Verification for everyone? An overview of Dynamic Logic

Alexandre Madeira

CIDMA, U. Aveiro, Portugal
QuantaLab INESC TEC, U. Minho
madeira@ua.pt

Abstract. This note, reporting the homonym keynote presented in the *International Symposium on Molecular Logic and Computational Synthetic Biology 2018*, traces an informal roadmap on Dynamic Logic (DL) field, focusing on its versatility and resilience to be adjusted and adopted in a wide class of application domains and computational paradigms. The exposition argues the room for developments on tagging DL to the analysis of synthetic biologic domain.

1 Introduction

Dynamic Logic [8] was introduced in the 70's by Pratt in [28] as a suitable logic to reason about, and verify, classic imperative programs. Since then, it evolved to an entire family of logics, which became increasingly popular for assertional reasoning about a wide range of systems and scenarios.

This talk guides an overview on this path prepared to the broad audience of this symposium, with interests and backgrounds ranging from formal Logics, control and systems theory to Synthetic Biology. Rather than to introduce technical aspects on the mentioned formalism, this presentation aims to raise the attention of the reader to the 'camaleonic' nature of DL, on its adoption on the verification of novel computational domains and paradigms, and in the way it can be yet extended to fit on the new challenges of synthetic biology.

This exposition starts by revisiting the roots of the topic, namely by (i) the generic ideas of the calculus of Floyd and Hoare on classic imperative programs and, by (ii) introduce Modal Logic, with its Kripke semantics, as the natural formalism to reason about state transition systems. Then, recalling the seminal ideas of Pratt of using a modal logic to perform Floyd-Hoare reasoning, we briefly introduce the propositional and first order versions of DL (see [29] for an historical perspective on the development of the topic).

Then, we overview some of contributions on the topic developed by our group. The dynamisation method [21, 23] contributed on this direction with a systematic procedure to construct Multi-valued Dynamic Logics able to handle systems where the uncertainty is a prime concern. The method is parametric, and follows our own pragmatic approach to the application of logics to a wide range of complex computational systems: on the place of defining a dedicated logic for each specific application, we develop parametric methods to derive logics tailored to each situation or domain. The specificities involved in each situation should be taken in account on the definition of the parameter adopted for each derivation. The method reviewed in this talk, generates logics suitable to deal with systems involving graded computations. The grading of these logics is reflected in the costs, weight and certainty degrees of programs; but also in the assertions we can do, due to their multi-valued semantics (rather than the standard bivalent one). Beyond of standard Propositional Dynamic Logic [8], we can capture with this method, for instance, the Fuzzy Dynamic Logics presented in [12, 16]. But other logics capable to reason with systems involving resource consuming computations, or assertions graded in discrete truth spaces, can also be achieved as well (cf. [21]).

This generic method have been also adjusted to build multi-valued variants of other families of logics. We discuss in this talk two possible specialisations: one to reason with systems involving knowledge - with its tuning to a method to build Multi-valued Dynamic Epistemic Logics (developed in collaboration with M. Martins and M. Benevides [2]; and, another one. to reason on weighted programs on means of intervals of weight, rather than points (developed in collaboration with R. Santiago, M. Martins and B. Bedregal [30, 31]).

2 The seminal roots

Floyd-Hoare Calculus

As mentioned above, the works of Floyd and Hoare were determinant on the adventure of the formal verification discipline in software engineering. The standard concept of software corrections emerged from the ideas of [7, 11], by means of the notion of Hoare triple:

$$\{\phi\}\,\pi\,\{\varphi\}$$

Formally, a triple $\{\phi\}\,\pi\,\{\varphi\}$ is valid if any terminating execution of π from a state satisfying ϕ, results in a state satisfying φ. Actually this notion of the program correctness w.r.t. a specification underlies, not only the modern techniques of software verification, but also the principles design-by-contract development and specification methods based in the state transitions with pre and post conditions. The Floyd-Hoare logic (Fig. 2) is a syntactic calculus to prove the correctness of a complex program by decomposing it into simpler ones. The intuitions for the set of axioms and inference rules is easy. For instance, the axiom (**assign**) just states that a condition φ is satisfied after an assignment $x := e$, whenever before of this assignment, the formula obtained by replacing in φ all the occurrences of x by the expression e, was already true. Axiom (**empty**) is also natural, since the program *skip* does not change states. As in the other natural deduction systems, the idea of this calculus is to decompose the proof of compound programs, into a set of simpler proof obligations, by creating a proof tree which leafs are axioms. This is clearly reflected in the (**comp**) and (**if_then_else**) inference rules. The rule (**weak**) allows to manipulate the triple conditions by strengthening

Axioms:

$$(\textbf{assign})\frac{}{\{\varphi[e/x]\}\; x := e\; \{\varphi\}} \qquad (\textbf{empty})\frac{}{\{\phi\}\; \mathit{skip}\; \{\phi\}}$$

Inference rules:

$$(\textbf{weak})\frac{\phi \to \phi' \quad \{\phi'\}S\{\varphi'\} \quad \varphi' \to \varphi}{\{\phi\}S\{\varphi\}} \qquad (\textbf{comp})\frac{\{\phi\}S\{\xi\} \quad \{\xi\}T\{\varphi\}}{\{\phi\}S;T\{\varphi\}}$$

$$(\textbf{if\_then\_else})\frac{\{\phi \wedge \alpha\}S_1\{\varphi\} \quad \{\phi \wedge \neg\alpha\}S_2\{\varphi\}}{\{\phi\}\textbf{if}\;\alpha\;\textbf{then}\;S_1\;\textbf{else}\;S_2\{\varphi\}}$$

Fig. 1. Fragment of the Floyd-Hoare Calculus

preconditions and weakening postconditions. Using this rules we are able to validate Hoare triples. For instance,

$$\{x = 1\}\textbf{if}\; x < 2\;\textbf{then}\; x := x + 1\;\textbf{else}\; x := x * x\{x = 2\}$$

can be proved with the deduction:

$$\frac{\dfrac{\dfrac{}{\{x{=}1\}x{:=}x{+}1\{x{=}2\}}}{\{x{=}1\wedge x{<}2\}x{:=}x{+}1\{x{=}2\}} \qquad \dfrac{}{\{x{=}1\wedge x{\geq}2\}x{:=}x{*}x\{x{=}2\}}}{\{x = 1\}\textbf{if}\; x < 2\;\textbf{then}\; x := x + 1\;\textbf{else}\; x := x * x\{x = 2\}}$$

The left leaf is closed by axiom (**assign**), since $(x = 2)[x + 1/x] \Leftrightarrow x = 1$. For the right one, we just have to note that $x = 1 \wedge x < 2 \Leftrightarrow false$, and therefore the triple is vacuously satisfied, since there is no any state satisfying the precondition $false$.

Modal Logic

The long tradition in the study of logics to reasoning in scenarios involving change, come since the age of Aristotle. This family of logics, known as Modal logics represents a classic topic in Logic and Philosophy. The developments of Kripke in the 60's in semantics for these logics, based in transition structures, endow such formalisms with the suitable ingredients to reasoning about state-based systems. This section briefly review the basic definition of propositional multi-modal logic.

Signature for of this logic are pairs (Prop, A) where Prop, A are disjoint sets of propositions, and modalities. The (Prop, A)-formulas are defined by the grammar

$$\varphi ::= p \,|\, \langle a\rangle\varphi \,|\, [a]\varphi \,|\, \neg\varphi \,|\, \varphi \vee \varphi \,|\, \varphi \wedge \varphi$$

where $p \in \text{Prop}$ and $a \in A$.

Models of this logic are state transition structures, with propositions locally assigned to states. Formally, a (Prop, A)-model is a tuple $M = \big(W, V, R\big)$ where

– W is a set

- $V : \mathrm{Prop} \to \mathcal{P}(W)$ is a function
- $R = (R_a \subseteq W \times W)_{a \in A}$ is an A-family of binary relations

Finally, we recall the notion of modal satisfaction. The satisfaction of a (Prop, A)-formula φ in a state w of a (Prop, A)-model M is recursively defined as follows:

- $M, w \models p$ iff $w \in V(p)$
- $M, w \models \langle a \rangle \varphi$ iff there is a $w' \in W$ such that $(w, w') \in R_a$ and $M, w' \models \varphi$
- $M, w \models [a]\varphi$ iff for any $w' \in W$ such that $(w, w') \in R_a$ we have $M, w' \models \varphi$
- $M, w \models \neg\varphi$ iff it is false that $M, w \models \varphi$
- $M, w \models \varphi \wedge \varphi'$ iff $M, w \models \varphi$ and $M, w \models \varphi'$
- $M, w \models \varphi \vee \varphi'$ iff $M, w \models \varphi$ or $M, w \models \varphi'$

Propositional Dynamic Logic

Being programs a paradigmatic example of state-transition systems, modal logic emerged as natural formalism to reason about it. Particularly, it provided solid theoretic field, to support the verifications in Floyd-Hoare triples. Moreover, as observed by V. Pratt in the seminal work [28] Floyd-Hoare logic is purely syntactic, and Modal logic can be considered as an alternative to Floyd-Hoare logic.

In a first view, the multi-modal logic presented above would be enough to reason about programs, by considering the class of possible programs as the set of modalities. Fortunately programs are structured terms. This allows us to deal with these objects in a systematic way, a key factor on the definition of dynamic logics. Assuming a set of atomic programs Π, the universe of the (composed) programs can be defined with the following grammar:

$$\pi ::= \pi_0 \mid \pi + \pi \mid \pi; \pi \mid \pi^* \mid ?\chi$$

for $\pi_0 \in \Pi$ and χ a formula in the logic. The connectives of the terms are the usual Kleene operators, namely $+$ represents the non-deterministic choice, ; the sequential composition and $*$ the reflexive iterative operator. Additionally we have the operator ? for tests, that is necessary to represent conditionals. Note that this grammar actually provides an abstract computational language, able to represent the standard imperative language commands. For instance we have that **if** χ **then** π **fi** $\equiv (?\chi; \pi) + (?\neg\chi)$, that **if** χ **then** π **else** π' **fi** $\equiv (?\chi; \pi) + (?\neg\chi; \pi')$ and that **while** χ **do** π **od** $\equiv (?\chi; \pi)^*; ?\neg\chi$.

Fixing this abstract model of computation, we are in condition to adjust multi-modal logic into a formalism to reasoning about programs. Firstly, signatures are pairs (Prop, Π) where Prop is a set of propositions and Π is a set of atomic programs names. Models are Kripke structures tuples $\big(W, V, R\big)$ where:

- W is a set
- $V : \mathrm{Prop} \to \mathcal{P}(W)$ is a function
- $R = (R_\pi \subseteq W \times W)$, $\pi \in \Pi$

Observe that these models only interprets atomic programs, since $\overline{R}$ can be extended to the interpretation of composed programs, with the usual relational operators. Namely, we have $\overline{R}(\pi_0) = R_{\pi_0}$, $\overline{R}(\pi + \pi') = \overline{R}(\pi) \cup \overline{R}(\pi')$, $\overline{R}(\pi; \pi') = \overline{R}(\pi) \cdot \overline{R}(\pi')$ and $\overline{R}(\pi^*) = \overline{R}(\pi)^\star = \bigcup_{n \in \mathbb{N}} \overline{R}(\pi^n)$, where $\pi^{n+1} = \pi; \pi^n$. Finally we have the interpretations of tests as the co-reflexive $\overline{R}(\chi?) = \{(w, w) | M, w \models \chi\}$. Now, we defined the satisfaction relation as above, just replacing the cases of modal operators by

– $M, w \models \langle\pi\rangle\varphi$ iff there is a $w' \in W$ such that $(w, w') \in \overline{R}_\pi$ and $M, w' \models \varphi$;
– $M, w \models [\pi]\varphi$ iff for any $w' \in W$ such that $(w, w') \in \overline{R}_\pi$ we have $M, w' \models \varphi$.

Shifting to the first-order case

As suggested, propositional dynamic logic provides the essential machinery to reason about abstract programs. The regular modalities reflect the abstract structure of the programs control, where the standard imperative commands can be easily accommodated. We observe here that above freedom on what an atomic program is a key factor to the versatility of this logic, to be adapted to new computational domains. Let us firstly focus in the verification of a classic imperative programs. For this case atomic programs are, naturally, variables assignments. As usual, the states in our models should correspond to valuations of program data variables. Hence, the atomic propositions used in the propositional case are here replaced by data predicates. For sake of simplicity, we assume that all the programs variables are numerical $\mathbb{R}$ variables.

Formally, signatures are sets of data variables Var. The set of programs is defined as in the propositional case, but considering assignments $x := \theta$, with θ a term defined with Var and the arithmetic operations $\{+, -, \times, \cdots\}$,on place of atomic programs $\pi \in \Pi$. As mentioned, semantic states are variables assignments $w \in \mathbb{R}^{\mathrm{Var}}$. The interpretation of programs is now given by an interpretation $\rho \subseteq \mathbb{R}^{\mathrm{Var}} \times \mathbb{R}^{\mathrm{Var}}$ exactly defined as the propositional $\bar{R}$, but considering the interpretation of base programs $\rho_{x:=\theta} = \{(u, v) | v(x) = \theta$ and for any $y \in \mathcal{V} \setminus \{x\}, u(y) = v(y)\}$.

Hence, we can use this modal logic to support the verification of Floyd-Hoare triples. For instance the validity of formula

$$x = 1 \to [(x < 2)?; x := x + 1 + (\neg\,(x < 2))?; x := x * x]x = 2$$

or, equivalently

$$x = 1 \to [\mathbf{if}\, x < 2\, \mathbf{then}\, x := x + 1\, \mathbf{else}\, x := x * x]x = 2$$

corresponds to the verification of the triple

$$\{x = 1\}\mathbf{if}\, x < 2\, \mathbf{then}\, x := x + 1\, \mathbf{else}\, x := x * x\{x = 2\}$$

done above. This is an useful fact that relates Floyd-Hoare logic and first-order dynamic logic: for any Floyd Hoare triple $\{\psi\}\pi\{\varphi\}$, $\{\psi\}\pi\{\varphi\}$ is verified iff the formula $\psi \to [\pi]\varphi$ is valid.

Note that this principle can be extended to other variants of Hoare and dynamic logics. Whenever a new dynamic logic is defined, a new Floyd-Hoare is created for free.

Less conventional variants

As stated in the introduction, the resilience of dynamic logic on being adjusted to new computational paradigms and domains is a key factor for its adoption in a wide multitude of contexts. Actually, the way we construct the first-order dynamic logic from its propositional version, by preserving all of its structure, with the exception of its atomic programs (and respective interpretation), not only justify the big family of dynamic logics we have today, but opens the door for further versions and variants. Actually, as it will be discussed, the DL adequacy and resilience on being adapted to a wide range of computational systems, relies on real understanding of what is the nature of the atomic programs involved in each context. On place of considering programs as the standard variables assignments, we can consider, for instance systems of differential equations flowing in a given domain (e.g. a time constraint or a data predicate). This is the base idea of differential dynamic logic of A. Platzer [27]. By considering as atomic programs these evolutions, we are in the presence of a logic to reason and verify continuous systems. But if we consider also discrete assignments we have a suitable logic to reason in hybrid systems.

A logic to reason about quantum programs and quantum protocols can be also achieved if we consider, as basic programs, quantum measurements and unitary transformations. Such is the idea behind the works of S. Smets and A. Baltag in Quantum Dynamic Logic [1]. The game logics of Parikh [26] and the Dynamic Epistemic logics (revisited bellow)[6] are two well established logic fields, where the same analogy can be done.

3 Parametric Generation of Dynamic Logics

This section overviews the dynamisation method, a systematic method to construct Multi-valued Dynamic Logics that we introduced in [22, 23]. This method is parametrized by an action lattice [13]. Despite of its distinct original purposes, this algebraic structure showed to be very useful in the context of our work, on providing a generic support for the computational space (as a Kleene algebra) and for the truth spaces (as residuated lattice) of the logics build trough our constructions.

Definition 1 ([13]). *An* action lattice *is a tuple*

$$\mathbf{A} = (A, +, ;, 0, 1, *, \rightarrow, \cdot)$$

where A is a set, 0 *and* 1 *are constants,* $*$ *is an unary operation in A and* $+, ;, \rightarrow$ *and* $\cdot$ *are binary operations in A satisfying the axioms enumerated in Figure 1, where the relation $\leq$ is induced by $+$: $a \leq b$ iff $a + b = b$.*

$$a + (b + c) = (a + b) + c \qquad (1)$$
$$a + b = b + a \qquad (2)$$
$$a + a = a \qquad (3)$$
$$a + 0 = 0 + a = a \qquad (4)$$
$$a; (b; c) = (a; b); c \qquad (5)$$
$$a; 1 = 1; a = a \qquad (6)$$
$$a; (b + c) = (a; b) + (a; c) \qquad (7)$$
$$(a + b); c = (a; c) + (b; c) \qquad (8)$$
$$a; 0 = 0; a = 0 \qquad (9)$$
$$1 + a + (a^*; a^*) \leq a^* \qquad (10)$$
$$a; x \leq x \Rightarrow a^*; x \leq x \qquad (11)$$
$$x; a \leq x \Rightarrow x; a^* \leq x \qquad (12)$$
$$a; x \leq b \Leftrightarrow x \leq a \rightarrow b \qquad (13)$$
$$a \rightarrow b \leq a \rightarrow (b + c) \qquad (14)$$
$$(x \rightarrow x)^* = x \rightarrow x \qquad (15)$$
$$a \cdot (b \cdot c) = (a \cdot b) \cdot c \qquad (16)$$
$$a \cdot b = b \cdot a \qquad (17)$$
$$a \cdot a = a \qquad (18)$$
$$a + (a \cdot b) = a \qquad (19)$$
$$a \cdot (a + b) = a \qquad (20)$$

Fig. 2. Axiomatisation of action lattices

As discussed bellow, the structure of an action lattice plays a double role in our method: it will support the model for computations, and of truth space. The operation $+$, plays a double role, the non-deterministic choice, in the interpretation of programs, and the logical disjunction, in the interpretation of sentences. Operations $*$ and ; are taken to interpret the iterative application and sequential composition of actions and, the operations $\rightarrow$ and $\cdot$ interpret the logical implication and conjunction.

We explore [22] an extensive set of action lattice. Here we will just recall four of them. Firstly, we consider the two elements boolean algebra

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

with the standard boolean connectives and with $\top^* = \bot * = \top$. Moreover, by explicitly introducting a denotation for a truth value *unknown*, we can consider the three elements linear lattice

$$\mathbf{3} = (\{\top, u, \bot\}, \vee, \wedge, \bot, \top, *, \rightarrow, \wedge)$$

where

$\vee$	$\bot$	u	$\top$
$\bot$	$\bot$	u	$\top$
u	u	u	$\top$
$\top$	$\top$	$\top$	$\top$

$\wedge$	$\bot$	u	$\top$
$\bot$	$\bot$	$\bot$	$\bot$
u	$\bot$	u	u
$\top$	$\bot$	u	$\top$

$\rightarrow$	$\bot$	u	$\top$
$\bot$	$\top$	$\top$	$\top$
u	$\bot$	$\top$	$\top$
$\top$	$\bot$	u	$\top$

$*$	
$\bot$	$\top$
u	$\top$
$\top$	$\top$

In order to consider a linear discrete lattice with a finite number of points, we can consider *Wajsberg hoops* [3] enriched with a suitable star operation. For a fix natural $k > 0$ and a generator a, we define the structure $\mathbf{W}_k = (W_k, +, ;, 0, 1, ^*, \rightarrow, \cdot)$, where $W_k = \{a^0, a^1, \cdots, a^k\}$, $1 = a^0$ and $0 = a^k$, and for any $m, n \leq k$, $a^m + a^n = a^{min\{m,n\}}$, $a^m; a^n = a^{min\{m+n,k\}}$, $(a^m)^* = a^0$, $a^m \rightarrow a^n = a^{max\{n-m,0\}}$ and $a^m \cdot a^n = a^{max\{m,n\}}$. For instace, the underlying order of the Wajsberg hoop $\mathbf{W}_5$ is $\mathbf{W}_5$ is $a^5 < a^4 < a^3 < a^2 < a^1 < a^0$.

Moreover, we can also consider continuous structures for the truth degrees and weight for our logics. For instance, the Łukasiewicz arithmetic lattice is the structure

$$\mathbf{Ł} = ([0,1], \max, \odot, 0, 1, *, \to, \min)$$

where $x \to y = \min(1, 1-x+y)$, $x \odot y = \max(0, y+x-1)$ and $x^* = 1$.

Now, fixing an action lattice $\mathbf{A} = (A, +, ;, 0, 1, *, \to, \cdot)$ as parameter, we will construct the multi-valued dynamic logic $\mathcal{DL}(\mathbf{A})$ (as proposed in [17]). Signatures of $\mathcal{DL}(\mathbf{A})$ are pairs (Π, Prop) where Π denotes the set of atomic computations and Prop the set of propositions. Then, the *set of Π-programs* $\mathrm{Prg}(\Pi)$, are defined by the grammar

$$\pi ::= \pi_0 \,|\, \pi;\pi \,|\, \pi + \pi \,|\, \pi^*, \text{ where } \pi_0 \in \Pi$$

Given a signature (Π, Prop), the set of formulas $\mathrm{Fm}^{\mathcal{DL}}(\Pi, \mathrm{Prop})$ is given by the grammar

$$\rho ::= \top \,|\, \bot \,|\, p \,|\, \rho \vee \rho \,|\, \rho \wedge \rho \,|\, \rho \to \rho \,|\, \rho \leftrightarrow \rho \,|\, \langle \pi \rangle \rho \,|\, [\pi]\rho$$

with $p \in \mathrm{Prop}$ and $\pi \in \mathrm{Prg}(\Pi)$.

Now we have to introduce the models for $\mathcal{DL}(\mathbf{A})$. As expected, graded computations will be interpreted in state transition systems with weights in the transitions, usually represented by adjency matrices. On this view, our method takes advantage of the Conway matricial constructions over Kleene algebras i.e. in the structure

$$\mathbb{M}_n(\mathbf{A}) = (M_n(\mathbf{A}), +, \mathbf{;}, \mathbf{0}, \mathbf{1}, \mathbf{*})$$

defined as in [4, 14]. Namely with:

- $M_n(\mathbf{A})$ is the space of $(n \times n)$-matrices over $\mathbf{A}$
- for any $A, B \in M_n(\mathbf{A})$, define $M = A{+}B$ by $M_{i,j} = A_{i,j} + B_{i,j}$, $i, j \leq n$.
- for any $A, B \in M_n(\mathbf{A})$, define $M = A \mathbf{;} B$ by $M_{i,j} = \sum_{k=1}^{n}(A_{i,k}; B_{k,j})$ for any $i, j \leq n$.
- $\mathbf{1}$ and $\mathbf{0}$ are the $(n \times n)$-matrices defined by $\mathbf{1}_{i,j} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$ and $\mathbf{0}_{i,j} = 0$, for any $i, j \leq n$.
- for any $M = [a] \in \mathbb{M}_1(\mathbf{A})$, $M^{\mathbf{*}} = [a^*]$;
  for any $M = \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] \in M_n(\mathbf{A})$, $n > 1$, where A and D are square matrices, define
  $$M^{\mathbf{*}} = \left[\begin{array}{l|r} F^{\mathbf{*}} & F^{\mathbf{*}} \mathbf{;} B \mathbf{;} D^{\mathbf{*}} \\ \hline D^*; C; F^* & D^{\mathbf{*}} {+} (D^{\mathbf{*}} \mathbf{;} C \mathbf{;} F^{\mathbf{*}} \mathbf{;} B \mathbf{;} D^{\mathbf{*}}) \end{array}\right]$$
  where $F = A + B \mathbf{;} D^{\mathbf{*}} \mathbf{;} C$. Note that this construction is recursively defined from the base case (where $n = 2$) where the operations of the base action lattice $\mathbf{A}$ are used.

As showed in [14], the structure $\mathbb{M}_n(\mathbf{A})$ is also a Kleene algebra, and therefore, figures as a suitable space to represent, manipulate and interpret programs. Enriching the interpretation of basic programs with graded interpretations for the propositions, we get the models for a signature (Π, Prop). Formally, the $\mathcal{DL}(\mathbf{A})$ models for (Π, Prop) are tuples

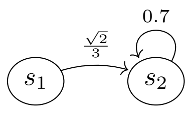
$$\mathcal{A} = (W, V, (\mathcal{A}_\pi)_{\pi \in \Pi})$$

where W is a finite set (of states), $V : \text{Prop} \times W \to A$ is a function, and $\mathcal{A}_\pi \in \mathbb{M}_n(\mathbf{A})$, with n standing for the cardinality of W.

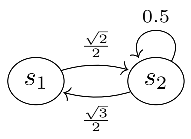
As expected, the interpretation of a program $\pi \in \text{Prg}(\Pi)$ in a model $\mathcal{A} \in \text{Mod}^{\mathcal{DL}}(\Pi, \text{Prop})$ is recursively defined, from the set of atomic programs $(\mathcal{A}_\pi)_{\pi \in \Pi}$, with $\mathcal{A}_{\pi;\pi'} = \mathcal{A}_\pi \,\boldsymbol{;}\, \mathcal{A}_{\pi'}$, $\mathcal{A}_{\pi+\pi'} = \mathcal{A}_\pi + \mathcal{A}_{\pi'}$ and $\mathcal{A}_{\pi^*} = \mathcal{A}_\pi^{\boldsymbol{*}}$ together with the constants interpretations $\mathcal{A}_1 = \mathbf{1}$ and $\mathcal{A}_0 = \mathbf{0}$.

The reader can easily observe that the models of $\mathcal{DL}(\mathbf{2})$ corresponds exactly to the standard PDL. More interesting instantiations can be found in [22].

In order to illustrate the running concepts, let us consider the consider the $\big(\{p, q\}, \{\pi, \pi'\}\big)$-model $\mathcal{A} = (\{s_1, s_2\}, V, (\mathcal{A}_p)_{p \in \{\pi, \pi'\}})$ of $\mathcal{DL}(\text{Ł})$ with $V(p, s_1) = 0.1$, $V(q, s_1) = 0.5$, $V(p, s_2) = \frac{\pi}{4}$ and $V(q, s_2) = 0.75$ and

$\mathcal{A}_\pi :$ 

$\begin{bmatrix} 0 & \frac{\sqrt{2}}{3} \\ 0 & 0.7 \end{bmatrix}$

$\mathcal{A}_{\pi'} :$ 

$\begin{bmatrix} 0 & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{3}}{2} & 0.5 \end{bmatrix}$ (21)

Then, for instance the program $\mathcal{A}_{\pi+\pi'}$ is interpreted by

$$\mathcal{A}_{\pi+\pi'} = \max(\mathcal{A}_\pi, \mathcal{A}_{\pi'}) = \max\left(\begin{bmatrix} 0 & \frac{\sqrt{2}}{3} \\ 0 & 0.7 \end{bmatrix}, \begin{bmatrix} 0 & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{3}}{2} & 0.5 \end{bmatrix}\right) = \begin{bmatrix} 0 & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{3}}{2} & 0.7 \end{bmatrix} \tag{22}$$

The last ingredient for the definition of $\mathcal{DL}(\text{Ł})$ is the graded satisfaction. Here, on place of being a satisfaction relating each state with the formulas there satisfied, we have a function that assigns the 'satisfaction degree' of a formula in a given state state. The operations of the action lattices have to play the truth space role, on the interpretation of logic connectives. Formally, the graded satisfaction relation for a model $\mathcal{A} \in \text{Mod}^{\mathcal{DL}}(\Pi, \text{Prop})$, with $\mathbf{A}$ complete, consists of a function

$$\models : W \times \text{Fm}^{\mathcal{DL}}(\Pi, \text{Prop}) \to A$$

recursively defined as follows:

– $(w \models \top) = \top$
– $(w \models \bot) = \bot$
– $(w \models p) = V(p, w)$, for any $p \in \text{Prop}$
– $(w \models \rho \wedge \rho') = (w \models \rho) \cdot (w \models \rho')$
– $(w \models \rho \vee \rho') = (w \models \rho) + (w \models \rho')$
– $(w \models \rho \to \rho') = (w \models \rho) \to (w \models \rho')$

- $(w \models \rho \leftrightarrow \rho') = (w \models \rho \to \rho'); (w \models \rho' \to \rho)$
- $(w \models \langle \pi \rangle \rho) = \sum_{w' \in W} \big(\mathcal{A}_\pi(w, w'); (w' \models \rho)\big)$
- $(w \models [\pi]\rho) = \prod_{w' \in W} \big(\mathcal{A}_\pi(w, w') \to (w' \models \rho)\big)$

We say that ρ is *valid* when, for any any model $\mathcal{A}$, and for each state $w \in W$, $(w \models \rho) = \top$.

Returning to our running example, we can calculate the satisfaction degree of the formula $\langle \pi + \pi' \rangle (p \to q))$ in the state s_1 as follows:

$$\begin{aligned}(s_1 \models \langle \pi + \pi' \rangle (p \to q)) &= \max(0 \odot (0.1 \to 0.5), \tfrac{\sqrt{2}}{2} \odot (0.75 \to \tfrac{\pi}{4}))\\ &= \tfrac{\sqrt{2}}{2} \odot (0.75 \to \tfrac{\pi}{4})\\ &= \tfrac{\sqrt{2}}{2} \odot \min(1, 1 - 0.75 + \tfrac{\pi}{4})\\ &= \tfrac{\sqrt{2}}{2}\end{aligned}$$

Therefore, we conclude with a degree of certainty $\frac{\sqrt{2}}{2}$ that, after executing $\pi + \pi'$ from the state s_1, we have $p \to q$.

Reasoning with systems involving knowledge

The complexity of the current information systems, involving processes with complex network of heterogeneous learning agents, raises for further generalisations of Multi-agent Epistemic Logics, including weighted versions. Hence, the building logics on-demand principle, inherent to dynamisation, appear as an adequate technique to be used is this domain. In this section, we review a variant of dynamisation tailored to the generation of graded dynamic epistemic logics introduced in [2].

Firstly let us recall the basis of Multi-agents Epistemic Logic (DEL). Signature of DEL are pairs (Prop, Ag) where Prop is a set of propositions and Ag a finite set of agents. Note that this can be seen as propositional dynamic logic signatures which atomic programs are the agent knowledge relations. The (Prop, Ag) formulas of DEL are defined by the grammar

$$\varphi ::= p \mid \top \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid K_a\varphi \mid B_a\varphi \mid C_G\varphi$$

where $p \in \text{Prop}$, $a \in \text{Ag}$ and $G \subseteq \text{Ag}$. The intuitive meaning of the epistemic modalities is the following: $K_a\varphi$ means that agent a knows φ; $B_a\varphi$ means that agent a believes that φ; and the common knowledge operator $C_G\varphi$ - means that all the members of the group of agents G knows φ and each member of the group knows that all the members of the groups know φ, etc.

The models are just special models of PDL. Formally, *multi-agent epistemic model* is a tuple $\mathcal{E} = (W, (R_a)_{a \in \text{Ag}}, V)$ defined as in PDL but assuming that, for any agent $a \in \text{Ag}$, R_a is an equivalence relation. The interpretation of knowledge modalities is defined by

- $\mathcal{M}, s \models K_a\phi$ iff for all $s' \in S : sR_as' \Rightarrow \mathcal{M}, s' \models \phi$
- $\mathcal{M}, s \models B_a\phi$ iff there is an$s' \in S$ such that sR_as' and $\mathcal{M}, s' \models \phi$

– $\mathcal{M}, s \models C_G\phi$ iff for all $s' \in S$, $sR^*_G s' \Rightarrow \mathcal{M}, s' \models \phi$

The similarities with PDL are straightforward. Modality K_a corresponds to the modality $[a]$ for an atomic program a. Its dual, the modality B_a, corresponds to the modality $\langle a \rangle$ for an atomic program a. Modality C_G is captured by the modality $[(\sum_{a\in G} a)^*]$.

In order to get some intuitions on this logic, let us recall the well know example of the envelops used in [6]. Three envelopes containing **0**, **1** and **2** euros are given to the agents **a**na, **b**ob and **c**lara. Each agent just knowns the content of her envelop. Using proposition Prop $= \big\{E_x | E \in \{1,2,3\}, x \in \{a,b,c\}\big\}$ referring that "agent x has envelop E, and representing states by the order of envelops, e.g. the state 012 represents the case that agent **a** has **0**, agent **b** has **1** and **c** has **2**, we can represent epistemic state of each agent as follows[1]:

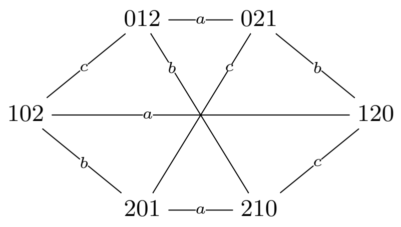


Fig. 3. anna's, **b**ob's and **c**lara's epistemic model [6]

Hence, we have, for instance, that $012 \models B_b 0_a$ and $012 \models B_a K_c 2_c$ hold. Redefining our dynamisation method for this specific seetings we obtain a method to build graded dynamic epistemic logics. The satisfaction relation for the epistemic modalities takes now the form:

– $(w \models K_a\ \varphi) = \bigwedge_{w'\in W} \big(R_a(w,w') \to (w' \models \varphi)\big)$
– $(w \models B_a\ \varphi) = \bigvee_{w'\in W} \big(R_a(w,w'); (w' \models \varphi)\big)$
– $(w \models C_G\ \varphi) = \bigwedge_{w'\in W} \big(R^*_G(w,w') \to (w' \models \varphi)\big)$

These logics are prepared to deal with agents with graded beliefs (on place of bivalent ones). Let us revisit the example above, by supposing that the agent **a**na 'suspect' that the envelop of **b**ob has a higher amount than the one of herself. In a scale from 0 to 5, her belief is 4; Conversely, her belief that the envelop received by **b**ob has a smaller value is 1. The epistemic perception of **a**na is depicted in the following picture. Again, we omit the reflexive loops in the picture (with value 5):[2]

[1] We omit the reflexive loops in the picture
[2] The complete treatment of this illustration is in [2];

$$\begin{tikzcd}
 & 012 \arrow[r, bend left, "4"] & 021 \arrow[l, bend left, "1"] & \\
102 \arrow[rrr, bend left, "4"] & & & 120 \arrow[lll, bend left, "1"] \\
 & 201 \arrow[r, bend left, "4"] & 210 \arrow[l, bend left, "1"] &
\end{tikzcd}$$

Fig. 4. anna's beliefs

Reasoning with interval approximations

There are some situations where only approximations for the transition weights are possible (e.g. when dealing weights over irrational numbers, we have no machine representation of transition weights; or due impreciseness in some measurements). On this purpose, for the specific case of Fuzzy Dynamic logic, we adjusted the dynamisation constructions to deal with intervals, rather than points. This results in a new family of dynamic logics whose assertions, and the satisfaction outcomes, are also intervals. This section informally overviews our work in Interval Dynamic Logic presented in [30, 31]. The presentation is guided to the case Ł but the same principle can be extended to other continuous action algebras.

In the sequel, for any closed interval X, we use $\underline{X}$ and $\underline{Y}$ to denote its left and right bounds, i.e. for $X = [a, b]$, $\underline{X} = a$ and $\underline{Y} = b$.

Our first concern is about the structure to interpret such kind of programs. The following result presented in [30] provides a Kleene algebra for that end:

Theorem 1 ([30]).

$$K(\widehat{Ł}) = \big(\mathbb{U}, Max, \bigodot, [0,0], [1,1], \widehat{*}\big)$$

where

- $\mathbb{U} = \{[a,b] \mid 0 \leq a \leq b \leq 1\}$
- $Max(X,Y) = [\max(\underline{X}, \underline{Y}), \max(\overline{X}, \overline{Y})]$
- $Min(X,Y) = [\min(\underline{X}, \underline{Y}), \min(\overline{X}, \overline{Y})]$
- $X \bigodot Y = [(\underline{X} \odot \underline{Y}), (\overline{X} \odot \overline{Y})] = [\max(0, \underline{X} + \underline{Y} - 1), \max(0, \overline{X} + \overline{Y} - 1)]$
- $X^{\widehat{*}} = [\underline{X}^*, \overline{X}^*] = [1,1]$.

is a Kleene algebra.

For instance, we can consider interval approximations of the weight transition structure presented above as

$$\mathcal{A}_\pi : \begin{tikzcd} s_1 \arrow[r, bend left, "{[0.4,0.5]}"] & s_2 \arrow[loop above, "{[0.7,0.7]}"] \end{tikzcd} \quad \begin{bmatrix} (0,0) & (0.4,0.5) \\ (0,0) & (0.7,0.7) \end{bmatrix} \quad \mathcal{A}_{\pi'} : \begin{tikzcd} s_1 \arrow[r, bend left, "{[0.6,0.8]}"] & s_2 \arrow[loop above, "{[0.5,0.5]}"] \arrow[l, bend left, "{[0.7,0.9]}"] \end{tikzcd} \quad \begin{bmatrix} (0,0) & (0.6,0.8) \\ (0.7,0.9) & (0.5,0.5) \end{bmatrix}$$

Using the $K(\widehat{Ł})$ operations, we can also interpret (composed) programs. For instance $\mathcal{A}_{\pi+\pi'}$ is

$$\max\left(\begin{bmatrix} (0,0) & (0.4,0.5) \\ (0,0) & (0.7,0.7) \end{bmatrix}, \begin{bmatrix} (0,0) & (0.6,0.8) \\ (0.7,0.9) & (0.5,0.7) \end{bmatrix}\right) = \begin{bmatrix} (0,0) & (0.6,0.8) \\ (0.7,0.9) & (0.7,0.7) \end{bmatrix}$$

It is expected to extend the structure $K(\widehat{Ł})$ with an interpretation of the implication, in order to have an action lattice for intervals. The natural candidate is $X \Rrightarrow Y = [(\overline{X} \to \underline{Y}), (\underline{X} \to \overline{Y})] = [\min(1, 1 - \overline{X} + \underline{Y}), \min(1, 1 - \underline{X} + \overline{Y})]$. However, the reader can easily observe that axioms (13) and (15) does not hold in $\widehat{Ł}$ (c.f. [30] for a complete discussion) . Hence, despite of its Kleene algebra structure, $\widehat{Ł}$ it is not an action lattice. We studied in [30], an weakness of action lattices, called *quasi-action lattice*, that capture $\widehat{Ł}$. Fortunately, this structures still have good properties to serve as parameter of dynamisation method. For instance, by considering a valuation $V : \text{Prop} \to \mathbb{U}$ with $V(p, s_1) = [0.1, 0.1]$, $V(q, s_1) = [0.5, 0.5]$, $V(p, s_2) = [0.7, 0.8]$ and $V(q, s_2) = [0.75, 0.75]$, we can calculate the degree of satisfaction of the sentence $\langle \pi + \pi' \rangle (p \to q)$ from the state s_1 as:

$$\begin{aligned}
&(s_1 \models_{\widehat{Ł}} \langle \pi + \pi' \rangle (p \to q)) \\
&\quad = \max([0,0] \odot ([0.1, 0.1] \Rrightarrow [0.5, 0.5]), [0.6, 0.8] \odot ([0.5, 0.5] \Rrightarrow [0.7, 0.8])) \\
&\quad = \max([0,0], [0.6, 0.8] \odot [0.5 \ \to 0.7, 0.5 \ \to 0.8]) \\
&\quad = [0.6, 0.8] \odot [1, 1] \\
&\quad = [0.6, 0.8].
\end{aligned}$$

4 Further extensions and applications?

As suggested along the paper, the pattern of *changing the atomic programs to adapt the computing paradigm* is not only recognised in our methods to build graded dynamic logics, but in most of variants of dynamic logics in the literature. This motivates our position that the shape of dynamic logic provides the *de facto* essence of what a logic for programs is. When invited to make a personal overview in dynamic logic in the International Symposium on Molecular Logic and Computational Synthetic Biology 2018, the authors main motivation was to open the discussion of what should be the suitable atomic programs, for a further dynamic logic tailored to synthetic biology. The same exercise have been done by the group on finding new dynamic logics for other domains and applications, including reactive processes [17, 9, 10], petri-nets with failures [15]. We have also explored this 'logic-on-demand' strategy in other modal logics. Our long term research in the parametric generation of hybrid logics [24, 19, 25] supports the formal development of a wide range of reconfigurable systems from the design to the verification stage [20]. Moreover, we extended the parametric generation of Dynamic Epistemic Logics in [18] by considering structured representation of states.

Exploiting the limits of our methods on building dynamic logics prepared to deal with paraconsistencies in behaviours or in knowledge acquirement, is a research line that we intend to develop. This will certainly be useful for the application domains of this symposium. The recent contributions within the group in paraconsistent hybrid logic [5] provides an interesting starting point for this agenda. Shifting the paraconsistency of atomic modalities to composed programs is, however, challenging. Specific questions as 'what is a paraconsistent program' should be answered. More precisely, the understanding of what is a paraconsistent execution of a program, if the paraconsistency is inherent to the atomic programs, or if it results from a 'paraconsistent control', due non conventional interpretation of the Kleene operators, are questions to be studied in this line.

Acknowledgements. The author would sincerely thanks the invitation of the *MLCSB'18* to present this personal perspective on Dynamic Logic, a topic with which he has been involved in the last years.

This work is financed 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 project `POCI-01-0145-FEDER-016692` and `UID/MAT/04106/2019`, in a contract foreseen in nos. 4–6 of art. 23 of the DL 57/2016, changed by DL 57/2017.

References

1. Baltag, A., Smets, S.: Quantum logic as a dynamic logic. Synthese **179**(2), 285–306 (2011). https://doi.org/10.1007/s11229-010-9783-6, `https://doi.org/10.1007/s11229-010-9783-6`
2. Benevides, M., Madeira, A., Martins, M.: A family of graded epistemic logics. Electr. Notes Theor. Comput. Sci. **338**, 45–59 (2018). https://doi.org/10.1016/j.entcs.2018.10.004, `https://doi.org/10.1016/j.entcs.2018.10.004`
3. Blok, W.J., Ferreirim, I.M.A.: On the structure of hoops. algebra universalis **43**(2-3), 233–257 (2000). https://doi.org/10.1007/s000120050156, `http://dx.doi.org/10.1007/s000120050156`
4. Conway, J.H.: Regular Algebra and Finite Machines. Printed in GB by William Clowes & Sons Ltd (1971)
5. Costa, D., Martins, M.A.: Paraconsistency in hybrid logic. J. Log. Comput. **27**(6), 1825–1852 (2017). https://doi.org/10.1093/logcom/exw027, `https://doi.org/10.1093/logcom/exw027`
6. van Ditmarsch, H., van der Hoek, W., Kooi, B.: Dynamic Epistemic Logic. Synthese Library Series, volume 337, Springer, The Netherland (2008)

7. Floyd, R.W.: Assigning meanings to programs. Proceedings of Symposium on Applied Mathematics **19**, 19–32 (1967), `http://laser.cs.umass.edu/courses/cs521-621.Spr06/papers/Floyd.pdf`
8. Harel, D., Kozen, D., Tiuryn, J.: Dynamic Logic. MIT Press (2000)
9. Hennicker, R., Madeira, A.: Institutions for behavioural dynamic logic with binders. In: Hung, D.V., Kapur, D. (eds.) Theoretical Aspects of Computing - ICTAC 2017 - 14th International Colloquium, Hanoi, Vietnam, October 23-27, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10580, pp. 13–31. Springer (2017)
10. Hennicker, R., Madeira, A., Knapp, A.: A hybrid dynamic logic for event/data-based systems. In: Hahnle, R., van der Aalst, W. (eds.) Fase 2019. Lecture Notes in Computer Science, vol. 11424. Springer (in print)
11. Hoare, C.A.R.: An axiomatic basis for computer programming. Commun. ACM **12**(10), 576–580 (Oct 1969). https://doi.org/10.1145/363235.363259, `http://doi.acm.org/10.1145/363235.363259`
12. Hughes, J., Esterline, A.C., Kimiaghalam, B.: Means-end relations and a measure of efficacy. Journal of Logic, Language and Information **15**(1-2), 83–108 (2006). https://doi.org/10.1007/s10849-005-9008-4, `http://dx.doi.org/10.1007/s10849-005-9008-4`
13. Kozen, D.: On action algebras, manuscript in: Logic and Flow of Information, Amsterdam, 1991
14. Kozen, D.: A completeness theorem for Kleene algebras and the algebra of regular events. Inf. Comput. **110**(2), 366–390 (1994)
15. Leandro Gomes, A.M., Benevides, M.: Logics for petri nets with propagating failures. FSEN19 - Fundamentals of Software Engineering. Lecture Notes in Computer Science (in print)
16. Liau, C.: 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, 7th International Workshop, RSFDGrC '99, Yamaguchi, Japan, November 9-11, 1999, Proceedings. Lecture Notes in Computer Science, vol. 1711, pp. 294–303. Springer (1999). https://doi.org/10.1007/978-3-540-48061-7-36, `http://dx.doi.org/10.1007/978-3-540-48061-7-36`
17. Madeira, A., Barbosa, L.S., Hennicker, R., Martins, M.A.: A logic for the stepwise development of reactive systems. Theor. Comput. Sci. **744**, 78–96 (2018). https://doi.org/10.1016/j.tcs.2018.03.004, `https://doi.org/10.1016/j.tcs.2018.03.004`
18. Madeira, A., Benevides, M., Martins, M.: Epistemic logics with structured states. Electr. Notes Theor. Comput. Sci. (in print)
19. Madeira, A., Martins, M.A., Barbosa, L.S., Hennicker, R.: Refinement in hybridised institutions. Formal Asp. Comput. **27**(2), 375–395 (2015). https://doi.org/10.1007/s00165-014-0327-6, `https://doi.org/10.1007/s00165-014-0327-6`
20. Madeira, A., Neves, R., Barbosa, L.S., Martins, M.A.: A method for rigorous design of reconfigurable systems. Sci. Comput. Program. **132**, 50–76 (2016). https://doi.org/10.1016/j.scico.2016.05.001, `https://doi.org/10.1016/j.scico.2016.05.001`
21. Madeira, A., Neves, R., Martins, M.A.: An exercise on the generation of many-valued dynamic logics. J. Log. Algebr. Meth. Program. **85**(5), 1011–1037 (2016). https://doi.org/10.1016/j.jlamp.2016.03.004, `https://doi.org/10.1016/j.jlamp.2016.03.004`

22. Madeira, A., Neves, R., Martins, M.A.: An exercise on the generation of many-valued dynamic logics. Journal of Logical and Algebraic Methods in Programming pp. – (2016). https://doi.org/http://dx.doi.org/10.1016/j.jlamp.2016.03.004, `http://www.sciencedirect.com/science/article/pii/S2352220816300256`
23. Madeira, A., Neves, R., Martins, M.A., Barbosa, L.S.: A dynamic logic for every season. In: Braga, C., Martí-Oliet, N. (eds.) Formal Methods: Foundations and Applications - 17th Brazilian Symposium, SBMF 2014, Maceió, AL, Brazil, September 29-October 1, 2014. Proceedings. Lecture Notes in Computer Science, vol. 8941, pp. 130–145. Springer (2014). https://doi.org/10.1007/978-3-319-15075-8_9, `https://doi.org/10.1007/978-3-319-15075-8\_9`
24. Martins, M.A., Madeira, A., Diaconescu, R., Barbosa, L.S.: Hybridization of institutions. In: Corradini, A., Klin, B., Cîrstea, C. (eds.) Algebra and Coalgebra in Computer Science - 4th International Conference, CALCO 2011, Winchester, UK, August 30 - September 2, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6859, pp. 283–297. Springer (2011). https://doi.org/10.1007/978-3-642-22944-2_20, `https://doi.org/10.1007/978-3-642-22944-2\_20`
25. Neves, R., Madeira, A., Martins, M.A., Barbosa, L.S.: Proof theory for hybrid(ised) logics. Sci. Comput. Program. **126**, 73–93 (2016). https://doi.org/10.1016/j.scico.2016.03.001, `https://doi.org/10.1016/j.scico.2016.03.001`
26. Parikh, R.: The logic of games and its applications. In: Selected Papers of the International Conference on "Foundations of Computation Theory" on Topics in the Theory of Computation. pp. 111–139. Elsevier North-Holland, Inc., New York, NY, USA (1985), `http://dl.acm.org/citation.cfm?id=4030.4037`
27. Platzer, A.: Logical Foundations of Cyber-Physical Systems. Springer (2018). https://doi.org/10.1007/978-3-319-63588-0, `https://doi.org/10.1007/978-3-319-63588-0`
28. Pratt, V.R.: Semantical considerations on floyd-hoare logic. In: 17th Annual Symposium on Foundations of Computer Science, Houston, Texas, USA, 25-27 October 1976. pp. 109–121. IEEE Computer Society (1976). https://doi.org/10.1109/SFCS.1976.27, `https://doi.org/10.1109/SFCS.1976.27`
29. Pratt, V.R.: Dynamic logic: A personal perspective. In: Madeira, A., Benevides, M.R.F. (eds.) Dynamic Logic. New Trends and Applications - First International Workshop, DALI 2017, Brasilia, Brazil, September 23-24, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10669, pp. 153–170. Springer (2017). https://doi.org/10.1007/978-3-319-73579-5_10, `https://doi.org/10.1007/978-3-319-73579-5\_10`
30. Santiago, R., Bedregal, B., Madeira, A., Martins, M.A.: On interval dynamic logic: Introducing quasi-action lattices. Science of Computer Programming **175**, 1 – 16 (2019). https://doi.org/https://doi.org/10.1016/j.scico.2019.01.007, `http://www.sciencedirect.com/science/article/pii/S0167642319300103`
31. Santiago, R.H.N., Bedregal, B.R.C., Madeira, A., Martins, M.A.: On interval dynamic logic. In: Ribeiro, L., Lecomte, T. (eds.) Formal Methods: Foundations and Applications - 19th Brazilian Symposium, SBMF 2016, Natal, Brazil, November 23-25, 2016, Proceedings. Lecture Notes in Computer Science, vol. 10090, pp. 129–144 (2016). https://doi.org/10.1007/978-3-319-49815-7_8, `https://doi.org/10.1007/978-3-319-49815-7\_8`