T U M

I N S T I T U T F Ü R I N F O R M A T I K

Properties of Hybrid Systems - a Computer Science Perspective

Thomas Stauner

TUM-I0017
November 00

T E C H N I S C H E U N I V E R S I T Ä T M Ü N C H E N

TUM-INFO-11-I0017-0/1.-FI

Druck: Institut für Informatik der
Technischen Universität München

Properties of Hybrid Systems - a Computer Science Perspective*

Thomas Stauner
Institut für Informatik, Technische Universität München
D-80290 München, Germany
`http://www4.in.tum.de/~stauner/`
Email: `stauner@in.tum.de`

Abstract

Motivated by the work on hybrid, i.e. mixed discrete and continuous, systems, we introduce a set of important properties of such systems and classify them. For the properties of stability and attraction which are central for continuous systems we discuss their relationship to discrete systems usually studied in computer science. An essential result is that the meaning of these properties for discrete systems vitally depends on the used topologies.

Based on the classification we discuss the utility of a notion of refinement which we will use in the future for property preserving transformations of hybrid systems. Finally, two proof rules and some specializations of them are introduced for proving stability and attraction. The rules result from adapting known techniques from systems theory and are applied to small examples.

*This work was supported with funds of the Deutsche Forschungsgemeinschaft under reference number Br 887/9 within the priority program *Design and design methodology of embedded systems*.

Contents

1 Introduction

The development of hybrid, i.e., mixed discrete and continuous, systems [GNRR93] occurring for example in robotics and process engineering is an interdisciplinary task. It mainly involves engineers from computer science and control theory. Developing such systems as well as designing formal methods, such as validation and refinement techniques for them requires a deeper understanding of their essential properties.

Regarding the work on hybrid systems one notices that properties of such systems which are examined in case studies often reflect the background of the research groups working on them. Computer scientists often focus on safety properties, e.g. that a certain set of states and certain variable domains are never left. People from control theory often put their emphasis on stability properties. While these distinct focuses are sometimes due to the specific characteristics of the regarded systems, they often also result from a lack of knowledge of the respective other domain. To alleviate this shortcoming we define and classify a set of important properties of control systems within a general framework which should be familiar for computer scientists. With the continuing integration of software and its physical environment in many systems we conjecture that properties which have only been of interest for continuous systems so far will also become important for the software part of embedded systems, which stresses the need for proof methods. For the properties of *stability* and *attraction* the paper identifies topologies where stability is a safety property and attraction is a persistence property in computer science terminology [CMP91].

The properties the paper considers result from the evaluation of nine hybrid systems case studies and a number of text books on control theory. The classification of the properties and the case study evaluation serve as reference for judging the utility of a notion of refinement for hybrid systems which is based on trace inclusion and will be the basis for future work. The result here is that not all but essential classes of properties are preserved by the intended refinement notion.

Finally, the paper proposes two general proof methods for stability and attraction together with some specializations. The methods result from adapting methods Liapunov like proof methods from general systems theory to our framework [MT75]. When developing specializations of the methods we outline parallels to abstraction in computer science and to Galois connections in particular. The proof methods are applied to two examples, a hybrid *electronic height control system* taken from a case study performed with BMW [SMF97] and a purely discrete self-stabilizing algorithm [Dij82], which is interesting in this context, because it shows the relevance of attraction also for computer science.

Overview. Section 2 defines our underlying system model which is the basis for the rest of the paper. Section 3 lists and defines the properties which resulted from our evaluation of case studies and text books. Furthermore, it contains the classification of the properties, sorts them according to their relevance as derived from the case studies and examines the utility of trace inclusion as refinement notion for hybrid systems. In Section 4 some proof concepts for stability and attraction are introduced and parallels to computer science are drawn. The proof methods are applied to examples in Section 5. Section 6 discusses the paper's contribution, compares it with related work, and outlines future work. As the paper requires some familiarity with basic topology, the necessary concepts are introduced in Appendix C.

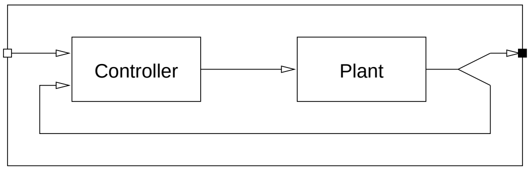


Figure 1: General structure of a closed-loop control system.

2 Systems under Consideration

The properties we will present all assume a system structure as the one depicted in Figure 1. Its basic elements are a controlling device (Controller), the physical environment (Plant) and a feedback loop. Such systems are called feedback control systems. If there is no feedback from the plant's output to the controlling device, the system is called a *supervisory* or *open loop* control system [PH88].

On a very abstract level we can regard a system as a nondeterministic function mapping a cause to a set of possible effects:[1]

$$Sys \in \mathcal{C} \to \wp(\mathcal{E})$$

Pairs (c, e) with $e \in Sys(c)$ are called *behaviors* of the system.

Black-box behavior. In the following we will use two specializations of this system model. The first one describes the black-box behavior (or I/O behavior) of a system as a function:

$$Sys_{IO} \in \mathcal{I}^{\mathbb{R}_+} \to \wp(\mathcal{O}^{\mathbb{R}_+})$$

where $\mathcal{I}$ is the input domain and $\mathcal{O}$ is the output domain, i.e. $\mathcal{C}$ has been instantiated to $\mathcal{I}^{\mathbb{R}_+}$ and $\mathcal{E}$ to $\mathcal{O}^{\mathbb{R}_+}$, where $\alpha^{\mathbb{R}_+}$ denotes the set of functions from the non-negative real numbers $\mathbb{R}_+$ to α. Elements of $\alpha^{\mathbb{R}_+}$ are called *hybrid streams*, *hybrid traces* or *trajectories* and functions on streams, like Sys_{IO}, are also called *stream processing functions* [Bro97]. We require Sys_{IO} to be total in its input, i.e. $Sys_{IO}(\iota) \neq \emptyset$ for all $\iota \in \mathcal{I}^{\mathbb{R}_+}$. Furthermore, to model realistic systems Sys_{IO} must be *(weakly) time guarded*, i.e. its current output may not depend on future inputs:

$$\iota_1\!\downarrow_{[0,t]} = \iota_2\!\downarrow_{[0,t]} \quad \Rightarrow \quad Sys_{IO}(\iota_1)\!\downarrow_{[0,t]} = Sys_{IO}(\iota_2)\!\downarrow_{[0,t]}$$

where $\xi\!\downarrow_M$ denotes the restriction of function ξ to arguments in the set M and is extended to sets of functions in a pointwise manner. Throughout the chapter variable t is used to denote a point in time, $t \in \mathbb{R}_+$.

White-box behavior. Some of the properties we will examine require a state-based (or glass-box) system description, or come in two variants, one depending only on the interface behavior and the other depending on the system state. We formalize state-based systems as follows:

$$\begin{aligned} &Sys_S \in \mathcal{S} \to \mathcal{I}^{\mathbb{R}_+} \to \wp(\mathcal{S}^{\mathbb{R}_+}) \\ &Out \in \mathcal{S} \to \mathcal{O} \end{aligned}$$

Again $\mathcal{I}$ is the input domain, $\mathcal{S}$ is the state space. Depending on the initial state in $\mathcal{S}$, Sys_S maps an input trajectory to a set of possible state trajectories. Thus $\mathcal{C}$ is instantiated to $\mathcal{S} \times \mathcal{I}^{\mathbb{R}_+}$ and $\mathcal{E}$ to $\mathcal{S}^{\mathbb{R}_+}$, here. All state trajectories are required to start with the prescribed initial state, $\forall \sigma \in Sys_S(s, \iota).\, \sigma(0) = s$. Function Out maps the current state to the current output in $\mathcal{O}$. Thus, it performs some state-to-output conversion. It is extended in time by pointwise extension, $Out^\dagger(\sigma)(t) = Out(\sigma(t))$ for all $t \in \mathbb{R}_+$. The black box behavior of a state based system with initial state s is defined by the sequential composition of $Sys_S(s)$ and the time extension of Out:

[1] Although control theory usually focuses on deterministic systems, we employ a more general nondeterministic approach which is closer to models in the field of computer science.

$$Sys_{IO} = Sys_S(s); Out^\dagger$$

where sequential composition of a nondeterministic system A and a deterministic system B is defined as $A; B(\alpha) = \{\gamma \mid \exists\beta.\beta \in A(\alpha) \ \wedge \ \gamma = B(\beta)\}$. For a set of initial states $S \subseteq \mathcal{S}$ we define $Sys_S(S)$ by pointwise extension. Like for black-box descriptions, we also require that state-based system descriptions $Sys_S(s)$ be total in the input and time-guarded for any start state s.

Furthermore, we demand that Sys_S is *time invariant*, meaning that the system does not depend on absolute timing. Instead, a system's state trajectories are completely determined by the initial state and the input trajectory. Formally, we demand that a left shift of the input results in the same left shift of the state trajectory for the shifted start state:

$$\sigma \in Sys_S(s, \iota) \Rightarrow \sigma^{-u} \in Sys_S(\sigma(u), \iota^{-u})$$

for any $u \geq 0$, where φ^{-u} is the left shift of φ by u, $\varphi^{-u}(t) = \varphi(u + t)$. Because of time invariance it is sensible to regard a disturbance of a system, i.e. an unforeseen change in its state, as reinitialization of the system. Hence, the behavior of system Sys_S resulting from disturbing it by setting its state to s' at time t is defined by $Sys_{disturbed}(s_0, \iota, s', t) = \{\sigma \mid \exists\sigma_1 \in Sys_S(s_0, \iota).\, \exists\sigma_2 \in Sys_S(s', \iota^{-t}).\, \sigma\downarrow_{[0,t)} = \sigma_1\downarrow_{[0,t)} \ \wedge \ \sigma^{-t}\downarrow_{[0,\infty)} = \sigma_2\}$. When we consider a system's reaction to disturbances, it therefore suffices to consider disturbances of the initial state. System behavior after later disturbances can be inferred from its reaction to initial disturbances.

Note that systems with a semantics of one of the above two types, i.e. black-box or white-box, can be specified with formal, graphical notations such as HyCharts [GSB98].

3 Properties

Based on the system model introduced above a number of important properties of control systems are defined and classified in this section. Relative to this classification we then discuss the utility of trace inclusion as refinement notion. The properties have been extracted from the evaluation of nine hybrid systems case studies, which were taken from papers and from student projects performed within various departments of the Technische Universität München [BHKT98, Eng97, Ger97, Sta97, Rap98, Ant96, Abr96, SB98, BGM93], and from text books on control theory, in particular [Föl90, Föl87, PH88, Vac95, Lei87]. Using case studies as one source was done to be able to estimate the practical relevance of the properties.

The case studies. The topics and domains of the case studies are as follows. [BHKT98] introduce a hybrid benchmark problem from the field of chemical engineering and list required properties of an implementation. The introductory paper [Eng97] explains hybrid systems along an example from process automation and lists system properties whose analysis is required to be feasible with (present and future) methods for hybrid systems. In [Ger97] elements of the design of an autonomous cruise control system, which adjust a car's speed and distance to the car in front, are discussed and vital properties are analyzed analytically and with prototypes, respectively. [Sta97] employs the *HyTech* model checker [HHWT95] to verify some properties of a system from the automotive field. A simplified version of this system also appears in this paper. In [Rap98] a further system stemming from automotive electronics is modeled with MatrixX/Betterstate [Int00, Int98]. [Ant96] models a system of conveyor belts from the process automation domain. [Abr96] introduces a steam boiler example as a benchmark for formal methods for embedded systems. In [SB98] a problem from robotics is discussed, an architecture for its solution is developed and validated with simulations. [BGM93] analyze the behavior of a control system for a hopping robot by systematic simulation.

Formalization. Unless otherwise mentioned all the following definitions assume that systems are given as nondeterministic functions from a cause to a set of effects. Thus, they apply to I/O based as well as to state-based system descriptions.

3.1 Robustness

An important property of control systems is that they satisfy some required properties in spite of deviations between the model of the system and the real system [Fri86]. Deviations may range from inaccurate parameters, e.g. due to aging effects, to structural mistakes, like an incorrect model of the plant.

Hence, robustness of a system depends on the system itself, the regarded property, and on the regarded deviations. Let D be the set of systems deviating from Sys in the way that is supposed to be considered. For instance, D may be defined relying upon some metric on systems. We demand that $Sys \in D$, i.e. no deviation from the perfect system is also allowed. Furthermore, let $valid_{\Psi}(Sys)$ be an evaluation function which is true iff system Sys satisfies property Ψ. We define robustness as follows.

Definition 1 (Robustness.) *A system Sys with deviations D, $Sys \in D$,* robustly satisfies *property Ψ iff $\forall Sys' \in D.\, valid_{\Psi}(Sys')$ holds.*

Thus robustness is parameterized with the notion of validity of the considered object property Ψ.

[MT75] introduces the related notion of structural stability. There, a topology is defined on the set of deviating systems. Based on this topology, the authors demand that small deviations in the system result in small output deviations (see also Section 3.3).

Note that robustness is related to nondeterminism. For universal properties, i.e. properties which have to hold for any execution of a system, we may subsume the behavior of all the deviating systems in one nondeterministic system, $Sys'(c) = \bigcup_{Sys \in D} Sys(c)$ for causes $c \in \mathcal{C}$. To prove robustness of the universal property w.r.t. D it then suffices to show that Sys' satisfies the property.

3.2 Optimality

Apart from finding a solution to a given problem, control theory is interested in identifying the optimal solution for the problem w.r.t. some cost function. Possible aims e.g. are minimizing energy consumption or maximizing throughput of a plant.

For a given system Sys, a set of alternative systems A, and a cost function c from the set of systems to a linearly ordered set with order $<$ we define optimality as follows.

Definition 2 (Optimality.) *System Sys is* optimal *w.r.t. alternatives A and cost function c iff $\forall Sys' \in A.\, c(Sys) \leq c(Sys')$.*

Here, $\leq$ denotes the reflexive closure of $<$. The set of alternative systems A may e.g. be defined as the set of all those systems which satisfy a given abstract specification. In practice cost functions are often defined based on the system's output.

3.3 Stability

The general idea of stability is based on the desire that small disturbances in the causes should only cause small disturbances in the effects. To formalize closeness of one cause (or effect) to another one we use the notion of neighborhood which is induced by the topologies considered for causes and effects (see Appendix C for basic concepts of topology). We write $N(\alpha)$ for the set of all neighborhoods of α.

Definition 3 ((General) Stability.) *For a system $Sys \in \mathcal{C} \to \wp(\mathcal{E})$ between topological spaces $(\mathcal{C}, \mathcal{O}_\mathcal{C})$ and $(\mathcal{E}, \mathcal{O}_\mathcal{E})$, the tuple of sets of causes and effects $(C, E) \subseteq \mathcal{C} \times \mathcal{E}$ is* stable *w.r.t. these spaces iff $\forall \alpha \in N(E). \exists \beta \in N(C). \forall b \in \beta. Sys(b) \subseteq \alpha$.*

The definition requires that for any neighborhood of the effects E there is a neighborhood of the causes C such that the effects resulting from any of these causes are in the considered neighborhood of E. Note that disturbances and their "size" are a vague concept. The stability definition tries to grasp the notion of "small disturbances of causes" resulting in "small disturbances of effects" by universally quantifying over the neighborhoods $N(E)$. For a "small" neighborhood $\alpha \in N(E)$, i.e. one that does not contain much more elements than E, stability provides that there is a $\beta \in N(C)$, which may also be small, such that disturbed causes in β result in disturbances of the effects which are in α. Hence, stability allows to conclude that there exists a neighborhood of C in which disturbances of causes must be in order to ensure that the resulting disturbances of the effects remain within a desired neighborhood of E.

Mesarovic et al. [MT75] furthermore demand that $Sys(C) \subseteq E$ which, as we will see in Section 3.3.2, corresponds to the stability of invariant sets in the state-based case. If enforced, this demand excludes that (C, E) is stable although C is mapped to a set which is disjoint from E, but contained in every neighborhood of E.

In topologies where C and E are open, (C, E) is stable iff $\forall c \in C. Sys(c) \subseteq E$ (for the proof see Theorem 4 in Appendix A). In particular, this includes the discrete topology (cf. Appendix C). This result is a consequence of the definition of neighborhood which for open sets X permits $X \in N(X)$ (see Definition 13). However, note that [MT75] does not permit $X \in N(X)$ in his definition of stability. In the standard case of control theory where stability of points is studied w.r.t. the usual topology on the Euclidean space, such singletons are closed sets and hence the definition of neighborhoods for open sets is immaterial there.

Note that in this general form the concept of stability is related to the continuity of functions on the reals. In standard textbooks on Analysis continuity of a function on the reals $f \in \mathbb{R} \to \mathbb{R}$ at x_0 is usually defined as $\forall \epsilon > 0. \exists \delta > 0. \forall x. |x - x_0| < \delta \to |f(x) - f(x_0)| < \epsilon$ [Kön90]. In terms of neighborhoods on the reals this can be expressed as $\forall \alpha \in N(f(x_0)). \exists \beta \in N(x_0). \forall x. x \in \beta \to f(x) \in \alpha$ which corresponds to the general notion of stability from above.

In the following we will outline a number of specializations of this general stability definition.

3.3.1 Stability of Trajectories

Depending on the instantiation of causes and effects in the general definition of stability from above and depending on employing an I/O-based or a state-based system model, we obtain a number of different notions of stability.

Stability of (sets of) trajectories basically expresses that the *traces* of the system are in the neighborhood of some desired traces for small disturbances in the system's causes. An I/O-based and a state-based formalization for stability of trajectories of a tuple (C, E) of sets of

	$C \subseteq .$	$E \subseteq .$				
$\mathrm{Tr}_{\mathrm{IO}}$	$\mathcal{I}^{\mathbb{R}_+}$	$\mathcal{O}^{\mathbb{R}_+}$	$\forall \alpha \in N(E).\, \exists \beta \in N(C).$		$\forall c \in \beta.$	$Sys_{IO}(c) \subseteq \alpha$
Tr_{S}	$\mathcal{S}$	$\mathcal{S}^{\mathbb{R}_+}$		$\forall \iota \in I.$		$Sys_S(c, \iota) \subseteq \alpha$
$\mathrm{Pt}_{\mathrm{IO}}$	$\mathcal{I}^{\mathbb{R}_+}$	$\mathcal{O}$				$Sys_{IO}(c) \subseteq \alpha^{\mathbb{R}_+}$
Pt_{S}	$C = E \subseteq \mathcal{S}$			$\forall \iota \in I.$		$Sys_S(c, \iota) \subseteq \alpha^{\mathbb{R}_+}$

Table 1: Notions of stability.

causes and effects is given in the first two lines of Table 1. Note that the considered effects are traces, here. Furthermore, note that for state-based system descriptions our definition is parameterized with the set of inputs I for which stability must hold. This differs from usual definitions like those in [MT75, MW95], which completely disregard external input, for state-based system descriptions

Stability of trajectories is of particular interest in connection with so called *limit cycles*. We will consider this in see Section 3.5.2.

Poisson stability. A rather different notion of stability is referred to as *Poisson stability* [Lei87]. Informally, a system is stable in the sense of Poisson iff there is a point in the system's trajectory whose neighborhoods are visited by the trajectory infinitely often.

For I/O based systems Poisson stability can be defined as follows. A definition for state based systems is similar.

Definition 4 (Poisson stability.) *For a system $Sys_{IO} \in \mathcal{I}^{\mathbb{R}_+} \to \wp(\mathcal{O}^{\mathbb{R}_+})$ on the topological space $(\mathcal{O}^{\mathbb{R}_+}, \mathcal{O}_{\mathcal{O}^{\mathbb{R}_+}})$ a set of trajectories $Sys_{IO}(\iota)$ is positively Poisson stable iff $\exists t_0.\, \forall \alpha \in N(Sys_{IO}(\iota)(t_0)).\, \forall t.\, \exists t' \geq t.\, Sys_{IO}(\iota)(t') \subseteq \alpha$.*

According to Leipholz an application area for Poisson stability is the stability of planet orbits in astronomy [Lei87]. There, it is usually required that *every* point of the orbit is visited infinitely often.

In topologies where the $Sys_{IO}(\iota)(t)$ are open sets for all $t \in \mathbb{R}_+$, Poisson stability has some similarity to *reactivity properties* in computer science, as defined in [CMP91]. Informally reactivity properties express that a certain predicate *always* holds *eventually*, i.e. it must hold infinitely often. In topologies where the $Sys_{IO}(\iota)(t)$ are open, Poisson stability similarly expresses that some $Sys_{IO}(\iota)(t)$ is visited infinitely often. Hence, Poisson stability may be regarded as expressing a reactivity property. However, reactivity properties in general are not restricted to expressing that some already visited set of states is visited infinitely often. Instead the set of states which is required to be visited infinitely often can be defined independently from the system's trajectories.

3.3.2 Stability of Points

Like stability of trajectories, stability of (sets of) points can also be derived from Definition 3. For stability of points, however, a notion of neighborhood of points instead of neighborhood of trajectories is used.

Informally, stability of (sets of) points expresses that the effects of a system *always* are in the neighborhood of some desired points for small disturbances in the system's causes. In contrast to stability of trajectories which regards traces as a whole, stability of points views the effects of a system in a pointwise manner, namely for each point in time. In the sense of Leipholz [Lei87] stability of points is a purely *geometric* notion of stability, while stability of trajectories is a *kinematic* notion of stability, because is also takes the time at which the individual points are reached into consideration. Depending on the topology for trajectories stability of (sets of) points and stability of trajectories are equivalent, namely in topologies which only consider the points visited and not the time when they are visited. The formalization of stability of a tuple (C, E) of sets of causes and effects is given in the bottom two lines of Table 1 for I/O-based and state-based systems, respectively.

State-based stability of points. In the following we will focus on state-based stability of (sets of) points, which is the stability notion encountered most frequently in applications. It is concerned with the effect of disturbances in the system state. According to the table, for a state-based system $Sys_S \in \mathcal{S} \to \mathcal{I}^{\mathbb{R}_+} \to \wp(\mathcal{S}^{\mathbb{R}_+})$ a set $S \subseteq \mathcal{S}$ is stable w.r.t. the topology $\mathcal{O}_\mathcal{S}$ on $\mathcal{S}$ and the inputs $I \subseteq \mathcal{I}^{\mathbb{R}_+}$ iff the formula in the fourth line of the table holds (for $S = C = E$). Again note that the definition is parameterized with the set of inputs I for which stability must hold.

While the definition only consideres disturbances in a system's initial state explicitly, we can infer that disturbances in the state at any point in time cause the same (shifted) behavior as a likewise disturbed initial state. This is due to our definition of disturbed systems which in turn is motivated by time invariance of the considered systems (Section 2).

Liapunov stability as defined e.g. in [Lei87] is a special case of the definition given here for deterministic systems and the usual metric on the real line. Typically Liapunov stability of a point $x \in \mathbb{R}$ is defined as follows: $\forall \epsilon > 0. \exists \delta > 0. \forall x_0. |x - x_0| < \delta \Rightarrow \forall t > 0. |x - Sys_S(x_0)(t)| < \epsilon$, where Sys_S is deterministic and gets no external input. If we take $\{x\}$ as the considered set of causes and effects and if we employ the notion of neighborhood that is induced by taking, as usually, the sets $B_\delta(y) = \{y' \mid |y - y'| < \delta\}$ as base for the topology on $\mathbb{R}$, the equivalence to our definition of stability is straightforward.

The stability definitions in [MT75, MW95] additionally require that the regarded set is invariant, i.e. that it is never left again once it has been entered.

Definition 5 (Invariance.) *A set $A \subseteq \mathcal{S}$ is* invariant *for system Sys_S and the inputs $I \subseteq \mathcal{I}^{\mathbb{R}_+}$ iff $\forall s \in A. \forall \iota \in I. Sys_S(s, \iota) \subseteq A^{\mathbb{R}_+}$.*

In topologies where A is open, invariance of A is equivalent to the stability of A (see Theorem 5 in Appendix A). When associating disturbances with neighborhoods this can informally be explained as follows. In topologies of this kind there is a neighborhood of A (i.e. a disturbance of A) which remains within A. Hence, the smallest possible disturbance is having no disturbance at all. Or, more informally, there are no "small" disturbances which leave A. This suits well to discrete systems where all disturbances can be regarded as equally big, i.e. for discrete systems the only small disturbance also is having no disturbance at all. From a computer science perspective invariance of A is a safety property [CMP91]. Hence, in topologies where A is open stability of A is a safety property.

A notion contrary to stability is *chaos* [Wig90]. One precondition for chaotic behavior is sensitive dependence on initial conditions for an invariant set A. Informally, this expresses that for any two arbitrarily close points in A system traces starting in them separate after some time.

3.4 Attraction

Often we do not only want that disturbances only have a limited effect (stability), but also that a system returns to some desired trajectory or state after a disturbance. Informally, attraction of a set means that despite of some initial disturbance the set is always reached

	$C \subseteq .$	$E \subseteq .$	stability/attraction of (sets of) ...
1	$\mathcal{I}^{\mathbb{R}_+}$	$\mathcal{O}^{\mathbb{R}_+}$	trajectories, I/O-based
2	$\mathcal{S}$	$\mathcal{S}^{\mathbb{R}_+}$	trajectories, state-based
3	$\mathcal{I}^{\mathbb{R}_+}$	$\mathcal{O}$	points, I/O-based
4	$C = E \subseteq \mathcal{S}$		points, state-based

Table 2: Instantiations of causes and effects and resulting notion of stability and attraction.

<table>
<tr><td>1</td><td rowspan="4">$\exists \alpha \in N(C).$</td><td></td><td rowspan="4">$\forall c \in \alpha. \forall \beta \in N(E).$</td><td>$\forall e \in Sys_{IO}(c).$</td><td rowspan="4">$\exists t.$</td><td rowspan="2">$e\downarrow_{[t,\infty)} \in \beta\downarrow_{[t,\infty)}$</td></tr>
<tr><td>2</td><td>$\forall \iota \in I.$</td><td>$\forall e \in Sys_S(c, \iota).$</td></tr>
<tr><td>3</td><td></td><td>$\forall e \in Sys_{IO}(c).$</td><td rowspan="2">$e\downarrow_{[t,\infty)} \in \beta^{\mathbb{R}_+}\downarrow_{[t,\infty)}$</td></tr>
<tr><td>4</td><td>$\forall \iota \in I.$</td><td>$\forall e \in Sys_S(c, \iota).$</td></tr>
</table>

Table 3: Notions of attraction.

after a finite or infinite amount of time. Like for stability, a number of variants of attraction result depending on whether we regard attraction of trajectories or of points for I/O-based or for state-based systems. Table 3 contains the formulas defining these variants of attraction and Table 2 lists the instantiation of causes and effects for each line of Table 3, and the resulting notion of attraction. In the following we give an overview over these notions and will then consider attraction of points in the state-based case, which occurs frequently, in more detail.

3.4.1 Attraction of Trajectories

In practice, attraction of trajectories is often of particular interest for periodic trajectories, expressing e.g. the repeated execution of some required task. For I/O-based systems attraction of trajectories is defined as follows.

Definition 6 (Attractive trajectories (I/O-based).) *For an I/O-based system Sys_{IO} the set of output trajectories $E \subseteq \mathcal{O}^{\mathbb{R}_+}$ is attractive w.r.t. the set of input trajectories $C \subseteq \mathcal{I}^{\mathbb{R}_+}$ and the topologies on $\mathcal{O}^{\mathbb{R}_+}$ and $\mathcal{I}^{\mathbb{R}_+}$ iff $\exists \alpha \in N(C). \forall c \in \alpha. \forall \beta \in N(E). \forall e \in Sys_{IO}(c). \exists t. e\downarrow_{[t,\infty)} \in \beta\downarrow_{[t,\infty)}$ (first line of Table 3).*

This requires that there is a neighborhood α of the input trajectories C such that for every input in α and for every neighborhood β of the output trajectories E the system's reaction to this inputs is such that for each of the system's output trajectories there is a point in time t such that the output trajectory is in β for the remaining time $[t, \infty)$. From a more abstract point of view the formula expresses that for some neighborhood of the input trajectories the system's output converges to the dynamics described by E as time progresses.

The state-based version of attraction of sets of trajectories is similar. Here, however, the set of causes consists of initial states, the set of effects consists of traces in the state space and the definition is parameterized with the set of inputs I for with attraction is supposed to hold (see line 2 of Table 3 for the formula).

3.4.2 Attraction of Points

Attraction of trajectories requires that the evolution of the effects converges to the attractive trajectories. If the attractive trajectory e.g. is a sine function, this means that the system's output must become similar to this sine as time progresses.[2] In contrast attraction of (a set of) points denotes that the effects must reach every neighborhood of the considered points as time progresses. It does not constrain the evolution of the output within this set. In the bottom two lines of the formalizations in Table 3 this becomes apparent in the last column. Here, it is only required that the rest of the output is some trajectory that completely lies within neighborhood β of the set of effects E, which is denoted by considering all trajectories over β, $\beta^{\mathbb{R}_+}$, in the formula. Hence, like for stability, attraction of points can be regarded as geometric notion of attraction whereas attraction of trajectories may be seen as a kinematic notion.

We do not go into detail on the I/O-based version of attraction of (sets of) points which is formalized in the third line of Table 3, but focus on the state-based version now, because it is encountered most frequently in control theory.

State-based attraction of points. For state-based systems attraction of points is defined as follows.

Definition 7 (Attractive set.) *The set* $A \subseteq \mathcal{S}$ *is* attractive *w.r.t. the topology* $\mathcal{O}_{\mathcal{S}}$ *on* $\mathcal{S}$ *and the inputs* $I \subseteq \mathcal{I}^{\mathbb{R}_+}$ *iff* $\exists \alpha \in N(A).\, \forall \iota \in I.\, \forall s \in \alpha.\, \forall \beta \in N(A).\, \forall \sigma \in Sys_S(s, \iota).\, \exists t.\, \forall t' \geq t.\, \sigma(t') \in \beta.$

A *is* globally attractive *iff* $\forall \iota \in I.\, \forall s \in \mathcal{S}.\, \forall \alpha \in N(A).\, \forall \sigma \in Sys_S(s, \iota).\, \exists t.\, \forall t' \geq t.\, \sigma(t') \in \alpha.$

Global attractiveness means that each system behavior remains inside any neighborhood of A eventually for any starting state. Hence, it expresses a kind of convergence to set A. (Normal) attractiveness only requires that there is a neighborhood of A such that system behaviors starting in it exhibit this convergence to A. Again we have parameterized our definition of attraction with a set I of allowed external inputs.

In topologies where A is open global attractiveness is equivalent to the property that A is already reached in finite time ($t \in \mathbb{R}_+$) and not left again, i.e. to $\forall \iota \in I.\, \forall s \in \mathcal{S}.\, \forall \sigma \in Sys_S(s, \iota).\, \exists t.\, \forall t' \geq t.\, \sigma(t') \in A$. Hence, the notion of asymptotically approaching A expressed via the neighborhoods in attraction is replaced by truly reaching A eventually in this property (Theorem 6 in Appendix A). Again, this suits well to discrete systems. The idea of attraction is that we are ensured to get arbitrarily close to the attractive set A. For discrete systems all states different from A can be regarded as being far away from it. The only states close to A are those inside it. Thus, for topologies where A is open attraction is a *persistence property* [CMP91] when regarded from a computer science point of views, i.e. it expresses that "*eventually* the system *always* remains in A". Persistence properties contain a safety as well as a liveness part. In the example of Section 5.2 we will encounter a discrete system with a persistence property which can be expressed as attraction of a set A on a topology where A is open.

Definition 8 (Asymptotic stability.) A *is called* asymptotically stable *iff it is stable and attractive.* A *is called* asymptotically stable in the large *iff it is stable and globally attractive.*

Note that in our general setting attraction does not imply stability. The reason is that attraction is interested in system behavior as time goes to infinity, whereas stability considers the whole time axis. However, for most control systems studied in practice attraction implies stability. Therefore, many control theory text books do not introduce the notion of attraction for its own, but only introduce asymptotic stability.

Asymptotic stability in the sense of Liapunov is a special case of the definition given here for deterministic systems and the usual metric on the real line. For a stable point x asymptotic

[2] The notion of similarity results from the underlying topology on the set of causes.

stability usually requires that the following holds [Lue79]: $\exists \delta > 0. \forall x_0. |x - x_0| < \delta \Rightarrow lim_{t \to \infty} Sys_S(x_0)(t) = x$, where Sys_S is deterministic and gets no external input. To see that this is a special case of the definition in line four of Table 3, we need to take $\{x\}$ as the considered set of causes and effects, and employ the usual topology on the real line as outlined in Section 3.3.2. As $lim_{t \to \infty} Sys_S(x_0)(t) = x$ is equivalent to $\forall \epsilon > 0. \exists t'. \forall t \geq t'. |Sys_S(x_0)(t) - x| < \epsilon$, the correspondence to our definition is straightforward.

3.5 Further Properties

Whereas the properties above are rather general and meaningful for most systems, there are a lot of further, more detailed requirements on individual systems. Classes of such properties will be discussed in the following.

3.5.1 Universal Properties

A characteristic of many properties is that they constrain *all* possible behaviors of a system. Formally, such properties can be written as $\forall c \in \mathcal{C}. \forall e \in Sys(c).(c, e) \in \Phi$, where $\Phi \subseteq \mathcal{C} \times \mathcal{E}$. In computer science properties of this kind are often formalized with linear time temporal logics (LTL) [MP92].

A very important example for such properties are invariants which demand that a certain time-independent constraint be always satisfied [CMP91]. Typically invariants constrain the value of variables. An example is the property that the temperature in a nuclear reactor may never exceed a certain threshold. Further examples for properties expressable in LTL are bounded response requirements, and safety and liveness formulas in general [CMP91].

3.5.2 Existential Properties

Existential properties can be divided into two relevant groups. First, we have properties which demand that certain behavior exists. Such properties involve existential quantification over causes and effects (see (1) below). Second, there are properties which require that causes exist which are guaranteed to lead to some desired effect. For these properties existential quantification over the causes and universal quantification over the effects is involved (see (2) below).

(1) Existence of behavior. The only property of this kind which was regarded in some of the evaluated case studies is the existence of periodic behavior when there are no external disturbances. For instance, this property is relevant for the hopping robot [BGM93], which is required to hop constantly as long as it is not perturbed from the outside.

For state-based system descriptions the existence of periodic behavior can formally be written as follows

$$\exists s_0. \exists \sigma \in Sys_S(s_0, 0). \exists t. \exists \tau \in \mathcal{O}^{[0,t)}. \sigma = \tau^\infty$$

where $0 \in \mathcal{I}^{\mathbb{R}_+}$ denotes a neutral input, i.e. no external disturbances, $\mathcal{O}^{[0,t)} = [0, t) \to \mathcal{O}$, and τ^∞ denotes the trajectory resulting from the infinite repetition of τ. An I/O-based version of this property can be given in a similar manner. Such periodic trajectories in reaction to neutral input are called *limit cycles* [Lei87]. In practice, the non-existence of stable limit cycles is sometimes used as an indirect evidence for the stability of the invariant sets of non-linear control systems. [Föl87] states that, as a rule of thumb, the invariant sets of non-linear control systems are asymptotically stable in the large if no stable limit cycles exists.

In an analogy to computer science the existential path quantifier of CTL [Eme90] can be used to express similar existential properties.

(2) Existence of Causes. In the regarded case studies no property of this type was examined. However, the properties of *controllability* and *observability*, which are important for state based controller design [Oga87, Föl90, Son90, PH88], are of this kind. In contrast to all the properties regarded before they only refer to the plant, not to the whole control system, and they presuppose a state based model of the plant. Thus, the term system refers to the plant in the remainder of this paragraph.

Controllability means that there is an input trajectory such that a certain system state can be reached from the initial state within finite time. In an adaption from [Son90] to nondeterministic systems, we can define controllability from state set S_1 to state set S_2 as follows:

$$\exists t.\, \exists \iota \in \mathcal{I}^{\mathbb{R}_+}.\, \forall s \in S_1. Sys_S(s,\iota)(t) \subseteq S_2$$

Note that the input trajectory after time t is irrelevant for controllability, since we are dealing with time guarded systems. If S_2 is a one element set, this is similar to the definition of controllability for deterministic systems [Son90], because we required that Sys be total in its inputs. A system is called *fully controllable* iff any system state can be controlled to any other system state.

Observability denotes that for any two distinct states there is an input such that a finite observation of the system output suffices to detect the distinctness of the states. Like in our treatment of controllability we define observability (or *distinguishability*) of sets of system states first. Two disjoint sets of states S_1 and S_2 are distinguishable iff

$$\forall s_1 \in S_1.\, \forall s_2 \in S_2.\, \exists \iota.\, \exists t.\, (Sys_S(s_1); Out^\dagger)(\iota)(t) \cap (Sys_S(s_2); Out^\dagger)(\iota)(t) = \emptyset$$

I.e. for any two differing start states s_1 and s_2 from S_1 and S_2 there exists an input such that the observable output of the system starting in s_1 is disjoint from that of the system starting in s_2 after some finite time. Note that disjointness is required to ensure that *all* nondeterministic choices Sys_S can make for the two start states are different. Due to time invariance observability provides that states from S_1 and S_2 can also be distinguished at any later time instant, not only when they are initial states. Finally, a system is called *fully observable* if every two distinct states can be distinguished, formally:

$$\begin{aligned}&\forall s_1, s_2 \in \mathcal{S}.\, \exists \iota.\, \exists t \in \mathbb{R}_+.\, s_1 \neq s_2 \Rightarrow \\ &\qquad (Sys_S(s_1); Out^\dagger)(\iota)(t) \cap (Sys_S(s_2); Out^\dagger)(\iota)(t) = \emptyset\end{aligned}$$

If a plant is not controllable or not observable we cannot design controllers which are able to drive it into every desired state, because either the desired state is unreachable from the current state of the plant, or we cannot determine the current state due to lacking observability, or both. In this sense unobservability can be seen as a lack of (appropriate) sensor information, and uncontrollability can be interpreted as a lack of (appropriate) actuators. Such a lack may be caused by the underlying physics, for instance some quantities may be difficult to measure, but also by economic constraints, like too expensive actuators. Controllability and observability are central prerequisites for state-based design techniques of controllers [Oga87]. If applicable, such methods e.g. allow to compute stable optimal controllers for a certain purposes. However, they have not been considered in any of our case studies.

In computer science properties of this kind can be expressed with *alternating-time temporal logic (ATL)* as defined in [AHK97]. In contrast to CTL, ATL allows to distinguish between the system and its environment in path quantification. Thus, we can express that for all possible moves of the system the environment can select moves (or inputs in our context) which ensure that a certain property holds.

3.6 Classification of the Properties and its Consequences

3.6.1 Classification

The classification of properties we propose is based on the semantic models relative to which the validity of the properties is defined.

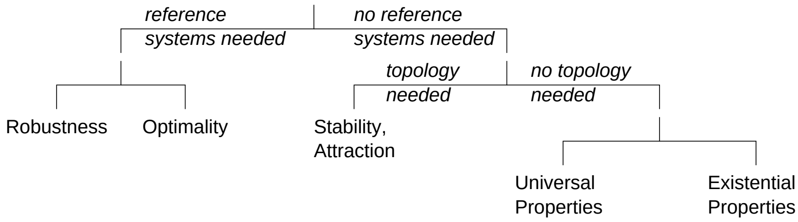


Figure 2: Classification of properties.

Robustness	3
Optimality	2
Stability	6
Attraction	1
Universal properties	8
Existential properties	2

Table 4: Number of times a property was referenced at last once in a case study.

The validity of robustness and optimality of a system must be determined relative to a set of systems (Fig. 2, left branch). For robustness, an evaluation function for the regarded object property is necessary additionally. Optimality instead requires a cost function. For the other properties in Section 3 no reference systems are needed to determine their validity (Fig. 2, right branch). Determining stability or attraction requires that topologies for the input and output space (or the state space, respectively) of the regarded system are given (Fig. 2, left branch at the second level). For the properties of Section 3.5 no topologies are necessary to determine their validity (Fig. 2, right branch at the second level). The properties of Section 3.5 are all evaluated in the same domain, namely w.r.t. a given system. As already indicated in the previous section, we partition this subclass further into the set of properties which constrain all behaviors of a system (universal properties) and the set of properties which demand existence of some specific behavior (existential properties). The existential properties can furthermore be divided into those demanding the existence of some behavior and those demanding the existence of causes enforcing certain effects (see Section 3.5.2).

3.6.2 Rating

In order to get a vague estimate of the relevance of the listed properties in practice, for each kind of property the number of case studies in which at least one property of that kind was considered was counted. For instance, if case study A considered three universal properties and two stability properties, the counter for the number of case studies considering universal properties and the counter for those considering stability properties are increased by one. Hence, the maximum possible count for a property class is nine, the number of all evaluated case studies. We do not use the *total* number of times a certain property class was considered for our rating, because this would assign inadequately high numbers to invariance properties, since a greater number of properties of this kind were listed in two of the case studies [BHKT98, Eng97].

Table 4 lists the resulting rating. Although attraction is considered in only one case study it probably also is relevant in the case studies which only considered stability, because in cases where stability is desired, attraction of the stable set usually is also desirable. The eight case studies regarding universal properties all regarded invariance properties (Section 3.5.1). Two of them also regarded an invariance property for a reduced set of causes $\mathcal{C}$. Namely, they required that for certain kinds of causes the system's state or output variables never exceed given values. In all two case studies mentioning or examining existential properties this regarded existential property was the existence of limit cycles.

Although the significance level of the given rating is low because only nine were examined, we think that there is a clear tendency indicating that universal properties, in particular invariance properties, and stability are most important. This clearly does not imply that the other properties are irrelevant, but it indicates that it is sensible to put priority on work on methods for establishing or maintaining these properties. Another possible conclusion is that these properties were regarded so frequently, because the best analysis methods exist for them. We think this can only be true by part. First, some of the regarded case studies merely explain a system and desirable properties without validating them, or, besides validating some properties, describe others as also important, but do not analyze them. These mentionings of properties also appear in our rating. In this respect the case studies only partially depend on existing methods. Second, the analysis method used most frequently in the case studies was simulation based testing which can in principle be applied to all the properties. Nevertheless, note that from a theoretical point of view tests are not helpful w.r.t. unbounded liveness properties or for (infinite time) attraction, since in theory judging the correctness of an output would require infinite observation.

3.6.3 Consequences for Refinement

Here we want to consider which properties are maintained under *refinement*. The notion of refinement we employ is based on set inclusion, i.e. we say that relation A is a refinement of relation B, written as $A \preceq B$, iff $A \subseteq B$. Consequently, for I/O based system descriptions this means that $Sys'_{IO} \preceq Sys_{IO}$ iff $\forall \iota \in \mathcal{I}.\, Sys'_{IO}(\iota) \subseteq Sys_{IO}(\iota)$. For state-based system descriptions we have $Sys'_S \preceq Sys_S$ iff $\forall s \in \mathcal{S}, \iota \in \mathcal{I}.\, Sys'_S(s, \iota) \subseteq Sys_S(s, \iota)$. This expresses that the traces of the original system include those of the refined system, while both systems must be total in their input (and start states) because of our definition of systems (Section 2). Note that this notion of refinement is common in computer science. For instance, it is studied in [Bro99, Bro97].

From the definition of universal properties it is obvious that they are preserved under this notion of refinement. For stability and attraction it is also easy to see that refinement maintains them. Simply note that for a refined system we can choose the same neighborhoods of causes in a proof of stability or attraction as for the original system and end up with a subset of the traces of the original system for the same causes (and inputs, in the case of state-based system descriptions) then. Hence, traces in this subset are also contained in all the sets and neighborhoods of sets in which the original trace set is.

Similarly, controllability and observability are preserved under refinement. Informally the reason is that they involve existential quantification over the input and (implicit) universal quantification over the output. As systems, whether refined or not, are required to be total in their input, inputs which provided controllability or observability for the original system also do so for the refined system with its smaller, but non-empty, set of traces for the same inputs.

Existential properties requiring the existence of effects, like e.g. the existence of periodic behavior or limit cycles (Section 3.5.2) are not maintained by refinement. Again this is straightforward, because in the refined system exactly the "good verdict" may have been removed from the possible effects of the original system for given causes (and input).

The situation is more difficult for the properties robustness and optimality, because we also have to consider sets of reference systems, the object property and the cost function, respectively, here. Let us first define what refinement means for reference systems. A set of reference systems R' is called a refinement of a set R of reference systems iff $\forall Sys' \in R'. \exists Sys \in R. Sys' \preceq Sys$. Thus, we require that for each system Sys' in the refined set there is a system in the original set which is refined by Sys'. Based on this definition the following is an immediate consequence of the definition of robustness, because the definition involves universal quantification over the reference systems. Provided system Sys with deviations D, $Sys \in D$, robustly satisfies Ψ and provided that Ψ is preserved by refinement, then the refined system Sys' with refined deviations D', $Sys' \in D'$, also robustly satisfies Ψ. As we have seen above Ψ is preserved under refinement if, for instance, it is a universal property.

Optimality in general is not preserved by refinement. Here, the character of the cost function is crucial. For instance, a reasonable class of cost functions are functions which determine a system's quality as the supremum of the cost of all possible effects for given causes (and input). If we only refine the optimal system, but not the set of alternative systems, optimality is maintained, since the cost for the refined system can only have decreased while costs for the alternative systems remained equal. However, if the alternative systems are refined optimality of a given system need not be preserved w.r.t. the new alternatives. Namely, there could be a refined alternative system from which expensive effects have been removed such that this system is now preferable to the old optimal system (or even to a refined version of it).

Contrasting these results with our rating of properties yields that the chosen refinement notion preserves most central properties. Merely existential properties with existential quantification over effects are not maintained and for robustness and optimality the situation depends on the object property and the cost function, respectively.

4 Some Proof Concepts

In practice we have to distinguish between two kinds of applications of hybrid systems. First, there may be a control task that is realized by the hybrid system, e.g. guaranteeing that a certain physical quantity has a prescribed value. We also call such a system a *control centered hybrid system*. Second, the system's task may be to implement a certain process, e.g. to control the appropriate consecution of different process phases such as heating material, shaping it and cooling it again. We also call such a system a *process centered hybrid system*. Note that this is not a clear distinction, control centered hybrid systems may appear within larger process centered systems. Similarly, process centered systems may be a part of control centered systems. Robotic hand regrasping is an example for such a hierarchy [SHB+99]. The overall task is to guarantee that an object is held in a stable position by a robotic hand. A subtask consists of moving some fingers in order to obtain a better grasp. For both kinds of systems different classes of properties are important. For control centered hybrid systems stability and attraction are vital properties. In process centered control systems safety properties, like "buffer capacities are never exceeded", and liveness properties, like "the next phase is started eventually", play a major role. Here, safety and liveness properties are not considered in greater detail, because their underlying proof principles are largely familiar to computer scientists. Besides that, there already is significant work on this topic, like e.g. [Pnu94] where *computational induction* is introduced to prove safety properties of hybrid systems or [Lam93] where TLA (*Temporal Logic of Actions*) with its existing proof methods is extended to hybrid systems.

In the following we will present proof methods for state-based stability and attraction of (sets of) points. This kind of stability and attraction is most extensively studied in standard text books on control theory. Furthermore, we will give specializations of the general methods which are helpful in applications and outline parallels to similar proof methods for discrete and continuous systems. The resulting methods are applied in a small example to prove stability and attraction of the EHC system. To emphasize parallels to computer science and demonstrate the applicability of our methods to discrete systems, we consider a classic self-stabilizing system from computer science as a further example.

4.1 State-based Stability

Proofs of stability in control theory usually consist of finding a continuous[3] monotonously decreasing function from the system states to some assessment space, usually the real numbers, which has a unique minimum for the state whose stability must be shown. According to the work of Liapunov existence of such a *Liapunov function* implies the stability of the unique minimum. In a physical interpretation Liapunov functions can often be regarded as energy functions. They associate the amount of energy in the system with each of its states. If energy is never added by the evolution of the control system it will be stable.[4] From a more abstract point of view, the Liapunov function can be seen as an abstraction that maps the real system to a qualitatively equivalent system [MW95].

The following theorem may be seen as a generalization of the classical Liapunov theory. It is adapted from [MT75]. In the theorem we write $\langle v \rangle$ for $\{v' \mid v' \sqsubseteq v\}$ for partial order $\sqsubseteq$ and f^{-1} for the inverse of a function f. f^{-1} is extended to sets by pointwise extension. As the theorem is rather technical by part, it is explained in the following paragraph and Sections 4.1.1 and 4.1.2 introduce specializations which also help to understand its last two requirements.

Theorem 1 *The set A is stable w.r.t. system $Sys_S \in \mathcal{S} \to \mathcal{I}^{\mathbb{R}_+} \to \wp(\mathcal{S}^{\mathbb{R}_+})$, topology $\mathcal{O}_\mathcal{S}$, and inputs in I if there exists a function $L \in \mathcal{S} \to V$ with:*

1. *V is a partially ordered set with partial order $\sqsubseteq$. $V^+ \subseteq V$ is the subset of elements $v \in V$ for which there is a neighborhood of A such that the inverse image under L of all elements $v' \sqsubseteq v$ is not a proper subset of the neighborhood, formally $V^+ = \{v \in V \mid \exists \alpha \in N(A). \neg(L^{-1}(\langle v \rangle) \subsetneq \alpha)\}$.*
2. *L is monotonously decreasing along the traces of Sys_S, formally* $\forall s \in \mathcal{S}, \iota \in I. \forall (s, \iota, \sigma) \in Sys_S. \forall t, t' \in \mathbb{R}_+. t' \geq t \Rightarrow L(\sigma(t')) \sqsubseteq L(\sigma(t))$.
3. $\forall v \in V^+. \exists \alpha \in N(A). \forall x \in \alpha. L(x) \sqsubseteq v$
4. $\forall \alpha \in N(A). \exists v \in V^+. \forall x. L(x) \sqsubseteq v \Rightarrow x \in \alpha$

Proof. Let $\alpha \in N(A)$ be an arbitrary neighborhood of A. Application of 4 and 3 yields that there is a $\beta \in N(A)$ and a $v \in V^+$ with $x \in \beta \Rightarrow L(x) \sqsubseteq v$, and therefore $x \in \alpha$. For $x \in \beta$ and $(x, \iota, \sigma) \in Sys_S$ monotonicity yields $\forall t. L(\sigma(t)) \sqsubseteq v$. Hence, $\sigma \in \alpha^{\mathbb{R}_+}$. $\square$

If existing, such a function L is called a *Liapunov function*. Informally the combination of the last two requirements expresses that for any neighborhood of A there exists a smaller neighborhood whose L-image is bounded from above by some $v \in V^+$. In terms of the inverse images under L the last two requirements can equivalently be written as $\forall v \in V^+. \exists \alpha \in N(A). \alpha \subseteq L^{-1}(\langle v \rangle)$ and $\forall \alpha \in N(A). \exists v \in V^+. L^{-1}(\langle v \rangle) \subseteq \alpha$, respectively. The set V^+ eliminates all those elements from V which are not helpful in the proof of stability, because they constrain the considered sets of points too much, namely to the inside of all neighborhoods of A. V^+ is needed to simplify the application of the theorem. Namely,

[3]Unless otherwise mentioned, we use to the topological definition of continuity.

[4]Note that adding energy does not violate the physical law of energy preservation, because the control system may use external sources for increasing the energy.

specializations of the theorem usually use sets V with bottom element $\bot$ and mappings L with $L^{-1}(\bot) = A$. If requirement three quantified over all $v \in V$, it would not be satisfiable for these specializations in topologies on $\mathcal{S}$ in which A is a closed set.

Note that the above theorem has some parallel to the proof method given in [Sin92] to prove *atomicity* of an invariant, i.e. to prove that the invariant set of a dynamical system is a singleton. This parallel is due to both proof methods using a comparison function L to show that the considered system has a non-expansive (in the case of stability) or contractive character (in the case of atomicity).

In the following we consider two specializations which are helpful for applications and which make parallels to the notion of abstraction in computer science explicit.

4.1.1 Liapunov Functions and Galois Connections

The last two requirements in Theorem 1 suggest that there are two total mappings from V^+ to the neighborhoods of A (Requirement 3) and back again (Requirement 4). Such mappings are similar to abstraction via Galois connections [CC92], which is a common technique in computer science. Informally a Galois connection is a pair of monotonous mappings, an *abstraction function* and a *concretisation function*, from a "concrete" partially ordered set to an "abstract" partially ordered set and back again. The compositions of these mappings are required to loose information in a way consistent with the regarded partial orders. We will make the similarity between Liapunov functions and Galois connections explicit now. This idea leads to a specialization of Theorem 1 which is developed in the following.

Assumptions. Let $(V, \sqsubseteq)$ be a complete partial order which furthermore is densely ordered by the strict version $\sqsubset$ of $\sqsubseteq$ and has a least element $\bot$. Let L be a mapping from $\mathcal{S}$ to V such that the given topology on $\mathcal{S}$ is the coarsest topology which makes L continuous w.r.t. the interval topology induced by $\sqsubseteq$ on V (see Appendix C for the definition of interval topology). L is required to be onto and the L image of all elements in A must be $\bot$. Furthermore, V and L must be chosen such that $V^+ = V \setminus \{\bot\}$, where V^+ is defined as in Theorem 1.[5] We define function $abs \in N(A) \to V^+$ by $abs(\alpha) = sup\{v \in V^+ \mid \alpha \supseteq L^{-1}(\langle v \rangle_o)\}$, where $\langle v \rangle_o$ is the set of all those v' which are strictly "less than" v, $\langle v \rangle_o = \{v' \mid v' \sqsubset v\}$. Function $conc \in V^+ \to N(A)$ is given by $conc(v) = L^{-1}(\langle v \rangle_o)$. Lemma 1 in Appendix A proves that *abs* and *conc* are well-defined.

Theorem 2 *Functions abs and conc are a* Galois connection *between the spaces* $(N(A), \supseteq)$ *and* $(V^+, \sqsupseteq)$*, where* $\sqsupseteq$ *is given by* $v \sqsupseteq v' :\Leftrightarrow v' \sqsubseteq v$.[6]

By the definition of Galois connections [CC92] this means that monotonicity of *abs* and *conc*, and $\alpha \supseteq conc(abs(\alpha))$ (*extensivity*) and $abs(conc(v)) \sqsupseteq v$ (*reductivity*) must be proven. This in done in Theorem 7 of Appendix A. In fact, a stronger variant of reductivity, $abs(conc(v)) = v$, is proven there, which implies that *abs* and *con* even are a *Galois surjection* on the considered spaces.

Specialization for Stability. The given definition of *abs* and *conc* allows to derive stability of A from the monotonicity of L w.r.t. the system traces of Sys_S. The proof is given in Appendix A, Theorem 8. This amounts to a specialization of Theorem 1, as a L function with the properties described in the assumptions above also satisfies the requirements of that theorem. Namely, *abs* and *conc*, which are defined depending on L, immediately help to satisfy Requirements 3 and 4 of Theorem 1.

[5] This can be achieved by choosing L such that $L^{-1}(\bot) = A$.

[6] Actually *abs* and *conc* make up a *dual* Galois connection [CC92], because in the usual terminology of [NNH98] and [CC92] our *abs* is the concretisation and *conc* the abstraction.

Note that monotonicity of L can also be interpreted as stability of $\perp$ w.r.t. the L-image of Sys_S in V. We will consider this in more detail in Section 4.1.2. Furthermore, note that the stability proof also works if V is not complete and densely ordered, but just a partially ordered set with bottom element. In this case the supremum operator, *sup*, in the definition of *abs* must be replaced by Hilbert's non-deterministic choice operator and *abs* no longer needs to be monotonic w.r.t. $\supseteq$. Hence, *abs* and *conc* no longer build a Galois connection. Besides that, to proof stability L need not be onto. This is only necessary to show that *abs* and *conc* are a Galois connection.

Using the interval topology on V together with the other requirements we stated for V and L implies that $L^{-1}(\perp)$ is a closed set in $\mathcal{S}$. If we choose L such that $L^{-1}(\perp) = A$, this suits well to standard control theory books. There, Liapunov functions usually use $V = \mathbb{R}_+$, the considered set A is a singleton, and therefore closed w.r.t. the natural topology on the Euclidean space, and $L(x)$ has its unique minimum for $\{x\} = A$.

Thinking in terms of abstraction and concretisation functions can help us to find Liapunov functions. Namely, they lead our intuition to looking for equivalence classes in $\mathcal{S}$ whose elements are then defined to produce the same L-value. In our view it is more constructive to search for a continuous L function such that the above *abs* is well-defined than to find a function satisfying the last two requirements of Theorem 1, since these requirements are very abstract.

4.1.2 Liapunov Functions and Homeomorphisms

A further specialization of Theorem 1 results if we demand that L be a *homeomorphism*, i.e. a function which is one-to-one and onto and has a continuous inverse L^{-1}. In detail the assumptions are as follows.

Assumptions. Let V be partially ordered by $\sqsubseteq$ and have a least element $\perp$. Let $L \in \mathcal{S} \to V$ be a homeomorphism, where continuity of L^{-1} is required w.r.t. the topology on $\mathcal{S}$ and the left topology on V (defined in Appendix C, Example 2). This implies that L also is continuous w.r.t. the same topologies [Eng89]. Furthermore, L must be monotonous w.r.t. the traces of Sys_S (see requirement two in Theorem 1), and it must map all elements in A to $\perp$. Note that together with L being one-to-one and onto this implies that A may only contain one element. For V^+ defined as in Theorem 1 the above requirements for V and L imply that $V^+ = V$ holds, since $\langle\perp\rangle$ is an open set w.r.t. the left topology and $L^{-1}(\langle\perp\rangle) \in N(A)$ therefore is the smallest neighborhood of A.

Remark that in contrast to the previous paragraph we here employ the left topology on set V instead of the interval topology. We do so in order to give an example of a class of L functions for which $L^{-1}(\perp)$ is an open set, which suits well in cases where A is open and $L^{-1}(\perp) = A$. Nevertheless, the arguments presented here can easily be adapted to totally ordered sets V with the interval topology.

Specialization for Stability. Theorem 9 in Appendix A shows that these properties imply the requirements of Theorem 1. As a consequence the above properties are sufficient to conclude the stability of A w.r.t. Sys_S and the topology on $\mathcal{S}$, and we have a specialization of Theorem 1.

Abstraction. From a computer science perspective L can be seen as an abstraction mapping the *concrete* system Sys_S to an *abstract* system $AbsSys \in V \to \mathcal{I}^{\mathbb{R}_+} \to \wp(V^{\mathbb{R}_+})$ such that L *commutes* between Sys_S and $AbsSys$ (see Figure 3) and furthermore preserves topological properties on $\mathcal{S}$ and V. Such a mapping is called a *(topological) conjugacy* in [Aki93]. Here, the abstract system $AbsSys$ is defined as $AbsSys(v, \iota) = L(Sys_S(L^{-1}(v), \iota))$

$$
\begin{tikzcd}
\mathcal{S} \arrow[r, "\mathrm{Sys}(.,\iota)"] \arrow[d, "\mathrm{L}"'] & \mathcal{P}(\mathcal{S}^{\mathcal{R}_+}) \arrow[d, "\mathrm{L}"] \\
\mathrm{V} \arrow[r, "\mathrm{AbsSys}(.,\iota)"'] & \mathcal{P}(\mathrm{V}^{\mathcal{R}_+})
\end{tikzcd}
$$

Figure 3: Commuting diagram for L.

where L is extended to trajectories and sets of trajectories in a pointwise and elementwise way. L commutes between $Sys_{\mathcal{S}}$ and $AbsSys$, i.e. $L(Sys_{\mathcal{S}}(s,\iota)) = AbsSys(L(s),\iota)$, because of the definition of $AbsSys$ and because L is one-to-one and onto. In our context, stability of $L(A)$ for the abstract system $AbsSys$ implies stability of A for the concrete system Sys. Due to using the left topology on V stability of $L(A)$ is a consequence of monotonicity of L w.r.t. the traces of $Sys_{\mathcal{S}}$. To see this, note that stability of $L(A) = \{\bot\}$ can be written as $\forall v.\, \exists v'.\, \forall \iota \in I.\, \forall v'' \sqsubseteq v'.\, \forall \eta \in Abs(v'',\iota).\, \forall t.\, \eta(t) \sqsubseteq v$, where we implicitly associate the neighborhood $\langle v \rangle$ of $\{\bot\}$ with every $v \in V$. By the definition of $AbsSys$ this formula for stability of $\{\bot\}$ is a direct consequence of the monotonicity of L along the traces of $Sys_{\mathcal{S}}$ (requirement 2 of Theorem 1).

We remark that, as stability depends on topology, it is not surprising that studying stability properties by abstraction requires to also relate the topologies on the abstract and the concrete space which is done by a homeomorphism and also by the particular Galois connection given above. In case of the homeomorphism continuity of L and L^{-1} establish the needed relationship between open sets in both spaces, while in the case considered in Section 4.1.1 continuity of L establishes the connection from open sets in the abstract space to open sets in the concrete space and requiring that the topology on the concrete space is the coarsest which makes L continuous establishes the other direction.

Note that finding a L function which defines a Galois connection as explained in Section 4.1.1 can be regarded as a greater abstraction than finding a homeomorphism, because in the case of a homeomorphism the concrete space $\mathcal{S}$ and the abstract space V are isomorphic. In contrast an L function that (only) leads to a Galois connection allows to identify elements of the concrete space in the abstract space. Hence, information is really lost in this case.

4.2 Attraction

Control theory usually considers attraction together with stability, i.e. asymptotic stability. Like in Section 4.1 Liapunov functions are also used in order to proof asymptotic stability. However, for asymptotic stability the functions must obey additional restrictions which ensure that the Liapunov function indeed reaches a certain unique minimum. For continuous time systems a sufficient criterion is that L is continuously differentiable with the derivative being strictly less than zero everywhere except of at the minimum [Lue79].

In our general framework, where attraction does not necessarily imply stability, the following theorem allows us to proof attraction of points for state-based systems. It corresponds to Theorem 1 with a different third requirement.

Theorem 3 *The set A is globally attractive w.r.t. system $Sys_{\mathcal{S}} \in \mathcal{S} \to \mathcal{I}^{\mathbb{R}_+} \to \wp(\mathcal{S}^{\mathbb{R}_+})$, the topology $\mathcal{O}_{\mathcal{S}}$ and the inputs in I if there exists a function $L \in \mathcal{S} \to V$ with:*

1. *V is a partially ordered set with partial order $\sqsubseteq$. $V^+ \subseteq V$ is the subset of elements $v \in V$ for which there is a neighborhood of A such that the inverse image under L of*

all elements $v' \sqsubseteq v$ is not a proper subset of the neighborhood, formally $V^+ = \{v \in V \mid \exists \alpha \in N(A).\, \neg(L^{-1}(\langle v \rangle) \subsetneqq \alpha)\}$.

2. *L is monotonously decreasing along the traces of Sys_S, formally $\forall s \in \mathcal{S}, \iota \in I. \forall \sigma \in Sys_S(s,\iota). \forall t, t' \in \mathbb{R}_+.\, t' \geq t \Rightarrow L(\sigma(t')) \sqsubseteq L(\sigma(t))$.*

3. *$\forall v \in V^+, s \in \mathcal{S}, \iota \in I. \forall \sigma \in Sys_S(s,\iota). \exists t.\, L(\sigma(t)) \sqsubseteq v$.*

4. *$\forall \alpha \in N(A). \exists v \in V^+. \forall x.\, L(x) \sqsubseteq v \Rightarrow x \in \alpha$.*

Proof. Let $\alpha \in N(A)$ be an arbitrary neighborhood of A and $s \in \mathcal{S}$, $\iota \in I$ arbitrary. Application of 4 and 3 yields that for all $\sigma \in Sys_S(s,\iota)$ there is a t such that $L(\sigma(t)) \sqsubseteq v$. Monotonicity yields $L(\sigma(t')) \sqsubseteq v$ for all $t' \geq t$ and hence $\sigma(t') \in \alpha$ for all $t' \geq t$. $\square$

Similar to Theorem 1 V^+ is needed here, because otherwise the theorem would be too restrictive for the frequently occurring case where V contains a bottom element $\perp$ with $L^{-1}(\perp) = A$ and A is a closed set. Requirement three expresses that any $v \in V^+$ is eventually reached by all trajectories of Sys_S. Obviously, the existence of a Liapunov function $L \in \mathcal{S} \to V$, as defined by Theorem 1, which also satisfies requirement three of Theorem 3, proves asymptotic stability of the considered set A. Hence, we can also use Galois connections of the kind defined in Section 4.1.1 or homeomorphisms as defined in Section 4.1.2 to prove attraction, provided they also satisfy requirement three from above.

Theorem 3, in general, does not imply Theorem 1. Informally the reason is that for attraction a function L suffices which, for every trace, is constant in the beginning and converging to a specific minimum as time goes to infinity. In particular if the chosen function L does not fit to the topology on $\mathcal{S}$ in the way demanded by the third requirement of Theorem 1, the value of L need not yield any information on the neighborhood of A in which a system trace is at a certain time instant.

Note that requirements two and three of the theorem together yield that $\forall v \in V^+, \iota \in I, s \in \mathcal{S}. \forall \sigma \in Sys_S(s,\iota). \exists t. \forall t' \geq t.\, L(\sigma(t')) \sqsubseteq v$ holds, which exactly is what is needed in the proof. If V has a least element $\perp$ and $V^+ = V$ this property is equivalent to global attraction of $\{\perp\}$ w.r.t. the left topology on V induced by $\sqsubseteq$ (see Definitions 7 and Appendix C, Example 2).[7] This underlines that, just as for stability, L is an abstraction mapping which allows to carry over attraction in V to attraction in $\mathcal{S}$.

4.2.1 Specializations and Parallels to Discrete and Continuous Systems

A problem for the practical utility of Theorem 3 is that requirement three is not very handy. In fact proving convergence, which is expressed in the requirement, usually is difficult. This, however, can be alleviated by selecting a set V for which there is a rich theory for convergence. We therefore want to consider a few specializations here.

Discrete systems. In the context of discrete-time and discrete-event systems and if A is open w.r.t. the topology on $\mathcal{S}$, Theorem 3 can be specialized as follows. We choose V as a well-founded set with bottom element, e.g. $V = \mathbb{N}$ with bottom element $\perp = 0$ and $\sqsubseteq$ the less or equal relation $\leq$. The inverse image of $\perp$ under L must be A which implies $V^+ = V$. We then require that for every trace there is an infinite, strictly monotonously increasing sequence of time instants $t_i \in \mathbb{R}_+$ at which the system performs its moves. L must be monotonously decreasing along the traces of Sys_S and strictly decreasing at every t_i unless the L-images of the states reach $\perp$ (from then on monotonicity suffices). As our traces σ are infinite and as well-founded sets only contain finite descending sequences, it

[7] A similar result holds for the interval topology.

follows that $L(\sigma)$ becomes constantly $\perp$ starting from some time t, i.e. convergence of the L-image of Sys_S to $\perp$ is ensured. Hence, these requirements imply requirements two and three of Theorem 3. Requirement four is a consequence of $L^{-1}(\perp) = A$, because for any neighborhood of A we can choose $v = \perp$ to satisfy the requirement. This specialization is applied in the example of Section 5.2. Note that this kind of L functions correspond to Floyd functions used in computer science to prove liveness properties [Cou90, Flo67], as [Sin92] mentions in a discrete time context. However, in comparison to Floyd functions we also require that L remains non-increasing when set A has been reached. This is needed to also proof the safety part contained in a persistence property, which is to what attraction corresponds to from a computer science point of view (Section 3.4.2). In the example of Section 5.2 this becomes apparent.

Continuous-time systems. Classical strictly monotonous Liapunov functions are a second special case of the above theorem [Lue79]. Here, the considered set A is a singleton and therefore closed w.r.t. the natural topology on the Euclidean space used for $\mathcal{S}$. $V = \mathbb{R}_+$ with $\perp = 0$, for $\sqsubseteq$ the less or equal relation $\leq$ is used, and $V^+ = V \setminus \perp$. L is required to be continuously differentiable along the traces of Sys_S with continuous derivative $\frac{d}{dt}L(\sigma(t)) < 0$ for all $\sigma(t) \notin A$ and $\frac{d}{dt}L(\sigma(t)) = 0$ for $\sigma(t) \in A$. Furthermore, L must have its unique minimum 0 for A, $L^{-1}(0) = A$. Convergence of L to 0 is a consequence of the strict monotonicity of L *and* the continuity of $\dot{L}$. Strict monotonicity alone only implies convergence to *some* value. Requirement four is satisfied for L, because of continuity of L by topological arguments roughly similar to those given in Section 4.1.1.

Specialization with uniform monotonicity. For topologies in which A is not open a similar more general specialization of the above theorem is obtained by again using $V = \mathbb{R}_+$, but by requiring a kind of uniform monotonicity of L along the traces of Sys_S. This specialization is an adaption of the criterion given in [SG95] for attraction for discrete-time systems. Here, L must be chosen such that $L^{-1}(0) = A$ and $V^+ = V \setminus \{0\}$ holds. Then, requirements two and three of Theorem 3 are replaced by the following: $\forall s \in \mathcal{S}, \iota \in I. \forall \sigma \in Sys_S(s, \iota). \exists k \in [0, 1). \forall t, t' \in \mathbb{R}_+. t' \geq t \Rightarrow L(\sigma(t')) \leq k^{t'-t} L(\sigma(t))$ Obviously, this implies monotonicity of L along the traces of Sys_S (second requirement of Theorem 3). To see that any $v \in V^+$ is reached or underpassed by any trace (third requirement of Theorem 3), let $v \in V^+$ be arbitrary and let $\sigma \in Sys_S(s, \iota)$ be a trace for start state s and input $\iota \in I$. Due to the above claim, there is a $k \in [0, 1)$ such that $\forall t. L(\sigma(t)) \leq k^t L(\sigma(0))$. For $t \to \infty$ $k^t L(\sigma(0))$ converges to 0 which implies that $L(\sigma(t))$ reaches v or falls below v eventually. Thus, with requirement four of Theorem 3 remaining unchanged this yields a sound specialization of the above theorem.

5 Applications

5.1 Stability and Attraction for the EHC

As example we consider the stability and attraction of the tolerance interval of an electronic height control system (EHC) under certain circumstances explained below. The purpose of this system is to control the chassis level of an automobile by a pneumatic suspension such that the level remains within a given tolerance interval. The abstract model of this system which regards only one wheel was first presented in [SMF97].

The proof of stability and attraction can already be done early in the development. We therefore consider a model with only one component here, which contains a very abstract model of the control logic and the physics of the suspension system. Figure 4 depicts the component's interface. Channel *height* outputs the chassis level, input *dist* denotes external

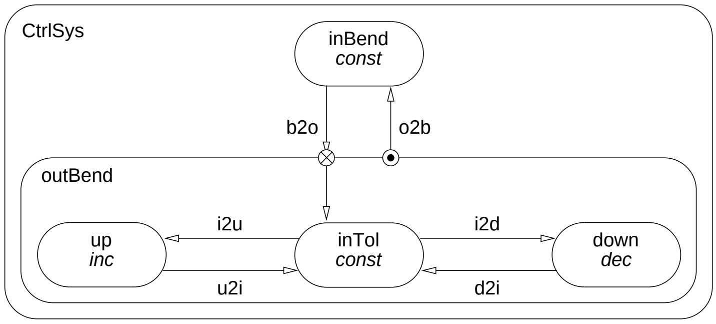
Figure 4: Interface of the EHC system.

disturbances and the toggling of input *bend* reports whether a curve is entered or left. Figure 5 defines its behavior as a HySChart, i.e. as an extended, hierarchic state transition diagram with continuous activities and invariants associated with the states [GSB98, Sta00]. Once stability and attraction is shown, this model can be refined further with rules which we will develop in the future. In particular this will include refining the system into the controller and suspension's dynamics, and discretizing the controller.

The primary aim of the EHC is to compensate different load situations of the car. At the considered stage it is not supposed to actively influence driving dynamics by e.g. adjusting the suspension to different speeds or lateral accelerations. In fact it is explicitly required that the actuators are turned off when the car is going through bends. This explains the two states *inBend*, where control is suspended, and *outBend*, where the controller may modify the chassis level, in Figure 5.

Because of control being suspended in state *inBend* no region A around the tolerance interval can be stable or attractive for all kinds of disturbances. If A (or a given neighborhood of it) has been reached, a curve may be entered and the disturbances occurring in the curve may cause that the chassis level is outside A (or the regarded neighborhood) when the bend ends. This contradicts attraction. Stability is violated in a similar way. Nevertheless, this behavior of the EHC is acceptable if after a disturbance and despite of (finite) bends the chassis level does not diverge further but approaches the tolerance interval again. We summarize this in requiring that set A, whose formal definition we give in the following paragraph, is stable and globally attractive w.r.t. the system of Figure 5 and the set of inputs I which expresses that bends are entered and left infinitely often with entries (exists) and exits (entries) of bends separated by at least time bs, and in which the chassis level is never influenced from outside. Hence, we only consider initial disturbances of the chassis level. As the system is time invariant, this ensures that whenever the chassis level is outside the tolerance interval it approaches the tolerance interval again, provided bends satisfy the claimed assumptions. Note that we forthermore require that $bs > \epsilon_{bend}$ holds, i.e. the time between togglings of *bend* must be greater than the maximum delay with which the system reacts to them.

Definitions. Formally, I is defined as follows: $I = \{(bend, dist) \in (\mathbb{B} \times \mathbb{R})^{\mathbb{R}_+} \mid \forall t \in \mathbb{R}_+ . (t = 0 \vee lim_{x \nearrow t} bend(x) = \neg bend(t)) \Rightarrow \exists b \in \mathbb{B}. \exists t' \geq t + bs. (bend, dist)\downarrow_{[t,t')} = (b, 0)^{\mathbb{R}_+}\downarrow_{[t,t')} \wedge (bend, dist)(t') = (\neg b, 0)\}$ This expresses that input *bend*, i.e. the signal whose toggling reports entry or exit of a bend, is toggled infinitely often with two consecutive togglings separated at least by time bs, and that input $dist$, denoting external disturbances of the chassis level, is constantly 0, i.e. there is no disturbance. Note that set I could also have been defined with some hybrid temporal logic, HySCs, or HySCharts. The proposed attractive and stable set A is $A = \{k.s \in \mathcal{S} \mid s_{height} \in [lb - \epsilon_{height}, ub + \epsilon_{height}]\}$ (see Figure 6 for a visualization of the interval and the relevant constants). The state space $\mathcal{S}$ of *CtrlSys* consists of the control state k and the ranges of the variables *height*, a, *bend* and $dist$. The regarded topology on $\mathcal{S}$ is constructed as follows. For *height* we use the topology generated by the base $\mathcal{B}_h = \{\{x \mid d(x, A) \in b\} \mid b \in \mathcal{B}_{\mathbb{R}_+}\}$, where $\mathcal{B}_{\mathbb{R}_+}$ is a base for the natural topology on $\mathbb{R}_+$ and $d(x, A) = inf\{|x - a| \mid a \in A\}$ (Euclidean distance). The resulting topology is similar to the natural topology on $\mathbb{R}$, but coarser. It does not distinguish between elements of A and between deviations from A to higher or to lower values. This is consistent with our view of the EHC system, because all values of *height* inside A are regarded as desirable, while for deviations from A it is not important whether *height* is too high or too low. The



Definition of actions, invariants, and activities:

$b2o$	$\equiv$	$bend?$
$o2b$	$\equiv$	$b2o$
$i2u$	$\equiv$	$height \leq lb$
$i2d$	$\equiv$	$height \geq ub$
$u2i$	$\equiv$	$height \geq lb + c$
$d2i$	$\equiv$	$height \leq ub - c$
$CtrlSys_{inv}$	$\equiv$	$True$
$inBend_{inv}$	$\equiv$	$bend\not{?}\epsilon_{bend}$
$outBend_{inv}$	$\equiv$	$inBend_{inv}$
$inTol_{inv}$	$\equiv$	$height \in (lb - \epsilon_{height}, ub + \epsilon_{height})$
up_{inv}	$\equiv$	$height < lb + c + \epsilon_{height}$
$down_{inv}$	$\equiv$	$height > ub - c - \epsilon_{height}$
$const$	$\equiv$	$a = 0 \ \wedge \ \dot{height} = a + dist$
inc	$\equiv$	$a \in [cp_-, cp_+] \ \wedge \ \dot{height} = a + dist$
dec	$\equiv$	$a \in [ev_-, ev_+] \ \wedge \ \dot{height} = a + dist$

Figure 5: HySChart for the control system *CtrlSys*. ($lb, ub, cp_-, cp_+, ev_-, ev_+, c, \epsilon_{height}$ and ϵ_{bend} are constants.)

other parts of the data-state space as well as the control-state space are immaterial here and we therefore use the coarsest topology for them, i.e. for variable a, for instance, we use the topology $\mathcal{O} = \{\emptyset, \mathbb{R}_+\}$. The topology $\mathcal{O}_\mathcal{S}$ we use on $\mathcal{S}$ is the Tychonoff (or product) topology of these topologies (see Appendix C, Definition 17).

Stability. To proof stability of A we use a Liapunov function of the kind described in Section 4.1.1. $L \in \mathcal{S} \to V$ is defined by $L(s) = d(s_{height}, A)$, i.e. it is the distance of *height* to A. We use $V = \mathbb{R}_+$ with order $\leq$ on the reals. V is complete and densely ordered, has least element 0, and $V^+ = V \setminus \{0\}$ (see Theorem 1 for the definition of V^+). L is onto and $L(s) = 0$ if $s_{height} \in A$. The left topology on V is equal to the natural topology on $\mathbb{R}_+$. Applying Theorem 12 proves that topology $\mathcal{O}_\mathcal{S}$ is the coarsest topology on $\mathcal{S}$ which makes L continuous. In fact, this by part motivated our choice of the topology for the chassis level.

Using the result of Section 4.1.1 stability of A can now be established by proving that L is monotonously decreasing along the traces of *CtrlSys* and for inputs I. The proof proceeds by computational induction over the traces of *CtrlSys*. Here, we argue informally. For a state $s \in \mathcal{S}$ at time t *CtrlSys* either performs a discrete transition, or time passes until time $t' > t$. As no action modifies the value of s_{height} L remains constant, and hence monotonous, whenever a transition is taken. If control is in state *inTol* or in *inBend* and time passes s_{height} also remains constant. If control is in *up* and time passes, its invariant, $height < lb + c + \epsilon_{height}$, must hold and *height* is increased. Thus, provided the constants are

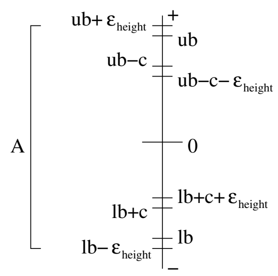


Figure 6: Relevant intervals and constants for the chassis level.

chosen such that $lb + c + \epsilon_{height} < ub + \epsilon_{height}$, L decreases until *height* reaches $lb - \epsilon_{height}$ and then remains constant. If control is in *up* and the invariant does not hold, i.e. *height* already is too high, no time can pass in the state and a transition is taken. The situation is similar for control being in *down*. Hence, L is monotonous.

Attraction. For attraction we additionally have to prove that every $v \in V^+$ is reached. Again, we argue informally. Let $k.s \in \mathcal{S}$ be some state. If $k.s$ is in A nothing remains to be shown as $L(k.s) \leq v$ already holds. If in the regarded state the height is below $lb - \epsilon_{height}$, there are two possibilities depending on whether the control-state is in *inBend* or in one of the substates of *outBend*. In the first case some time will pass with *height* remaining constant. Then, at most time ϵ_{bend} after the next toggling of *bend* in the input, control changes to *inTol* and we have the second case. In the second case the control-state immediately changes to *up*, because the transitions leading to *up* are enabled and the invariants of *inTol* and *down* are false. Control will remain in *up* until either the height is in A or a bend occurs. In case of a bend the minimal separation bs between *bend* signals in the input ensures that time $bs - \epsilon_{bend}$ passes in *up* (or A is reached) before state *inBend* is entered.[8] Note that we required that $bs > \epsilon_{bend}$ above. As long as control is in *up* the height decreases at least with rate cp_-. Thus, between two bends the height is always decreased by at least $cp_- \cdot (bs - \epsilon_{bend})$ in *up* unless the height is in A. As bends are left infinitely often, the regarded v is reached after finitely many curves, and therefore after a finite amount of time. Note that the height, of course, also is decreased in *up* if no bends occur. Finally, if in the regarded state the height is above $up + \epsilon_{height}$, the argument is analogous to the above case. This ends the proof.

Remarks. We could also have incorporated the model of disturbances into the system model and could have examined the resulting (input-free) model. For complex disturbances for hybrid systems this may often be necessary, as certain control states can only occur under certain circumstances of the environment. For the EHC, e.g., leaving a bend is only possible if a bend has been entered. Here, not incorporating the inputs into the state enforced us to consider infinitely many changes in the bend signal only because we connot decide which signal models leaving a bend and which denotes entry.

Furthermore, we could also use a control-state depended Liapunov function. This was not needed here, but it would be necessary if we allowed (small) disturbances inside and outside bends (where disturbances inside bends are required to vanish).

Proofs of stability and attraction are also possible for more general models of external disturbances. However, such proofs require a a lot of technical detail. We rather decided to give a simple example here to demonstrate proofs of stability and attraction for hybrid systems.

[8] Only time $bs - \epsilon_{bend}$ not time bs is guaranteed to pass in *up*, because the reaction to the *bend* signal may be delayed by at most ϵ_{bend}.

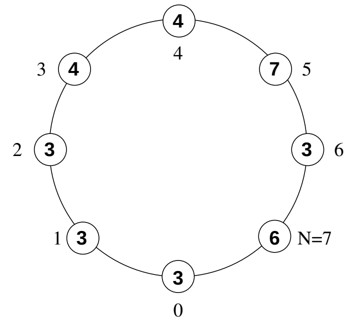


Figure 7: An example configuration for N=7 and M=7.

5.2 Attraction for Self-Stabilizing Algorithms

The example in this section shows how a correctness proof for a so called self-stabilizing algorithm can be modified to result in a proof of attraction. Due to the used topologies attraction will turn out to be equivalent to the original safety- and liveness properties subsumed in self-stabilization.

In [Dij82] Dijkstra defines *self-stabilizing systems* as systems which are guaranteed to arrive at a legitimate state after a finite number of steps regardless of their initial state and which then remain in such legitimate states forever. Note that this term may not be confused with the definitions of stability we have given here. In our terms self-stabilization rather is related to attraction than to stability. In terms of [CMP91] self-stabilization is a persistence property. Similar to attraction, self-stabilization can be useful as a substitute for an initialization procedure or for error recovery. References on self-stabilizing systems can e.g. be found in [Ali99].

The system. The system we consider here is taken from [Dij82]. It consists of a ring of $N+1$ processes, $N > 0$, numbered from 0 to N in clockwise direction. Each process has a variable with a value in $\{0, \dots, M\}$, where $M \geq N$. A process $(i+1) mod (N+1)$ may read and write its own variable and read the variable of its left-hand neighbor i. The legitimate states of the system are those where there is exactly one *token* in the ring.

The system evolves as follows. Process 0 holds a token iff its variable *is equal* to that of its left-hand neighbor. When holding the token it can make a step and increment its variable by 1 modulo $M+1$. Every other process holds a token iff its variable *is different* from that of its left-hand neighbor. In this case it may make a step and set its variable to the value of his left-hand neighbor's variable. The process which actually executes its step is selected by a fair nondeterministic scheduler. In the example of Figure 7 process numbers are written besides the processes. The values of their variables are written in bold font inside the processes. In the depicted configuration processes $3, 5, 6$ and 7 are enabled, i.e. hold a token.

Dijkstra proves that for any initial state the system eventually reaches a legitimate state and remains within the set of legitimate states forever [Dij82]. We will follow the more formal proof in [Mer98] which uses the standard technique for proving liveness properties, namely to define a valuation function from the state space to a well-founded set that decreases with every step of the system, where an infinite number of steps is assumed to exist. As we will see, after a slight modification this valuation function can be used as a Liapunov function that proves the attraction of the set of legitimate states. We want to avoid to introduce an extra discrete time system model only for this algorithm. Therefore, we embed the algorithm in our dense time model by using a modified scheduler which schedules the next process to proceed at a nondeterministically selected finite time after the current step. During intervals between the selected time instances all processes are idle, the system state remains constant.

Solution principle. Informally the algorithm works as follows. The new values created by process 0 spread out from 0 to its right, i.e. there is a "prefix" of processes in the ring having the same value as process 0. The values held by the variables of processes outside the prefix are potentially "bad", they might enable to many processes. The legitimate states are those, where there either is only one value outside the prefix (then the process to the right of the prefix is enabled), or the prefix is the whole ring. In this case process 0 is enabled. Values different from the one in the prefix are eliminated as time elapses. They vanish at process N, because process 0 creates no new value for them. If the value in the prefix also occurs in one of the processes outside the prefix, process 0 will create a new value (and hence a new token) when it reads the value from process N. However, none of the processes ever creates a new value not in the prefix. (In the case of process 0 the prefix changes so that the statement remains correct.) As $M \geq N$, the value of process 0's variable (by incrementing) eventually reaches a value that is different from all other values in the ring. This value will spread out, all other values will vanish until the prefix is the whole ring and we have reached a legitimate state. As no further tokens are ever introduced, the system remains in this state forever.

Conventional proof (outline). [Mer98] formalizes this argument with the following definitions: For a system state $s \in \mathcal{S} = \{0, 1, \ldots, M\}^{N+1}$, the set $\mathit{prefix}(s) \subseteq \{0, 1, \ldots, N\}$ contains the numbers of all those processes which, together with all their left-hand neighbors up to process 0, hold the same value as process 0. In the example configuration $\mathit{prefix}(s) = \{0, 1, 2\}$ and $s = (3, 3, 3, 4, 4, 7, 3, 6)$. $\mathit{others}(s) \subset \{0, 1, \ldots, M\}$ is the set of values held by processes not in the prefix. In our example $\mathit{others}(s) = \{3, 4, 6, 7\}$.[9] At any system state $\mathit{minfree}(s) \in \{0, 1, \ldots, M\}$ is the smallest value by which the value of process 0's variable must be incremented (modulo $M+1$) to get a value that is different from all values in others. Hence, $\mathit{minfree}$ is the number of times process 0 has to increase its variable until a value is reached which is not present elsewhere in the ring. In the example $\mathit{minfree}(s) = 2$, because 5 is not in others. The bit vector $\mathit{enabs}(s) \in \mathbb{B}^N$ is defined such that component i of the vector is true iff process i is enabled. The status of process 0 is not part of enabs. In the example $\mathit{enabs}(s) = (f, f, t, f, t, t, t)$, where we write t for true and f for false. Based on these definitions [Mer98] uses valuation function $\mathit{meas}(s) = (\mathit{minfree}(s), \mathit{enabs}(s))$ from system states to the well-founded set $(\{0, 1, \ldots, M\} \times \mathbb{B}^N, \sqsubset)$, where $\sqsubset$ denotes the lexicographic order derived from the usual less-than order on $\{0, 1, \ldots, M\}$ and the order $\sqsubset_{\mathbb{B}}$ on the booleans given by $f \sqsubset t$.[10]

The proof in [Mer98] consists of two basic parts. First, it is shown that once a legitimate state is reached, the system remains in legitimate states forever. Second, [Mer98] shows that the system eventually reaches a state where only one process is enabled. The second part proceeds by proving that steps of non-zero processes decrease enabs without increasing $\mathit{minfree}$ while steps of process 0 decrease $\mathit{minfree}$ as long as a non-zero process exists which is enabled. Hence, it is proved that meas strictly decreases with every step of the system which starts in a non-legitimate state. As meas is strictly decreasing for steps from these states and as $\sqsubset_{lex}$ is well-founded a finite number of steps must lead to a state where meas no longer strictly decreases, i.e. to a legitimate state. Note that when a legitimate state is reached succeeding steps of the system may increase meas.

Proof based on attraction. In order to prove the same property of self-stabilization we

[9] Value 3 is held by process 6 which is not in the prefix.

[10] For partially ordered sets $(A, \sqsubset_A)$ and $(B, \sqsubset_B)$ the lexicographic order $\sqsubset_{lex}$ on $A \times B$ is defined by $(a, b) \sqsubset_{lex} (a', b')$ iff $a \sqsubset_A a'$ or $a = a'$ and $b \sqsubset_B b'$. The lexicographic order is well-founded if the orders $\sqsubset_A$ and $\sqsubset_B$ are well-founded [Win93].

can employ the specialization of Theorem 3 for discrete systems introduced in Section 4.2.1 with a slight variant of valuation function *meas* as Liapunov function. The set A, whose attractiveness we show, consists of all those system states in which exactly one process is enabled. The topology we use on $\mathcal{S}$ is defined by $\mathcal{O}_{\mathcal{S}} = \{\emptyset, A, \mathcal{S}\}$. As A is an open set in $\mathcal{O}_{\mathcal{S}}$, proving attraction of A shows that A is entered in finite time and never left again, which exactly is the goal behind self-stabilization. The Liapunov function L we use results from a modification of *meas*. We extend the assessment space by a new bottom element, $V = \{0, 1, \ldots, M\} \times \mathbb{B}^N \cup \{\perp\}$. The lexicographic order $\sqsubset$ is extended to V by defining $x \sqsubset_V y$ iff $x = \perp$ or $x \sqsubseteq_{lex} y$. Therefore, $\perp$ is the least element in V and $(V, \sqsubset_V)$ again is a well founded set. Now $L \in \mathcal{S} \to V$ is defined by $L(s) = meas(s)$ iff more than one process is enabled in s and $L(s) = \perp$ otherwise. Obviously, $\sqsubseteq_V$ is a partial order on V, where $\sqsubseteq_V$ is the reflexive closure of $\sqsubset_V$. $V^+ = V$ by definition of L and the topology on $\mathcal{S}$. Furthermore, L is monotonously decreasing along the traces of the system, because of the respective property of *meas* as explained above and because of the new element $\perp$. Note that adding a bottom element is necessary, because as soon as the first legitimate state is reached *meas* no longer is non-increasing. By adding the bottom element we simply identify all legitimate states in the assessment space V of the system. Third, as long as $\perp$ is not reached L is strictly decreasing at the time instants where a step of the system is scheduled, because *meas* also is. Finally, $L^{-1}(\perp) = A$ holds by definition of L.

Hence, function L satisfies the properties required in the specialization of Theorem 3 for discrete systems in Section 4.2.1 and A therefore is an attractive set. Openness of A in the considered topology yields that attraction implies that A is eventually entered and never left again, regardless of the initial state. Consequently the system is self-stabilizing.

Note that L also satisfies requirement 3 in Theorem 1, because for any v we can choose $A \in N(A)$. Hence, A is also stable which, together with the openness of A, implies that it is an invariant set, i.e. it is never left by the system. Furthermore, note that we could also have used a finer topology such as the topology $\{S_0, S_1, \ldots, S_{N+1}\}$, where S_i is the set of states with at most i tokens. Hence, $S_0 = \emptyset$, $S_1 = A$ and $S_{N+1} = \mathcal{S}$. With this topology the proof would be the same, because A is open in it as well.

6 Discussion and Further Work

6.1 Contribution

Based on an abstract system model which is suitable for hybrid systems and close to models in computer science we have formalized important properties of control systems. The properties have been extracted from the evaluation of nine case studies and a number of text books on control theory. The properties have been classified w.r.t. the semantic models relative to which they are defined. Rating the properties according to the number of case studies in which they were considered identified invariance properties as the most important ones. Furthermore, the rating revealed that a refinement notion based on trace inclusion preserves at least those properties which were regarded most frequently in the case studies.

For the properties of stability and attraction the vital role topology plays in their definition was made obvious. Topologies were identified under which stability and attraction are equivalent to invariance and persistence, respectively, which are important classes of properties in computer science. Due to their importance in control theory proof methods for stability and attraction were examined in greater detail. This study resulted in an adaption of the general Liapunov-like proof methods from [MT75] to our model of hybrid systems. Furthermore, we were able to make parallels between the Liapunov theory and abstraction in computer science explicit. Namely, circumstances were identified under which Liapunov functions define Galois connections.

We applied the developed proof methods to two example system. First, we considered a small hybrid example system and proved stability and attraction for specific environment assumptions and w.r.t. a certain meaningful topology for the system. The work at the example revealed that the imprecision which is present in the model w.r.t. the values (and times, respectively) for which discrete transitions are taken, makes the proof more difficult. In fact considerable time had to be spent to identify circumstances in which stability can be proven and is not violated by the imprecision in the model. Although this is undesirable, it merely reflects that stability properties of an ideal, exact model can not necessarily be transfered to a real model. Apparently, additional environment assumptions and/or a more liberal interpretation of the considered properties are necessary to carry over results from a precise to an imprecise model.

As a second example we demonstrated how the developed methods can be applied to the purely discrete example of a self-stabilizing algorithm and contrasted it with the conventional proof. While the proof is not simplified by our method it is nevertheless interesting to see how the intuition of bad and good system configurations, which are eventually reached, is reflected in the topology underlying the proof. This indicates that topology, which in this respect can be seen as identifying equivalence classes, also has relevance for practical computer science systems, not only for theoretical results.

Note that the outlined proof methods for stability and attraction are not meant to replace the variety of existing techniques. Instead existing techniques can and should be used to construct optimal stable controllers in single control modes. The techniques of the kind presented here are only necessary if stability has to be ensured in the presence of switching between control modes. For instance in process automation there are many interesting example systems where stability in only one mode is desired, e.g. the wire stretching plant in [PPS00]. However, the proof methods for stability and attraction developed here could be particularly useful in applications were there are also disturbances in the discrete state space. An example is the work on robotic hand regrasping [SHB+99], where controllers are designed which are supposed to compensate discrete errors resulting from incorrect environment models.

With the continuing integration of software and its physical environment in many systems we conjecture that properties which have only been of interest for continuous systems so far, like stability and attraction, will also become important for the software part of embedded systems. Furthermore, progress in mobile, self-configuring systems and in component-based software engineering may also make stability and attraction important for computer science.

6.2 Related Work

Dynamical systems theory. [MT75] studies similar properties from a very general system theory point of view. In particular [MT75] also introduces an extension of the classical Liapunov theory (see e.g. [Lue79]) for their systems. We build on this result by extending it to nondeterministic systems, which play an important role in the early development phases of discrete (computer science) systems. [MW95] also puts the Liapunov theory into a more general framework and regards it as one way of defining abstractions, i.e. qualitatively equivalent comparison systems to the system under study. This work, however, remains limited to systems operating on metric spaces as it assumes metric spaces as basis for the abstraction mappings it defines.

[Sin96] mentions that there is a correspondence between invariance in control theory and safety in computer science, and between attraction in control theory and liveness in computer science without going into further detail. Furthermore, Sintzoff notes the similarity between Liapunov functions in control theory and Floyd functions used in computer science to prove termination of programs [Sin92]. With our formal definitions of control systems' properties we make these correspondences precise by identifying classes of topologies where they become apparent.

[SG95] and [Geu96] regard dynamical systems from an abstract, systems theory point of view, based on predicate transformers ([SG95]) or iterated relations ([Geu96]). The authors define the concepts of invariance, fullness of invariants (i.e. invariants are not empty) and atomicity of invariants (invariants are singletons) and (finite time) attraction. Furthermore, invariants and attractors are identified as necessary or potential, corresponding to universal or existential path quantification over a system's computations. [Geu96] mentions the correspondence between necessary invariants and invariants in linear-time temporal logic in computer science, and between necessary attractors and termination in computer science. Furthermore, the relationship between the notion of *chaos* from dynamical systems theory and fullness and atomicity of invariants is elaborated by the authors.

The classification into necessary and potential properties is similar to our partitioning of properties in universal and existential properties. However, for existential properties we furthermore distinguish between system inputs and the system output or state, which is in spirit of alternating-time temporal logic (ATL) [AHK97]. This is necessary to classify the control theory properties of controllability and observability.

The proof methods for atomicity of invariants and attraction given in [SG95, Geu96] are also Liapunov-like criteria. In Section 4.2.1 a criterion from [SG95] was used to derive a specialization of our general proof method for attraction.

In [Geu96] the exposition of the properties is motivated by finding out which kind of dynamics are created by (discrete) dynamical systems. This classification of dynamics serves as the basis for analyzing the effect of composition operators on the dynamics. In contrast this work focuses towards hybrid systems development. We are mainly interested in properties actually considered for hybrid systems during their design. This explains why we based our exposition of properties to a large part on the evaluation of case studies. Continuous dynamics which play an important role here, are not considered in [Geu96] and correspondingly topologies on the input, output or state space are not essential for the meaning of properties there.

[Aki93] develops the topological foundations of general dynamical systems starting from iterated relations. Although invariance and attraction, also in the context of Liapunov theory, are considered there, the theory does not seem to be immediately useful for the application to hybrid systems. It rather supports a deeper understanding of dynamical systems in general.

Computer science. The role of topology in computer science, namely for the classification into safety and liveness properties, is explained in [AS85]. Note that the specific topology used in this context is defined on the space of traces, not on the state space, like in our state-based versions of stability and attraction. [Mis96] discusses advantages and disadvantages of using domain theory instead of topology as a basis for computer science.

A number of temporal logics have been defined for hybrid systems, most notably the *hybrid temporal logic* of [HMP93], the extension of the duration calculus in [CRH93], hybrid TLA+ [Lam93] and [Fri98]. [HMP93] and [CRH93] are interval temporal logics, [Fri98] defines a first order logic suitable for hybrid systems. However, these papers do not study specific properties of hybrid systems like stability and attraction.

Refinement is a common topic in computer science [Hoa86, Mai87, AL91, Bro93]. An overview over notions of behavior refinement can be found in [vdB00]. The notion of refinement we employ is taken from [Bro99, Bro97].

Control theory. [Bra94] defines a method for proving stability of hybrid systems. It basically consists of requiring the existence of Liapunov functions for each individual control mode which furthermore must be non-increasing whenever the control mode is changed. This is strongly focused at proving the stability of continuous variables in the presence of mode switching. It does not allow to incorporate a topology on the control state space. All control states are regarded as equivalent. Hence, it mainly concentrates on the continuous fragment of the dynamics of hybrid systems, discrete jumps in the (otherwise continuous) variables are not considered in the underlying system model and the proposed proof method.[11] Thus, it can be regarded as specialization of this work, which definitely is highly relevant for practice.

6.3 Further Work

In the future we will use the results on refinement obtained here in order to develop refinement rules which support the transition from abstract, hybrid models capturing system requirements to implementation oriented discrete time models. Bridging the gap from hybrid models to discrete time models then allows to use conventional techniques in the remainder of the development process for the discrete time parts [PSS00]. Furthermore, we will also consider rules for directly refining hybrid models.

Besides that, to be more useful in practice specializations of the general proof methods developed here are necessary. Such specialization should be driven by deficits encountered and experience gained in the practical development of hybrid system. Apart from that it is also necessary to validate within case studies whether existing proof principles from computer science which have been adapted to hybrid systems (e.g. [Pnu94, Lam93]) are suitable in applications.

Acknowledgment. We thank Alexander Pretschner for his valuable feedback on draft versions of this paper. Further thanks go to Michel Sintzoff for his helpful hints on related work in particular concerning dynamical systems theory.

References

[Abr96] J.-R. Abrial. Steam-boiler control specification problem. In *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control*, LNCS 1165. Springer-Verlag, 1996.

[AHK97] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, pages 100–109. IEEE Computer Society Press, 1997.

[Aki93] E. Akin. *The General Topology of Dynamical Systems.* American Mathematical Society, 1993.

[AL91] M. Abadi and L. Lamport. The existence of refinement mappings. *Theoretical Computer Science*, 82(2):253–284, May 1991.

[Ali99] L. O. Alima. *Self-Stabilization by Self-Stabilizing Waves.* PhD thesis, Université catholique de Louvain, Faculté des Sciences Appliquées, Département d'Ingéniere Informatique, Belgium, 1999.

[11] However, it seems they can be integrated easily.

[Ant96] F. Antosch. *Systematische Maschinenmodellierung: Modellierung eines Glideliners (Fa. Krones) nach der ROOM Methode.* Diplomarbeit, Lehrstuhl für Informationstechnik im Maschinenwesen, Technische Universität München, 1996.

[AS85] B. Alpern and F. B. Schneider. Defining liveness. *Information Processing Letters*, 21(4):181–185, October 1985.

[BGM93] A. Back, J. Guckenheimer, and M. Myers. A dynamical simulation facility for hybrid systems. In *Proc. of Hybrid Systems I*, LNCS 736. Springer-Verlag, 1993.

[BHKT98] H. Brettschneider, H.-M. Hanisch, S. Kowalewski, and J. Thieme. Diskontinuierlicher Verdampfer: Ein erweitertes Benchmark-Beispiel für ereignisdiskrete und hybride Systeme. Presentation at the 1998 Workshop of GMA Fachausschuß 1.8 "Methoden der Steuerungstechnik", 3 1998.

[Bra94] M. S. Branicky. Stability of switched and hybrid systems. In *Proc. 33rd IEEE Conf. Decision and Control*, pages 3498–3503, Lake Buena Vista, FL, 1994.

[Bro93] Manfred Broy. (inter-)action refinement: The easy way. In *Program Design Calculi, NATO ASI*, volume 118 of *Series F: Computer and Systems Sciences.* IOS Press, 1993.

[Bro97] M. Broy. Refinement of time. In *ARTS'97*, volume 1231 of *LNCS*. Springer-Verlag, 1997.

[Bro99] M. Broy. A logical basis for modular systems engineering. In *Proc. of the 1998 NATO ASI International Summer School on Calculational System Design*, volume 173 of *Series F: Computer and Systems Sciences.* IOS Press, 1999.

[CC92] P. Cousot and R. Cousot. Abstract Interpretation and applications to logic programs. *Journal of Logic Programming*, 13(2-3):103–180, 1992.

[CMP91] E. Chang, Z. Manna, and A. Pnueli. The safety-progress classification. In F.L. Bauer, W. Brauer, and H. Schwichtenberg, editors, *Logic and Algebra of Specifications*, NATO Advanced Science Institutes Series. Springer-Verlag, 1991.

[Cou90] P. Cousot. Methods and logics for proving programs. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics.* Elsevier, 1990.

[CRH93] Zhou Chaochen, A. P. Ravn, and M. R. Hansen. An extended duration calculus for hybrid real-time systems. In *Hybrid Systems*, LNCS 736. Springer-Verlag, 1993.

[Dij82] E. W. Dijkstra. Self-stabilization in spite of distributed control (EWD 391). In E. W. Dijkstra, editor, *Selected Writings on Computing: A Personal Perspective*, Texts and Monographs in Computer Science. Springer-Verlag, 1982.

[Eme90] E. A. Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics.* Elsevier, 1990.

[Eng89] Ryszard Engelking. *General Topology*, volume 6 of *Sigma Series in Pure Mathematics.* Heldermann Verlag, Berlin, 1989.

[Eng97] S. Engell. Modellierung und Analyse hybrider dynamischer Systeme. *at – Automatisierungstechnik*, 45(4), 1997.

[Flo67] R. W. Floyd. Assigning meaning to programs. In *Mathematical aspects of computer science: Proc. American Mathematics Soc. symposia*, volume 19. American Mathematical Society, 1967.

[Föl87] O. Föllinger. *Nichtlineare Regelungen I.* R. Oldenbourg Verlag, München, fourth edition, 1987.

[Föl90] O. Föllinger. *Regelungstechnik.* Hüthig Buch Verlag, Heidelberg, sixth edition, 1990.

[Fri86] B. Friedland. *Control System Design – an introduction to state-space methods.* McGraw-Hill Inc., 1986.

[Fri98] V. Friesen. A logic for the specification of continuous systems. In *Proc. of Hybrid Systems: Computation and Control (HSCC'98)*, LNCS 1386. Springer-Verlag, 1998.

[Ger97] S. Germann. *Modellbildung und modellgestützte Regelung der Fahrzeuglängsdynamik.* Fortschrittsberichte VDI, Reihe 12: Verkehrstechnik/Fahrzeugtechnik, Bericht 307. VDI Verlag, 1997.

[Geu96] F. Geurts. *Compositional Analysis of Iteratted Relations: Dynamics and Computations.* PhD thesis, Université catholique de Louvain, Faculté des Sciences Appliquées, Département d'Ingéniere Informatique, Belgium, 1996.

[GNRR93] R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, editors. *Hybrid Systems I*, LNCS 736. Springer-Verlag, 1993.

[GSB98] R. Grosu, T. Stauner, and M. Broy. A modular visual model for hybrid systems. In *Proceedings of Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT'98)*, volume 1486 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998.

[HHWT95] T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. A user guide to HyTech. In E. Brinksma, W.R. Cleaveland, K.G. Larsen, T. Margaria, and B. Steffen, editors, *TACAS 95: Tools and Algorithms for the Construction and Analysis of Systems*, Lecture Notes in Computer Science 1019, pages 41–71. Springer-Verlag, 1995.

[HMP93] T.A. Henzinger, Z. Manna, and A. Pnueli. Towards refining temporal specifications into hybrid systems. In R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, editors, *Hybrid Systems I*, Lecture Notes in Computer Science 736, pages 60–76. Springer-Verlag, 1993.

[Hoa86] C. A. R. Hoare. *Communicating Sequential Processes.* Prentice-Hall, 1986.

[Int98] Integrated Systems Inc. Integrated Systems: Products - BetterState. `http://www.isi.com/Products/BetterState/`, 1998.

[Int00] Integrated Systems Inc. Integrated Systems: Product families - MATRIXx. `http://www.isi.com/Products/MATRIXx/`, 2000.

[Kön90] K. Königsberger. *Analysis I.* Springer-Verlag, 1990.

[Lam93] Leslie Lamport. Hybrid systems in TLA+. In R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, editors, *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993.

[Lei87] H. Leipholz. *Stability Theory.* John Wiley & Sons Ltd., second edition, 1987.

[Lue79] D. G. Luenberger. *Introduction to Dynamic Systems – Theory, Models, and Applications.* John Wiley & Sons Ltd., 1979.

[Mai87] M. Main. Trace, failure and testing equivalences for communicating processes. *Int. Journal of Prallel Programming*, 16(5):383–400, 1987.

[Mer98] S. Merz. On the verification of a self-stabilizing algorithm. Available at http://www.pst.informatik.uni-muenchen.de/~merz/papers/dijkstra.ps.gz, 1998.

[Mis96] M. W. Mislove. Topology, domain theory and theoretical computer science. Preprint, available at http://www.math.tulane.edu/ftp/pub/mwm/topandcs.ps.gz, 1996.

[MP92] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems.* Springer-Verlag, 1992.

[MT75] M. D. Mesarovic and Y. Takahara. *General Systems Theory: Mathematical Foundations*, volume 113. Academic Press, 1975. Mathematics in Science and Engineering.

[MW95] A. N. Michel and K. Wang. *Qualitative theory of dynamical systems – The role of stability preserving mappings.* Marcel Dekker Inc., 1995.

[NNH98] F. Nielson, H. R. Nielson, and C. Hankin. *Principles of Program Analysis.* Springer-Verlag, 1998.

[Oga87] K. Ogata. *Discrete-Time Control Systems.* Prentice-Hall, 1987.

[PH88] C. L. Phillips and R. D. Harbor. *Feedback Control Systems.* Prentice-Hall, 1988.

[Pnu94] A. Pnueli. Development of hybrid systems. In *Proc. of Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT'94)*, LNCS 863. Springer-Verlag, 1994.

[PPS00] I. Péter, A. Pretschner, and T. Stauner. Heterogeneous development of hybrid systems. In *Proc. of GI workshop Rigorose Entwicklung software-intensiver Systeme.* Ludwig-Maximilians-Universität München, Institut für Informatik, Bericht 0005, September 2000.

[PSS00] A. Pretschner, O. Slotosch, and T. Stauner. Developing correct safety critical, hybrid, embedded systems. In *Proc. of New Information Processing Techniques for Military Systems.* NATO Research and Technology Organization, October 2000.

[Rap98] M. Rappl. Strukturierte Analyse und Design hybrider Systeme mit MatrixX und BetterState. Diplomarbeit, Fakultät für Informatik, Technische Universität München, 1998.

[SB98] T. Schlegl and M. Buss. Hybrid closed-loop control of robotic hand regrasping. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 3026–3032, Leuven, Belgium, 1998.

[SG95] M. Sintzoff and F. Geurts. Analysis of dynamical systems using predicate transformers: Attraction and composition. In *Analysis of Dynamical and Cognitive Systems*, LNCS 888. Springer-Verlag, 1995.

[SHB+99] T. Schlegl, S. Haidacher, M. Buss, F. Freyberger, F. Pfeiffer, and G. Schmidt. Compensation of discrete contact state errors in regrasping experiments with the TUM-hand. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*, pages 118–123, Kyongju, Korea, 1999.

[Sin92] M. Sintzoff. Invariance and contraction by infinite iterations of relations. In *Research Directions in High-Level Parallel Programming Languages*, LNCS 574. Springer-Verlag, 1992.

[Sin96] M. Sintzoff. Abstract verification of structured dynamical systems. In *Proc. of Hybrid Systems III*, LNCS 1066. Springer-Verlag, 1996.

[SMF97] T. Stauner, O. Müller, and M. Fuchs. Using HyTech to verify an automotive control system. In *Proc. Int. Workshop on Hybrid and Real-Time Systems (HART'97)*, volume 1201 of *LNCS*, pages 139–153. Springer-Verlag, 1997.

[Son90] E. D. Sontag. *Mathematical Control Theory*. Springer-Verlag, 1990.

[Sta97] T. Stauner. Specification and verification of an electronic height control system using hybrid automata. Diploma thesis, Technische Universität München, 1997.

[Sta00] T. Stauner. Extending hycharts with state-invariants. In *Proc. of GI workshop Rigorose Entwicklung software-intensiver Systeme*. Ludwig-Maximilians-Universität München, Institut für Informatik, Bericht 0005, September 2000.

[Vac95] R. J. Vaccaro. *Digital Control*. McGraw-Hill Inc., 1995.

[vdB00] M. v. d. Beeck. Behaviour specifications: Semantics, equivalence and refinement. In *Proc. of GROOM 2000 (Grundlagen Objekt-Orientierter Modellierung)*. Institut für Informatik, WWU Münster, 2000.

[Wig90] S. Wiggins. *Introduction to Applied Nonlinear Dynamical Systems and Chaos*. Texts in Applied Mathematics (TAM) 2. Springer-Verlag, 1990.

[Win93] G. Winskel. *The Formal Semantics of Programming Languages*. MIT Press, 1993.

A Proofs

Theorem 4 *In topologies where C and E are open, (C, E) is stable iff $Sys(C) \subseteq E$.*

Proof. "$\Rightarrow$": With choosing $\alpha = E$ stability yields $\forall b \in \beta.\, Sys(b) \subseteq \alpha$ for a $\beta \in N(C)$. By the definition of neighborhood $C \subseteq \beta$ holds. Hence, we also get that all causes in C produce effects in α, $\forall b \in C.\, Sys(b) \subseteq \alpha$.

"$\Leftarrow$": For every neighborhood of E we can choose $\beta = C$. $\square$

Theorem 5 *In topologies where set A is open, invariance of A is equivalent to the (state-based) stability of A.*

Proof. "$\Rightarrow$": For every neighborhood of A we can choose $\beta = A$.

"$\Leftarrow$": With choosing $\alpha = A$ stability yields $\forall \iota \in I.\, \forall s \in \beta.\, Sys_S(s, \iota) \subseteq \alpha^{\mathbb{R}_+}$ for a $\beta \in N(A)$. By definition of state based system descriptions $Sys_S(s, \iota)(0) = s$ holds. Hence, we get $\beta \subseteq A$ and, by applying the definition of neighborhood, $\beta = A$, which completes the poof. $\square$

Theorem 6 *In topologies where A is open global attractiveness is equivalent to the property that A is reached in finite time and not left again.*

Proof. "$\Rightarrow$": We select the particular neighborhood A of A in the definition of global attractiveness to proof the claim.

"$\Leftarrow$": Due to the definition of neighborhood, $A \subseteq \alpha$ holds for any neighborhood of A. As there is a t such that A is reached and not left again after t, this also holds for any of A's neighborhoods. $\square$

Lemma 1 *The functions abs and conc as defined in Section 4.1.1 are well-defined.*

Proof. We start with function *abs*. To see that the supremum in the definition of $abs(\alpha)$ exists in V for $\alpha \in N(A)$ first note that V is complete. Second, the regarded set $\{v \in V^+ \mid \alpha \supseteq L^{-1}(\langle v\rangle_o)\}$, whose supremum is sought, is non-empty. To prove this we can construct an element of this set: As the topology on $\mathcal{S}$ is the coarsest which makes L continuous, Theorem 12 implies that every open set α can be written as $\alpha = \bigcup_{j \in J} L^{-1}(I_j)$ for an index set J and open intervals I_j. As $\alpha \supseteq A$, there exists a I_j with $\perp \in I_j$. Hence, I_j can be written as $\langle v\rangle_o$ for some $v \neq \perp$. This implies that $\alpha \supseteq L^{-1}(\langle v\rangle_o)$ holds for v, i.e. the set whose supremum is needed is non-empty, and it furthermore guarantees that the supremum is not equal to $\perp$. Thus, $abs(\alpha) \in V^+$.

For function *conc* we have that set $conc(v)$ is open for $v \in V^+$, because L is continuous. Furthermore, $A \subseteq conc(v)$ because $\forall s \in A.\, L(s) = \perp$. Hence, $conc(v) \in N(A)$ for $v \in V^+$. $\square$

Theorem 7 *The functions abs and conc as defined in Section 4.1.1 are a Galois connection between $(N(A), \supseteq)$ and $(V^+, \sqsupseteq)$, i.e. $(N(A), \supseteq)$ and $(V^+, \sqsupseteq)$ are partially ordered sets, abs and conc are monotonous, and $\alpha \supseteq conc(abs(\alpha))$ (*extensivity*) and $abs(conc(v)) \sqsupseteq v$ (*reductivity*) holds. In fact even $abs(conc(v)) = v$ is valid.*

Proof. $(N(A), \supseteq)$ and $(V^+, \sqsupseteq)$ obviously are partially ordered sets (V^+ even is densely ordered by assumption). The abstraction function is monotonous: $\alpha \supseteq \beta \Rightarrow abs(\alpha) \sqsupseteq abs(\beta)$, because for $v = abs(\beta)$ $\alpha \supseteq \beta \supseteq L^{-1}(\langle v\rangle_o)$ holds, which implies that the supremum in the definition of $abs(\alpha)$ is at least v.

The concretisation function is monotonous: $v_1 \sqsupseteq v_2 \Rightarrow conc(v_1) \supseteq conc(v_2)$ holds, because of the monotonicity of L^{-1} and $\langle v_1\rangle_o \supseteq \langle v_2\rangle_o$.

Extensitivity, i.e. $\alpha \supseteq conc(abs(\alpha))$, holds: For $v = abs(\alpha)$ the definition of supremum together with $\sqsupseteq$ being total and *abs* being monotonous implies that $\forall v' \in \langle v\rangle_o.\, \alpha \supseteq L^{-1}(\langle v'\rangle_o)$ is valid. Hence, α also is a superset of the union of these inverse images, $\alpha \supseteq \bigcup_{v' \in \langle v\rangle_o} L^{-1}(\langle v'\rangle_o)$. The union can be rewritten as $L^{-1}(\{v'' \mid \exists v'.\, v \sqsupset v' \sqsupset v''\})$. As V is densely ordered by $\sqsupset$, there is a v' between any two elements of V (Theorem 10). Therefore, $\{v'' \mid \exists v'.\, v \sqsupset v' \sqsupset v''\} = \{v'' \mid v \sqsupset v''\} = \langle v\rangle_o$ holds, which finally yields that $\alpha \supseteq L^{-1}(\langle v\rangle_o) = conc(v)$.

$abs(conc(v)) = v$, which implies reductivity, holds because L is onto: For $\alpha = conc(v)$ and $v' = v$ surely $\alpha = L^{-1}(\langle v\rangle_o) \supseteq L^{-1}(\langle v'\rangle_o)$ holds. Hence, $abs(\alpha) \sqsupseteq v'$ must hold. Due to L being onto there can be no v'' with $v'' \sqsupset v'$ and $\alpha \supseteq L^{-1}(\langle v''\rangle_o)$. In more detail this is, because for $v'' \sqsupset v'$ there is a $\bar{v}$, $v'' \sqsupset \bar{v} \sqsupset v'$, such that, due to L being a function and onto, $L^{-1}(\bar{v}) \cap L^{-1}(\langle v'\rangle_o) = \emptyset$ holds, which in turn implies that $L^{-1}(\langle v''\rangle_o) \supsetneqq L^{-1}(\langle v'\rangle_o)$. Thus, $v' = v$ is the supremum required in the definition of *abs*. $\square$

Theorem 8 *Given the abstraction and concretisation functions abs and conc as defined in Section 4.1.1 monotonicity of L along the system traces of Sys_S implies the stability of A w.r.t. the topology considered on $\mathcal{S}$.*

Proof. Let $\alpha \in N(A)$. For $v = abs(\alpha)$ monotonicity of L along the traces of Sys_S implies that $\forall s \in \mathcal{S}. L(s) \sqsubset v \Rightarrow \forall \sigma \in Sys_S(s,\iota). \forall t. L(\sigma(t)) \sqsubseteq L(s)$ for all $\iota \in I$ since $\sigma(0) = s$ for $\sigma \in Sys_S(s,\iota)$. Hence, selecting $\beta = conc(v)$ yields that $\forall s \in \beta. \forall \sigma \in Sys_S(s,\iota). \forall t. L(\sigma(t)) \subseteq L(s) \sqsubset v$. By the definition of *conc*, $L(\sigma(t)) \sqsubset v$ implies $\sigma(t) \in conc(v)$. Together with $\alpha \supseteq conc(abs(\alpha))$ (extensivity) this implies $\sigma(t) \in \alpha$ for all t, $\sigma \in Sys_S(s,\iota)$, $s \in \beta$ and $\iota \in I$. $\square$

Theorem 9 *Function L, as defined in Section 4.1.2, satisfies the requirements of Theorem 1.*

Proof. Requirements one and two trivially hold. To verify the third requirement, let $v \in V^+$. We can then choose $\alpha = L^{-1}(\langle v \rangle)$ which is open because of the continuity of L and which contains A because $L(A) = \bot \in \langle v \rangle$. As L-values of elements of α are less or equal v w.r.t. $\sqsubseteq$, the third requirement holds. For the fourth requirement we choose $L(\alpha)$ for $\alpha \in N(A)$ where L is extended to sets by pointwise extension. As L^{-1} is continuous, $L(\alpha)$ is open in V and can therefore be written as the union of a family of sets in the base of the left topology of V, i.e. $L(\alpha) = \bigcup_{v \in J} \langle v \rangle$ where $J \subseteq V^+ = V$. We select a $v \in J$ and get $L^{-1}(\langle v \rangle) \subseteq \alpha$ as demanded by the fourth requirement. $\square$

B Sets and Orders

Definition 9 (Order.) *Let X be a set and $\sqsubseteq$ a relation on X, $\sqsubseteq \subseteq X \times X$. $\sqsubseteq$ is an* order *iff it is transitive, reflexive and antisymmetric. If for all $x, y \in X$ either $(x,y) \in \sqsubseteq$ or $(y,x) \in \sqsubseteq$ holds, the order is* total, *otherwise it is* partial. *Usually infix notation is used for orders, i.e. we write $x \sqsubseteq y$ for $(x,y) \in \sqsubseteq$. $(X, \sqsubseteq)$ is called a* partially ordered set, *iff $\sqsubseteq$ is a partial order on X. It is called a* totally ordered set, *iff $\sqsubseteq$ is a total order on X. If X has a least element w.r.t. $\sqsubseteq$ this element is called bottom element and denoted by $\bot$.*

Definition 10 (Complete partial order (Cpo).) *A partially ordered set $(X, \sqsubseteq)$ is called a* complete partial order *iff if there is a least upper bound for any increasing chain $x_1 \sqsubseteq x_2 \sqsubseteq \ldots$ in X.*

For an order $\sqsubseteq$ on X we write $\sqsubset$ to denote the irreflexive relation $\sqsubseteq \setminus id_X$, where id_X is the identity relation on X.

Definition 11 (Densely ordered set [Eng89].) *Let $(X, \sqsubseteq)$ be a totally ordered set. Let $A, B \subseteq X$ be a partitioning of X, i.e. $A \cup B = X$, such that $A, B \neq \emptyset$ and $x \in A$ and $y \in B$ implies $x \sqsubset y$. X is called* densely ordered *by $\sqsubset$ if for any such partitioning either A has no largest element or B has no smallest element.*

Theorem 10 *For a set X densely ordered by $\sqsubset$, for all $x, z \in X$ with $x \sqsubset z$ there is a $y \in X$ with $x \sqsubset y \sqsubset z$.*

Proof. The proof proceeds by contraposition. Assume there is no y between x and z. Let $A = \{a \in X \mid a \sqsubseteq x\}$ and $B = \{b \in X \mid x \sqsubset b\}$. Clearly, $A \cup B = X$ and $a \in A \ \wedge \ b \in B \Rightarrow a \sqsubset b$ holds. Furthermore, A contains a largest element, x. Due to the assumption, B contains z as its smallest element. This contradicts the assumption that X is a densely ordered set. $\square$

C Some Topology

This section introduces some basic notions from topology. Most of the definition are taken or adapted from [Eng89].

Definition 12 (Topological space [Eng89].) *For a set X and a family of subsets $\mathcal{O} \subseteq \wp(X)$, $(X, \mathcal{O})$ is a* topological space *iff*

- $\emptyset \in \mathcal{O}$ *and* $X \in \mathcal{O}$,
- *for* $U_1 \in \mathcal{O}$ *and* $U_2 \in \mathcal{O}$, $U_1 \cap U_2 \in \mathcal{O}$,
- *for* $\mathcal{A} \subseteq \mathcal{O}$, $\bigcup \mathcal{A} \in \mathcal{O}$.

X is also called a *space*, $\mathcal{O}$ is called a *topology* on X, and the elements of $\mathcal{O}$ are called *open* sets w.r.t. the topology. A set $U \subseteq X$ is called *closed* if its complement $X \setminus U$ is an open set.

For two topologies $\mathcal{O}_1$ and $\mathcal{O}_2$ on a set X topology $\mathcal{O}_1$ is *coarser* than topology $\mathcal{O}_2$ iff $\mathcal{O}_1 \subseteq \mathcal{O}_2$.

Definition 13 (Neighborhood of a point.) *For $x \in X$, a set $U \in \mathcal{O}$ is a* neighborhood *of x iff $x \in U$ [Eng89]. We define $N(x)$ to denote the set of all neighborhoods of x, formally $N(x) = \{O \in \mathcal{O} \mid x \in O\}$.*

The notion of neighborhood is extended so sets in an elementwise manner.

Definition 14 (Neighborhood of a set.) *$U \in \mathcal{O}$ is a* neighborhood *of $Y \subseteq X$ iff U is a neighborhood for every element of Y, formally $Y \subseteq U$. The set of all neighborhoods of Y is defined as $N(Y) = \{O \in \mathcal{O} \mid Y \subseteq O\}$.*

Note that this notion of neighborhood implies that every open set is a neighborhood of itself: $Y \in \mathcal{O} \Rightarrow Y \in N(Y)$

Example 1 (Discrete topology.) *$(X, \wp(X))$ is a topological space, where $\wp(X)$ is the set of all subsets of X. $\mathcal{O} = \wp(X)$ is called the* discrete topology *on X [Eng89]. In the discrete topology every set $U \subseteq X$ is open* and *closed.*

Definition 15 (Base for a topological space.) *A family $\mathcal{B} \subseteq \mathcal{O}$ of open sets is a* base for the topological space $(X, \mathcal{O})$ *iff every non-empty open set can be represented by a union of sets in $\mathcal{B}$.*

[Eng89] states that any base has the properties:

(B1) for any $B_1, B_2 \in \mathcal{B}$ and for all $x \in B_1 \cap B_2$ there is a $B \in \mathcal{B}$ such that $x \in B \subseteq B_1 \cap B_2$

(B2) for any $x \in X$ there is a $B \in \mathcal{B}$ such that $x \in B$

Theorem 11 (Topology generated by a base.) *Let $\mathcal{B}$ be a family of subsets of X with the properties (B1) and (B2). Then the set $\mathcal{O} \subseteq \wp(X)$ consisting of all subsets of X which are unions of sets in $\mathcal{B}$ is a topology on X and $\mathcal{B}$ is a base for this topology. $\mathcal{O}$ is called the* topology generated by base $\mathcal{B}$.

Proof. A proof is given in [Eng89]. □

Example 2 (Left topology.) *Let X be a set partially ordered by $\sqsubseteq$. The sets $\langle x \rangle = \{x' \mid x' \sqsubseteq x\}$ for every $x \in X$ generate a topology on X. This topology is called the* left topology on X induced by $\sqsubseteq$ *[Eng89].*

To show that the proposed sets indeed generate a topology we apply Theorem 11, which amounts to showing that the sets satisfy the assumptions of the theorem. For the first assumption holds as $x \in \langle x_1 \rangle \cap \langle x_2 \rangle$ implies that $\langle x \rangle \subseteq \langle x_1 \rangle \cap \langle x_2 \rangle$ because of transitivity of $\sqsubseteq$. The second assumption is valid because $x \in \langle x \rangle$ for all $x \in X$.

Example 3 (Interval topology.) *Let X be a set containing at least two elements and totally ordered by $\sqsubseteq$ (i.e. for $x, y \in X$ $x \sqsubseteq y$ or $y \sqsubseteq x$ holds). $\sqsubseteq$ is also called a* linear order. *We write $x \sqsubset y$ for $x \sqsubseteq y \ \wedge \ x \neq y$. For $x, y \in X$ with $x \sqsubset y$ the sets $(x, y) = \{z \in X \mid x \sqsubset z \sqsubset y\}$, $(\leftarrow, y) = \{z \in X \mid z \sqsubset y\}$ and $(\rightarrow, x) = \{z \in X \mid x \sqsubset z\}$ are called* intervals. *The set of all intervals on X generates a topology which is called the* interval topology on X induced by $\sqsubseteq$ *[Eng89].*

To prove that the intervals generate a topology we apply Theorem 11, i.e. we have to show that the set of all intervals satisfies properties (B1) and (B2) from above. (B1) holds as $x \in (a, b) \cap (c, d)$ for $a, c \in X \cup \{\leftarrow\}$ and $b, c \in X \cup \{\rightarrow\}$ implies $x \in (max\{a, c\}, min\{b, d\})$ where $max\{\leftarrow, y\} = y$ for $y \in X \cup \{\leftarrow\}$ and min is defined analogously. (B2) is valid since for $x \in X$ we can select an arbitrary $y \in X \setminus \{x\}$ (X has at least two elements) such that either $x \in (\leftarrow, y)$ or $x \in (y, \rightarrow)$ holds.

Definition 16 (Continuous function.) *A function $f \in X \rightarrow Y$ between two topological spaces X and Y is called continuous if the inverse image of every open set of Y is open in X.*

Note that this definition is equivalent to the one usually used in computer science. There continuity amounts to the preservation of least upper bounds of increasing chains by monotonous functions on Cpo's. If the Scott topologies induced by the order on the domain and range of the function are used, our topological definition and the definition based on Cpo's are equivalent [Win93].

Theorem 12 (Topology generated by a function.) *Let f be a function from X to topological space Y and let $\mathcal{B}_Y$ be a base for the topology on Y. Then the topology $\mathcal{O}_X$ generated by the base $\mathcal{B}_X = \{f^{-1}(B) \mid B \in \mathcal{B}_Y\}$ is the coarsest topology on X which makes f continuous.*

Note that $f^{-1}(B)$ is the inverse image of B under f.

Proof. First, we show that the $\mathcal{B}_X$ generate a topology. Let $x \in f^{-1}(B_1) \cap f^{-1}(B_2)$, $B_1, B_2 \in \mathcal{B}_Y$. Hence, $f(x) \in B_1 \cap B_2$. As $\mathcal{B}_Y$ is a base, there is a $B \in \mathcal{B}_Y$ with $f(x) \in B \subseteq B_1 \cap B_2$. Thus, $x \in f^{-1}(B)$, $f^{-1}(B) \in \mathcal{B}_X$ and $f^{-1}(B) \subseteq B_1 f^{-1}(B_1) \cap f^{-1}(B_2)$ by some set arithmetic. Furthermore, for every $x \in X$ there is a $f^{-1}(B) \in \mathcal{B}_X$ with $x \in f^{-1}(B)$, because of the respective property of $\mathcal{B}_Y$ on Y. Therefore, Theorem 11 can be applied and yields that $\mathcal{B}_X$ generates a topology on X.

By the definition of base and continuity it is easy to see that f is continuous w.r.t. the topologies generated by $\mathcal{B}_X$ and $\mathcal{B}_Y$.

Third, we show that $\mathcal{B}_X$ is the coarsest topology that makes f continuous. Let $\mathcal{O}$ be a topology on X such that f is continuous and let α be an open set w.r.t. the topology generated by $\mathcal{B}_X$, i.e. $\alpha = \bigcup U_J$ for some $U_J \subseteq \mathcal{B}_X$. By definition each $U \in U_J$ is given as $U = f^{-1}(B)$ for some $B \in \mathcal{B}_Y$. By the assumption f is continuous w.r.t. $\mathcal{O}$ which implies that these U are open w.r.t. $\mathcal{O}$. Hence, their union α is also open and correspondingly the topology generated by $\mathcal{B}_X$ is coarser than $\mathcal{O}$. $\square$

Definition 17 (Tychonoff topology.) *Let $\{X_i\}_{i\in I}$ be a family of topological spaces. The topology on the product space $X = \Pi_{i\in I} X_i$ which is generated by the base $\mathcal{B}_X = \{\bigcap_{k=1}^{\ell} \pi_{i_k}^{-1}(o_{i_k}) \mid o_{i_k} \in \mathcal{O}_{X_{i_k}} \wedge i_1, \ldots, i_\ell \in I\}$, where $\mathcal{O}_{X_i}$ is the topology on X_i and π_i is the projection from X to X_i, is called the* Tychonoff topology *on X.*

Due to the definition given in [Eng89] for topologies generated by *families* of mappings, the Tychonoff topology is generated by the family of projections $\{\pi_i\}_{i\in I}$ (see also Theorem 12 for topologies generated by *single* functions).