *formal methods*. This is done in the framework of institutions, too. Compared to this work, we intended to be more formal and hence more precise when defining what constitutes an FDT and the models we consider. Also, in [AR97] methodological aspects are considered, whereas we restricted ourselves to characterize FDTs for timed systems more precisely.

In [Bro96] several models, as e.g. *data models* or *system–component models*, are presented. The models are oriented towards components of distributed systems and the mathematical model of ongoing behavior is based on streams of messages exchanged between the components. In contrast, our definitions are more oriented towards sequences of states. Although these approaches are interchangeable to some extent, the technical expositions are different. As a second point, [Bro96] aims at a comprehensive set of mathematical models as a formal foundation of software engineering, whereas we are interested in the formal analysis of T-FDTs.

5 Concluding Remarks

Starting from a general definition of an FDT similar to an institution, we derived a definition of an *FDT for timed systems*. This was done by requiring that signatures contain an explicit subsignature to express time and that this subsignature is interpreted as a partial order by the structures. Other function and predicate symbols have to be interpreted as functions and relations over time. These, together with the partial order constitute our main models, i.e. the behavior models. Derived notions are state and system models.

The sentences over the signatures have been restricted in no way, henceforth our characterization is independent of a concrete syntax. Therefore, we believe that it is even possible to show that graphical notations are T-FDTs.

In this paper we have outlined the proof that the temporal logic MTL is actually a T-FDT. In [KDE98] we present complete and detailed proofs not only for MTL being a T-FDT but also for the two temporal logics CTL* [Eme90] and MTL-$\int$ [LH95]. There we also provide evidence that presumably all kinds of temporal logics can be embedded in the framework given by T-FDTs. In [EDK98] we show that *Abstract State Machines* (ASMs), a formalism completely different from temporal logics, is also an instantiation of the general notion of T-FDTs. This indicates, that quite different description techniques fit into our framework.

When showing, that a description technique is actually a T-FDT we made the experience, that already by this activity a lot of information about and a deep understanding of this FDT could be gained. To get an even deeper understanding, and to get it in a more systematic way, we are currently formulating criteria investigating different aspects of T-FDTs. An initial set of such criteria has already been developed and applied to the three FDTs considered in this paper. This set covers e.g. such different aspects as expressiveness, properties of the time model, and compositionality. We intend to give precise definitions of as much as possible of the criteria collected in [DKZ97]. It should be clear, that not all of these criteria, especially those investigating methodological aspects, can be formalized within the framework presented here. But let us note that our framework has been quite adequate for the criteria formalized up to now.

Acknowledgments
We would like to thank J. Avenhaus and K. Madlener for fruitful discussions on this and earlier versions of this paper.

Appendix

For the sake of self–containment of this report we present some basic definitions of category theory, model theory, and algebraic specification. Further material can be found for example in [Lan71], [CK90], and [LEW96], respectively.

A Category Theory

Definition 23 (Category). *A* category *is given by a class of* objects *and a class of* arrows *or* morphisms *such that:*

1. *Each arrow has a* domain *and a* codomain *which are objects. One writes $f : A \to B$ or $A \xrightarrow{f} B$ if A is the domain of the arrow f, and B its codomain. One also writes $A = dom(F)$ and $B = cod(f)$.*
2. *Given two arrows f and g with $cod(f) = dom(g)$, the* composition *of f and g, written $f \cdot g$ (or $g \circ f : A \to C$), is defined and has domain $dom(f)$ and codomain $cod(g)$: $A \xrightarrow{f} B \xrightarrow{g} C$*
3. *Composition is* associative*, that is: given $f : A \to B$, $g : B \to C$ and $h : C \to D$, then $(f \cdot g) \cdot h = f \cdot (g \cdot h)$.*
4. *For every object A there is an* identity *arrow id_A (or 1_A), satisfying $f \cdot id_B = f$ and $id_A \cdot f = f$ for $f : A \to B$.*

For a category $\mathbb{C}$ let $\mid \mathbb{C} \mid$ denote its class of objects. For the sake of simplicity we sometimes omit the bars and write $A \in \mathbb{C}$ for an object A in $\mathbb{C}$. Analogously, we write $f : A \to B \in \mathbb{C}$ for a morphism $f : A \to B$ of $\mathbb{C}$. Given two objects A and B, the collection of all morphisms f such that $f : A \to B$ is denoted by $\mathbb{C}[A, B]$.

Example 1. Standard examples for categories are:

1. $\mathbf{0}$ is the empty category.
2. Set is the category, which has all sets as its objects and functions from one set to another as morphisms. Hereby we assume the standard Zermelo-Fraenkel axioms (ZFC) for set theory.
3. Top is the category of topological spaces and continuous functions.
4. Grp is the category of groups and group homomorphisms.

For a category $\mathbb{C}$, the category with the same objects, but all the morphisms reversed is denoted by $\mathbb{C}^{op}$. It is called the *opposite category* of $\mathbb{C}$.

Definition 24 (Functor). *A morphism $F : \mathbb{C} \to \mathbb{D}$ from a category $\mathbb{C}$ to a category $\mathbb{D}$ is called a* covariant functor *if it satisfies the following conditions. It consists of two mappings, one relating objects and the other one relating morphisms, usually denoted by the same name.*

1. *For each morphism $f : A \to B \in \mathbb{C}$ there is a morphism $F(f) : F(A) \to F(B) \in \mathbb{D}$.*
2. *$F(id_A) = id_{F(A)}$ for each $A \in \mathbb{C}$.*
3. *$F(f \cdot g) = F(f) \cdot F(g)$ for each $f : A \to B$, $g : B \to C \in \mathbb{C}$.*

F is called a contravariant *functor, if F is a functor from $\mathbb{C}^{op}$ to $\mathbb{D}$.*

In the following we simply speak of functors instead of covariant functors. Often, application of functors is written by juxtaposition, i.e. we write for example Ff instead of $F(f)$. A functor $F : \mathbb{C} \to \mathbb{D}$ is *faithful* if for all objects $A, B \in \mathbb{C}$ and all $f, g \in \mathbb{C}[A, B]$, $Ff = Fg$ implies $f = g$; it is *full* if for all objects $A, B \in \mathbb{C}$ and every $h \in \mathbb{D}[FA, FB]$ there is $g \in \mathbb{C}[A, B]$ such that $h = Fg$.

Taking categories themselves as objects and functors as morphisms, the category $\textsc{Cat}$ is obtained. Strictly speaking, this is the category of all *small* categories to avoid paradoxes as in set theory. This is discussed in more detail for example in [Lan71].

A morphism $f : A \to B$ is an *isomorphism* if there exists a morphism $g : B \to A$ with

$$f \cdot g = id_A \text{ and } g \cdot f = id_B.$$

It is easy to show that there exists at most one g satisfying this condition, and this unique arrow is called the *inverse* f^{-1} of f. Two objects A and B of a category $\mathbb{C}$ are called *isomorphic*, if there is an isomorphism $f : A \to B$. Note that every functor preserves isomorphisms.

Definition 25 (Subcategory). *A* subcategory $\mathbb{B}$ *of a category $\mathbb{C}$, denoted by $\mathbb{B} \subseteq \mathbb{C}$, is a collection of some of the objects and some of the arrows of $\mathbb{C}$, which includes with each arrow $f : A \to B$ both A and B, with each object A its identity arrow id_A, and with each pair of composable arrows $f : A \to B$ and $g : B \to C$ their composite $f \cdot g : A \to C$.*

The functor $\mathbb{B} \to \mathbb{C}$ which sends each object and each arrow of $\mathbb{B}$ to itself (in $\mathbb{C}$) is called the *inclusion functor*. This functor is clearly faithful. $\mathbb{B}$ is called a *full* subcategory of $\mathbb{C}$ if the inclusion functor $\mathbb{B} \to \mathbb{C}$ is full.

Definition 26 (Coproduct). *The diagram $A \xrightarrow{i} C \xleftarrow{j} B$ is called an* coproduct diagram *if for any pair of arrows $f : A \to D$ and $g : B \to D$ there is a uniquely $h : C \to D$ with $f = i \cdot h$ and $g = j \cdot h$. In this case the object C is unique determined up to isomorphism. It is written $C = A \amalg B$ and is called* coproduct object *or simply* coproduct *of A and B.*

$$\begin{tikzcd}
A \arrow[r, "i"] \arrow[dr, "f"'] & A \amalg B \arrow[d, dashed, "h"] & B \arrow[l, "j"'] \arrow[dl, "g"] \\
& D &
\end{tikzcd}$$

Note that in SET the coproduct $A \amalg B$ coincides with the disjoint union $A \uplus B$.

B Model Theory

Definition 27 (Signature). *A signature $\Sigma = (S, F, P, D)$ consists of*

1. *a finite set S of sorts,*
2. *a set F of function symbols, a set P of relation symbols, and*
3. *a set D of declarations $f : \overline{s} \to s$ and $p : \overline{t}$ for each $f \in F$ and each $p \in P$, respectively, where $\overline{s} \in S^*$, $\overline{t} \in S^+$, and $s \in S$, such that for each function or relation symbol there is exactly one declaration in D.*

If not stated otherwise, the components of a signature Σ are always (S, F, P, D) as described above. Sometimes we identify Σ with its disjoint union $S \uplus F \uplus P \uplus D$.

Definition 28 (Subsignature). *A* subsignature *Σ' of Σ is a signature (S', F', P', D') such that $S' \subseteq S$, $F' \subseteq F$, $P' \subseteq P$, and $D' \subseteq D$. This is denoted by $\Sigma' \subseteq \Sigma$.*

For $\Sigma' \subseteq \Sigma$ the difference $\Sigma \setminus \Sigma'$ is defined component wise. This difference is sometimes called *enrichment* of Σ'. Note that $\Sigma \setminus \Sigma'$ must not be a signature.

Definition 29 (Signature Morphism). *Given two signatures $\Sigma, \Sigma' \in$ SIG, a* signature morphism *$\sigma : \Sigma \to \Sigma'$ with $\sigma = (\sigma_S, \sigma_F, \sigma_P)$ consists of functions $\sigma_S : S \to S'$, $\sigma_F : F \to F'$, and $\sigma_P : P \to P'$ such that for each $f : s_1, \ldots, s_n \to s \in \Sigma$ and $p : s_1, \ldots, s_n \in \Sigma$ we have $\sigma_F(f) : \sigma_S(s_1), \ldots, \sigma_S(s_n) \to \sigma_S(s)$ and $\sigma_P(p) : \sigma_S(s_1), \ldots, \sigma_S(s_n)$ in Σ'.*

If it is clear from the context, the indices of the components of a signature morphism are omitted, e.g. we write $\sigma(f)$ instead of $\sigma_F(f)$.

Definition 30 (Signature Category). SIG *is the category of all signatures with signature morphisms as morphisms.*

Definition 31 (Structure). *Let $\Sigma \in$ SIG be a signature. A* structure *$\mathfrak{A} = (A, \_^{\mathfrak{A}})$ over Σ consists of a family $A = (A_s \mid s \in S)$ of non-empty carrier sets*[3] *and a mapping $\_^{\mathfrak{A}}$ which is compatible with the declarations of Σ, i.e.*

[3] For the sake of simplicity we identify sometimes A with its disjoint union $\biguplus\{A_s \mid s \in S\}$.

1. *there is a function* $f^{\mathfrak{A}} : A_{\overline{s}} \to A_s$ *for each* $f : \overline{s} \to s \in \Sigma$,
2. *there is a relation* $p^{\mathfrak{A}} \subseteq A_{\overline{s}}$ *for each* $p : \overline{s} \in \Sigma$,

where $A_{\overline{s}}$ *with* $\overline{s} = (s_1, \ldots, s_n)$ *stands for* $\prod(A_{s_i} \mid i = 1, \ldots, n)$.

Definition 32 (Homomorphism). *Given two structures* $\mathfrak{A}, \mathfrak{B}$ *over the signature* Σ, *a family* $h = (h_s \mid s \in S)$ *of mappings* $h_s : A_s \to B_s$ *is called a* homomorphism *iff it satisfies the following conditions:*

1. $h_s(f^{\mathfrak{A}}(\overline{a})) = f^{\mathfrak{B}}(h_{\overline{s}}(\overline{a}))$ *for all* $f : \overline{s} \to s \in \Sigma$ *and* $\overline{a} \in A_{\overline{s}}$, *and*
2. $\overline{a} \in p^{\mathfrak{A}}$ *implies* $h_{\overline{s}}(\overline{a}) \in p^{\mathfrak{B}}$ *for all* $p : \overline{s} \in \Sigma$ *and* $\overline{a} \in A_{\overline{s}}$,

where $h_{\overline{s}}$ *with* $\overline{s} = (s_1, \ldots, s_n)$ *stands for* $\prod(h_{s_i} \mid i = 1, \ldots, n)$.

Definition 33 (Structure Category). $\mathrm{STR}(\Sigma)$ *is the category of all structures over the signature* Σ *with homomorphisms as morphisms.*

For a signature morphism $\sigma : \Sigma \to \Sigma'$, the structures over the signatures are related by the so-called *forgetful functor*.

Definition 34 (Forgetful Functor). *Given the signature morphism* $\sigma : \Sigma \to \Sigma'$ *the* forgetful functor $\_|\sigma : \mathrm{STR}(\Sigma') \to \mathrm{STR}(\Sigma)$ *is defined for all structures* $\mathfrak{A}' \in \mathrm{STR}(\Sigma')$ *by* $\mathfrak{A}'|\sigma = \mathfrak{A}$ *with*

1. $A_s = A'_{\sigma(s)}$ *for all* $s \in S$,
2. $f^{\mathfrak{A}} = \sigma(f)^{\mathfrak{A}'}$, $p^{\mathfrak{A}} = \sigma(p)^{\mathfrak{A}'}$ *for all* $f \in F$, $p \in P$.

A homomorphism $h' : \mathfrak{A}' \to \mathfrak{B}' \in \mathrm{STR}(\Sigma')$ *is mapped to the corresponding homomorphism* $h'|\sigma : \mathfrak{A}'|\sigma \to \mathfrak{B}'|\sigma \in \mathrm{STR}(\Sigma)$.

For a class $\mathfrak{K} \subseteq \mathrm{STR}(\Sigma)$ of structures, $\mathfrak{K}|\sigma$ denotes the class $\{\mathfrak{A}|\sigma \mid \mathfrak{A} \in \mathfrak{K}\}$.

Str denotes the contravariant functor $Str : \mathrm{SIG} \to \mathrm{CAT}$ assigning the category $\mathrm{STR}(\Sigma)$ to each signature $\Sigma \in \mathrm{SIG}$ and the forgetful functor $\_|\sigma : \mathrm{STR}(\Sigma') \to \mathrm{STR}(\Sigma)$ to each signature morphism $\sigma : \Sigma \to \Sigma'$.

Let $\mathcal{F}$ be a FDT. The terms *model class* and *theory* are defined as in first-order logic. Let $\Sigma \in \mathrm{SIG}$ be a signature. For a set $\Phi \subseteq Sen_{\mathcal{F}}(\Sigma)$ of sentences, $Mod_{\mathcal{F}}(\Phi) := \{\mathfrak{A} \mid \mathfrak{A} \in Str_{\mathcal{F}}(\Sigma), \mathfrak{A} \models_{\mathcal{F},\Sigma} \varphi$ for all $\varphi \in \Phi\}$ denotes the *model class* of Φ. For a class $\mathfrak{K} \subseteq Str_{\mathcal{F}}(\Sigma)$ of structures the set $Th_{\mathcal{F}}(\mathfrak{K}) := \{\varphi \mid \varphi \in Sen_{\mathcal{F}}(\Sigma), \mathfrak{A} \models_{\mathcal{F},\Sigma} \varphi$ for all $\mathfrak{A} \in \mathfrak{K}\}$ denotes the *theory* of $\mathfrak{K}$. If $\mathfrak{K}$ contains only one element $\mathfrak{A}$ we write $Th_{\mathcal{F}}(\mathfrak{A})$. Analogously, we write $Mod_{\mathcal{F}}(\varphi)$ instead of $Mod_{\mathcal{F}}(\{\varphi\})$. Two sentences $\varphi, \psi \in Sen_{\mathcal{F}}(\Sigma)$ are *equivalent* (in $\mathcal{F}$) iff $Mod_{\mathcal{F}}(\varphi) = Mod_{\mathcal{F}}(\psi)$. Two structures $\mathfrak{A}$ and $\mathfrak{B}$ are *elementarily equivalent* (in $\mathcal{F}$) :iff every sentence that is true in $\mathfrak{A}$ is true in $\mathfrak{B}$ and vice versa. We will express this relationship between structures by $\sim_{\mathcal{F}}$. Two classes $\mathfrak{K}, \mathfrak{K}' \subseteq Sen_{\mathcal{F}}(\Sigma)$ are *elementarily equivalent* (in $\mathcal{F}$) :iff $Th_{\mathcal{F}}(\mathfrak{K}) = Th_{\mathcal{F}}(\mathfrak{K}')$

References

[ABL96] J.-R. Abrial, E. Boerger, and H. Langmaack, editors. *Formal Methods for Industrial Applications, Specifying and Programming the Steam Boiler Control*, volume 1165 of *LNCS*. Springer, 1996.

[ACJ+96] M. A. Ardis, J. A. Chaves, L. J. Jagadeesan, P. Mataga, C. Puchol, M. G. Staskauskas, and J. Von Olnhausen. A framework for evaluating specification methods for reactive systems, experience report. *IEEE TSE*, 22(6):378–389, 1996.

[AH90] R. Alur and T. A. Henzinger. Real-time logics: Complexity and expressiveness. In *Proc. of 5th LICS*, pages 390–401. IEEE Computer Society Press, 1990.

[AR97] E. Astesiano and G. Reggio. Formalism and method. In M. Bidoit and M. Dauchet, editors, *TAPSOFT'97: Theory and Practice of Software Development, Intl. Joint Conf. CAAP/FASE*, volume 1214 of *LNCS*, pages 93–114. Springer, 1997.

[Bar85] J. Barwise. *Model-theoretic logics*. Perspectives in mathematical logic. Springer, 1985.

[Bro96] M. Broy. Formal description techniques - how formal and descriptive are they? In R. Gotzhein and J. Bredereke, editors, *FORTE IX, Theory, application and tools*, pages 95–110. Chapman and Hall, 1996.

[CK90] C. C. Chang and H. J. Keisler. *Model Theory*, volume 73 of *Studies in Logic and the Foundation of Mathematics*. Elsevier, 1990.

[DKZ97] T. Deiß, M. Kronenburg, and D. Zeckzer. A catalogue of criteria for evaluating formal methods and its application. SFB 501 Bericht 04/97, Fachbereich Informatik, Universität Kaiserslautern, 1997.

[EDK98] R. Eschbach, T. Deiß, and M. Kronenburg. Abstract State Machines as an example of a formal description technique for timed systems. SFB 501 Bericht 07/98, Fachbereich Informatik, Universität Kaiserslautern, 1998.

[Eme90] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, chapter 16, pages 995–1072. Elsevier, 1990.

[GB92] J. A. Goguen and R. M. Burstall. Institutions: Abstract model theory for specification and programming. *JACM*, 39(1):95–146, 1992.

[KDE98] M. Kronenburg, T. Deiß, and R. Eschbach. Temporal logics as examples of formal description techniques for timed systems. SFB 501 Bericht 06/98, Fachbereich Informatik, Universität Kaiserslautern, 1998.

[Koy92] R. Koymans. *Specifying Message Passing and Time-critical Systems with Temporal Logic*, volume 651 of *LNCS*. Springer, 1992.

[Lam94] L. Lamport. The temporal logic of actions. *ACM TOPLAS*, 16(3):872–923, 1994.

[Lan71] S. Mac Lane. *Categories for the Working Mathematician*. Springer, 1971.

[LEW96] J. Loeckx, H.-D. Ehrich, and M. Wolf. *Specification of Abstract Data Types*. Wiley-Teubner, 1996.

[LH95] Y. Lakhneche and J. Hooman. Metric temporal logic with durations. *TCS*, 138:169–199, 1995.