Soft Computing (2021) 25:6789–6805
https://doi.org/10.1007/s00500-021-05593-9

FOCUS

A semantics and a logic for *Fuzzy Arden Syntax*

Leandro Gomes[1] · **Alexandre Madeira**[2] · **Luís Soares Barbosa**[3]

Published online: 13 February 2021

Abstract
Fuzzy programming languages, such as the *Fuzzy Arden Syntax* (FAS), are used to describe behaviours which evolve in a fuzzy way and thus cannot be characterized neither by a Boolean outcome nor by a probability distribution. This paper introduces a semantics for FAS, focusing on the *weighted parallel* interpretation of its conditional statement. The proposed construction is based on the notion of a *fuzzy multirelation* which associates with each state in a program a fuzzy set of weighted possible evolutions. The latter is parametric on a residuated lattice which models the underlying semantic 'truth space'. Finally, a family of dynamic logics, equally parametric on the residuated lattice, is introduced to reason about FAS programs.

1 Introduction

Facing the complex nature of modern societies, computational systems with a complex, unconventional behaviour have become the norm rather than the exception in software engineering design. Often some notion of *weight* (e.g. a cost, a probability, a continuous effect, etc.) has to be brought into the picture leading to a broader understanding of what a computation may stand for. This entails the need not only for suitable programming languages able to capture forms of probabilistic, fuzzy or hybrid computation, but also for formalisms for rigorous design, verification and refinement for such programs. While in the "classical world" Kleene algebras (Kozen 1994, 2000) and dynamic logics (Pratt 1991; Harel et al. 2000; Fischer and Ladner 1979) provide the underlying formal setting to establish program correctness, suitable generalizations to different forms of weighted computation are still object of intense research.

Uncertainty is a key ingredient (Kozen 1985; Qiao et al. 2008; den Hartog and de Vink 2002; McIver et al. 2008; Foster et al. 2016) in this setting. Probabilities often emerge as a natural way to describe behaviour in a variety of application scenarios. Typically, probabilistic programs are syntactically equipped with an operator to describe a probabilistic choice between two executions (Qiao et al. 2008). In the following expression, for example,

$$(x := x + 1) +_{0.6} (y := y \times 2),$$

operator $+_{0.6}$ represents the nondeterministic execution of the assignments $x := x + 1$ and $y := y \times 2$ with probabilities 0.6 and 0.4, respectively. Programs of this kind resemble a "yes–no" question, modelled by a Bernoulli distribution of a random variable, whose answer takes the value 1 with probability p and 0 with probability $1 - p$. A typical illustrative, real-world scenario is a sequence of (possible biased) coin tosses, as represented by a binary tree in Fig. 1. Note that, despite the uncertainty associated with each event, the result is always Boolean, and therefore, at each stage, only one of the branches provides an outcome.

Another class of systems in which uncertainty plays a major role is fuzzy control systems (Zadeh 1965), which typically express situations that cannot be conceptualized in terms of a Boolean choice. Fuzzy programming languages (FPL) are precisely suited to describe such systems, which are present in a variety of applications, ranging from medical diagnosis (Vetterlein et al. 2010) to robotics (Cingolani and Alcalá-fdez 2013).

The Fuzzy Arden Syntax (FAS) (Vetterlein et al. 2010) is an example of a FPL designed for medical diagnosis. Actu-

Communicated by Tomas Veloz.

✉ Leandro Gomes
leandro.r.gomes@inesctec.pt

Alexandre Madeira
madeira@ua.pt

Luís Soares Barbosa
lsb@di.uminho.pt

[1] HASLab INESC TEC, Univ. Minho, Braga, Portugal

[2] CIDMA, Univ. Aveiro, Aveiro, Portugal

[3] HASLab INESC TEC, INL and Univ. Minho, Braga, Portugal

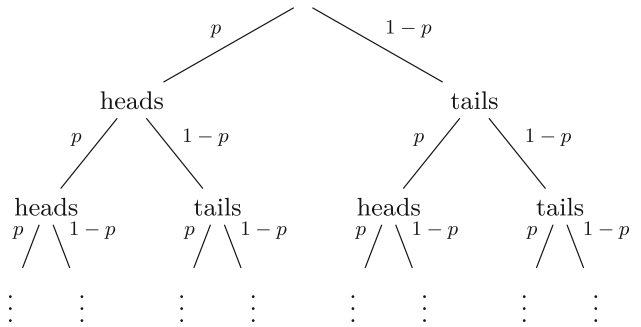

Fig. 1 Binary tree representing a sequence of (possible biased) coin tosses

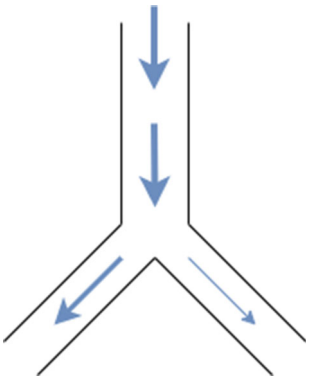

Fig. 2 Conditionals in a fuzzy programming language have a parallel flavour as in a course of water flowing through a 'Y-shaped' pipe

ally, it extends the Arden Syntax (AS) to cater for vague or uncertain information, typically arising in clinical situations. A typical example is the following: if the body temperature of a patient reaches $37.4°C$, and some automatic control system defines a lower limit of $37.5°C$ for identifying a fever condition, it may be worthwhile to consider that the patient is actually reaching a temperature worth of attention. A sharp, Boolean reasoning will not be adequate to capture this sort of cases. Actually, often in real life, it is helpful to infer information from facts that are not exactly true or false, but so up to a certain degree.

FAS is based on fuzzy set theory (Zadeh 1965) and fuzzy logic (Caicedo and Rodriguez 2010; Bou et al. 2011; Hansoul and Teheux 2013): its data types have been generalized to represent truth values between the extremes false and true, and the operations on these types were adapted accordingly. Intuitive and very close to natural language, it has been successfully used to design knowledge-based components in medical decision support systems (Starren et al. 1994; Anand et al. 2018; Samwald et al. 2012).

Programs in FAS are written as units that may interact with each other to process clinical information. Composition is basically achieved through the conditional `if-then-else` and `switch` statements. Being fuzzy, they entail a sort of *parallel* execution of possible alternative paths. Differently from what happens in probabilistic programming, there is no (probabilistic) choice of alternative courses of action. Actually, execution proceeds in parallel with distinct associated weights — just as a liquid flowing through different channels at the same time, after a split point, to use a popular metaphor depicted in Fig. 2. The distinct thicknesses of the arrows represent, as expected, the distinct 'flows' of liquid (weights).

This paper introduces a proper semantics for conditional statements in FAS as a form of *weighted parallelism*. The underlying mathematical structure is a *fuzzy multirelation*

$$R \subseteq W \times \mathbf{L}^W \tag{1}$$

which relates each state (typed as W) to a function from W to a 0-*bounded right residuated lattice* $\mathbf{L}$, i.e. a reduct of a 0-bounded residuated lattice without the left residual $\leftarrow$[1] and with the additional axiom of the absorbent property of 0 on ; (axiom (11)). Such a structure is used to capture the fuzzy evolution of the system.

The concept is based on *binary multirelations*, which were first used in the context of game logics to distinguish between *angelic* and *demonic* non-determinism (Parikh 1983, 1985; Rewitzky 2003), i.e. between an internally controlled choice and a totally external, thus uncontrollable one. Multirelations are also used to base predicate transformer semantics (Rewitzky and Brink 2006).

In a fuzzy multirelation, on the other hand, each state is related to a fuzzy set of reachable states leading to multiple branches that can be explored in parallel, each of them associated with a (possibly different) weight. Such weights are values taken from a truth space modelled by $\mathbf{L}$.

The idea is close to *probabilistic multirelations* (Tsumagari 2012), but considering a more generic algebra to evaluate truth degrees, and agnostic with respect to some properties, such as convexity. Note that, making $\mathbf{L} = \mathbf{2}$, the Boolean lattice, inclusion (1) boils down to

$$R \subseteq W \times \mathbf{2}^W \tag{2}$$

i.e. the standard notion of a multirelation associating states to sets of states.

To better understand the intuitive meaning of a fuzzy multirelation, consider the following simple example. Let $W = \{w_0, w_1, w_2\}$ and choose the carrier of $\mathbf{L}$ to be the real interval $[0, 1]$. Figure 3 depicts a fuzzy multirelation $R = \{w_0, \varphi\}$ such that $\varphi : W \to \mathbf{L}$ is defined as $\varphi(w_0) = 0$, $\varphi(w_1) = 0.4$ and $\varphi(w_2) = 0.6$.

[1] Note that while the most common nomenclature for $\leftarrow$ ($\rightarrow$) is *right* (*left*) *division*, we follow reference Kozen (1993), where $\leftarrow$ ($\rightarrow$) is called *left* (*right*) *residual*.

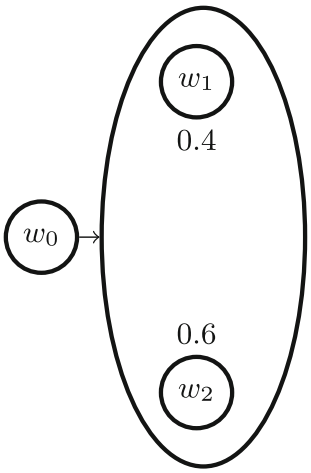

Fig. 3 Picturing fuzzy multirelation R

Fuzzy multirelation R may be thought as representing the following `if-then-else` statement in FAS:

Example 1 `if (Temperature is inFever_condition) then medicine:=5 else medicine:=0`

where w_0 is the initial state, and w_1, w_2 correspond to the states achieved up execution of, respectively, assignments `medicine:=5` and `medicine:=0`. Values $\varphi(w_1) = 0.4$, $\varphi(w_2) = 0.6$, emerging from the evaluation of condition `Temperature is in Fever_condition`, capture the fact that `Temperature` has probably not reached the limit of a fever condition but is close to that limit. Function φ is interpreted as the fuzzy set of states reachable in parallel from w_0. If, instead, the condition above was evaluated to a Boolean value, the conditional statement would behave as an **if–then–else** in a standard imperative programming languages. In the present scenario, only assignment `medicine:=5` would be executed.

Note that this fuzzy behaviour is completely different from the one of a conditional both in a classical and probabilistic programming language. Actually, the sort of conditional we are interested here in does not reduce to a non-deterministic or even a probabilistic choice (McIver et al. 2013). On the contrary, a fuzzy conditional corresponds to a form of parallel composition in which branches are parallel courses of action executed with different weights. This is also different, of course, from the usual parallel composition which models a crisp notion of two programs running in parallel (Peleg 1987; Hoare et al. 2011; Furusawa and Struth 2015), or two actions being processed at the same time (Prisacariu 2010).

The choice of a concrete $\mathbf{L}$ depends, of course, on the problem to be addressed. Therefore, the paper develops a generic semantics which is parametric on the definition of $\mathbf{L}$. This semantics is then extended to a family of *-free dynamic logics, also parametric in $\mathbf{L}$, to reason about FAS programs.

Structure. This paper is organized as follows. Section 2 introduces *fuzzy multirelations* and their algebra, parametric on a 0-bounded right residuated lattice $\mathbf{L}$, as the main mathematical component of a semantics and a logic for FAS. This is used in the following section to provide a formal (denotational) semantics for the language. Section 4 reports on the paper's second contribution: the development of a family of *-free dynamic logics, $\mathcal{L}(\mathbf{L})$, parametrized by $\mathbf{L}$, for reasoning about FAS programs. Its syntax, semantics, satisfaction relation and axiomatization are discussed in detail. Finally, Sect. 6 sums up related research, concludes and enumerates some topics for future work.

2 The algebra of fuzzy multirelations

This section introduces *fuzzy multirelations* parametric, as referred above, on a 0-bounded right residuated lattice, and their algebra.

2.1 Preliminaries

In order to capture different interpretations of the fuzzy connectives underlying the FAS programming constructors, the whole formal treatment of FAS in this paper is parametrized by an 0-*bounded right residuated lattice* (Galatos et al. 2007). This structure plays a double role in the sequel: as a computational model, to interpret programs, and as a truth space, to give meaning to variables in the corresponding *-free dynamic logic.

Definition 1 (0-**bounded right residuated lattice** (Galatos et al. 2007)) A 0-*bounded right residuated lattice* is a tuple $\mathbf{L} = (L, +, ;, 1, 0, \rightarrow, \cdot)$, where L is a set, $1, 0$ are constants, and $+, ;, \rightarrow$ and $\cdot$ are binary operations over $\mathbf{L}$ satisfying the axioms in Fig. 4. The order relation $\leq$ is induced by $+$ as $a \leq b \Leftrightarrow a + b = b$. This structure is a reduct of a 0-bounded residuated lattice (Galatos et al. 2007) without the left residual $\leftarrow$, and with the additional constant 0 and (11) as an additional axiom. Note that by including $\leftarrow$ and the correspondent residuation axiom, (11) could be derived and thus omitted from the axiomatization. In other terms, $\mathbf{L}$ is a reduct of a *FL-algebra* (Galatos et al. 2007) without $\leftarrow$. Since this paper will only consider such class of residuated lattices, we refer to this structure simply as *right residuated lattice*. $\mathbf{L}$ is called a $\mathbb{I}$-right residuated lattice when the identity of the ; operator coincides with the greatest element, i.e. $1 = \top$. As this will always be the case in this paper, prefix $\mathbb{I}$ will be omitted in the sequel. A right residuated lattice is *complete* if every sublattice has both supremum and infimum. Notation $\mathbf{L}$ will be used to refer to both the lattice and its carrier, interchangeably, whenever clear from the context.

$$a + (b + c) = (a + b) + c \quad (3)$$
$$a + b = b + a \quad (4)$$
$$a + a = a \quad (5)$$
$$a + 0 = 0 + a = a \quad (6)$$
$$a; (b; c) = (a; b); c \quad (7)$$
$$a; 1 = 1; a = a \quad (8)$$
$$a; (b + c) = (a; b) + (a; c) \quad (9)$$
$$(a + b); c = (a; c) + (b; c) \quad (10)$$
$$a; 0 = 0; a = 0 \quad (11)$$
$$a; x \leq b \Leftrightarrow x \leq a \rightarrow b \quad (12)$$
$$a \cdot (b \cdot c) = (a \cdot b) \cdot c \quad (13)$$
$$a \cdot b = b \cdot a \quad (14)$$
$$a \cdot a = a \quad (15)$$
$$a + (a \cdot b) = a \quad (16)$$
$$a \cdot (a + b) = a \quad (17)$$

Fig. 4 Axioms for 0-bounded right residuated lattice (based on Kozen (1993))

The intuition behind these operators come from the double role $\mathbf{L}$ plays, as mentioned above. Operators $+$, $\cdot$ and $\rightarrow$ are taken as logical disjunction (`or`), conjunction (`and`) and implication, respectively. Having a monoidal structure, both $+$ and ; admit a distributed version denoted by $\sum$ and $\prod$, respectively. The generation of dynamic logics illustrated in Sect. 4 will be parametric on the class of complete right residuated lattices, since completeness is required to ensure the existence of arbitrary suprema. The following are typical examples of such structures.

Example 2 (**2** - *The Boolean lattice.*) The Boolean lattice with the standard Boolean connectives,

$$\mathbf{2} = (\{\top, \bot\}, \vee, \wedge, \bot, \top, \rightarrow, \wedge) .$$

Example 3 (***Ł*** - *The Łukasiewicz arithmetic lattice)* The *Lukasiewicz arithmetic lattice*

$$Ł = ([0, 1], \max, \odot, 0, 1, \rightarrow, \min)$$

where

$$x \odot y = \max(0, y + x - 1),$$
$$x \rightarrow y = \min(1, 1 - x + y).$$

Example 4 (**G** - *Gödel algebra)*] A Gödel algebra is defined as

$$\mathbf{G} = ([0, 1], \max, \min, 0, 1, \rightarrow, \min)$$

where

$$x \rightarrow y = \begin{cases} 1, & \text{if } x \leq y \\ y, & \text{if } y < x \end{cases}$$

Example 5 (**Π** - *the* **Π** *product algebra)* The product **Π**-algebra

$$\mathbf{\Pi} = ([0, 1], \max, \cdot, 0, 1, \rightarrow, \min)$$

where $\cdot$ is the usual multiplication of real numbers and

$$x \rightarrow y = \begin{cases} 1, & \text{if } x \leq y \\ y/x, & \text{if } y < x \end{cases}$$

for/the real division.

These lattices provide precisely the operators required for combining truth values underlying FAS programming constructs. The very notion of a fuzzy set is parametric on $\mathbf{L}$.

Definition 2 (**Fuzzy set and fuzzy relation** (Zadeh 1965)) Let W, W_1, W_2 be sets and $\mathbf{L}$ a complete right residuated lattice. A *fuzzy subset* of W is a function $\varphi : W \rightarrow \mathbf{L}$, with $\varphi(w)$ defining the *membership degree of* x *in* φ. The set of all fuzzy subsets of W is denoted by $\mathbf{L}^W$. A *fuzzy binary relation* over W_1, W_2 is a function $\mu : W_1 \times W_2 \rightarrow \mathbf{L}$ returning the weight assigned to the pair (w_1, w_2).

Fuzzy sets model collections of objects each of them associated with a degree of membership; fuzzy relations do the same for pairs of values. The qualifier extends, as expected, to multirelations. Recall that a binary multirelation over a set W (Rewitzky 2003) is a subset of the Cartesian product $W \times \mathbf{2}^W$. Given multirelations R, S, their *sequential composition* is given by

$$R \circ S = \{(w, U) \mid \exists_V . (w, V) \in R \ \wedge \ \exists_{F:V \rightarrow U} . (\forall_{v \in V} . (v, F(v)) \in S) \wedge U = \bigcup F(V)\}$$

and the *parallel composition* (Peleg 1987) is defined as

$$R||S = \{(a, B \cup C) \mid (a, B) \in R \wedge (a, C) \in S\}$$

2.2 Fuzzy binary multirelations

The semantics of FAS programs proposed in this paper is based on fuzzy multirelations, which we introduce in this section.

Definition 3 (Fuzzy binary multirelation) Let W be a set and $\mathbf{L}$ a complete right residuated lattice. A *fuzzy binary multirelation* over W is a set of pairs $R \subseteq W \times \mathbf{L}^W$. The space of $\mathbf{L}$-valued fuzzy multi-relations over a set W, $2^{W \times \mathbf{L}^W}$, will be denoted by $M^{\mathbf{L}}(W)$, or simply $M(W)$, whenever $\mathbf{L}$ is clear from the context.

Given two fuzzy (binary) multirelations R and S, their *sequential composition* is given by

$$R \circ S = \{\ (a,\varphi) \mid \exists_{\varphi'}.(a,\varphi') \in R \wedge \exists_{F:W\to\mathbf{L}^W}.\forall_{b\in W}.\ (b,F(b)) \in S \wedge \varphi(u) = \sum_b (\varphi'(b);\ F(b)(u))\ \} \tag{18}$$

and the *parallel composition* is given by

$$R \uplus S\ =\ \{(a,\varphi\cup\varphi') \mid (a,\varphi)\in R \wedge (a,\varphi')\in S\} \tag{19}$$

where $(\varphi\cup\varphi')(w) = \varphi(w) + \varphi'(w)$, for each $w \in W$, with symbol $+$ on the right hand side is sum on lattice $\mathbf{L}$ (as in Definition 1). The new multirelation collects pairs from R and S whose first component is the same in both relations. Note this definition generalizes the one given by Peleg (1987) for sharp multirelations.

The set $M^{\mathbf{L}}(W)$ of fuzzy multirelations over $\mathbf{L}$, together with operator $\uplus$, provides a suitable semantic structure for computations in FAS.

Definition 4 (Algebra of fuzzy multirelations)
Given a set W and a complete right residuated lattice $\mathbf{L}$, the algebra of fuzzy multirelations over $\mathbf{L}$ is the structure

$$\mathbb{M} = (M^{\mathbf{L}}(W), \cup, \circ, \uplus, \emptyset, 1_\circ, 1_\uplus)$$

where:

- $\cup$ is the binary set union;
- $\circ$ and $\uplus$ correspond to sequential (18) and parallel (19) composition of multirelations, respectively;
- $1_\circ = \{(w,\delta_w) \mid w\in W\}$ is the identity of $\circ$, with $\delta_w : W \to \mathbf{L}$ defined as

$$\delta_w(w') \ =\ \begin{cases} \top & \text{if } w' = w \\ \bot & \text{otherwise} \end{cases}$$

- $1_\uplus = \{(w,\underline{0}) \mid w\in W\}$ is the identity of $\uplus$, where $\underline{0} : W \to \mathbf{L}$ is the fuzzy set defined as $\underline{0}(w) = \bot$, $\forall w \in W$;

Notation $\underline{r}$ stands for the constant function on $r \in \mathbb{R}$, i.e. the function that returns $r \in \mathbb{R}$ for every input.

In the context of program semantics, multirelation $1_\circ$ can be regarded as the semantics of program statement **skip**, a common constructor in imperative programming languages, although not considered in FAS.

Theorem 1 *For any complete right residuated lattice* $\mathbf{L}$, $\mathbb{M}$ *is a* proto-trioid, *i.e. an algebra satisfying the following axioms:*

$$R\cup(S\cup T) = (R\cup S)\cup T \tag{20}$$
$$R\cup S = S\cup R \tag{21}$$
$$R\cup R = R \tag{22}$$
$$R\cup\emptyset = R \tag{23}$$
$$(R\circ S)\circ T \subseteq R\circ(S\circ T) \tag{24}$$
$$R\circ 1_\circ = 1_\circ\circ R = R \tag{25}$$
$$(R\circ S)\cup(R\circ T) \subseteq R\circ(S\cup T) \tag{26}$$
$$(R\cup S)\circ T = (R\circ T)\cup(S\circ T) \tag{27}$$
$$R\circ\emptyset = \emptyset\circ R = \emptyset \tag{28}$$
$$R\uplus(S\uplus T) = (R\uplus S)\uplus T \tag{29}$$
$$R\uplus S = S\uplus R \tag{30}$$
$$R\uplus R = R \tag{31}$$
$$1_\uplus\uplus R = R \tag{32}$$
$$R\uplus(S\cup T) = (R\uplus S)\cup(R\uplus T) \tag{33}$$

Proof $(M^{\mathbf{L}}(W),\cup,\emptyset)$ is a semilattice with least element $\emptyset$, thus satisfying axioms (20)–(23). The proofs of axioms (24)–(28) are not straightforward and require particular attention.
(24):
Let $(a,\varphi)\in(R\circ S)\circ T$. Then,

$$\exists_{\varphi'}.\ (a,\varphi')\in(R\circ S)\ \wedge\ \exists_{F:W\to\mathbf{L}^W}.\ \forall_{b\in W}.\ (b,F(b))\in T\ \wedge\ \varphi(u) = \sum_b \varphi'(b);\ F(b)(u)$$

and, similarly,

$$\exists_{\varphi''}.\ (a,\varphi'')\in R\ \wedge\ \exists_{F':W\to\mathbf{L}^W}.\ \forall_{a'\in W}.(a',F'(a'))\in S\ \wedge\ \varphi'(b) = \sum_{a'}\varphi''(a');\ F'(a')(b)$$

Thus,

$$\begin{aligned}
\varphi(u) &= \sum_b\left(\sum_{a'}(\varphi''(a');\ F'(a')(b));\ F(b)(u)\right)\\
&= \sum_b\left(\sum_{a'}(\varphi''(a');\ F'(a')(b);\ F(b)(u))\right)\\
&= \sum_{a'}\left(\sum_b(\varphi''(a');\ F'(a')(b);\ F(b)(u))\right)\\
&= \sum_{a'}\left(\varphi''(a');\ \left(\sum_b(F'(a')(b);\ F(b)(u))\right)\right)\\
&= \sum_{a'}(\varphi''(a');\ F''(a')(u))
\end{aligned}$$

where $F'' : W \to \mathbf{L}^W$ is defined by

$$F''(a')(u) = \sum_{a'}(\varphi''(a');\ F''(a')(u))$$

Clearly $(a', F''(a')) \in S \circ T$ entails $(a, \varphi) \in R \circ (S \circ T)$.
(25):
Case 1: $\underline{1_\circ \circ R \subseteq R}$.
Suppose $(a, \varphi) \in 1_\circ \circ R$. Then,

$$(\exists_{\varphi'}.\ (a, \varphi') \in 1_\circ)\ \wedge\ \exists_{F:W\to\mathbf{L}^W}.$$
$$\forall_{b\in W}.\ (b, F(b)) \in R\ \wedge\ \varphi(u) = \sum_b \varphi'(b);\ F(b)(u)$$
$$\Leftrightarrow$$
$$(\exists_{\varphi'}.\ (a, \varphi') \in 1_\circ)\ \wedge\ \exists_{F:W\to\mathbf{L}^W}.$$
$$\forall_{b\in W}.\ (b, F(b)) \in R\ \wedge\ \varphi(u) = F(a)(u)$$

because

$$\varphi(u) = \sum_b \varphi'(b);\ F(b)(u)\ =\ \varphi'(a);\ F(a)(u)$$
$$=\ \top;\ F(a)(u) = F(a)(u)$$

with $(a, F(a)) \in R$. Thus, $(a, \varphi) \in R$.
Case 2: $\underline{R \subseteq 1_\circ \circ R}$:
Conversely, let $(a, \varphi) \in R$, and define $F : W \to \mathbf{L}^W$ as $F(b) = \begin{cases} \varphi & \text{if } b = a \\ \underline{0} & \text{otherwise} \end{cases}$
such that $(b, F(b)) \in R$. Thus, φ is a function defined as follows

$$\varphi(u) = \sum_b (\varphi'(b);\ F(b)(u))$$
$$= \sum_b (\delta_a(b);\ F(b)(u)),$$
$$\text{with } \Big(a, \sum_b (\delta_a(b);\ F(b)\Big) \in (1_\circ \circ R)$$

Thus, $(a, \varphi) \in 1_\circ \circ R$. Let us now prove $R \circ 1_\circ = R$.
Case 1: $\underline{R \circ 1_\circ \subseteq R}$:
Suppose $(a, \varphi) \in R \circ 1_\circ$. Then,

$$\exists_{\varphi'}.\ (a, \varphi') \in R \wedge \exists_{F:W\to\mathbf{L}^W}.$$
$$\forall_{b\in W}.\ (b, F(b)) \in 1_\circ\ \wedge\ \varphi(u) = \sum_b (\varphi'(b);\ \delta_b(u))$$
$$\Leftrightarrow$$
$$\exists_{\varphi'}.(a, \varphi') \in R \wedge \exists_{F:W\to\mathbf{L}^W}.$$
$$\forall_{b\in W}.\ (b, F(b)) \in 1_\circ\ \wedge\ \varphi(u) = \varphi'(b)$$

because

$$\varphi(u) = \sum_b (\varphi'(b);\ \delta_b(u))$$
$$= \varphi'(b);\ \delta_b(b)$$
$$= \varphi'(b);\ \top$$
$$= \varphi'(b)$$

with $(a, \varphi') \in R$. Thus, $(a, \varphi) \in R$.
Case 2: $\underline{R \subseteq R \circ 1_\circ}$:
Let $(a, \varphi) \in R$, and define $F : W \to \mathbf{L}^W$ as $F(b)(u) = \delta_b(u)$. Following a similar argument the one above,

$$\varphi(u)\ =\ \sum_b (\varphi(b);\ F(b)(u))\ =\ \sum_b \varphi(b);\ \delta_b(u)$$

with $\big(a, \sum_b \varphi(b);\ \delta_b\big) \in (R \circ 1_\circ)$.
(26):
Assume $(a, \varphi) \in (R \circ S) \cup (R \circ T)$. Then, $(a, \varphi) \in R \circ S \vee (a, \varphi) \in R \circ T$. That means

$$\exists_{\varphi'}.(a, \varphi') \in R\ \wedge$$
$$\Big(\exists_{F:W\to\mathbf{L}^W}.\forall_{b\in W}.\big((b, F(b)) \in S \vee (b, F(b)) \in T\big)\Big)\wedge$$
$$\varphi(u) = \sum_b (\varphi'(b);\ F(b)(u))$$
$$\Leftrightarrow$$
$$\exists_{\varphi'}.(a, \varphi') \in R \wedge \big(\exists_{F:W\to\mathbf{L}^W}.\forall_{b\in W}.(b, F(b)) \in S \cup T\big)\wedge$$
$$\varphi(u) = \sum_b (\varphi'(b);\ F(b)(u))$$
$$\Leftrightarrow$$
$$(a, \varphi) \in R \circ (S \cup T)$$

(27):
We start by proving $(R \cup S) \circ T \subseteq (R \circ T) \cup (S \circ T)$. Let us assume $(a, \varphi) \in (R \cup S) \circ T$. Then

$$\exists_{\varphi'}.(a, \varphi') \in (R \cup S) \wedge \exists_{F:W\to\mathbf{L}^W}.$$
$$\forall_{b\in W}.\ (b, F(b)) \in T \wedge \varphi(u) = \sum_b (\varphi'(b);\ F(b)(u))$$
$$\Leftrightarrow$$
$$\exists_{\varphi'}.\big((a, \varphi') \in R \vee (a, \varphi') \in S\big) \wedge \exists_{F:W\to\mathbf{L}^W}.$$
$$\forall_{b\in W}.\ (b, F(b)) \in T \wedge \varphi(u) = \sum_b (\varphi'(b);\ F(b)(u))$$
$$\Leftrightarrow$$
$$\big(\exists_{\varphi'}.(a, \varphi') \in R \wedge \exists_{F:W\to\mathbf{L}^W}.$$
$$\forall_{b\in W}.\ (b, F(b)) \in T \wedge \varphi(u) = \sum_b (\varphi'(b);\ F(b)(u))\big) \vee$$
$$\big(\exists_{\varphi'}.(a, \varphi') \in S \wedge \exists_{F:W\to\mathbf{L}^W}.$$
$$\forall_{b\in W}.\ (b, F(b)) \in T \wedge \varphi(u) = \sum_b (\varphi'(b);\ F(b)(u))\big)$$
$$\Leftrightarrow$$
$$\big((a, \varphi) \in R \circ T\big) \vee \big((a, \varphi) \in S \circ T\big)$$
$$\Leftrightarrow$$

$$(a,\varphi) \in (R \circ T) \cup (S \circ T).$$

Conversely, assume $(a,\varphi) \in (R \circ T) \cup (S \circ T)$. Then

$$\begin{aligned}
&(a,\varphi) \in R \circ T \vee (a,\varphi) \in Q \circ T \\
&\quad \Leftrightarrow \\
&\big(\exists_{\varphi'}.\ (a,\varphi') \in R \wedge \exists_{F:W\to\mathbf{L}^W}. \\
&\qquad \forall_{b\in W}.\ (b,F(b)) \in T \wedge \varphi(u) = \sum_b (\varphi'(b);\ F(b)(u))\big) \vee \\
&\big(\exists_{\varphi'}\ sf(a,\varphi') \in S \wedge \exists_{F:W\to\mathbf{L}^W}. \\
&\qquad \forall_{b\in W}.\ (b,F(b)) \in T \wedge \varphi(u) = \sum_b (\varphi'(b);\ F(b)(u))\big) \\
&\quad \Leftrightarrow \\
&\exists_{\varphi'}.\ ((a,\varphi') \in R \vee (a,\varphi') \in S) \wedge \exists_{F:W\to\mathbf{L}^W}. \\
&\qquad \forall_{b\in W}.\ (b,F(b)) \in R \wedge \sum_b (\varphi'(b);\ F(b)(u)) \\
&\quad \Leftrightarrow \\
&\exists_{\varphi'}.\ (a,\varphi') \in R \cup S \wedge \exists_{F:W\to\mathbf{L}^W}. \\
&\qquad \forall_{b\in W}.\ (b,F(b)) \in T \wedge \sum_b (\varphi'(b);\ F(b)(u)) \\
&\quad \Leftrightarrow \\
&\qquad (a,\varphi) \in (R \cup S) \circ T
\end{aligned}$$

(28):
Let us prove $\emptyset \circ R \subseteq \emptyset$. Suppose $(a,\varphi) \in \emptyset \circ R$. Then,

$$\begin{aligned}
&\exists_{\varphi'}.\ (a,\varphi') \in \emptyset \wedge \exists_{F:W\to\mathbf{L}^W}. \\
&\qquad (b,F(b)) \in R \wedge \varphi(u) = \sum_b (\varphi'(b);\ F(b)(u))
\end{aligned}$$

But $\nexists_{\varphi'}.(a,\varphi') \in \emptyset$ since $\emptyset$ is the empty set. So there is no $(a,\varphi) \in \emptyset \circ R$, and thus, $(a,\varphi) \in \emptyset$.

To prove that $R \circ \emptyset \subseteq \emptyset$, assume $(a,\varphi) \in R \circ \emptyset$. Then,

$$\begin{aligned}
&\exists_{\varphi'}.(a,\varphi') \in R \wedge \exists_{F:W\to\mathbf{L}^W}. \\
&\qquad \forall_{b\in W}.\ (b,F(b)) \in \emptyset \wedge \varphi(u) = \sum_b (\varphi'(b);\ F(b))
\end{aligned}$$

But there is no $(b,F(b)) \in \emptyset$ since $\emptyset$ is the empty set. Hence $(a,\varphi) \in \emptyset$. The two converse inclusions are trivial.
(29)–(31):

These axioms come directly from Definition 3 and the properties of a residuated lattice.
(32):

Consider an arbitrary fuzzy multirelation R over a set of states W. Then,

$$\begin{aligned}
&R \uplus 1_{\uplus} \\
=\ &\quad \{\text{ definition of } \uplus \text{ (19) and } 1_{\uplus}\} \\
&\{(a,\varphi_R \cup \{(a,\emptyset) \mid a \in W\}) \mid (a,\varphi_R) \in R\} \\
=\ &\quad \{\text{ for any } a \in W,\ \varphi_R(a) + 0 \text{ by (6)}\} \\
&\{(a,\varphi_R) \mid (a,\varphi_R) \in R\} \\
=\ &\quad \{\text{ identity}\} \\
&R
\end{aligned}$$

(33):
First we prove $R \uplus (S \cup T) \subseteq R \uplus S \cup R \uplus T$. Assume $(a,\varphi) \in R \uplus (S \cup T)$. Then,

$$\begin{aligned}
&\exists_{\varphi_1,\varphi_2}.(a,\varphi_1) \in R \wedge (a,\varphi_2) \in S \cup T\ \wedge\ \varphi = \varphi_1 + \varphi_2 \\
&\quad \Rightarrow \\
&\exists_{\varphi_1,\varphi_2}.\Big(\big((a,\varphi_1) \in R \wedge (a,\varphi_2) \in S\big) \vee \\
&\qquad \big((a,\varphi_1) \in R \wedge (a,\varphi_2) \in T\big)\Big)\ \wedge\ \varphi = \varphi_1 + \varphi_2 \\
&\quad \Rightarrow \\
&\qquad \big((a,\varphi) \in R \uplus S\big) \vee \big((a,\varphi) \in R \uplus T\big) \\
&\quad \Rightarrow \\
&\qquad (a,\varphi) \in (R \uplus S) \cup (R \uplus T)
\end{aligned}$$

Conversely, suppose that $(a,\varphi) \in (R \uplus S) \cup (R \uplus T)$. Then,

$$\begin{aligned}
&\qquad (a,\varphi) \in R \uplus S \vee (a,\varphi) \in R \uplus T \\
&\quad \Rightarrow \\
&\exists_{\varphi_1,\varphi_2}.\Big(\big((a,\varphi_1) \in R\ \wedge\ (a,\varphi_2) \in S\big) \\
&\qquad \vee \big((a,\varphi_1) \in R \wedge (a,\varphi_2) \in T\big)\Big)\ \wedge\ \varphi = \varphi_1 + \varphi_2 \\
&\quad \Rightarrow \\
&\exists_{\varphi_1,\varphi_2}.\Big((a,\varphi_1) \in R \\
&\qquad \wedge\ \big((a,\varphi_2) \in S\ \vee\ (a,\varphi_2) \in T\big)\Big)\ \wedge\ \varphi = \varphi_1 + \varphi_2 \\
&\quad \Rightarrow \\
&\exists_{\varphi_1,\varphi_2}.\Big((a,\varphi_1) \in R\ \wedge\ (a,\varphi_2) \in S \cup T\Big)\ \wedge\ \varphi = \varphi_1 + \varphi_2 \\
&\quad \Rightarrow \\
&\qquad (a,\varphi) \in R \uplus (S \cup T)
\end{aligned}$$

□

3 The *Fuzzy Arden Syntax*

The *Fuzzy Arden Syntax* is a fuzzy programming language optimized for the design of fuzzy control systems for clinical

decision-making. After a brief overview of its syntax, this section introduces a denotational semantics for FAS based on fuzzy multirelations.

3.1 Syntax

FAS is syntactically supported by the following elements:

- A set X of *variables* (e.g. `Temp`, `O2_low`);
- A set F of *function symbols*. Each $(F_n)_{n\in\mathbb{N}_0} \subseteq F$ denotes a family of function symbols with arity $n \in \mathbb{N}_0$. Symbols $c \in F_0$ are called constants. Semantically, a function symbol f in F is interpreted as $f(t_1, \ldots, t_n) : \mathbb{R}^n \to \mathbb{R}$.
- A set of P of *predicate symbols*. Each $(P_n)_{n\in\mathbb{N}_0} \subseteq P$ denotes a family of predicate symbols with arity $n \in \mathbb{N}_0$. Semantically, a predicate symbol p in P is interpreted as $p(t_1, \ldots, t_n) : \mathbb{R}^n \to \mathbf{2}$.

Notation $T(X)$ stands for the set of terms with variables in X. In particular: $T_F(X)$ represents its restriction to functional terms (e.g. `Temp+5`, `O2*2`,

`CO2_very_high-3` or `O2_low+5`). Similarly, $T_P(X)$ is the restriction to predicate terms, as in, for example, `O2_low` $<$ `90`).

A program π in FAS is either an *assignment*

$x := t$

a *conditional* statement

`if` p `then` π_1 `else` π_2 `endif`

or a *generalized conditional*

```
switch
        p1 then π1
        ...
        pn then πn
endswitch
```

where p_i, $1 \le i \le n$ are predicate terms in $T_P(V)$, $x \in X$ is a variable, $t \in T_F(X)$ a term over X. We denote the set of all assignments by At and the set of all programs by Prg(At).

To simplify reasoning about FAS programs at a later stage, it is useful to decompose FAS conditional and switch statements into more elementary operators,

namely tests, as well as parallel and sequential composition of programs. Such operators, although not explicitly part of the syntax, can easily be added to the language as generated by the following rule:

$\pi ::= \pi_0 \mid p? \mid \pi ; \pi \mid \pi || \pi$

where $\pi_0 \in$ At and $p?$ stands for a notion of test. The latter depends, of course, on the space of truth values in the proposed semantics, i.e. on the particular choice of a complete right residuated lattice $\mathbf{L}$. Hence, a conditional statement

`if` p `then` π_1 `else` π_2 `endif`

is encoded as

$p?; \pi_1 || (\texttt{not}\ \ p)?; \pi_2$

where `not` p denotes the negation of p and is defined in Sect. 3.2. Note that in this encoding parallel composition plays the role usually assigned to the choice operator $+$ in the decomposition of conditionals for classical imperative languages. The motivation for this definition is to suitably capture the behaviour of conditionals in FAS, as informally explained in the Introduction.

A few examples will help the reader to get familiar with the language. Thus, statement

```
Temp:=38
```

assigns the real number 38 to a variable of type `number` (i.e. in data domain $\mathbb{R}$). Similarly,

```
O2_low:=FUZZY SET((70,0),
  (75,1),(85,1),(90,0))
```

assigns a fuzzy set to the fuzzy variable `O2_low`. The notation used means that `O2_low` is defined as a fuzzy set, linear on the open intervals]70, 75[,]75, 85[, [85, 90] and constant on $]-\infty, 70[$ and $]90, +\infty[$. Formally,

$$\texttt{O2\_low}(r) := \begin{cases} r/5 - 14 & \text{if } 70 \le r \le 75 \\ 1 & \text{if } 75 \le r \le 85 \\ -r/5 + 18 & \text{if } 85 \le r \le 90 \\ 0 & \text{otherwise} \end{cases}$$

for each $r \in \mathbb{R}$, as plotted in Fig. 5.
Intuitively it gives, for each oxygen value, a degree which measures how "low" such value actually is. In other words, it defines numerically what it means for the variable to represent a low level of oxygen concentration.

Generically, a fuzzy set, i. e. a function $\sigma : \mathbb{R} \to [0, 1]$, is denoted by

`FUZZY SET`$((a_1, t_1), \ldots, (a_k, t_k))$

where $t_i = \sigma(a_i)$ for $i = 1, \ldots, k$, such that σ is linear on each open interval $]a_1, a_2[$, ..., $]a_{k+1}, a_k[$, and constant on $]-\infty, a_1[$ and $]a_k, \infty[$. Naturally, although the example

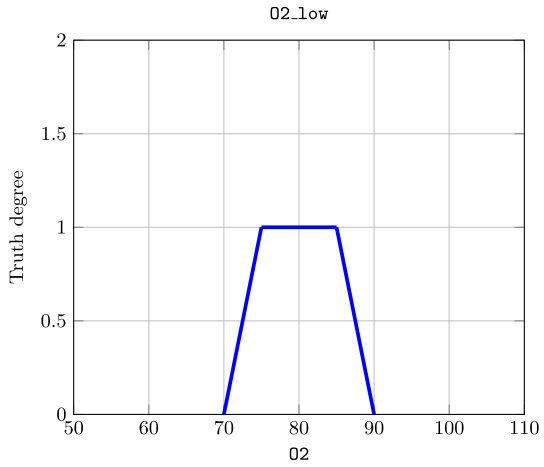

Fig. 5 Graph of the fuzzy variable `O2_low`

presented has a trapezoidal form, other types of fuzzy sets can be defined in FAS. For instance, we can have a fuzzy variable defined as the triangle (70, 0), (80, 1), (90, 0).

3.2 Semantics

Let $\mathbf{L}$ be a complete right residuated lattice and X a set of variables. Denotationally, programs in FAS are interpreted over computational *states* defined as weighted valuations of variables, i.e. functions

$$w : X \times \mathbb{R} \to \mathbf{L}$$

To illustrate the proposed semantics, we will resort, along this subsection, to the following example, taken from Vetterlein et al. (2010).

Example 6

```
O2_low:=FUZZY SET((70,0),(75,1),
  (85,1),(90,0))
if O2 is in O2_low
      then PIP_inc:=5  else PIP_inc:=0
```

The program represents an excerpt of a module of a medical fuzzy control system which provides support for mechanically ventilated patients after cardiac surgery. The code instantiates, in the first line, the fuzzy variable `O2_low` to a fuzzy set. The next instruction defines a fuzzy control rule, encoded in a `if-then-else` statement. Depending on the oxygen level of a patient, the rule makes a modification to the peak inspiratory pressure (`PIP_inc`).

In this setting, functional and predicate terms are interpreted as follows.

Definition 5 (Interpretation of functional terms) Let F be a set of function symbols and X a set of variables. The interpretation of terms $t \in T_F(X)$ in a state $w \in W$ is given by the map

$$[\![\_]\!]_w : T_F(X) \to \mathbf{L}^{\mathbb{R}}$$

defined as follows:

$$[\![x]\!]_w(r) = w(x)(r)$$

$$[\![\mathrm{c}]\!]_w(r) \;=\; \delta_c(r) \;=\; \begin{cases} \top & \text{if } r = c \\ \bot & \text{otherwise} \end{cases}$$

$$[\![\mathrm{f}(t_1, \ldots, t_n)]\!]_w(r) =$$

$$\sum_{i \in I} \left\{ \prod_{j=1}^{n} [\![t_j]\!]_w(r_j^i) \mid f(r_1^i, \ldots, r_n^i) = r \right\}$$

where I is the cardinality of the set of all possible solutions of $f(r_1^i, \ldots, r_n^i) = r$ in $\mathbb{R}$, with each f of arity n being interpreted as a function on real numbers $\mathbb{R}^n \to \mathbb{R}$ (e.g. $+, \times, ^2, \sqrt{}, \ldots$), $x \in X$ and $\mathrm{c}, \mathrm{f}(t_1, \ldots, t_n)$ are the syntactic representations of the constant $c \in \mathbb{R}$ and the functional term $f(t_1, \ldots, t_n) \in T_F(X)$, respectively.

Let us illustrate this semantics resorting to Example 6 and choosing $\mathbf{L}$ as the lattice $\mathbf{G}$ of Example 2. We already defined, in the previous subsection, the fuzzy variable `O2_low`.

Consider computing the value of term `O2_low+5` in state w_0. According to Definition 5, the term 5 is defined as

$$[\![5]\!]_{w_0}(r) \to \begin{cases} 1 & \text{if } r = 5 \\ 0 & \text{, otherwise} \end{cases}$$

The vertices are computed as follows:

$$\begin{aligned}
[\![\texttt{O2_low+5}]\!]_{w_0}(75) &= [\![\texttt{O2_low}]\!]_{w_0}(70);\, [\![5]\!]_{w_0}(5) \\
&= \min\{0, 1\} = 0 \\
[\![\texttt{O2_low+5}]\!]_{w_0}(80) &= [\![\texttt{O2_low}]\!]_{w_0}(75);\, [\![5]\!]_{w_0}(5) \\
&= \min\{1, 1\} = 1 \\
[\![\texttt{O2_low+5}]\!]_{w_0}(90) &= [\![\texttt{O2_low}]\!]_{w_0}(85);\, [\![5]\!]_{w_0}(5) \\
&= \min\{1, 1\} = 1 \\
[\![\texttt{O2_low+5}]\!]_{w_0}(95) &= [\![\texttt{O2_low}]\!]_{w_0}(90);\, [\![5]\!]_{w_0}(5) \\
&= \min\{0, 1\} = 0
\end{aligned}$$

For an intermediate value of r, e.g. $r = 77$, this yields

$$\begin{aligned}
[\![\texttt{O2_low+5}]\!]_{w_0}(77) &= [\![\texttt{O2_low}]\!]_{w_0}(72);\, [\![5]\!]_{w_0}(5) \\
&= \min\{0.4, 1\} = 0.4
\end{aligned}$$

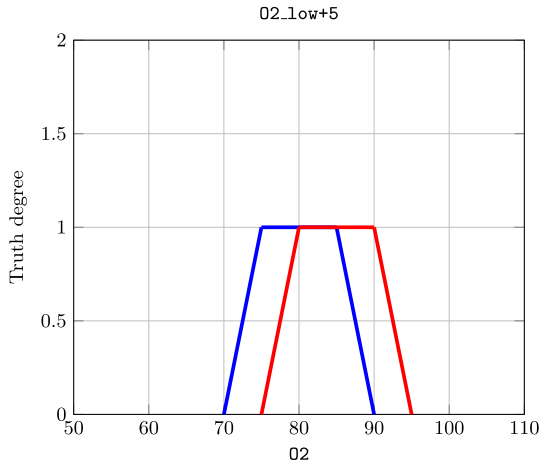

Fig. 6 Graph `O2_low` (blue line) and `O2_low+5` (red line)

Note that the interpretation of the term `O2_low+5` in w_0 is given by a horizontal translation of the graph of `O2_low` by 5 units, as depicted in Fig. 6:

Definition 6 (Interpretation of predicate terms) Let P be a set of predicate symbols. The interpretation of a predicate $p \in T_P(X)$ in a state $w \in W$ is given by the map

$$[\![\_]\!]_w : T_P(X) \to L$$

defined by

$$[\![p(t_1, \ldots, t_n)]\!]_w = \sum_{i \in I} \left\{ \prod_{j=1}^{n} [\![t_j]\!]_w(r_j^i) \mid p(r_1^i, \ldots, r_n^i) \right\}$$

where I is the cardinality of the set of all possible values $(r_1^i, \ldots, r_n^i) \in \mathbb{R}^n$ satisfying $p(r_1^i, \ldots, r_n^i)$, with p of arity n interpreted as a predicate (e.g. $\leq, =, \ldots$).

As an example, let us compute the interpretation of the predicate `O2_low` $\leq$ `75`, according to Definition 6.

$$\begin{aligned}
[\![\texttt{O2_low} \leq 75]\!]_{w_0} &= [\![\texttt{O2_low}]\!]_{w_0}(75); [\![75]\!]_{w_0}(75) + \cdots \\
&\quad + [\![\texttt{O2_low}]\!]_{w_0}(70); [\![75]\!]_{w_0}(75) \\
&= 1; 1 + \cdots + 0; 1 \\
&= \max\{\min\{1, 1\}, \ldots, \min\{0, 1\}\} \\
&= 1
\end{aligned}$$

For another example, consider predicate `O2 is in O2_low`, whose interpretation in the same state w_0 is the function

$$[\![\texttt{O2 is in O2\_low}]\!]_{w_0}(r) = \begin{cases} w_0(\texttt{O2\_low}, r) & \text{for the only } r.[\![\texttt{O2}]\!]_{w_0}(r) = \top \\ \bot & \text{otherwise} \end{cases}$$

We are now able to introduce the semantics of FAS programs as function

$$[\![\_]\!] : \mathrm{Prg}(\mathrm{At}) \longrightarrow M^{\mathbf{L}}(W)$$

Thus, *Assignment.*

$$[\![x := t]\!] \;=\; \{(w, \varphi(w, t)) \mid w \in W\}$$

where

$$\varphi(w, t) \;=\; \lambda\, w' . \begin{cases} \top & \text{if } \forall_{x' \neq x, r \in \mathbb{R}} . \, w'(x')(r) = w(x')(r) \\ & \qquad \wedge\, w'(x)(r) = [\![t]\!]_w(r) \\ \bot & \text{otherwise} \end{cases}$$

Note that although tha variables involved may be fuzzy, the assignment itself is a crisp binary relation on the set of states. Concretely, an *assignment* $x := t$ connects any state w with a set of states w' such that the weight of x in w' is the value of term t in w.

In Example 6, the assignment $[\![$`PIP_inc:=5`$]\!]$ attributes the term `5` to the variable `PIP_inc`, formally

$$[\![\texttt{PIP\_inc}]\!]_{w_1}(r) = [\![5]\!]_{w_0}(r)$$

for all $r \in \mathbb{R}$.

In order to capture conditional operators `if-then-else` and `switch` entail the need for a notion of test, whose syntax is, as mentioned above, $p?$, for a predicate $p \in T_P(X)$. Its interpretation is as follows:
Test.

$$[\![p?]\!] = \left\{ (w, \varphi) \mid \varphi(w') = \begin{cases} [\![p]\!]_w & \text{if } w = w' \\ \bot & \text{otherwise} \end{cases} \right\}$$

FAS supports the predicates composed with the usual logical connectives `and`, `or` and `not`.

Conjunction (`and`) and disjunction (`or`) are interpreted as operators $\cdot$ and $+$ on $\mathbf{L}$, respectively. Moreover, since the semantics is based on $\mathbb{L}$-residuated lattices, i.e. with 1 as the greatest element, "negation" of value $[\![p]\!]_w$, syntactically `not p`, is interpreted as $1 - [\![p]\!]_w$.
Conditional.

$$[\![\texttt{if } p \texttt{ then } \pi_1 \texttt{ else } \pi_2 \texttt{ endif}]\!] = [\![p?; \pi_1 || (\texttt{not } p)?; \pi_2]\!]$$

$$= [\![p?]\!] \circ [\![\pi_1]\!] \uplus [\![(\texttt{not}\, p)?]\!] \circ [\![\pi_2]\!]$$

Switch.

$$[\![\texttt{switch}\ p_i\ \texttt{then}\ \pi_i \cdots \texttt{endswitch}]\!] = [\![p_1?; \pi_1 || \ldots || p_n?; \pi_n]\!] = \biguplus_i ([\![p_i?]\!] \circ [\![\pi_i]\!])$$

Intuitively, both operators relate a state w with a fuzzy set of states w', which assigns a weight of execution to each π_i. Each one of these weights is the value of the evaluated predicate corresponding to the branch in which π_i is executed.

Although in standard imperative programming languages, the semantics of conditionals is expressed through choice (union), conditionals in FAS do not represent a choice, but a form of parallel evaluation. The parallel composition $\uplus$ suits better the idea that we want to capture, since the execution of those commands in FAS does not represent a choice. Consider the following simple example. Let $W = \{w_0, w_1, w_2\}$ be a set of states and consider fuzzy multirelations $R = \{(w_0, \varphi_1)\}$, with $\varphi(w_1) = 0.4$, and $S = \{(w_0, \varphi_2)\}$, with $\varphi(w_2) = 0.6$. The union of R and S is the set $\{(w_0, \varphi_1), (w_0, \varphi_2)\}$, which carries the standard interpretation of conditionals as a nondeterministic choice between (w_0, φ_1) and (w_0, φ_2). On the other hand, the parallel composition $R \uplus S$ represents a single execution $\{(w_0, \varphi_1 \cup \varphi_2)\}$ from state w_0 going simultaneously to states w_1 and w_2.

Let us now illustrate the suitability of the proposed semantics by computing the interpretation of the conditional fragment of our reference example, choosing again $\mathbf{L} = \mathbf{G}$.

$$\begin{aligned}
&[\![\texttt{if O2 is inO2_low}\\
&\texttt{then PIP_inc:=5 else PIP_inc:=0}]\!]\\
&= [\![p?; \texttt{PIP_inc:=5} \,||\, (\texttt{not}\ \ p)?; \texttt{PIP_inc:=0}]\!]\\
&= [\![p?]\!] \circ [\![\texttt{PIP_inc:=5}]\!] \uplus [\![(\texttt{not}\ \ p)?]\!] \circ [\![\texttt{PIP_inc:=0}]\!]\\
&= \Big\{(w, \varphi_1) \mid \exists_{\varphi_1'}.(w, \varphi_1') \in [\![p?]\!] \wedge\\
&\quad \exists_{F_1: W \to \mathbf{L}^W}. \forall_{b \in W}. (b, F'(b)) \in [\![\texttt{PIP_inc:=5}]\!]\\
&\quad \wedge\ \varphi_1(u) = \sum_b \big(\varphi_1'(b); F_1(b)(u)\big)\Big\}\\
&\quad \uplus \Big\{(w, \varphi_2) \mid \exists_{\varphi_2'}.(w, \varphi_2') \in [\![(\texttt{not}\ \ p)?]\!] \wedge\\
&\quad \exists_{F_2: W \to \mathbf{L}^W}. \forall_{b \in W}. (b, F_2(b)) \in [\![\texttt{PIP_inc:=0}]\!]\\
&\quad \wedge\ \varphi_2(u) = \sum_b \big(\varphi_2'(b); F_2(b)(u)\big)\Big\}\\
&= \{(w, \varphi_1) \mid \varphi_1(u) = \varphi_1'(w); F_1(w)(u)\}\\
&\quad \uplus \{(w, \varphi_2) \mid \varphi_2(u) = \varphi_2'(w); F_2(w)(u)\}\\
&= \{(w, \varphi_1) \mid \varphi_1(u) = [\![p]\!]_w; 1\}\\
&\quad \uplus \{(w, \varphi_2) \mid \varphi_2(u) = [\![\texttt{not}\ \ p]\!]_w; 1\}\\
&= \{(w, \underline{[\![p]\!]_w})\} \uplus \{(w, \underline{[\![\texttt{not}\ \ p]\!]_w})\}
\end{aligned}$$

where $p = \texttt{if O2 is in O2\_low}$.

4 A *-free dynamic logic for FAS

4.1 The logic

Each complete right residuated lattice $\mathbf{L}$ induces a *-free dynamic logic $\mathcal{L}(\mathbf{L})$ to reason about fuzzy imperative programs in FAS, with $\mathbf{L}$ giving a precise interpretation of "fuzziness". This section works out the details and exemplifies the logic in action through a number of examples.

Once a language for programs is fixed, as done in the previous section, the set of formulas for $\mathcal{L}(\mathbf{L})$ introduces an existential modality over FAS programs. Formally,

Definition 7 A *signature* for $\mathcal{L}(\mathbf{L})$ is a tuple

$$\Delta = ((F, P), \Pi)$$

where (F, P) is a data signature composed by function and predicate symbols, and $\Pi \subseteq \mathrm{At}$. The set of $\mathcal{L}(\mathbf{L})$-*formulae* for Δ, denoted by $\mathrm{Fm}^{\mathcal{L}(\mathbf{L})}(\Delta)$, is generated by the following rule

$$\rho ::= \top \mid \bot \mid p \mid \rho \vee \rho \mid \rho \wedge \rho \mid \rho \to \rho \mid \langle \pi \rangle \rho$$

for $p \in T_P(Y)$ and $\pi \in \mathrm{Prg}(\mathrm{At})$.

By abuse of notation, we use the same symbols $\top$ and $\bot$ to refer to both the greatest and lowest element of the lattice $\mathbf{L}$ and the logic formula, whenever clear from context.

Definition 8 Let $\mathbf{L}$ be a complete right residuated lattice. The *weighted satisfaction relation* for $\mathcal{L}(\mathbf{L})$ consists of a function

$$\models_{\mathcal{L}(\mathbf{L})} : W \times \mathrm{Fm}^{\mathcal{L}(\mathbf{L})}(\Delta) \to \mathbf{L}$$

recursively defined by

- $(w \models_{\mathcal{L}(\mathbf{L})} \top) = \top$
- $(w \models_{\mathcal{L}(\mathbf{L})} \bot) = \bot$
- $(w \models_{\mathcal{L}(\mathbf{L})} p) = [\![p]\!]_w$
- $(w \models_{\mathcal{L}(\mathbf{L})} \rho \wedge \rho') = (w \models_{\mathcal{L}(\mathbf{L})} \rho) \cdot (w \models_{\mathcal{L}(\mathbf{L})} \rho')$
- $(w \models_{\mathcal{L}(\mathbf{L})} \rho \vee \rho') = (w \models_{\mathcal{L}(\mathbf{L})} \rho) + (w \models_{\mathcal{L}(\mathbf{L})} \rho')$
- $(w \models_{\mathcal{L}(\mathbf{L})} \rho \to \rho') = (w \models_{\mathcal{L}(\mathbf{L})} \rho) \to (w \models_{\mathcal{L}(\mathbf{L})} \rho')$
- $(w \models_{\mathcal{L}(\mathbf{L})} \langle \pi \rangle \rho) = \sum_{\varphi \in \Pi_2[\![\pi]\!]} \Big(\sum_{u \in U} \big(\varphi(u); (u \models_{\mathcal{L}(\mathbf{L})} \rho)\big)\Big)$

In several situations, as the one of Example 6, some contextual assumptions must be taken into account. Thus, to verify a formula in $\mathcal{L}(\mathbf{L})$, we assume the validity of a set of contextual rules Γ. Notation

$$(\Gamma, w \models_{\mathcal{L}(\mathbf{L})} \rho)$$

stands for the value of $(w \models_{\mathcal{L}(\mathbf{L})} \rho)$ assuming Γ, i.e. given that $(w \models_{\mathcal{L}(\mathbf{L})} \Gamma) = \top$.

The (graded) satisfaction of $(w, g \models_{\mathcal{L}(\mathbf{L})} \langle\pi\rangle\rho)$ is given by the weight of some fuzzy set φ which is related to state w by some fuzzy multirelation, and the weight of formula ρ for some state u in the domain of φ (U).

4.2 Some properties of $\mathcal{L}(\mathbf{L})$

This section establishes a number of properties of $\mathcal{L}(\mathbf{L})$. The following two lemmas, the second one establishing auxiliary properties relevant to prove the axiomatization of $\mathcal{L}(\mathbf{L})$, can be proved along the lines of the second author previous work (Madeira et al. 2016).

Lemma 1 *Let* $\mathbf{L}$ *be a complete right residuated lattice. Then,*

$$(w \models_{\mathcal{L}(\mathbf{L})} \rho \to \rho') = \top \Leftrightarrow (w \models_{\mathcal{L}(\mathbf{L})} \rho) \leq (w \models_{\mathcal{L}(\mathbf{L})} \rho') \tag{34}$$

$$(w \models_{\mathcal{L}(\mathbf{L})} \rho \leftrightarrow \rho') = \top \Leftrightarrow (w \models_{\mathcal{L}(\mathbf{L})} \rho) = (w \models_{\mathcal{L}(\mathbf{L})} \rho') \tag{35}$$

Lemma 2 *The following properties hold for any right residuated lattice* $\mathbf{L}$*:*

$$a \leq b \;\&\; c \leq d \Rightarrow a + c \leq b + d \tag{36}$$

$$a; (b \cdot c) \leq (a; b) \cdot (a; c) \tag{37}$$

Additionally, for I finite,

$$\sum_{i \in I} (a_i \cdot b_i) \leq \sum_{i \in I} a_i \cdot \sum_{i \in I} b_i \tag{38}$$

Lemma 3 *Let* $\mathbf{L}$ *be a complete right residuated lattice. The following are valid formulæ in* $\mathcal{L}(\mathbf{L})$*:*

(3.1) $\langle\pi\rangle(\rho \vee \rho') \leftrightarrow \langle\pi_0\rangle\rho \vee \langle\pi_0\rangle\rho'$, $\pi_0 \in \mathrm{At}$
(3.2) $\langle\pi\rangle(\rho \wedge \rho') \to \langle\pi\rangle\rho \wedge \langle\pi\rangle\rho'$
(3.3) $\langle\pi_1; \pi_2\rangle\rho \leftrightarrow \langle\pi_1\rangle\langle\pi_2\rangle\rho$
(3.4) $\langle\pi\rangle\bot \leftrightarrow \bot$
(3.5) $\langle\pi_1 || \pi_2\rangle\rho \leftrightarrow \langle\pi_1\rangle\rho \vee \langle\pi_2\rangle\rho$

Proof **(3.1)**:

$$\begin{aligned}
& (w \models_{\mathcal{L}(\mathbf{L})} \langle\pi_0\rangle(\rho \vee \rho')) \\
= \;& \{ \text{definition of } \models_{\mathcal{L}(\mathbf{L})} \} \\
& \sum_{\varphi \in \Pi_2[\![\pi_0]\!]} \left(\sum_{u \in U} \left(\varphi(u); (u \models_{\mathcal{L}(\mathbf{L})} \rho \vee \rho') \right) \right) \\
= \;& \{ \text{definition of } \models_{\mathcal{L}(\mathbf{L})} \} \\
& \sum_{\varphi \in \Pi_2[\![\pi_0]\!]} \left(\sum_{u \in U} \left((\varphi(u); \left((u \models_{\mathcal{L}(\mathbf{L})} \rho) \right. \right. \right. \\
& \left. \left. \left. + (u \models_{\mathcal{L}(\mathbf{L})} \rho') \right) \right) \right) \\
= \;& \{ \text{by (9)} \} \\
& \sum_{\varphi \in \Pi_2[\![\pi_0]\!]} \left(\sum_{u \in U} \left(\varphi(u); (u \models_{\mathcal{L}(\mathbf{L})} \rho) \right. \right. \\
& \left. \left. + \varphi(u); (u \models_{\mathcal{L}(\mathbf{L})} \rho') \right) \right) \\
= \;& \{ \pi_0 \text{ atomic; thus, } w \text{ is related to a singleton } \{u\} \} \\
& \sum_{\varphi \in \Pi_2[\![\pi_0]\!]} \left(\varphi(u); (u \models_{\mathcal{L}(\mathbf{L})} \rho) + \varphi(u); (u \models_{\mathcal{L}(\mathbf{L})} \rho') \right) \\
= \;& \{ \text{by (3) and (4)} \} \\
& \sum_{\varphi \in \Pi_2[\![\pi_0]\!]} \left(\sum_{u \in U} \left(\varphi(u); (u \models_{\mathcal{L}(\mathbf{L})} \rho) \right) \right) \\
& + \sum_{\varphi \in \Pi_2[\![\pi_0]\!]} \left(\sum_{u \in U} \left(\varphi(u); (u \models_{\mathcal{L}(\mathbf{L})} \rho') \right) \right) \\
= \;& \{ \text{definition of } \models_{\mathcal{L}(\mathbf{L})} \} \\
& (w \models_{\mathcal{L}(\mathbf{L})} \langle\pi_0\rangle\rho) + (w \models_{\mathcal{L}(\mathbf{L})} \langle\pi_0\rangle\rho') \\
= \;& \{ \text{definition of } \models_{\mathcal{L}(\mathbf{L})} \} \\
& (w \models_{\mathcal{L}(\mathbf{L})} \langle\pi_0\rangle\rho \vee \langle\pi_0\rangle\rho')
\end{aligned}$$

Therefore, by (35),
$\langle\pi_0\rangle(\rho \vee \rho') \leftrightarrow \langle\pi_0\rangle\rho \vee \langle\pi_0\rangle\rho'$ holds.

(3.2):

$$
\begin{aligned}
& (w \models_{\mathcal{L}(\mathbf{L})} \langle\pi\rangle(\rho \wedge \rho')) \\
= \quad & \{ \text{definition of } \models_{\mathcal{L}(\mathbf{L})} \} \\
& \sum_{\varphi \in \Pi_2[\![\pi]\!]} \left(\sum_{u \in U} \left(\varphi(u); (u \models_{\mathcal{L}(\mathbf{L})} \rho \wedge \rho') \right) \right) \\
= \quad & \{ \text{definition of } \models_{\mathcal{L}(\mathbf{L})} \} \\
& \sum_{\varphi \in \Pi_2[\![\pi]\!]} \left(\sum_{u \in U} \left(\varphi(u); \left((u \models_{\mathcal{L}(\mathbf{L})} \rho) \cdot (u \models_{\mathcal{L}(\mathbf{L})} \rho') \right) \right) \right) \\
\leq \quad & \{ \text{by (37) and (36)} \} \\
& \sum_{\varphi \in \Pi_2[\![\pi]\!]} \left(\sum_{u \in U} \left((\varphi(u); (u \models_{\mathcal{L}(\mathbf{L})} \rho)) \right. \right. \\
& \qquad \left. \left. \cdot (\varphi(u); (u \models_{\mathcal{L}(\mathbf{L})} \rho')) \right) \right) \\
\leq \quad & \{ \text{by (38)} \} \\
& \sum_{\varphi \in \Pi_2[\![\pi]\!]} \left(\sum_{u \in U} \left(\varphi(u); (u \models_{\mathcal{L}(\mathbf{L})} \rho) \right) \right) \\
& \quad \cdot \sum_{\varphi \in \Pi_2[\![\pi]\!]} \left(\sum_{u \in U} \left(\varphi(u); (u \models_{\mathcal{L}(\mathbf{L})} \rho') \right) \right) \\
= \quad & \{ \text{definition of } \models_{\mathcal{L}(\mathbf{L})} \} \\
& (w \models_{\mathcal{L}(\mathbf{L})} \langle\pi\rangle\rho) \cdot (w \models_{\mathcal{L}(\mathbf{L})} \langle\pi\rangle\rho') \\
= \quad & \{ \text{definition of } \models_{\mathcal{L}(\mathbf{L})} \} \\
& (w \models_{\mathcal{L}(\mathbf{L})} \langle\pi\rangle\rho \wedge \langle\pi\rangle\rho')
\end{aligned}
$$

Therefore, by (34), $\langle\pi\rangle(\rho \wedge \rho') \rightarrow \langle\pi\rangle\rho \wedge \langle\pi\rangle\rho'$ holds.

(3.3):

$$
\begin{aligned}
& (w \models_{\mathcal{L}(\mathbf{L})} \langle\pi_1; \pi_2\rangle\rho) \Leftrightarrow \\
& \quad \exists_{\varphi' \in \Pi_2[\![\pi_1]\!], F: W \rightarrow \mathbf{L}^W} . \forall_{b \in W} . \\
& \quad (b, F(b)) \in [\![\pi_2]\!] \ \wedge \ \varphi(u) = \sum_b (\varphi'(b); F(b)(u))
\end{aligned}
$$

This yields.

$$
\begin{aligned}
& (w \models_{\mathcal{L}(\mathbf{L})} \langle\pi_1; \pi_2\rangle\rho) \\
= \quad & \{ \text{definition of } \models_{\mathcal{L}(\mathbf{L})} \text{ and } [\![\_]\!] \} \\
& \sum_{u \in U} \left(\left(\sum_b (\varphi'(b); F(b)(u)) \right); (u \models_{\mathcal{L}(\mathbf{L})} \rho) \right) \\
= \quad & \{ \text{by (4)} \} \\
& \sum_b \left(\left(\sum_{u \in U} (\varphi'(b); F(b)(u)) \right); (u \models_{\mathcal{L}(\mathbf{L})} \rho) \right) \\
= \quad & \{ \text{by (10)} \} \\
& \sum_b \left(\sum_{u \in U} (\varphi'(b); F(b)(u); (u \models_{\mathcal{L}(\mathbf{L})} \rho) \right) \\
= \quad & \{ \text{by (9)} \} \\
& \sum_b \left(\varphi'(b); \left(\sum_{u \in U} (F(b)(u); (u \models_{\mathcal{L}(\mathbf{L})} \rho)) \right) \right)
\end{aligned}
$$

But since $(w, \varphi') \in \Pi_2[\![\pi_1]\!]$ and $(b, F(b)) \in [\![\pi_2]\!]$, for all $b \in W$, we have

$$
\begin{aligned}
& \sum_b \left(\varphi'(b); \left(\sum_{u \in U} (F(b)(u); (u \models_{\mathcal{L}(\mathbf{L})} \rho)) \right) \right) \\
& \quad = (w \models_{\mathcal{L}(\mathbf{L})} \langle\pi_1\rangle\langle\pi_2\rangle\rho)
\end{aligned}
$$

Hence, by (35), $\langle\pi_1; \pi_2\rangle\rho \leftrightarrow \langle\pi_1\rangle\langle\pi_2\rangle\rho$ is valid.

(3.4):

$$
\begin{aligned}
& (w \models_{\mathcal{L}(\mathbf{L})} \langle\pi\rangle\bot) \\
= \quad & \{ \text{definition of } \models_{\mathcal{L}(\mathbf{L})} \} \\
& \sum_{\varphi \in \Pi_2[\![\pi]\!]} \left(\sum_{u \in U} \left(\varphi(u); (u \models_{\mathcal{L}(\mathbf{L})} \bot) \right) \right) \\
= \quad & \{ \text{definition of satisfaction} \} \\
& \sum_{\varphi \in \Pi_2[\![\pi]\!]} \left(\sum_{u \in U} \left(\varphi(u); \bot \right) \right) \sum_{\varphi \in \Pi_2[\![\pi]\!]} \left(\sum_{u \in U} \bot \right) \\
= \quad & \{ \text{by (6)} \} \\
& \bot
\end{aligned}
$$

Therefore, by (35), $\langle\pi\rangle\bot \leftrightarrow \bot$ is valid.

(3.5):

$$
\begin{aligned}
& (w \models_{\mathcal{L}(\mathbf{L})} \langle \pi_1 || \pi_2 \rangle \rho) \\
= \quad & \{ \text{ definition of } \models_{\mathcal{L}(\mathbf{L})} \} \\
& \sum_{\varphi \in \Pi_2[\![\pi_1 || \pi_2]\!]} \left(\sum_{u \in W} (\varphi(u); u \models_{\mathcal{L}(\mathbf{L})} \rho) \right) \\
= \quad & \{ \text{ definition of } [\![\_]\!] \} \\
& \sum_{\varphi_1 \cup \varphi_2 \in \Pi_2[\![\pi_1 || \pi_2]\!]} \left(\sum_{u \in W} \left((\varphi_1 \cup \varphi_2)(u); (u \models_{\mathcal{L}(\mathbf{L})} \rho) \right) \right) \\
= \quad & \{ (\varphi_1 \cup \varphi_2)(u) = \varphi_1(u) + \varphi_2(u) \} \\
& \sum_{\substack{\varphi_1 \in \Pi_2[\![\pi_1]\!] \\ \wedge \varphi_2 \in \Pi_2[\![\pi_2]\!]}} \left(\sum_{u \in W} \left((\varphi_1(u) + \varphi_2(u)); (u \models_{\mathcal{L}(\mathbf{L})} \rho) \right) \right) \\
= \quad & \{ \text{ by (9)} \} \\
& \sum_{\substack{\varphi_1 \in \Pi_2[\![\pi_1]\!] \\ \wedge \varphi_2 \in \Pi_2[\![\pi_2]\!]}} \left(\sum_{u \in W} \left(\varphi_1(u); (u \models_{\mathcal{L}(\mathbf{L})} \rho) \right. \right. \\
& \qquad \left. \left. + \varphi_2(u); (u \models_{\mathcal{L}(\mathbf{L})} \rho) \right) \right) \\
= \quad & \{ \text{ by (3), (4) and definition of } || \} \\
& \sum_{\varphi_1 \in \Pi_2[\![\pi_1]\!]} \left(\sum_{u \in W} \left(\varphi_1(u); (u \models_{\mathcal{L}(\mathbf{L})} \rho) \right) \right) \\
& + \sum_{\varphi_2 \in \Pi_2[\![\pi_2]\!]} \left(\sum_{u \in W} \left(\varphi_2(u); (u \models_{\mathcal{L}(\mathbf{L})} \rho) \right) \right) \\
= \quad & \{ \text{ definition of } \models_{\mathcal{L}(\mathbf{L})} \} \\
& w \models_{\mathcal{L}(\mathbf{L})} \langle \pi_1 \rangle \rho + w \models_{\mathcal{L}(\mathbf{L})} \langle \pi_2 \rangle \rho \\
= \quad & \{ \text{ definition of } \models_{\mathcal{L}(\mathbf{L})} \} \\
& w \models_{\mathcal{L}(\mathbf{L})} \langle \pi_1 \rangle \rho \vee \langle \pi_2 \rangle \rho
\end{aligned}
$$

Therefore, by (35), $\langle \pi_1 || \pi_2 \rangle \rho \leftrightarrow \langle \pi_1 \rangle \rho \vee \langle \pi_2 \rangle \rho$ is valid. □

5 An illustration

In order to illustrate the logic at work, let us consider a simpler version of Example 6, which consists of a fuzzy control system suggesting a value for the *peak inspiratory pressure* (PIP) based on the O2 level measured in a patient after a cardiac surgery. The situation is described in FAS by the following conditional statement:

$$
\begin{aligned}
\rho \stackrel{\text{abv}}{=} & \texttt{ if O2 is in O2\_low} \\
& \texttt{ then PIP:=0 else PIP:=5}
\end{aligned}
$$

Let us also assume that, in an initial state w_0, the patient has a O2 level of 89 [2]. Our goal is to verify how 'normal' the O2 level of the patient is after running the controller from state w_0.

Formally, this problem corresponds to computing the value of the following $\mathcal{L}(\mathbf{L})$ formula

$$
w_0 \models_{\mathcal{L}(\mathbf{L})} \langle \rho \rangle \, \phi \tag{39}
$$

where $\phi \stackrel{\text{abv}}{=} \texttt{O2 is in O2\_normal}$.

In order to shorten the proof, we use the encoding defined in Sect. 3.1 to represent program ρ as

$$
\psi?; \pi_1 || (\texttt{not } \psi)?; \pi_2
$$

where

$$
\begin{aligned}
\pi_1 &\stackrel{\text{abv}}{=} \texttt{PIP\_inc:=0} \\
\pi_2 &\stackrel{\text{abv}}{=} \texttt{PIP\_inc:=5} \\
\psi &\stackrel{\text{abv}}{=} \texttt{O2 is in O2\_low}
\end{aligned}
$$

The value of

$$
w_0 \models_{\mathcal{L}(\mathbf{L})} \langle \psi?; \pi_1 || (\texttt{not } \psi)?; \pi_2 \rangle \, \phi \tag{40}
$$

is obtained through the satisfaction relation established in Definition 8, as follows.

[2] Note that the system only suggests a modification in the value of PIP_inc and such alteration is carried out manually by some health professional.

$$
\begin{aligned}
& w_0 \models_{\mathcal{L}(\mathbf{L})} \langle \psi?; \pi_1 || (\texttt{not}\ \ \psi)?; \pi_2 \rangle\ \phi \\
= \quad & \{ \text{Lemma 3} \} \\
& w_0 \models_{\mathcal{L}(\mathbf{L})} \langle \psi? \rangle \langle \pi_1 \rangle\ \phi\ +\ w_0 \models_{\mathcal{L}(\mathbf{L})} \langle (\texttt{not}\ \ \psi)? \rangle \langle \pi_2 \rangle\ \phi \\
= \quad & \{ \text{definition of } \models_{\mathcal{L}(\mathbf{L})} \} \\
& \sum_{\psi_1 \in \Pi_2[\![\psi?]\!]} \left(\sum_{u \in U} \psi_1(u); \right. \\
& \left. \left(\sum_{\varphi_1 \in \Pi_2[\![\pi_1]\!]} \left(\sum_{u \in U} \varphi_1(u); u \models_{\mathcal{L}(\mathbf{L})} \phi \right) \right) \right) \\
& + \sum_{\psi_2 \in \Pi_2[\![(\texttt{not}\ \ \psi)?]\!]} \left(\sum_{u \in U} \psi_2(u); \right. \\
& \left. \left(\sum_{\varphi_2 \in \Pi_2[\![\pi_2]\!]} \left(\sum_{u \in U} \varphi_2(u); u \models_{\mathcal{L}(\mathbf{L})} \phi \right) \right) \right) \\
= \quad & \{ \text{definition of } [\![\_]\!] \} \\
& \psi_1(w_0); (\varphi_1(u_1); u_1 \models_{\mathcal{L}(\mathbf{L})} \phi) \\
& + \psi_2(w_0); (\varphi_2(u_2); u_2 \models_{\mathcal{L}(\mathbf{L})} \phi) \\
= \quad & \{ \text{definitions of } [\![\_]\!] \} \\
& \psi_1(w_0); (\top; u_1 \models_{\mathcal{L}(\mathbf{L})} \phi)\ +\ \psi_2(w_0); (\top; u_2 \models_{\mathcal{L}(\mathbf{L})} \phi) \\
= \quad & \{ (8) \} \\
& \psi_1(w_0); u_1 \models_{\mathcal{L}(\mathbf{L})} \phi\ +\ \psi_2(w_0); u_2 \models_{\mathcal{L}(\mathbf{L})} \phi
\end{aligned}
$$

Let us now interpret this problem using a concrete evaluation environment encoded in the following context rules

$$
\Gamma = \begin{cases} \texttt{PIP\_inc} = 0 \rightarrow \texttt{O2} = 89 \\ \texttt{PIP\_inc} = 5 \rightarrow \texttt{O2} = 92 \end{cases}
$$

which describe the impact of increasing the ventilator's `PIP_inc` on the O2 level of the patient. The increment of `PIP_inc` by 5 units results in an increase in the O2 from 89 to 92, whereas if no action is performed, this level does not suffer any modification. In turn, these new values for O2 entail a new value for the predicate `O2 is in O2_normal`.

It is relevant to mention that the output of the program and the concrete decision of the health professional operating the system do not necessarily coincide. The values $\psi_1(w_0)$ and $\psi_2(w_0)$, which correspond to the weights of the predicates `O2 is in O2_normal` and `not O2 is in O2_normal`, highlight which branch is more relevant to execute. This way, we are in presence of an uncertainty about which path will be chosen by the user. It is precisely over such an uncertainty that the logic reasons about. The actual execution of the command, which will be decided according to the weight of each branch, is not reflected in the logic.

Assuming Γ, we can discuss the problem instantiating $\mathbf{L}$ with different lattices, namely $\mathbf{Ł}$, $\mathbf{G}$ and $\mathbf{\Pi}$. Starting with $\mathbf{Ł}$, we obtain

$$0.2; 1 + 0.8; 0.8 = \max(0.2, 0.6) = 0.6$$

The value 0.6 is interpreted as the certainty that the execution of the fuzzy controller ρ adjusts the O2 level of the patient to 'normal'. The formula (40) evaluates precisely the impact of the suggestion made by the fuzzy controller on the O2 level of the patient. Note, in particular, that the values of predicates `O2 is in O2_low`, `not O2 is in O2_low` and `O2 is in O2_normal` affect the value of (40), with the 'else' branch having a stronger impact due to its higher weight compared to the 'then' branch.
Considering now the lattice $\mathbf{G}$, it yields

$$\max(\min(0.2, 1), \min(0, 0.8)) = \max(0.2, 0) = 0.2$$

Finally, the instantiation with $\mathbf{\Pi}$ entails

$$\max(0.2 \times 1, 0 \times 0.8) = \max(0.2, 0) = 0.2$$

Despite the strictness of the logic, the discrepancy between the values of the three lattices suggests that the truth space for each concrete application context needs to be carefully selected by the user.

6 Conclusion and further work

This paper introduced a formal semantics and a family of *-free dynamic logics for the Fuzzy Arden Syntax, parametric on an complete right residuated lattice which supports a truth space in which 'fuzziness' is interpreted. The semantics is based on the notion of *fuzzy binary multirelation* which models a program going from a state to a fuzzy set of states. On the logical side, the modal operators are adapted from Peleg (1987) to account for 'fuzziness' in the execution of programs.

Numerous variants of fuzzy, modal and dynamic logics, or combinations thereof, were previously considered for reasoning with fuzziness degrees either in the evaluation logical formulas, program executions or both. Some approaches add modalities to fuzzy logics over concrete lattices (e.g. Gödel or Łukasiewicz) and define corresponding Kripke models (Caicedo and Rodriguez 2010; Hansoul and Teheux 2013; Hájek 2010). Others, more generic, define many-valued modal logics (Bou et al. 2011) and propositional dynamic logics (Sedlár 2020), in which both the truth space of propositions and the semantics of transitions are values in an arbitrary mathematical structure: a residuated lattice, in the former, and a FL-algebra, in the latter. Still other approaches, aiming specifically at program verification purposes, are discussed in Madeira et al. (2016) and Gomes et al. (2019). The former

proposes a family of propositional dynamic logics, parametric on an action lattice, to interpret and reason about generic, weighted programs. The latter extends such a family by considering that both programs and propositions are built over a signature with variables and terms over them. In such a context, programs can represent uncertainties in their courses of execution or resources consumed. However, despite the weighted nature of both programs and predicates, the conditional statements are encoded, in those two approaches by choice, i.e. the $+$ operator of Kleene algebra, whose semantics is defined as fuzzy set union. Other papers, built with different application scenarios in mind, include Liau (1999), which uses logics to reason about expected utility of actions with applications to decision theory, and Teheux (2014), Di Nola et al. (2020), aiming to generalize PDL to include reasoning with Łukasiewicz logics and their algebraic counterparts. Another example of a PDL extension is Hughes and Kimiaghalam (2006), which adapts the classic PDL semantics to a probabilistic scenario, as a way to measure the efficacy of menas-end relations. In such proposal, probabilities are added to transition systems and formulas are evaluate as fuzzy predicates.

The attentive reader may observe that, contrary to what is common in first-order dynamic logic (e.g. Harel et al. 2000), our proposal is quantifier free. Actually there is not any technical problem in enriching the formalism with quantification. However, as variables in this paper are instantiated with function values, such an enrichment would enforce additional technicalities in the interpretation of formulas which could hinder our fundamental message.

Certainly, this approach can be extended in several directions. In particular, we are experimenting semantic variants for sequential composition of fuzzy multirelations through generalizing the three corresponding operators proposed for the crisp case, by Furusawa et al. (2017). We are also looking at capturing the semantics of FAS `aggregate` statement as a generic operation in the lattice.

We believe that the parametric nature of our framework may benefit research on fuzzy program analysis. Taking as parameter other instances of lattices beyond the ones discussed above will extend the range of applications to other non-trivial computational domains. The method scales to deal with, for example, resource management or minimization of energy costs, by considering the tropical algebra

$$\boldsymbol{R} = (R_+ \cup \{\infty\}, \min, +, \infty, 0, \rightarrow, \min)$$

or even to model a discrete truth space, by taking the algebra of Blok and Ferreirim (2000). Regarding the former interpretation, we would want to study in which terms such an instance of our logic is related with the PDL generalization presented in Behounek (2008).

Finally, we want to discuss the suitability of this framework to interpret other fuzzy programming languages. One such example is jFuzzyLogic (Cingolani and Alcalá-fdez 2013), an open-source Java library which provides an interface and an Eclipse plugin to facilitate the development of fuzzy control applications.

Acknowledgements This work was founded by the ERDF—European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation—COMPETE 2020 Programme and by National Funds through the Portuguese funding agency, FCT—Fundação para a Ciência e a Tecnologia, within projects `POCI-01-0145-FEDER-030947` and `POCI-01-0145-FEDER-029946`

Author contributions The three authors contributed to the planning, design and writing of the manuscript. The conception of the definitions, results and correspondent proofs were regularly discussed by the three authors. The first draft of the manuscript was written by Leandro Rafael Moreira Gomes, and all authors read and comment on each version and future directions to take. The final version was read and approved by all authors.

Compliance with ethical standards

Conflict of interest The authors have no conflict of interest to declare.

Ethics approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

Anand V, Carroll AE, Biondich PG, Dugan TM, Downs SM (2018) Pediatric decision support using adapted arden syntax. Artif Intell Med 92:15–23

Behounek L (2008) Modeling costs of program runs in fuzzified propositional dynamic logic. In: Hakl F (ed) Doktorandské dny 08. ICS AS CR and Matfyzpress, pp 6–14

Blok WJ, Ferreirim I (2000) On the structure of hoops. Algebra Univ 43:233–257

Bou F, Esteva F, Godo L, Rodríguez R (2011) On the minimum many-valued modal logic over a finite residuated lattice. J Logic Comput 21:739–790

Caicedo X, Rodriguez R (2010) Standard Gödel modal logics. Stud Logica 94:189–214

Cingolani P, Alcalá-fdez J (2013) jFuzzyLogic: a java library to design fuzzy logic controllers according to the standard for fuzzy control programming. Int J Comput Intell Syst 6:61–75

den Hartog J, de Vink EP (2002) Verifying probabilistic programs using a Hoare like logic. Int J Found Comput Sci 13(3):315–340

Di Nola A, Grigolia R, Vitale G (2020) Dynamic Łukasiewicz logic and dynamic MV-algebras. Int J Approx Reason 124:103–110

Fischer MJ, Ladner RE (1979) Propositional dynamic logic of regular programs. J Comput Syst Sci 18(2):194–211

Foster N, Kozen D, Mamouras K, Reitblatt M, Silva A (2016) Probabilistic NetKAT. In: Thiemann P (ed) Programming languages and systems—ESOP 2016, Held as Part of ETAPS 2016, Eindhoven, The Netherlands, Proceedings, volume 9632 of LNCS. Springer, Berlin, pp 282–309

Furusawa H, Kawahara Y, Struth G, Tsumagari N (2017) Kleisli, Parikh and Peleg compositions and liftings for multirelations. J Loginc Algebraic Methods Program 90:84–101

Furusawa H, Struth G (2015) Concurrent dynamic algebra. ACM Trans Comput Logic 16(4):30:1–30:38

Galatos N, Jipsen P, Kowalski T, Ono H (2007) Residuated Lattices: an algebraic glimpse at substructural logics. Elsevier, Amsterdam

Gomes L, Madeira A, Barbosa LS (2019) Generalising KAT to verify weighted computations. Sci Ann Comput Sci 29(2):141–184

Hansoul G, Teheux B (2013) Extending Łukasiewicz logics with a modality: algebraic approach to relational semantics. Studia Logica 101

Harel D, Kozen D, Tiuryn J (2000) Dynamic logic. MIT Press, Cambridge

Hoare T, Möller B, Struth G, Wehrman I (2011) Concurrent Kleene algebra and its foundations. J Logic Algebraic Methods Program 80(6):266–296

Hughes J, Kimiaghalam B (2006) Means-end relations and a measure of efficacy. J Logic Lang Inf 15:83–108

Hájek P (2010) On fuzzy modal logics s5(c). Fuzzy Sets Syst 161(18):2389–2396

Kozen D (1985) A probabilistic PDL. J Comput Syst Sci 30(2):162–178

Kozen D (1993) On action algebras. In: Logic and the flow of information. Amsterdam

Kozen D (1994) A completeness theorem for Kleene algebras and the algebra of regular events. Inf Comput 110:366–390

Kozen D (2000) On Hoare logic and Kleene algebra with tests. ACM Trans Comput Logic TOCL 1(212):1–14

Liau C-J (1999) Many-valued dynamic logic for qualitative decision theory. In: Zhong N, Skowron A, Ohsuga S (eds) New directions in rough sets, data mining, and granular-soft computing. Springer, Berlin, pp 294–303

Madeira A, Neves R, Martins MA (2016) An exercise on the generation of many-valued dynamic logics. J Log Algebraic Methods Program 1:1–29

McIver A, Gonzalia C, Cohen E, Morgan C (2008) Using probabilistic Kleene algebra for protocol verification. J Logic Algebraic Methods Program 76(1):90–111

McIver A, Rabehaja TM, Struth G (2013) Probabilistic concurrent Kleene algebra. In: Bortolussi L, Wiklicky H (eds) Proceedings QAPL 2013, Rome, Italy, volume 117 of *EPTCS*, pp 97–115

Parikh R (1983) Propositional game logic. In: 24th annual symposium on foundations of computer science, Tucson, Arizona, USA, 1983, SFCS'83. IEEE Computer Society, pp 195–200

Parikh R (1985) The logic of games and its applications. In: Karplnski M, van Leeuwen J (eds) Topics in the theory of computation, volume 102 of North-Holland mathematics studies. North-Holland, pp 111 – 139

Peleg D (1987) Concurrent dynamic logic. J ACM 34(2):450–479

Pratt VR (1991) Action logic and pure induction. In: van Eijck J (ed) Logics in AI, JELIA 1990 proceedings, volume 478 of LNCS (lecture notes in artificial intelligence). Springer, Berlin, pp 97–120

Prisacariu C (2010) Synchronous Kleene algebra. J Log Algebraic Methods Program 79(7):608–635

Qiao R, Wu J, Wang Y, Gao X (2008) Operational semantics of probabilistic Kleene algebra with tests. In: Proceedings—IEEE symposium on computers and communications, pp 706–713

Rewitzky I (2003) Binary multirelations. In de Swart HCM, Orlowska E, Schmidt G, Roubens M (eds) Theory and applications of relational structures as knowledge instruments, COST Action 274, TARSKI, revised papers, volume 2929 of Lecture notes in computer science. Springer, Berlin, pp 256–271

Rewitzky I, Brink C (2006) Monotone predicate transformers as up-closed multirelations. In: Schmidt RA (ed) Relations and Kleene algebra in computer science. Springer, Berlin, pp 311–327

Samwald M, Fehre K, de Bruin J, Adlassnig K-P (2012) The Arden Syntax standard for clinical decision support: experiences and directions. J Biomed Inform 45(4):711–718

Sedlár I (2020) Finitely-valued propositional dynamic logic. In: Olivetti N, Sandu G, Verbrugge R, Negri S (eds) Advances in modal logic, vol 13. College Publications, Bern, pp 561–579

Starren JB, Hripcsak G, Jordan D, Allen B, Weissman C, Clayton PD (1994) Encoding a post-operative coronary artery bypass surgery care plan in the Arden Syntax. Comput Biol Med 24(5):411–417

Teheux B (2014) Propositional dynamic logic for searching games with errors. J Appl Logic 12(4):377–394

Tsumagari N (2012) Probabilistic relational models of complete IL-semirings. Bull Inf Cybern 44:87–109

Vetterlein T, Mandl H, Adlassnig K-P (2010) Fuzzy Arden syntax: a fuzzy programming language for medicine. Artif Intell Med 49(1):1–10

Zadeh LA (1965) Fuzzy sets. Inf Control 8(3):338–353