A Formal Approach based on Fuzzy Logic for the Specification of Component-Based Interactive Systems

Vasileios Koutsoumpas
Technische Universität München, Munich, Germany
koutsoum@in.tum.de

Formal methods are widely recognized as a powerful engineering method for the specification, simulation, development, and verification of distributed interactive systems. However, most formal methods rely on a two-valued logic, and are therefore limited to the axioms of that logic: a specification is valid or invalid, component behavior is realizable or not, safety properties hold or are violated, systems are available or unavailable. Especially when the problem domain entails uncertainty, impreciseness, and vagueness, the appliance of such methods becomes a challenging task. In order to overcome the limitations resulting from the strict modus operandi of formal methods, the main objective of this work is to relax the boolean notion of formal specifications by using fuzzy logic. The present approach is based on Focus theory, a model-based and strictly formal method for component-based interactive systems. The contribution of this work is twofold: *i*) we introduce a specification technique based on fuzzy logic which can be used on top of Focus to develop formal specifications in a qualitative fashion; *ii*) we partially extend Focus theory to a fuzzy one which allows the specification of fuzzy components and fuzzy interactions. While the former provides a methodology for approximating I/O behaviors under imprecision, the latter enables to capture a more quantitative view of specification properties such as realizability.

1 Introduction

Formal methods are widely recognized as a powerful engineering method for the specification of interactive systems [3]. They follow the principle of "*correctness by construction*" and are therefore well suited for security-critical systems [11]. Although the promises of formal methods are well known [16], there are many limitations preventing the usage in industrial software development. The following limitations are generally identified in literature [4, 22] as the main blockers: *1*) *Limited scope*: Formal methods are not well suited to specifying user and environment interfaces and interactions; *2*) *Limited scalability*: As systems increase in size, the time and effort required to develop a formal specification grows disproportionately; *3*) *Limited expressiveness*: standard formal methods are not capable to quantify values between the "absolute truth" and the "absolute false".

Through the longtime experience obtained within the research projects SPES [21] and E-Energy[1], we empirically confirmed the presence and challenges of the above stated limitations for the avionic, automotive, and smart grid domain. Driven from the individual problems recognized in each domain, there is a natural question whether it is possible to extend standard formal methods to allow on the one hand to speed up the development of specifications while on the other hand the specification should remain formal enough to allow the promises of formal methods such as verification, model checking, etc. To advance this overarching question we distinguish between two major problem categories:

[1] http://www.e-energy.de/en/

B. Buhnova, L. Happe, J. Kofroň (Eds.):
Formal Engineering approaches to Software Components and Architectures (FESCA'15).
EPTCS ??, 2015, pp. 1–15, doi:10.4204/EPTCS.??.??

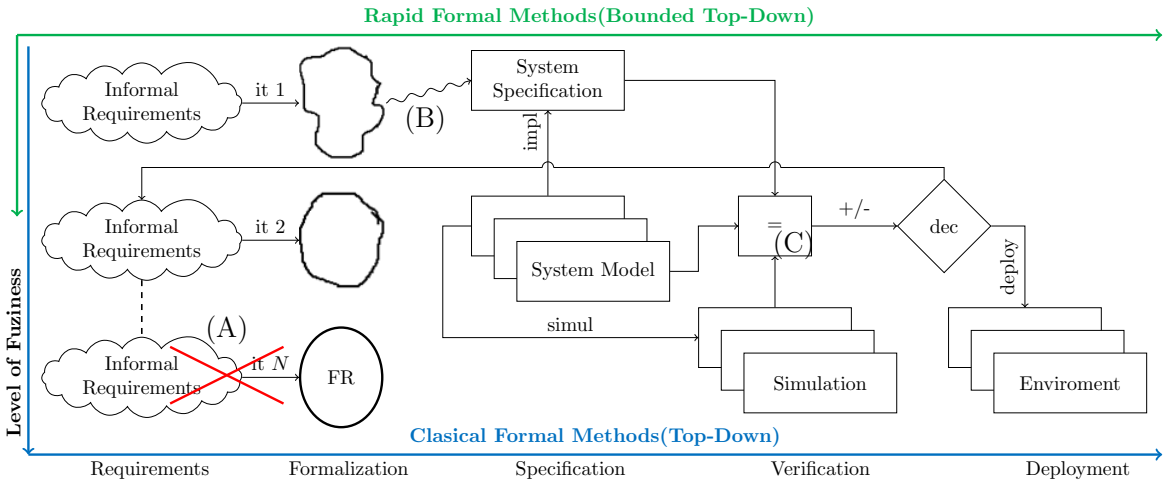


Figure 1: (A) Instead of an iterative refinement we suggest to proceed with formal modeling as early as possible (B) We propose a specification technique for: 1) formalizing qualitative properties of components and; 2) for approximating component behavior in terms of a rule base (C) we provide an equivalence model which allows to capture distances between specifications and systems

Problem Statement 1: Formal methods, such as Focus [3] or Z [13], permit the precise and unambiguous modeling of interactive component behavior. To achieve that, it's necessary to formalize the informal system requirements. Since vagueness, imprecision, and ambiguity are inherent in natural language, the informal system requirements suffer also from this. Thus, a tight feedback loop between detailed requirements specification and formal specification is observed and repeated until the formal specification becomes precise enough to continue with the implementation. Nevertheless, some system problems, particularly those drawn from the systems engineering domain, where the system's context includes user and environment interactions, may be difficult to model in crisp or precise terms. Furthermore, in order to meet the project's time constraints, it may be desirable that formal methods should commence as early as possible, even though the understanding of parts of the problem domain is only approximate. Hence, the first problem we deal with in this paper is visualized in Figure 1 and addresses the research question: How to soften the aforementioned tight feedback loop?

Problem Statement 2: Once a formal system specification is defined, standard verification systems (e.g. Isabelle [20]) return a boolean answer that indicates whether a system behavior conforms to its specification. Hence, two distinct behavior clusters are formed, namely that of correct and that of incorrect behaviors. However, not all correct behaviors are equally good, and not all incorrect behaviors are equally bad. Thus, a second research question rises whether it is possible to relax the strict boolean notion of formal methods to capture a more fine grained view as depicted in Figure 1 between specification and possible implementations. Such a view, allows for quantitative reasoning about specification properties such as realizability, safety, and liveness, to name only a few.

Motivation and Research Objective: The identified problems are closely related to the strict mathematical concepts used in formal methods. Most of them are based on crisp sets and on a two-valued logic, and are therefore limited to the axioms of that logic. Many researcher [15, 19] have successfully applied probabilistic and stochastic approaches to deal with uncertainty resulting from the lack of information. However, there is also another source of uncertainty, resulting from the inability to characterize

information. The latter is also the kind of uncertainty we address in this work. In recent years, there is a number of research attempts [4, 5, 12, 17, 18], which point out the need for emerging ideas and concepts to overcome these limitations. Indeed, most existing approaches, especially those addressing the second problem are based on distance specification [4, 5, 12]. Their attempt is to relax the boolean notion by defining custom distances for each specification property and to measure the corresponding deviation. The alternative we suggest in this work is an innovative approach where we use fuzzy logic to tackle with this problem. The overall idea is schematically depicted in Figure 1 and can be understood as a combination of rapid prototyping with formal methods. We call this engineering method Rapid Formal Methods (RFM). The research objective is to establish the basic foundations and concepts needed towards a complete theory for the specification of fuzzy interactive systems. Such a theory should provide the necessary concepts for developing softer specifications but also for modeling fuzzy interactions. The presentation of a complete theory within this paper is not possible and thus we concentrate on component behavior.

Structure: Section 2 presents the related foundations of Focus and fuzzy set theory. In addition the conventions made for this paper are declared. Section 3 describes how fuzzy logic applies on top of Focus to develop specifications based on qualitative properties. In Section 4 the concept of fuzzy components is introduced and the necessary formalisms are presented. Section 5 lists the related work and establishes a border between this and other approaches. Finally, Section 6 concludes the present work and describes possible future directions.

2 Preliminaries

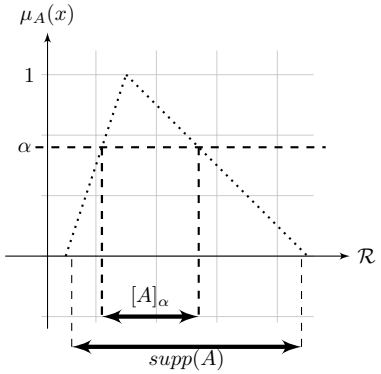
Focus Theory. We base our approach on Focus [3], a model-based and strictly formal software and systems engineering method for distributed interactive systems. The method builds on top of High-Order, two-valued, typed Logic (HOL [1]), which describes systems in terms of their structure (syntactic) and behavior (semantic). The system structure is determined by a static hierarchy of components, each defining an interface $I \blacktriangleright O$ through a set of typed input channels $I \in \mathbb{I}$ and typed output channels $O \in \mathbb{O}$.

The central concept of Focus is that of a stream, which is used to represent communication histories. Let M be a given set of messages. A stream s over the set M is a finite (M^*) or an infinite (M^∞) sequence of elements from M. Furthermore, the set of timed streams denoted by $M^{\aleph} =_{def} (M^*)^\infty$ represent an infinite history of finite communications over a channel that are carried out in a discrete time frame. The k-th sequence in a timed stream represents the sequence of messages exchanged on the channel in the k-th time interval.

Further, different components can be connected through I/O channels to describe component interaction through message exchange. Hence, component behavior is determined by a mapping from the set of possible input histories (streams over input channels $\vec{I}$) to the set of possible output histories (streams over output channels $\vec{O}$). Therefore, the semantic interface of a component is denoted by a set-valued function $F : \vec{I} \rightarrow \wp(\vec{O})$. For example, this mapping can be expressed by means of automata including states and transitions with guards over input histories and actions over output histories, but other description techniques such as table specifications [7] are supported in principle as well.

Fuzzy Set Theory. We assume that the reader has a basic knowledge of fuzzy set theory and fuzzy logic. For a detailed description, we refer to [14, 24, 25].

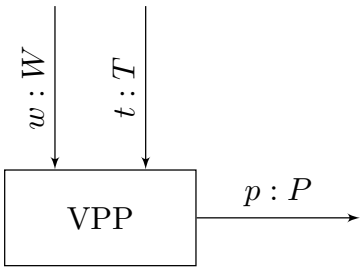
A fuzzy set μ of X is a function from the reference set X to the unit interval, formally $\mu : X \rightarrow [0,1]$. $\mathcal{F}(X)$ denotes the set of all fuzzy sets of X. The value $\mu(x)$ is called degree of truth and the function μ is called membership function.



A fuzzy set can be represented by a continuous membership function μ, or by a set A of ordered pairs. The latter is denoted by $A = \{(x, \mu_A(x)) \mid x \in X\}$. The set $supp(A) =_{def} \{x \in X \mid \mu_A(x) > 0\}$ is called support of A. The set $[A]_\alpha =_{def} \{x \in X \mid \mu(x) \geq \alpha\}$ is called α-cut of A. The fuzzy set A is often denoted by $\{\mu_A(x_1)/x_1, ..., \mu_A(x_n)/x_n\}$. Now let $X, Y \subseteq R$ be universal sets, then a fuzzy relation R is a fuzzy set given by $R = \{((x,y), \mu_R(x,y)) \mid (x,y) \in X \times Y\}$. Qualitatively, a fuzzy relation can be understood as an expression of the form $R =$ "x is heavier than y", where $x \in X$, $y \in Y$ and $R \subseteq X \times Y$. Finally, let $R_1(x,y) \subseteq X \times Y$ and $R_2(y,z) \subseteq Y \times Z$ be two fuzzy relations. The composition of them is denoted by $R_1 \circ R_2$ defined in $X \times Z$. The membership function of the composed relation is given by the max-min composition denoted by $\mu_{R_1 \circ R_2} = Sup_y Min[\mu_{R_1(x,y)}, \mu_{R_2(y,z)}]$.

Conventions. Throughout this paper we make usage of some basic operators on streams. Let s be a stream, then $(s.k)$ denotes the k-th element of the stream, $s@t$ denotes the element of a timed stream at time point t, $(s{\downarrow}k)$ denotes the sequence of the first k sequences/messages in the stream and $(\#s)$ is the number of elements in s. For an infinite stream $(\#s) = \infty$. Furthermore, we define the functions $max(s)/sup(s)$ and $min(s)/inf(s)$, returning the maximum/supremum and minimum/infimum element of a finite/infinite stream, respectively. By $s_1 \copyright s_2$ we denote the concatenation of two streams. In general, messages of any type are supported by streams but for readability we use only the set of real numbers $\mathbb{R}$. Types and sets used in any context, i.e. $x : T$ and $x \in T$, respectively, are by default to be understood as crisp. Fuzzy sets are always stated explicitly. Fuzzy types are recognized by the prefix $\langle \mathcal{F}\_ \rangle$, followed by the type-name. We define the domain and the range of a fuzzy set by $dom.\mu_A =_{def} A$ and $rng.\mu_A =_{def} \{\mu_A(x) \mid x \in A\}$.

3 Fuzzy Logic on Top of Focus



In this Section we apply fuzzy logic on top of Focus to develop soft specifications for interactive systems. Consider the following simple example of a Virtual Power Plant (VPP) which exchanges weather information with its environment (i.e. weather station) and produces power to supply a network of consumers. A system according to Focus is specified if the syntactic and the semantic interface are fully specified. The former specifies how the system interact $(I \blacktriangleright O)$ with its environment while the latter specifies the behavior of the component denoted by $B : \vec{I} \rightarrow \wp(\vec{O})$. Formalizing the behavior of a component is not always easy. In the given example one first has to decompose the system in its elementary building blocks, for example a set of solar panels. Afterwards the formalization by means of mathematical models like differential equations of each behavior is required. For a detailed overview on how to apply formal methods to smart grid systems and the coherent challenges, we refer to [9] and [10].

3.1 Syntactic Interface - I/O Specification

First we have to extend the syntactic interface for the introduced example. As illustrated above the $(I \blacktriangleright O)$ of the VPP consists of its input channels w,t, its output channel p, and the types of messages that are transmitted on them. Messages received on w/t are of type W/T respectively, and messages sent along p are of type P. Since channels are typed, and Focus uses crisp sets to define types, we introduce a new concept namely that of fuzzy properties and fuzzy ports.

Definition 1 (Fuzzy Property)**.** A fuzzy property $\widetilde{p}$ is a three-tuple $\langle X, \xi, \pi_\xi \rangle$, where X is the universe of discourse which can be referenced by $\widetilde{p}$, ξ is a linguistic term which characterizes the property and $\pi_\xi : X \to [0,1] \cup \{\perp\}$ is the membership function. The value $\pi_\xi(x_i)$ is an indicator to what degree the property holds for a given $x_i \in X$. A fuzzy property can be represented by a fuzzy set $X_\xi = \{(x, \pi_\xi(x)) \mid x \in X\}$, which is fully specified by the three-tuple. By $\mathcal{P}$ we denote the set of all fuzzy properties.

Example 1. The tuple $\langle T, HIGH, \pi_{HIGH} \rangle$, where $T = \{t \in \mathbb{R} \mid (-30 \leq t \leq 40)\}$ defines a property which describes the high temperature for the VPP. A possible representation could then be $T_{HIGH} = \{0/15, 0.3/20, 0.6/25, 0.9/30, 1/35\}$, where the temperature of 15°C are considered to be high with a degree of truth 0, the temperature of 20°C are considered to be high with a degree of truth 0.3, and so on.

Definition 2 (Total Fuzzy Property)**.** We say that a property $\widetilde{p} = \langle X, \xi, \pi_\xi \rangle$ is total, denoted by $Def_{total}(\widetilde{p})$ if:

$$Def_{total}(\widetilde{p}) \Rightarrow \forall x \in X \; \exists y \in [0,1] : \pi_\xi(x) = y \tag{1}$$

Definition 3 (Partial Fuzzy Property)**.** We say that a property $\widetilde{p} = \langle X, \xi, \pi_\xi \rangle$ is partial, denoted by $Def_{partial}(\widetilde{p})$ if:

$$Def_{partial}(\widetilde{p}) \Rightarrow \exists x \in X : \pi_\xi(x) = \perp \tag{2}$$

In example 1, the defined property is partial because $\exists t \in T | \pi_{HIGH}(t) = \perp$, e.g. $\pi_{HIGH}(28) = \perp$. Defining total properties is time intensive, mostly because of the partial known interaction with the environment. Additionally, the possible deployment of a system in multiple environments requires to define each property separately for each environment. We will show later in this paper how to overcome this issues by defining mapping strategies over I/O streams.

Definition 4 (Fuzzy Port)**.** A fuzzy port Θ_T over a type T is a set of fuzzy properties $\Theta_T = \{\widetilde{p} \in \mathcal{P}\}$, which satisfies the following two conditions:

- Each property type is a subset of T, formally:

$$\forall \widetilde{p} \in \Theta_T \to \widetilde{p}.X \subseteq T \tag{c1}$$

- Each property is uniquely characterized by its linguistic term, formally:

$$\forall \widetilde{p}_1, \widetilde{p}_2 \in \Theta_T \mid \widetilde{p}_1 \neq \widetilde{p}_2 \to \widetilde{p}_1.\xi \neq \widetilde{p}_2.\xi \tag{c2}$$

A fuzzy port Θ_T is said to be well defined, only if, c1 and c2 are satisfied, $\Theta_T \vdash c1 \wedge c2$. Graphically, a fuzzy input/output port is denoted by a white/black circle $(\circ)/(\bullet)$, respectively, at the boundary of a component. By IP_S/OP_S we denote the set of all fuzzy input/output ports for a given system S. Furthermore, by $\widetilde{p}^{\Theta_T}$ we denote the property $\widetilde{p}$ which belongs to the fuzzy port Θ_T. This notation is further generalized also for the elements $\xi^{\Theta_T}, \pi_\xi^{\Theta_T}$ of a property.

Since fuzzy ports are formally specified we can now connect channels with fuzzy ports. I/O channels can be connected to I/O fuzzy ports respectively through connections. A connection is defined as the

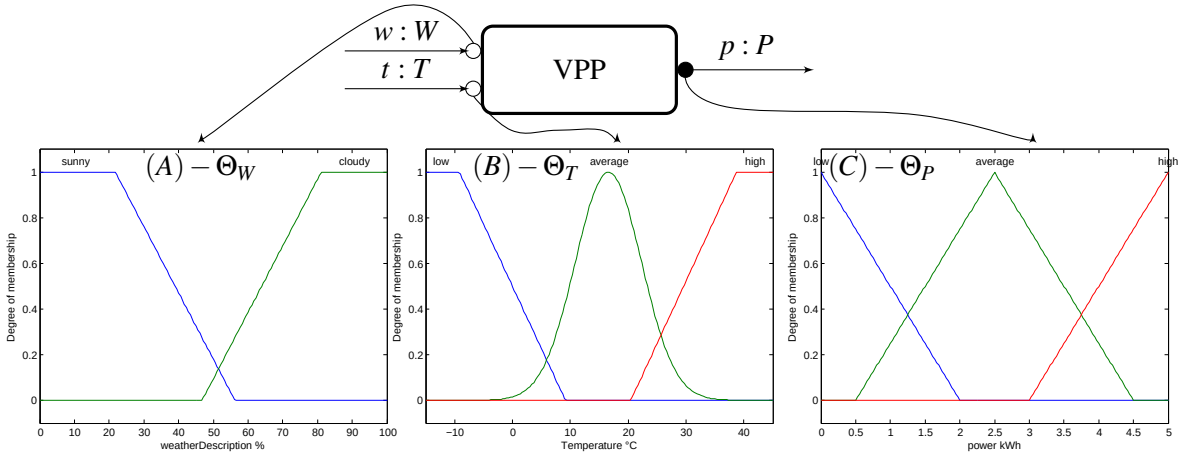


Figure 2: Syntactic Interface Specification for the VPP

binding of a concrete channel to a concrete fuzzy port. Note that not every channel can be connected to a concrete port. This is because ports and channels are specified separately. While the former is a characteristic of the component to be developed the latter may preexist i.e. consider we develop a component for an already existing system. Thus, following connectivity property has to hold:

Definition 5 (Connectivity)**.** A channel $c : C$ can be connected with a fuzzy port Θ_T only if: $C \subseteq T$. This property guarantees that each message transmitted over the channel c can be interpreted by the port Θ_T.

For the VPP example we define the set of fuzzy input ports $IP_{VPP} = \{\Theta_W, \Theta_T\}$, where $\Theta_W = \{W_{SUNNY}, W_{CLOUDY}\}$ (Figure 2-A) and $\Theta_T = \{T_{LOW}, T_{AVERAGE}, T_{HIGH}\}$ (Figure 2-B). The set of fuzzy output ports $OP_{VPP} = \{\Theta_P\}$ contains a single fuzzy port $\Theta_P = \{P_{LOW}, P_{AVERAGE}, P_{HIGH}\}$ (Figure 2-C). Figure 2 denotes the defined total properties of each fuzzy port. Intuitively a fuzzy port takes the role of an interpreter. For a given message received at some time point t over a channel c, the port gives all possible interpretations for each property. For example, the temperature of 23° can be interpreted to be high/average/low with degree of truth 0.2/0.6/0, respectively. Hence, given a port Θ_T and a measure $t \in T$, a port interpretation defines a total order $\leq$ on Θ_T, e.g. $T_{LOW} \leq T_{HIGH} \leq T_{AVERAGE}|_{t=23}$

Concluding, the syntactic interface of a component is fully specified if *1*) its I/O channels are specified and additionally to Focus theory *2*) the corresponding fuzzy I/O ports are well defined.

3.2 Semantic Interface - Behavior Specification

3.2.1 Rule Base Specification.

After specifying the syntactic interface of a system, we now specify the semantic by a rule base. Let $I = \{i_1 : I_1, ..., i_n : I_n\}$ and $O = \{o_1 : O_1, ..., o_m : O_m\}$ be a set of typed I/O channels. Furthermore, let $IP = \{\Theta_{I_1}, ..., \Theta_{I_n}\}$ and $OP = \{\Theta_{O_1}, ..., \Theta_{O_m}\}$ represent the well defined fuzzy ports that correspond to the typed I/O channels. For readability, we write $\widetilde{p}^i$ instead of $\widetilde{p}^{\Theta_{I_i}}$ to denote that a property $\widetilde{p} \in \Theta_{I_i}$. Then, a single rule for a specific $o \in O$ has generally the form:

$$R_r^o\text{: } \textbf{if } i_1@t \text{ is } \xi_{1,r}^{(1)} \text{ .. } \textbf{and } \ldots i_n@t \text{ is } \xi_{n,r}^{(n)} \textbf{ then } o@(t+1) \textbf{ is } \xi_r,\ r = 1,..,k \tag{3}$$

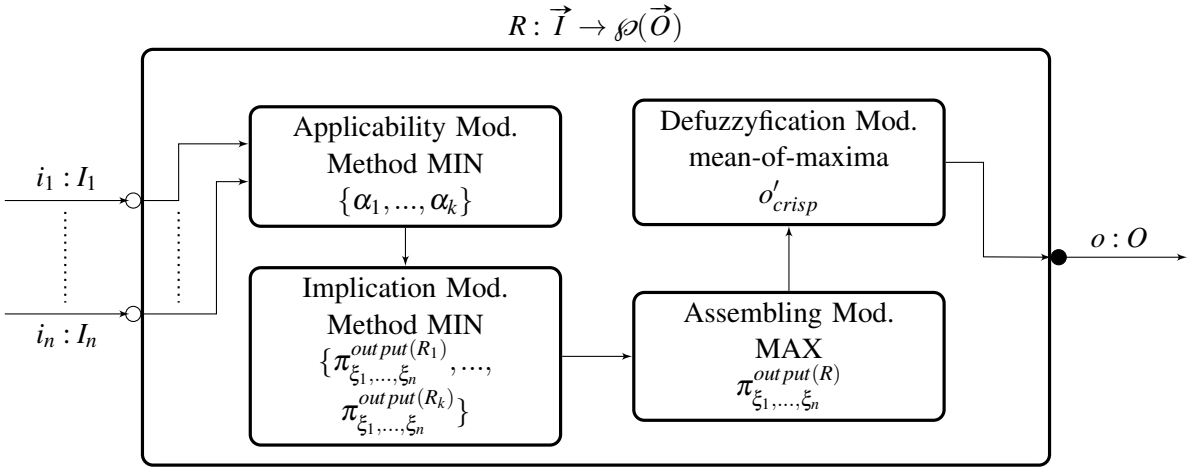


Figure 3: Behavior interpretation of a rule based specification

where $\xi^{(1)}_{1,r}, ..., \xi^{(n)}_{n,r}$, and ξ_r represent the linguistic terms that correspond to the fuzzy properties of a fuzzy port such that $\xi^{(j)}_{j,r} = \tilde{p}.\xi \mid \tilde{p} \in \Theta_{I_j}, j = 1, ..., n$ and $\xi_r = \tilde{p}.\xi \mid \tilde{p} \in \Theta_O$.

3.2.2 Behavior Specification

In the following we explain how the behavior function can be defined. Figure 3 depicts the required modules for the behavior specification. A tuple $\langle i_1@t, ..., i_n@t \rangle \in I_1 \times ... \times I_n$ denotes the measured input picked up by the syntactic interface at some time point t. For each rule R_r in the rule base we determine the degree to which the measured input fulfills the premise of the rule, called degree of applicability $\alpha_r = min\{\pi^{(1)}_{\xi_{1,r}}(i_1), ..., \pi^{(n)}_{\xi_{n,r}}(i_n)\}$. The applicability degrees are passed to the implication module where each rule R_r implies for the measured input the fuzzy output set $\pi^{output(R_r)}_{\xi_1,...,\xi_n} : O \to [0,1], o \longmapsto min\{\alpha_r, \pi_{\xi_r}(o)\}$. The output of a rule R_r is a fuzzy set of output values obtained by cutting of the fuzzy set π_{ξ_r} at the level of applicability α_r. The results are passed to the assembling module which combines all calculated fuzzy output sets (one for each rule) into a single fuzzy output set by determining the maximum $\pi^{output(R)}_{\xi_1,...,\xi_n} : O' \to [0,1], o' \longmapsto \max_{0 \le r \le k} \{\pi^{output(R_r)}_{\xi_1,...,\xi_n}\}$. The fuzzy output set is passed to the defuzzyfication module which decides for a crisp value o'_{crisp} by selecting the value with the maximum membership degree. In case where more values have the same degree the mean of maxima is selected. Finally, the crisp output value is passed to the output stream o. In case of multiple output channels the above procedure is repeated for each $o_i \in O, i = 1, ..., m$. Thus, the behavior of a system S is fully specified by the set of all output specifications $R_S = \{R^{o_1}, ..., R^{o_m}\}$. Given a set of timed input streams, the output streams are evaluated according to R_S for each time point.

Example 2. For the example depicted in Figure 2, let the fuzzy properties be defined according to the following scheme: $T_{HIGH} = \{\frac{0}{10}, \frac{0.4}{20}, \frac{0.6}{25}, \frac{0.8}{30}, \frac{1}{35}\}$, $T_{LOW} = \{\frac{0.2}{20}, \frac{0.4}{15}, \frac{0.6}{10}, \frac{0.8}{5}, \frac{1}{0}\}$, $W_{SUNNY} = \{\frac{0}{80}, \frac{0.4}{60}, \frac{0.6}{40}, \frac{0.8}{20}, \frac{1}{0}\}$, $W_{CLOUDY} = \{\frac{0}{20}, \frac{0.4}{40}, \frac{0.6}{60}, \frac{0.8}{80}, \frac{1}{100}\}$, $P_{HIGH} = \{\frac{0}{1}, \frac{0.4}{2}, \frac{0.6}{3}, \frac{0.8}{4}, \frac{1}{5}\}$, $P_{LOW} = \{\frac{0}{4}, \frac{0.4}{3}, \frac{0.6}{2}, \frac{0.8}{1}, \frac{1}{0}\}$.
Furthermore, let R^p be the rule base specification containing the following rules:

R_1: **if** t **is** $HIGH$ **and** w **is** $SUNNY$ **then** p **is** $HIGH$

R_2: **if** t **is** LOW **and** w **is** $CLOUDY$ **then** p **is** LOW

Given the tuple $\langle t@t_1, w@t_1 \rangle = \langle 20, 40 \rangle$ denoting the measured input at time point t_1, we are seeking for the output p. As a first step we calculate the degree of applicability for each rule. Thus, $\alpha_1 = min\{\pi^{(1)}_{HIGH}(20), \pi^{(2)}_{SUNNY}(40)\} = min\{0.4, 0.6\} = 0.4$ and $\alpha_2 = min\{\pi^{(1)}_{LOW}(20), \pi^{(2)}_{CLOUDY}(40)\} = min\{0.2, 0.4\} = 0.2$. By cutting of the fuzzy sets P_{HIGH}, P_{LOW} to the degree of applicability α_1 and α_2, respectively, we get the output value of each rule: $\pi^{output(R_1)}_{HIGH,SUNNY} = \{\frac{0}{1}, \frac{0.4}{2}, \frac{0.4}{3}, \frac{0.4}{4}, \frac{0.4}{5}\}$ and $\pi^{output(R_2)}_{LOW,CLOUDY} = \{\frac{0}{4}, \frac{0.2}{3}, \frac{0.2}{2}, \frac{0.2}{1}, \frac{0.2}{0}\}$. By assembling the fuzzy outputs of each rule we get: $\pi^{output(R)} = \{\frac{max(0.2,0)}{0}, \frac{max(0.2,0)}{1}, \frac{max(0.2,0.4)}{2}, \frac{max(0.2,0.4)}{3}, \frac{max(0,0.4)}{4}, \frac{max(0,0.4)}{5}\} = \{\frac{0.2}{0}, \frac{0.2}{1}, \frac{0.4}{2}, \frac{0.4}{3}, \frac{0.4}{4}, \frac{0.4}{5}\}$. Finally, applying the mean of maxima we get $o = 3.5$ which is the crisp output that is passed to the output channel p.

Theorem 1. *Every rule based behavior specification $R_S = \{R^{o_1}, ..., R^{o_m}\}$, where R^{o_i} is of the generally form given by equation 3, has a deterministic behavior interpretation $R : \vec{I} \to \wp(\vec{O})$, which defines a total deterministic Moore machine (Δ, Λ) with transition function:*

$$\Delta : (\Sigma \times (I \to M^*)) \to \wp(\Sigma \times (O \to M^*)) \tag{4}$$

The above theorem states that despite the fact that the rule based behavior specification relies on fuzzy properties, the component behavior from a black box point of view is not fuzzy at all. This implies, that the abstraction from a rule based behavior specification leads to a crisp deterministic interface behavior R. Consequently, tools like Autofocus [2] and theorem provers like Isabelle [20] can be further used for behavior analysis.

3.3 Mapping Strategies

The definition of total properties requires a total mapping from the reference set to the unit interval. This mapping may be achievable for static properties such as the speed of a car. However, most properties especially when modeling complex systems with environmental interactions are in nature not static. How high temperature should be interpreted depends highly on the geographically location the system will be deployed in. Furthermore, the temperature of 15°C may considered to be high in winter but only average in summer. Therefore, properties can be also time dependent. To deal with location and time dependency of properties we introduce the concept of mapping strategies. Such a strategy defines the membership function of a property according to the observed history of a channel. Thus, the property adapts to the location of a component. Additionally, a threshold for the history length may be declared to consider only recent interactions, this guarantees a smooth adaption of the membership function over time.

Definition 6 (Mapping Strategy)**.** A mapping strategy for a given property $\widetilde{p} = \langle X, \xi, \pi_\xi \rangle$(partial or total) is a high order function over a stream to a membership function for that property, formally:

$$mapstr_\xi : Stream\, X, \mathbb{N} \cup \{\infty\} \to (\pi_\xi : X \to [0,1]) \tag{5}$$

Example 3 (Mapping Strategy). For the VPP example the signature of a concrete mapping strategy for the property average temperature $T_{AVERGAGE}$ could be declared as:

fct $mapstr_{T_{AVERAGE}}$(*t* : Stream T, *n*: Nat) **fct** $\pi_\xi(x : \mathrm{T})\{$
ret gaussmf(min(t↓n), max(t↓n))}

4 Fuzzy Components

In Section 3 we showed that fuzzy logic is well suited for modeling soft properties and develop rule based specifications. We proved that the abstraction of a rule based behavior specification leads to a crisp deterministic interface behavior $R : \vec{I} \to \vec{O}$. However, not all correct behaviors are equally good, and not all incorrect behaviors are equally bad. Thus, we introduce the concept of fuzzy components and fuzzy behavior of them. This description yields a quantitative reasoning about component behaviors. Figure 4 depicts the extension of a component with deterministic behavior $b : \vec{I} \to \vec{O}$ to a fuzzy component with fuzzy behavior $\hat{b} : \vec{\vec{I}} \to \vec{\vec{O}}$ which is the subject of this section.

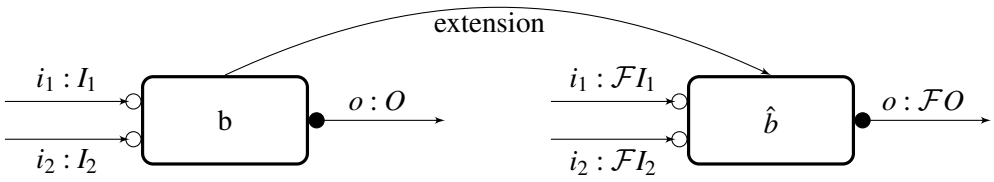


Figure 4: Fuzzy Components and Fuzzy Behavior

4.1 Basic Adaption

To enable fuzzy component behavior first we have to extend Focus theory in order to deal with fuzzy types. Thus, we introduce the notion of fuzzy types/channels. The prefix symbol $\mathcal{F}$ defines a fuzzy type as a total function from the crisp reference type T to the unit interval [0,1], denoted by $\mathcal{F}T : T \to [0,1]$. Now, let a set T_F of fuzzy types $\mathcal{F}T$ be given. By C_F we denote the set of fuzzy channels. Furthermore, we assume that we have given a fuzzy type assignment for the fuzzy channels: $f\_type : C_F \to T_F$. Given a set C_F of fuzzy channels, a valuation or history of a fuzzy channel is denoted by:

$$\vec{\vec{C_F}} = \{x : C_F \to M^{\mathcal{N}} : \forall c \in C_F : x.c \in \{dom.(f\_type(c))\}^{\mathcal{N}}\} \tag{6}$$

A valuation of a fuzzy channel $x \in \vec{\vec{C_F}}$ associates a stream s of elements of type $dom.f\_type(c)$ with each fuzzy channel $c \in C_F$. Throughout this chapter we work with a simple notation for streams over fuzzy channels which is described in the following. By $s.j$ we denote the j-th element of the stream s and by $acc_c(s.j)$ we denote the degree of membership of $s.j$ in $dom.f\_type(c)$. Informally, the value $acc_c(s.j)$ tell us to what degree element $s.j$ is accepted by channel c. If we combine two elements ($s.j$, $s.k \mid j,k \in \mathbb{N} \wedge j \neq k$) of a stream, their combination is rated according to the following scheme:

$$\begin{aligned} &\text{Lower: } acc_{\downarrow}(s.j,s.k) = (s.j \wedge s.k) = min\{(acc_c(s.j), acc_c(s.k)\} \\ &\text{Upper: } acc_{\uparrow}(s.j,s.k) = (s.j \vee s.k) = max\{(acc_c(s.j), acc_c(s.k)\} \end{aligned} \tag{7}$$

For a finite number of elements in a stream we define analogously the acceptance degree of a stream s by:

$$acc_{\downarrow}(s) = \min_{0 \leq j \leq \#s} \{acc(s.j)\} \mid acc_{\uparrow}(s) = \max_{0 \leq j \leq \#s} \{acc(s.j)\} \tag{8}$$

Since streams can have an infinite number of elements the above scheme converts to following equations for the infinite case:

$$acc_{\downarrow}(s) = \inf_{0 \leq j \leq \#s = \infty} \{acc(s.j)\} \mid acc_{\uparrow}(s) = \sup_{0 \leq j \leq \#s = \infty} \{acc(s.j)\} \tag{9}$$

Furthermore, we can combine not only elements of the same stream but also from different streams as well using the scheme above with following replacement in equation 7 $(s.j/s1.j, s.k/s2.k)$. Thus, two or more streams can be combined in order to evaluate the upper and lower acceptance bounds. It is noteworthy to mention that the acceptance degree is not limited to the specified upper and lower bounds in this paper. A statistical representation for the acceptance degree is possible as well (e.g. $\overline{acc(s)} = \frac{1}{\#s}\sum_{j=0}^{\#s} acc(s.j)$). Which representation is best suited depends highly on the system characteristics. Hence, while for a fault tolerant system like a VPP some could prefer the statistical mean representation for a safety critical system like an airplane the lower and upper bounds seems to be more appropriate. Finally, having established a strict notion for fuzzy types, channels and stream processing we introduce the notion of a fuzzy syntactic interface of a component:

Definition 7 (Fuzzy syntactic interface)**.** Given a set of fuzzy input channels I_F and a set of of fuzzy output channels O_F we introduce the notion of a fuzzy syntactic interface of a component by (I_F, O_F) or symbolic $(I_F \widetilde{\blacktriangleright} O_F)$.

4.2 Fuzzy Extension

Theorem 2 (Fuzzy Type Extension)**.** *Let $f : I^n \to O$ be a mapping from typed inputs $(i_1 : I_1, ..., i_n : I_n)$ to a single typed output $o : O$. If the input becomes fuzzy through a fuzzy type assignment of the form $(i_1 : \mathcal{F}I_1, ..., i_n : \mathcal{F}I_n)$ then the fuzzy type extension of O is given by:*

$$\mathcal{F}O(o) \stackrel{def}{=} sup\{min\{\mathcal{F}I_1(i_1), ..., \mathcal{F}I_n(i_n))\}| \\ (i_1, ..., i_n) \in I^n \text{ and } o = f(i_1, ..., i_n)\} \tag{10}$$

Example 4 (Stateless Fuzzy Behavior). We show how the extension principle is applied to a stateless adder with deterministic behavior $o = f(i_1, i_2)$. Let $i_1 : I_1$, $i_2 : I_2$ and $o : O$ be of type $I_1 = \{2,3,4\}$, $I_2 = \{6,7,8\}$ and $O = \{8,9,10,11,12\}$, respectively. We are seeking for the fuzzy output type $\mathcal{F}O$ if the input of f becomes fuzzy typed.

fct $f =(i_1 : \mathrm{I}_1, i_2 : \mathrm{I}_2)$ **out** $o : \mathrm{O}\{$
ret $i_1 + i_2;\}$

fct $\hat{f} =(i_1 : \mathcal{F}I_1, i_2 : \mathcal{F}I_2)$ **out** $o :$ **?**$\{$
ret $i_1 + i_2;\}$

Let, $\mathcal{F}I_1$= {0.5/2, 1/3, 0.5/4} be a fuzzy type representing the "fuzzy 3" and $\mathcal{F}I_2$= {0.5/6, 1/7, 0.5/8} another fuzzy type representing the "fuzzy 7". Now, according to Theorem 2:

$$\mathcal{F}O(o) = sup\{min\{\mathcal{F}I_1(i_1), \mathcal{F}I_2(i_2)) \,|\, i_1 \in I_1, i_2 \in I_2 \text{ and } o = f(i_1, i_2)\}$$

For $i_1 + i_2 = 9$ we receive:

$$\begin{aligned}\mathcal{F}O(i_1 + i_2 = 9) &= max\{min(\mathcal{F}I_1(3), \mathcal{F}I_2(6)), min(\mathcal{F}I_1(2), \mathcal{F}I_2(7))\} \\ &= max(min(1, 0.5), min(0.5, 1)) = 0.5\end{aligned}$$

Repeating for all $o \in O$ we obtain $\mathcal{F}O = \{0/8, 0.5/9, 1/10, 0.5/11, 0/12\}$, which is the fuzzy type representing the "fuzzy 10" depicted in figure 5-A.

4.3 Fuzzy Component Behavior

Recall from section 3 where component behavior was denoted by $B : \vec{I} \to \wp(\vec{O})$, meaning that input histories $\vec{I}$ are mapped to all possible output histories $\vec{O}$ over the set-valued function B we turn to the motivation of a general method which enables the mapping of fuzzy input histories to all possible fuzzy output histories over a set-valued function $\hat{B}$, denoted by $\hat{B} : \tilde{\vec{I}} \to \wp(\tilde{\vec{O}})$. A fuzzy behavior $\hat{B}$ is called deterministic if $\hat{B}(x)$ is a one element set for each fuzzy input history x. Such a behavior is equivalent to a function $\hat{b} : \tilde{\vec{I}} \to \tilde{\vec{O}}$ where $\hat{B}(x) = \{\hat{b}(x)\}$.

Definition 8 (Fuzzy Behavior Extension)**.** Let $b : \vec{I} \to \vec{O}$ be a mapping from input histories $\vec{I}$ to output histories $\vec{O}$. The fuzzy extension of b is given by:

$$\hat{b} : \tilde{\vec{I}} \to \tilde{\vec{O}}$$

where $\forall o \in O$ we apply the fuzzy type extension theorem 2.

Definition 9 (α-Realizability)**.** A fuzzy I/O behavior $\hat{B}$ is called α-realizable, if there exist a total function $\hat{b} : \tilde{\vec{I}} \to \tilde{\vec{O}}$ such that:

$$\forall x \in \tilde{\vec{I}} : \hat{b}(x) \in \hat{B}(x) \wedge acc(\hat{b}(x)) \geq \alpha \tag{11}$$

$[\hat{b}]_\alpha$ is called an α-realization of $\hat{B}$. By $[\![\hat{B}]\!]_\alpha$ we denote the set of all α-realizations of $\hat{B}$. An output history $y \in \hat{B}(x)$ is called α-realizable for a fuzzy I/O behavior with input x, if there exists a realization $[\hat{b}]_\alpha \in [\![\hat{B}]\!]_\alpha$ with $y = \hat{b}(x)$.

Example 5 (Stateful Fuzzy Behavior). Consider the following two programs (left: boolean, right:fuzzy) which is the stateful extension for the example 4.

fct $b = (i_1 : \mathrm{I}_1, i_2 : \mathrm{I}_2)$ **out** $o : \mathrm{O}$ {
  $\langle \mathrm{first}(i_1) + \mathrm{first}(i_2) \rangle ©$
  $b(\mathrm{rest}(i_1), \mathrm{rest}(i_2))$}

fct $\hat{b} = (i_1 : \mathcal{F}I_1, i_2 : \mathcal{F}I_2)$ **out** $o : \mathcal{F}O$ {
  $\langle \mathrm{first}(i_1) + \mathrm{first}(i_2) \rangle ©$
  $b(\mathrm{rest}(i_1), \mathrm{rest}(i_2))$}

Now let $i_1 = \langle 2,3,4,3,3,4,2,3 \rangle$ be an input stream of fuzzy type $\mathcal{F}I_1$ and $i_2 = \langle 7,6,6,7,6,7,9,7 \rangle$ another input stream of fuzzy type $\mathcal{F}I_2$. Then, the fuzzy behavior $\hat{b}(i_1, i_2)$ is 0.5-realizable but it is not 0.75-realizable as visualized in Figure 5-B.

Concluding this Section, we showed how to extend basic specification properties like realizability, in order to tackle with fuzzy behavior. In a similar way, theorem 2 and definition 8 provide the necessary tools for formalizing further specification properties such as safety, liveness, and fairness.

5 Related Work

In the last decade many research efforts are recorded in literature [4–6,12,15,17–19], where classical formal methods have been extended with probabilistic, stochastic, distance measurement, and multi-valued logic techniques in order to deal with uncertainties in modeling component-based interactive systems. However, uncertainty has two distinct facets: randomness and fuzziness both of which play basic roles in human reasoning, decision making and concept formation [26]. While the former handles partial knowledge (lack of essential information) the latter deals with partial truth (inability to characterize information). Thus, we intentionally leave probabilistic and stochastic systems outside the scope of this

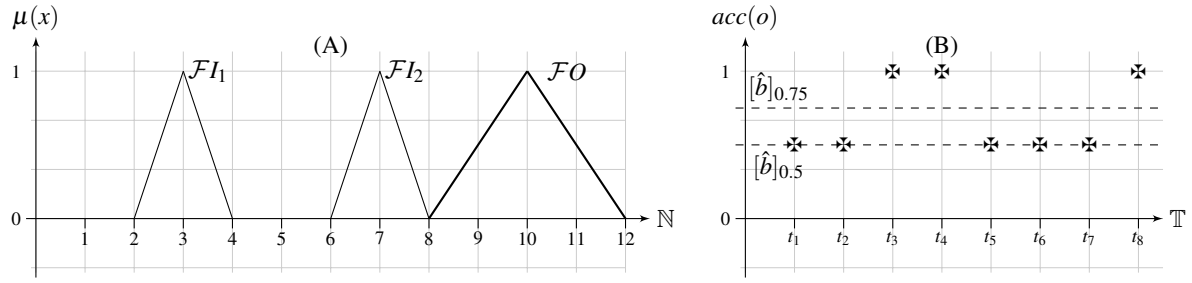


Figure 5: (A) Fuzzy type extension for a stateless adder. (B) α-realization of stateful adder with fuzzy behavior $\hat{b}$

paper, concentrating instead on how to deal with partial truth. For the specification and development of interactive systems in consideration of probabilistic effects we refer to Neubecks dissertation [19] where a theoretical framework for probabilistic systems is provided.

Chechik et al. [6] introduces the concept of multi-valued model-checking and describes a multi-valued symbolic model-checker, χChek for analyzing models that contain uncertainty or inconsistency. They develop a modeling language based on a generalization of Kripke structures, where both atomic propositions and transitions between states may take any of the truth values of a given multi-valued logic. In addition to the theoretical foundation they present a model-checking algorithm which is illustrated on some examples. Finally, the formalization of specification properties such us fairness in multi-valued model-checking is addressed. While Chechik et al. concentrate on logics with a finite set of truth values (a 3-valued logic is evaluated in their examples), we explore the case of continuous intervals of truth values. Furthermore, the concept of mapping strategies introduced in this paper enables the dynamic reconfiguration of specified intervals of truth values, which is also an extension to the aforementioned work.

With respect to formal specification based on fuzzy logic, Matthews et al. [18] suggests fuzzy set theory as a possible representation scheme to deal with uncertainty. The main contribution of their work is an extension of a set based specification language, namely Z. They develop a suitable fuzzy set notation within the existing syntax of Z. A summary of a toolkit is provided that defines the operators, measures and modifiers necessary for the manipulation of fuzzy sets and relations. In further work [17], Matthews illustrates how the toolkit can be used to specify a simple fuzzy expert system. However, their approach does not capture component interactions, which is the primary concern in this paper.

Cerny et al. [5] in a recent attempt pointed out that boolean notions of correctness are formalized by preorders on systems. To overcome the limitations of a two-valued logic, the authors introduce the notion of distances between two systems or between a system and a specification, and suggest quantitative simulation games as a framework for measuring such distances. They presented three particular distances: two for quantifying aspects of correct systems, namely coverage and robustness; and one for measuring the degree of correctness of an incorrect system. In a later work [4], the same authors extend the quantitative notion of simulation distances to automata with inputs and outputs. The introduced interface distance, allows for measuring the desirability of an interface w.r.t. a given specification. In a direct comparison with the work presented in this paper one could say that both approaches pursue the same objective, namely to relax the boolean notion in formal specifications. However, the common objective is addressed by two distinct approaches. While Cerny et al. define for each property of interest a simulation distance and measure afterwards the deviation of all models, we rely on fuzzy set theory to soften the boolean notion. Hence, we suggest to formalize properties in terms of a-cuts and acceptance

degrees on vague descriptions and measure to what degree a property of interest is fulfilled by concrete models (e.g. a-Realizabilty of two behaviors).

The restrictions of a two-valued logic are present also in systems with continuous behavior. Henzinger et al. presented in their recent paper [12] a model measuring framework for the hybrid case, where distances are represented by parametrized hybrid automata. Actually, they address the same problem as described in [5] for the hybrid case. In our approach, we consciously decided for fuzzy set theory because of the fuzzification property which allows the generalization of a distinct theory to a continuous one. Thus, the introduced concepts in this paper can be easily generalized to continuous behaviors. An interesting future research objective would be to analyze the trade-off between fuzzy and hybrid approaches, in general. While hybrid automata make use of differential equations to describe a state, fuzzy approaches use vague rules. What is the distance between fuzzy descriptions and differential equations?

6 Conclusion

Tool Support. The intention of this work was not to present a concrete tool which is part of a tool demonstration but rather to establish the underlying theory required for the development of such a tool. Thus, we abstract away from the implementation details and present only an overview of a prototype under development depicted in Figure 6. Xtext [8], a framework for development of programming languages and DSLs is the starting point. It is used for the development of a model based specification language with support to the introduced concepts in chapters 3,4. The model based specification language generates the required parser and linker. Additionally, an eclipse plugin is generated which enables full support for the specification language inside the eclipse IDE. Hence, the integrated specification editor is used to transform the informal specification (requirements) into a formal specification which conforms to the developed language. Once, the informal requirements are formalized a series of model transformations becomes available. On the one hand, the specification can be transformed to executable models (Java and Simulink) which allows automated simulation for the system under development. On the other hand a generic theorem prover Isabelle [20] is used for the verification and validation of system properties. Currently, there is only support for the introduced concepts in chapter 3, see for example Focus on Isabelle [23]. In particular, support for formal verification of fuzzy component behavior is a major future research direction. Hence, a primary concern is to develop/adapt a fuzzy theory toolbox in Isabelle which enables fuzzy reasoning inside the framework.

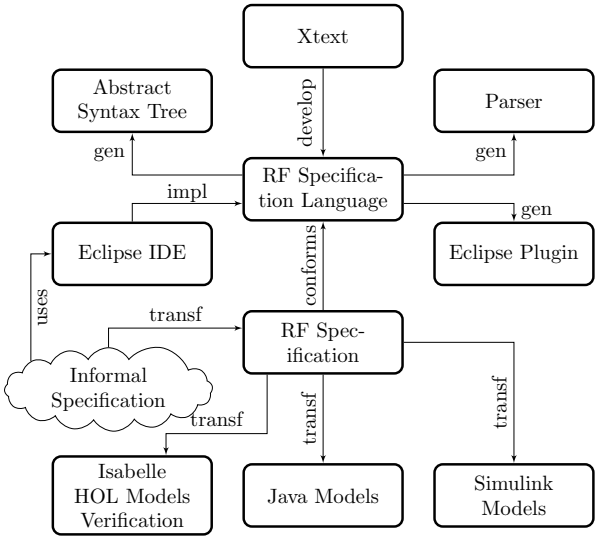


Figure 6: Tool Prototype

Summary. In chapter 3, we introduced a specification technique based on fuzzy logic for interactive systems. In particular, we showed that a fuzzy rule based specification can be represented in terms of a black box view as a deterministic behavior and can be therefore modeled in a deterministic fashion by

means of automata. The introduced technique is well suited for modeling especially user and environment interactions which are characterized by vagueness and uncertainty. The underlying Focus theory has been adapted to enable vague desriptions over fuzzy I/O ports. Finally, mapping strategy are introduced, which adapts fuzzy properties to the measured behavior over the I/O histories. Mapping strategies are well suited for formalizing self* properties.

In chapter 4, we introduced fuzzy components and fuzzy behavior of them. We established a basic notion for fuzzy types, channels and interfaces and provided basic operators on streams. A general method which enables the mapping of fuzzy input streams to fuzzy output streams over a set valued function is defined. The latter enables the modeling of fuzzy component behavior. Finally, we showed the fuzzy interpretation of basic specification properties like realizability.

Outlook. Concluding, we point out that our proposed method allows to capture certain system aspects which can not be represented by formal methods based on a two-valued logic. However, the work presented here is only an introduction towards a complete theory for fuzzy interactive systems. Basic system concepts as composition and decomposition, refinement, interface abstraction and architecture, to name only a few, have to be addressed in more detail. Last but not least from a more practical point of view specification techniques such as tables and diagrams and tool support in the form of AutoFocus [2] are future directions we have to go in order to set up more practical case studies to evaluate the expressiveness, completeness, and effectiveness of the introduced approach.

Acknowledgments.

The author address special thanks to Prof. Manfred Broy, Diego Marmsoler, Jonas Eckhardt, and Orestis Gkorgkas for their invaluable suggestions and the fruitful discussions on the topic.

References

[1] Peter B. Andrews (1986): *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof*. Academic Press Professional, Inc., San Diego, CA, USA.

[2] Manfred Broy, Franz Huber & Bernhard Schätz (1999): *AutoFocus - Ein Werkzeugprototyp zur Entwicklung eingebetteter Systeme*. *Informatik Forschung und Entwicklung* 14, pp. 121–134, doi:10.1007/s004500050132.

[3] Manfred Broy & Ketil Stølen (2001): *Specification and Development of Interactive Systems - Focus on Streams, Interfaces, and Refinement*. Monographs in Computer Science, Springer, doi:10.1007/978-1-4613-0091-5.

[4] Pavol Cerný, Martin Chmelik, Thomas A. Henzinger & Arjun Radhakrishna (2014): *Interface simulation distances*. *Theor. Comput. Sci.* 560, pp. 348–363, doi:10.1016/j.tcs.2014.08.019.

[5] Pavol Cerny, ThomasA. Henzinger & Arjun Radhakrishna (2010): *Simulation Distances*. In Paul Gastin & François Laroussinie, editors: *CONCUR 2010 - Concurrency Theory*, *Lecture Notes in Computer Science* 6269, Springer Berlin Heidelberg, pp. 253–268, doi:10.1007/978-3-642-15375-4_18.

[6] Marsha Chechik, Benet Devereux, Steve Easterbrook & Arie Gurfinkel (2003): *Multi-valued Symbolic Model-checking*. *ACM Trans. Softw. Eng. Methodol.* 12(4), pp. 371–408, doi:10.1145/990010.990011.

[7] Alan M. Davis (1988): *A Comparison of Techniques for the Specification of External System Behavior*. *Commun. ACM* 31(9), pp. 1098–1115, doi:10.1145/48529.48534.

[8] Moritz Eysholdt & Heiko Behrens (2010): *Xtext: implement your language faster than the quick and dirty way*. In: *Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion*, ACM, pp. 307–309, doi:10.1145/1869542.1869625.

[9] G. Hackenberg, M. Irlbeck, V. Koutsoumpas & D. Bytschkow (2012): *Applying formal software engineering techniques to smart grids*. In: *Software Engineering for the Smart Grid (SE4SG), 2012 International Workshop on*, pp. 50–56, doi:10.1109/SE4SG.2012.6225719.

[10] Georg Hackenberg, Maximilian Irlbeck, Vasileios Koutsoumpas & Denis Bytschkow (2014): *A Rapid Prototyping Approach for Smart Energy Systems Based on Partial System Models*. In: *Computer Software and Applications Conference Workshops (COMPSACW), 2014 IEEE 38th International*, pp. 596–601, doi:10.1109/COMPSACW.2014.100.

[11] Anthony Hall & Roderick Chapman (2002): *Correctness by Construction: Developing a Commercial Secure System*. *IEEE Software* 19(1), pp. 18–25, doi:10.1109/52.976937.

[12] Thomas A. Henzinger & Jan Otop (2014): *Model Measuring for Hybrid Systems*. In: *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control*, HSCC '14, ACM, New York, NY, USA, pp. 213–222, doi:10.1145/2562059.2562130.

[13] Jonathan Jacky (1996): *The Way of Z: Practical Programming with Formal Methods*. Cambridge University Press, New York, NY, USA, doi:10.1017/CBO9780511574924.

[14] Rudolf Kruse, Joan E. Gebhardt & F. Klowon (1994): *Foundations of Fuzzy Systems*, 1st edition. John Wiley & Sons, Inc., New York, NY, USA.

[15] Marta Kwiatkowska, Gethin Norman & David Parker (2004): *Probabilistic symbolic model checking with PRISM: a hybrid approach*. *International Journal on Software Tools for Technology Transfer* 6(2), pp. 128–142, doi:10.1007/s10009-004-0140-2.

[16] Luqi & Joseph A Goguen (1997): *Formal methods: promises and problems*. *Software, IEEE* 14(1), pp. 73–85, doi:10.1109/52.566430.

[17] C. Matthews (2002): *Fuzzy concepts and formal methods: a sample specification for a fuzzy expert system*. In: *Fuzzy Systems, 2002. FUZZ-IEEE'02. Proceedings of the 2002 IEEE International Conference on*, 2, pp. 1150–1155, doi:10.1109/FUZZ.2002.1006666.

[18] Chris Matthews & PaulA. Swatman (2000): *Fuzzy Concepts and Formal Methods: A Fuzzy Logic Toolkit for Z*. In: *ZB 2000: Formal Specification and Development in Z and B*, *Lecture Notes in Computer Science* 1878, Springer Berlin Heidelberg, pp. 491–510, doi:10.1007/3-540-44525-0_29.

[19] Philipp Neubeck (2012): *A Probabilitistic Theory of Interactive Systems*. Dissertation, Technische Universität München, München.

[20] Tobias Nipkow, Lawrence C. Paulson & Markus Wenzel (2002): *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*. *LNCS* 2283, Springer.

[21] Klaus Pohl, Harald Hönninger, Reinhold Achatz & Manfred Broy, editors (2012): *Model-Based Engineering of Embedded Systems, The SPES 2020 Methodology*. Springer, doi:10.1007/978-3-642-34614-9.

[22] Ian Sommerville (2006): *Software Engineering: (8th Edition) (International Computer Science)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[23] Maria Spichkova (2008): *Focus on Isabelle: From specification to verification*. *Department of Electrical and Computer Engineering, Concordia University, Tech. Rep.*

[24] L. A. Zadeh (1965): *Fuzzy sets*. *Information and control* 8(3), pp. 338–353, doi:10.1016/S0019-9958(65)90241-X.

[25] L. A. Zadeh (1999): *Fuzzy Sets As a Basis for a Theory of Possibility*. *Fuzzy Sets Syst.* 100, pp. 9–34, doi:10.1016/S0165-0114(99)80004-9.

[26] L.A Zadeh (1977): *Possibility theory vs. probability theory in decision analysis*. In: *Decision and Control including the 16th Symposium on Adaptive Processes and A Special Symposium on Fuzzy Set Theory and Applications, 1977 IEEE Conference on*, pp. 1267–1269, doi:10.1109/CDC.1977.271764.