# ReacTS 2026

Third International Workshop on

## Reconfigurable Transition Systems

Semantics, Logics and Applications

---

**Informal Pre-proceedings**

Valletta, Malta
24 November 2026

Edited by

Daniel Figueiredo
Duncan Paul Attard

(Please do not distribute)

# Preface

This volume contains the papers presented at the 3rd International Workshop on Reconfigurable Transition Systems: Semantics, Logics and Applications, taking place in Valletta, Malta on 24 November 2026.

This workshop aims to bring together the whole community of researchers working on different ways to model reconfigurable and reactive systems from a formal perspective. These approaches include hybrid logics, reactive frames, model-update logics, and topological and algebraic semantics, as well as formalisms designed for specific purposes, such as separation logic for software verification and dynamic epistemic logic for AI planning. Additionally, our goal is to develop novel approaches, explore potential applications, and share perspectives on the discipline.

The workshop received 13 registered submissions: 10 full papers and 3 short presentations. Of these, 11 were completed manuscripts. The final programme comprises 6 full papers and 3 short presentations. The completed submissions were reviewed by the Programme Committee and external reviewers. These informal pre-proceedings are being made available ahead of the workshop and bring together the versions supplied by the authors for workshop discussion. Authors of accepted full papers will be invited to submit their contributions for the LNCS post-proceedings, combining papers from workshops associated with SEFM 2026. Short presentations are included in these informal pre-proceedings but will not appear in the LNCS post-proceedings. Authors of selected contributions will also be invited to submit extended versions to a special issue of the Journal of Logical and Algebraic Methods in Programming (JLAMP).

We would like to thank all researchers who submitted their work to the workshop, all colleagues who served on the Programme Committee, and the external reviewers for helping us to prepare the workshop programme.

Particular thanks go to our invited speaker, Manuel A. Martins, from the University of Aveiro, Portugal, for presenting his research and sharing his perspectives on reconfigurable transition systems at ReacTS. We are extremely grateful to the local organisers at the University of Malta for their help with the practical arrangements at the Valletta Campus.

November 2026
Valletta, Malta

Daniel Figueiredo
Duncan Paul Attard

# Organisation

## Programme Chairs

| | |
|---|---|
| Daniel Figueiredo | University of Aveiro, Portugal |
| Duncan Paul Attard | University of Malta, Malta |

## Programme Committee

| | |
|---|---|
| Elli Anastasiadi | Aalborg University, Denmark |
| Luís Soares Barbosa | Universidade do Minho, Portugal |
| Benjamin Bedregal | Universidade Federal do Rio Grande do Norte, Brazil |
| Mario Benevides | Universidade Federal Fluminense, Brazil |
| Valentin Cassano | Universidad Nacional de Río Cuarto and CONICET, Argentina |
| Valentina Castiglioni | Eindhoven University of Technology, The Netherlands |
| Madalena Chaves | Centre Inria d'Université Côte d'Azur, France |
| Gabriel Ciobanu | Institute of Computer Science, Romanian Academy, Romania |
| Raul Fervari | Universidad Nacional de Córdoba and CONICET, Argentina |
| Sabine Frittella | Université d'Orleans, France |
| Andreas Herzig | CNRS, Université Paul Sabatier, France |
| Omar Inverso | Gran Sasso Science Institute (GSSI), Italy |
| Juha Kontinen | University of Helsinki, Finland |
| Alexandre Madeira | University of Aveiro, Portugal |
| Sérgio Marcelino | IT & Dep. Mathematics IST, University of Lisbon, Portugal |
| Vanina Martínez | IIIA-CSIC, Barcelona, Spain |
| Alessandra Palmigiano | Vrije Universiteit Amsterdam, The Netherlands |
| José Proença | FCUP, University of Porto, Portugal |
| Fernando R. Velázquez-Quesada | University of Bergen, Norway |
| Umberto Rivieccio | Universidad Nacional de Educación a Distancia, Spain |
| Regivan Santiago | Universidade Federal do Rio Grande do Norte, Brazil |
| François Schwarzentruber | ENS Rennes, France |

| | |
|---|---|
| Sonja Smets | ILLC, University of Amsterdam, The Netherlands |
| Ionuț Țuțu | Simion Stoilow Institute of Mathematics of the Romanian Academy, Romania |
| Amanda Vidal | Czech Academy of Sciences, Czech Republic |
| Fan Yang | Utrecht University, The Netherlands |

# Contents

Preface . . . . . i

**Keynote**

*Reconfigurable Fuzzy Graphs: Semantics, Algebra and Languages* . . . . . 1
Manuel A. Martins.

**Full papers**

*Introducing Relation-Changing Intuitionistic Fuzzy Graphs* . . . . . 2
Suene Duarte, Daniel Figueiredo, Manuel A. Martins, and Regivan Santiago.

*Weak algebraizability for logics of indicative conditionals* . . . . . 12
Miguel Muñoz Pérez.

*Opening the Box to a Probabilistic Dynamic Logic* . . . . . 28
Leandro Gomes and Alexandre Madeira.

*A Reusable Formalisation of Hennessy–Milner Logic for Arbitrary Transition Systems in Lean* . . . . . 46
Fabrizio Montesi, Marco Peressotti, and Alexandre Rademaker.

*Typed Modalities and Friends* . . . . . 60
Ionuţ Ţuţu and Claudia Elena Chiriţă.

*Bisimulation in Fuzzy Topological Semantics* . . . . . 74
Muhammad Afaq Khan, Manuel A. Martins, and Alexandre Madeira.

**Short presentations**

*Timed Reconfigurable Systems: A Tool for Encoding and Analysis* . . . . . 89
Joshua Dourado and Antonio César Castro Iglesias.

*A generalization of twist approach to sublogics* . . . . . 94
Alfredo Roque Freire.

*Reconfigurable Automata Meets Machine Learning: Modeling and Analizing Real Transit Scenarios* . . . . . 95
Antonio César Castro Iglesias, Joshua Dourado, and Mariana Ferreira.

# Reconfigurable Fuzzy Graphs: Semantics, Algebra and Languages

Manuel A. Martins

University of Aveiro, Portugal

**Abstract.** Reconfigurability extends relational models by allowing the structure that governs the dynamics of the system to evolve during execution. In this talk, I present a unified perspective on reactive graph-based systems, ranging from their original relational foundations to recent developments involving fuzzy, labelled, interval-valued and probabilistic reconfiguration. The presentation revisits the key concepts and semantics that support these models, highlighting connections with graph algebra, behavioural equivalence, dynamic fuzzy logic, and language acceptance. Through illustrative examples, I discuss the relationship between graph operations and language-theoretic constructions, as well as the role of finite configuration spaces in enabling static analysis via unfolding techniques. I conclude by outlining some open research directions, including reactive Kleene-type correspondences, equivalence notions, and verification methods, emphasising their relevance to the future development of reactive systems theory.

# Introducing Relation-Changing Intuitionistic Fuzzy Graphs

Suene Duarte[1], Daniel Figueiredo[2,3], Manuel A. Martins[2], and Regivan Santiago[4]

[1] UFERSA – Universidade Federal Rural do Semi-Árido, Rio Grande do Norte, Brazil
[2] CIDMA – Center for Research and Development in Mathematics and Application, University of Aveiro, Portugal
[3] AIBILI – Association for Innovation and Biomedical Research on Light and Image, Coimbra, Portugal
[4] Universidade Federal do Rio Grande do Norte, Departamento de Informática e Matemática Aplicada, Natal, Brazil

**Abstract.** Intuitionistic fuzzy graphs, as presented in [1], are a special case of fuzzy graphs where we consider a nonmembership fuzzy set for edges, in addition to the usual membership fuzzy set. In recent work, the authors have proposed relation-changing fuzzy graphs, obtaining an expressive structure which is able to describe complex systems with a compact representation. This work follows a similar direction and generalizes Intuitionistic fuzzy graphs by embedding relation-changing dynamics, in order to obtain intuitionistic fuzzy switch graphs. Moreover, we explore these new models with some examples.

**Keywords:** Intuitionistic graph · Fuzzy logic · Switch graph.

## 1 Introduction

Intuitionistic fuzzy graphs [1] generalize the usual fuzzy graph by considering both a membership and a nonmembership degree for each edge. These structures, based on the concept of intuitionistic fuzzy set [2], allow one to understand an intuitionistic fuzzy graph as a fuzzy relation of vertices. Another related concept is the notion of fuzzy graph [3], where membership functions are considered for both the set of vertices and the set of edges. However, as we follow the work of [4,5], we do not consider fuzziness of vertices but rather on edges only.

The goal of this work is to expand the framework of intuitionistic fuzzy graph to embed relation-changing features. This notion has been proposed by multiple authors to describe the possibility of a model to change its behavior, for instance by considering the possibility of a "reconfiguration" [6,7] or a system to have "memory" [8,9]. This kind of dynamical structure has been introduced and studied in diverse areas such as epistemic logic [10] and probabilistic, fuzzy and paraconsistent graphs [11,4,12].

In this paper, we propose two versions of relation-changing structures to generalize intuitionistic fuzzy graphs. We firstly introduce intuitionistic fuzzy

switch graphs as a generalization of classic switch graphs [13], where the set of edges is not fixed, but can vary whenever an edge is traversed. Switch graphs represent this relation-changing through a set of higher-level directed edges, that connect two other edges and indicate that crossing one source edge will impact the targeted one. The possibility of fully removing an edge from the model is then considered and explored by introducing intuitionistic reversal fuzzy switch graphs, based on the concept of reversal fuzzy switch graphs [4]. These models also admit edges to be restored, while maintaining the fuzzy measures for membership and nonmembership.

**Outline.** In the next section, we recall the notion of intuitionistic fuzzy graph and fuzzy definitions. Then, Section 3 presents intuitionistic fuzzy switch graphs and Section 4 presents intuitionistic reversal fuzzy switch graphs. The paper closes with some final considerations and directions to follow.

## 2 Intuitionistic fuzzy graphs and fuzzy operators

We start by recalling the concept of intuitionistic fuzzy graph introduced in [1] and then revisit some definitions related to fuzzy setting, that are used throughout the paper. We note that many authors conceive intuitionistic fuzzy graphs where the set of vertices is also conceived under an intuitionistic perspective. However, since we will not explore this, the definitions that follow were obtained by adapting the ones from [1] to our context.

**Definition 1.** *An* intuitionistic fuzzy graph *is a triple* $(V, \mu, \gamma)$ *such that* $V$ *is a non-empty set,* $\mu : V \times V \to [0, 1]$ *is the degree of membership and* $\gamma : V \times V \to [0, 1]$ *is the degree of nonmembership satisfying* $\forall s \in V \times V$, $\mu(s) + \gamma(s) \leqslant 1$.

The set of edges of an intuitionistic fuzzy graph $E \subseteq V \times V$ is defined as $E = \{e \in V \times V : \mu(e) > 0 \text{ or } \gamma(e) < 1\}$. Usually, in intuitionistic fuzzy graphs, the set $E$ of edges is fixed. However, since we will consider a scenario where the set of edges can change, we define $E$ in a different way.

**Definition 2.** *Let* $(V, \mu, \gamma)$ *be an intuitionistic fuzzy graph. A path (of length* $n$*) is a tuple* $(e_1, ..., e_n)$ *such that* $\forall i \in \{1, ..., n\}$, $e_i \in E$, *where* $e_i = (s_i, t_i)$ *and* $e_{i+1} = (s_{i+1}, t_{i+1})$ *are such that* $t_i = s_{i+1}$ *for every* $i < n$.

Now, we introduce some fuzzy operators that will be used in the following sections to define how a graph updates.

**Definition 3 ([14]).** *A* t-norm $T : [0, 1]^2 \to [0, 1]$ *is a function satisfying:*

- $T$ *is associative and commutative;*
- $\forall\ a \in [0, 1]$, $T(1, a) = a$;
- $T(a, b) \leqslant T(a', b')$ *whenever* $a \leqslant a'$ *and* $b \leqslant b'$.

*Moreover, for every $a, b \in [0,1]$, we can define $S : [0,1]^2 \to [0,1]$ as the function such that $S(a,b) = 1 - \big(T(1-a, 1-b)\big)$ and we say that $S$ is the corresponding $t-conorm$ or $s-norm$.*

**Definition 4 (Aggregation Function [15]).** *Let $A : [0,1]^n \to [0,1]$ be a monotonic function. $A$ is an $n-ary$ fuzzy aggregation if $A(0,...,0) = 0$ and $A(1,...,1) = 1$.*

*We say that $A$ is a disjunctive aggregation if $A(x_1,...,x_n) \geqslant \max(x_1,...,x_n)$ and it is a conjunctive aggregation if $A(x_1,...,x_n) \leqslant \min(x_1,...,x_n)$*

**Proposition 1.** *Consider $\bar{A}(x_1,...,x_n) = 1 - A(1-x_1,...,1-x_n)$. $A$ is a conjunctive aggregation if and only if $\bar{A}$ is a disjunctive aggregation.*

*Proof.* If $A$ is an aggregation, it is easy to check that $\bar{A}(0,...,0) = 0$ and $\bar{A}(1,...,1) = 1$. Moreover, if $x_1 \leqslant y_1, ..., x_n \leqslant y_n$ for $y_1, ..., y_n \in [0,1]$, then $\bar{A}(y_1,...,y_n) = 1 - A(1-y_1,...,1-y_n) \geqslant 1 - A(1-x_1,...,1-x_n) = \bar{A}(x_1,...,x_n)$, due to the monotonicity of $A$. Thus, $\bar{A}$ is an aggregation. The reciprocal is obtained because $\bar{\bar{A}} = A$.

Let A be a conjunctive aggregation. Then $\bar{A}(x_1,...,x_n) = 1 - A(1-x_1,...,1-x_n) \geqslant 1 - \min(1-x_1,...,1-x_n) = \max(x_1,...,x_n)$. The reciprocal result is obtained analogously for $A$ being disjunctive and noting that $\bar{\bar{A}} = A$.

Note that every t-norm $T$ is a conjunctive aggregation because, by definition, for every $a, b \in [0,1]$ $T(a,b) \leqslant T(a,1) = a$ and $T(a,b) \leqslant T(1,b) = b$, that implies $T(a,b) \leqslant \min(a,b)$. In the same way, every s-norm is a disjunctive aggregation.

## 3 Intuitionistic Fuzzy Switch Graph

Intuitionistic fuzzy switch graphs are introduced as a structure that comprises features of both switch graphs and intuitionistic fuzzy graphs, by combining the features of both structures using t-norm operators as parameter for updating membership and nonmembership functions.

Switch graphs consider the possibility of either temporarily removing an edge from a graph or restoring an edge. Since intuitionistic fuzzy graphs consider membership and nonmembership functions, the structure resulting from combining these frameworks admits that these functions may be updated. An edge being removed corresponds to its corresponding value in the membership function being decreased and, simultaneously, its corresponding value in the nonmembership function being increased. For an edge being restored, the opposite rationale is used.

**Definition 5.** *An* intuitionistic fuzzy switch graph *is a tuple $(V, S, \mu, \gamma, I)$ such that $V$ is a non-empty set of vertices (or states), $\mu : S_0 \to [0,1]$ is the degree of membership and $\gamma : S_0 \to [0,1]$ is the degree of nonmembership satisfying $\forall\ s \in S_0,\ \mu(s) + \gamma(s) \leqslant 1$ and $I : S \backslash S_0 \to \{on, off\}$, where $S$ is defined recursively as:*

- $S_0 \subseteq V \times V$
- $S_{n+1} \subseteq S_0 \times S_n \times \{\bullet, \circ\}$
- $S = \bigcup_{i \geqslant 0} S_i$

*where we impose* $\forall s, t \in S,\ (s, t, \circ) \in S \Rightarrow (s, t, \bullet) \notin S$.

*Finally, the set* $E = \{e \in S_0 : \mu(e) > 0 \text{ or } \gamma(e) < 1\}$ *is the edges.*

The difference between the structure of regular intuitionistic fuzzy graphs and intuitionistic fuzzy switch graphs is the higher-level edges, together with the *instantiation* function $I$. Semantically, higher-level edges govern the reconfiguration of the model, being triggered when the corresponding source arrow is crossed and affecting the pointed edge. Higher-level edges may point to other higher-level edges and, because of this, the instantiation function indicates how higher-levels can themselves be temporarily removed from the model or restored. Thus, for every $s \in S \backslash S_0$, $s$ is active if $I(s) = \text{on}$ while $I(s) = \text{off}$ indicates that $s$ is temporarily removed from the model and will not induce any change in the model. Also higher level edges whose third component is $\bullet$ are said to be activators while higher-level edges whose third component is $\circ$ are said to be inhibitors, as their role in the classical switch graphs is to respectively restore or remove the edge they target.

Note that $S$ will be used to denote both the set of edges and an $S$-norm. Nevertheless, the context will clarify whether it is being used to denote a set (of edges) or a function.

*Example 1.* Figure 1 illustrates one example of intuitionistic fuzzy switch graphs such that $V = \{x, y, z\}$, $S = \{(x, y), (y, z), (x, z), \big((x, y), (x, z), \bullet\big), \big((y, z), (x, z), \circ\big), \big((x, y), \big((y, z), (x, z), \circ\big), \circ\big)\}$, $I(s) = \text{on}$ for every $s \in S \backslash S_0$ and $(\mu(x, y), \gamma(x, y)) = (0.8, 0.2)$, $(\mu(y, z), \gamma(y, z)) = (0.6, 0)$ and $(\mu(x, z), \gamma(x, z)) = (0.2, 0.6)$.

The figure distinguishes between $S \backslash S_0$ edges whose third component is $\bullet$ and $\circ$ using a full or a hollow arrow head. Moreover, temporarily removed edges will be displayed with dashed lines.

Let $G = (V, S, \mu, \gamma, I)$ be an intuitionistic fuzzy switch graph. The reconfiguration of $G$ through an edge $s = (v, w)$ and a t-norm $T$ is $G^s_T = (V, S, \mu^s_T, \gamma^s_T, I^s)$ where $\mu^s_T : S_0 \to [0, 1],\ \gamma^s_T : S_0 \to [0, 1]$ are defined as:

$$\mu^s_T(t) = \begin{cases} T(\mu(s), \mu(t)), & \text{if } (s, t, \circ) \in S \text{ and } I((s, t, \circ)) = \text{on} \\ S\big(\mu(s), \mu(t)\big), & \text{if } (s, t, \bullet) \in S \text{ and } I((s, t, \bullet)) = \text{on} \\ \mu(t), & \textit{otherwise.} \end{cases}$$

$$\gamma^s_T(t) = \begin{cases} S\big(\gamma(s), \gamma(t)\big), & \text{if } (s, t, \circ) \in S \text{ and } I((s, t, \circ)) = \text{on} \\ T\big(\gamma(s), \gamma(t)\big), & \text{if } (s, t, \bullet) \in S \text{ and } I((s, t, \bullet)) = \text{on} \\ \gamma(t), & \textit{otherwise.} \end{cases}$$

and $I^s : S \backslash S_0 \to \{\text{on}, \text{off}\}$ is defined as:

$$I^s(t) = \begin{cases} \text{off}, & \text{if } (s,t,\circ) \in S \text{ with } I((s,t,\circ)) = \text{on} \\ \text{on}, & \text{if } (s,t,\bullet) \in S \text{ with } I((s,t,\bullet)) = \text{on} \\ I(t), & \textit{otherwise.} \end{cases}$$

At this point, we note that the condition $\forall s,t \in S$, $(s,t,\circ) \in S \Rightarrow (s,t,\bullet) \notin S$ in Definition 5 is only imposed for coherence, in order to guarantee that an activator and an inhibitor edge do not update the same edge simultaneously. We note that if one does not want to include this restriction, it is possible to consider a "preference" rule such as, for instance, if both $(s,t,\circ)$, $(s,t,\bullet) \in S$ and $I(s,t,\circ) = I(s,t,\bullet) =$on, $I(s,t,\circ)$ is only triggered if $I(t) =$on and $I(s,t,\bullet)$ is only triggered if $I(t) =$off. However, we do not explore this option further on this paper.

**Proposition 2.** *Let $G = (V,S,\mu,\gamma,I)$ be an intuitionistic fuzzy switch graph. The reconfiguration $G^s_T = (V,S,\mu^s_T,\gamma^s_T,I^s)$ of $G$ through an edge $s = (v,w)$ and a t-norm $T$ is an intuitionistic fuzzy switch graph.*

*Proof.* Since $V$ and $S$ do not change, we simply need to prove that, for every $t \in S_0$, $\mu^s_T(t)+\gamma^s_T(t) \leqslant 1$. Note that while $I^s$ may be distinct from $I$, this function still is defined from $S \backslash S_0$ to {on, off}.

By definition, since $\mu(t) + \gamma(t) \leqslant 1 \Rightarrow \mu(t) \leqslant 1 - \gamma(t)$ for every $t \in S_0$ and $T$ is monotonic, then $T(\mu(s),\mu(t)) \leqslant T(1-\gamma(s),1-\gamma(t))$. Analogously, $T(\gamma(s),\gamma(t)) \leqslant T(1-\mu(s),1-\mu(t))$.

If $(s,t,\circ) \in S$ and $I(s,t,\circ) =$on:
$\mu^s_T(t) + \gamma^s_T(t) = \overline{T(\mu(s),\mu(t)) + S}(\gamma(s),\gamma(t))$
$= T(\mu(s),\mu(t)) + 1 - T(1-\gamma(s),1-\gamma(t))$
$\leqslant T(1-\gamma(s),1-\gamma(t)) + 1 - T(1-\gamma(s),1-\gamma(t))$
$= 1$

If $(s,t,\bullet) \in S$ and $I(s,t,\bullet) =$on:
$\mu^s_T(t) + \gamma^s_T(t) = \overline{S(\mu(s),\mu(t)) + T}(\gamma(s),\gamma(t))$
$= 1 - T(1-\mu(s),1-\mu(t)) + T(\gamma(s),\gamma(t))$
$\leqslant T(1-\mu(s),1-\mu(t)) + 1 - T(1-\mu(s),1-\mu(t))$
$= 1$

Finally, if $(s,t,\bullet), (s,t,\circ) \notin S$ or $I(s,t,\circ) = I(s,t,\bullet) =$off, we trivially conclude $\mu^s_T(t) + \gamma^s_T(t) = \mu(t) + \gamma(t) \leqslant 1$.

Since the reconfiguration of an intuitionistic fuzzy switch graph still is an intuitionistic fuzzy switch graph, the set of edges $E^s_T = \{e \in S_0 : \mu^s_T(e) > 0 \text{ or } \gamma^s_T(e) < 1\}$ of $G^s_T$ is well defined.

Thus, we define a path (of length $n$) as a tuple $(e_1,...,e_n)$ such that $e_1 \in E$, $e^2 \in E^{e_1}$, ..., $\left((...(E^{e_1})^{\cdots})^{e_{n-1}}\right)$ where $e_i = (s_i,t_i)$ and $e_{i+1} = (s_{i+1},t_{i+1})$ are such that $t_i = s_{i+1}$ for every $i < n$.

*Example 2.* Consider the intuitionistic fuzzy switch graph shown in Figure 1 and the Gödel t-norm, defined as $T(a,b) = min(a,b)$, for $a,b \in [0,1]$. The update of

this intuitionistic fuzzy switch graph after traversing the edge $(x, y)$ is shown in Figure 2.

In this case, $\big((x, y), (x, z), \bullet\big)$ is triggered – changing the value of the pair $(\mu(x, z), \gamma(x, z))$ – as well as $\big((x, y), \big((y, z), (x, z), \circ\big), \circ\big)$ – that temporarily removed the edge $\big((y, z), (x, z), \circ\big)$ from the graph.

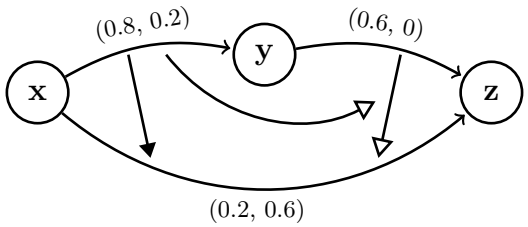



**Fig. 1.** Example of an Intuitionistic Fuzzy Switch Graph.

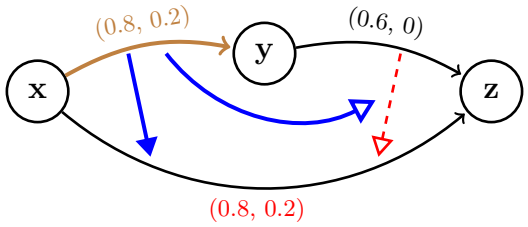



**Fig. 2.** Reconfiguration of the Intuitionistic Fuzzy Switch Graph shown in Figure 1.

## 4 Intuitionistic Reversal Fuzzy Switch Graph

The structures introduced in the previous section are able to combine the nature of switch graphs and intuitionistic fuzzy graphs, following [4], where reversal fuzzy switch graphs are also presented as a fuzzy version of switch graphs in a fuzzy setting.

Thus, based on the notion of reversal fuzzy switch graphs and intuitionistic fuzzy graphs, we propose a definition for intuitionistic reversal fuzzy switch graphs. These structures are based on the ones presented in the previous section but consider a more general class of fuzzy operators to parameterize structure update (aggregation functions in the place of t-norms) and, more distinguishably, consider the possibility of fully removing an $S_0$ edge from a model temporarily (not only changing its membership and nonmembership functions value). These

edges cannot be crossed but can be later restored, as defined for regular switch graphs.

This section details how these structures are formally defined.

**Definition 6.** *An* intuitionistic reversal fuzzy switch graph *is a tuple* $(V, S, \mu, \gamma, I)$ *such that* $V$ *is a non-empty set of vertices (or states),* $I : S \to \{on, off\}$*,* $\mu : S \to [0,1]$ *is the degree of membership and* $\gamma : S \to [0,1]$ *is the degree of nonmembership satisfying* $\forall\ s \in S,\ \mu(s) + \gamma(s) \leqslant 1$*, where* $S$ *is defined recursively as:*

- $S_0 \subseteq V \times V$
- $S_{n+1} \subseteq S_0 \times S_n \times \{\bullet, \circ\}$
- $S = \bigcup_{i \geqslant 0} S_i$

*where we impose* $\forall s, t \in S,\ (s, t, \circ) \in S \Rightarrow (s, t, \bullet) \notin S$*.*

*Finally, we define the set of edges of an intuitionistic reversal fuzzy switch graph* $E \subseteq S_0$ *as* $E = \{e \in S_0 : \mu(e) > 0 \text{ or } \gamma(e) < 1\}$ *and* $E^* = \{e \in S_0 : I(e) = on\}$*.*

Again, the notions of activator and inhibitor edges as well as of activating or temporarily removing an edge are analogous to the ones defined in the previous section.

Let $G = (V, S, \mu, \gamma, I)$ be an intuitionistic reversal fuzzy switch graph. The reconfiguration of $G$ through an edge $s = (v, w)$ and a conjunctive aggregation $A$ is $G_A^s = (V, S, \mu_A^s, \gamma_A^s, I^s)$ where $\mu_A^s : S \to [0,1]$, $\gamma_A^s : S \to [0,1]$ are defined as:

$$\mu_A^s(t) = \begin{cases} A(\mu(s), \mu(s,t,\circ), \mu(t)), & \text{if } (s,t,\circ) \in S \text{ and } I((s,t,\circ)) = \text{on} \\ \bar{A}\big(\mu(s), \mu(s,t,\bullet), \mu(t)\big), & \text{if } (s,t,\bullet) \in S \text{ and } I((s,t,\bullet)) = \text{on} \\ \mu(t), & otherwise. \end{cases}$$

$$\gamma_A^s(t) = \begin{cases} \bar{A}\big(\gamma(s), \gamma(s,t,\circ), \gamma(t)\big), & \text{if } (s,t,\circ) \in S \text{ and } I((s,t,\circ)) = \text{on} \\ A\big(\gamma(s), \gamma(s,t,\bullet), \gamma(t)\big), & \text{if } (s,t,\bullet) \in S \text{ and } I((s,t,\bullet)) = \text{on} \\ \gamma(t), & otherwise. \end{cases}$$

and $I^s : S \to \{\text{on}, \text{off}\}$ is defined as:

$$I^s(t) = \begin{cases} \text{off}, & \text{if } (s,t,\circ) \in S \text{ with } I((s,t,\circ)) = \text{on} \\ \text{on}, & \text{if } (s,t,\bullet) \in S \text{ with } I((s,t,\bullet)) = \text{on} \\ I(t), & otherwise. \end{cases}$$

An edge $s \in S_0$ is said to be active if $I(s) = \text{on}$ and inactive (or temporarily removed) if $I(s) = \text{off}$. Again, as in previous section, we do not explore scenarios without the imposed condition $\forall s, t \in S,\ (s,t,\circ) \in S \Rightarrow (s,t,\bullet) \notin S$ on this paper.

**Proposition 3.** *Let $G = (V, S, \mu, \gamma, I)$ be an intuitionistic reversal fuzzy switch graph. The reconfiguration $G^s_A = (V, S, \mu^s_A, \gamma^s_A, I^s)$ of $G$ through an edge $s = (v, w)$ and a conjunctive aggregation $A$ is an intuitionistic reversal fuzzy switch graph.*

*Proof.* The proof follows the same steps as the one for intuitionistic fuzzy switch graph. Although we are considering $n$-ary conjunctive aggregation functions instead of t-norms, this does not impact the reasoning as the only required condition is the monotonicity.

Because of this, the set of edges of $G^s_A$ is also well-defined and we denote it by $E^s_A$. We denote by $(E^s_A)^*$ the set $\{e \in E^s_A : I^s(e) = \text{on}\}$, defined as the active edges of $G^s_A$.

**Definition 7.** *Let $(V, S, \mu, \gamma, I)$ be an intuitionistic reversal fuzzy switch graph. A path (of length $n$) is a tuple $(e_1, ..., e_n)$ such that $e_1 \in E^*$, $e_2 \in (E^{e_1}_A)^*$, ..., $e_n \in \left(\left(...(E^{e_1}_A)^{\cdots}_{\cdots}\right)^{e_{n-1}}_A\right)^*$, where $e_i = (s_i, t_i)$ and $e_{i+1} = (s_{i+1}, t_{i+1})$ are such that $t_i = s_{i+1}$ for every $i < n$.*

*Example 3.* Consider the intuitionistic reversal fuzzy switch graph illustrated in Figure 3, that is based on the intuitionistic fuzzy switch graph shown in Figure 1 but with the following additional specifications:

$(\mu\big((x, y), (x, z), \bullet\big), \gamma\big((x, y), (x, z), \bullet\big)) = (0.4, 0.6)$,
$(\mu\big((y, z), (x, z), \circ\big), \gamma\big((y, z), (x, z), \circ\big)) = (0.5, 0.5)$ and
$(\mu\big((x, y), \big((y, z), (x, z), \circ\big), \circ\big), \gamma\big((x, y), \big((y, z), (x, z), \circ\big), \circ\big)) = (0.1, 0.1)$.
Moreover, $I(s) = \text{on}$ for every $s \in S$.

The update of this intuitionistic reversal fuzzy switch graph after traversing the edge $(y, z)$ considering the aggregation $A(a, b, c) = b \times \min(a, c)$ is shown in Figure 4. When $\big((y, z), (x, z), \circ\big)$ is triggered, the value of the pair $(\mu(x, z), \gamma(x, z))$ is updated and the edge $(x, z)$ is also temporarily removed from the graph.

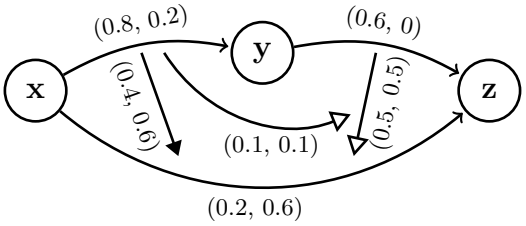



**Fig. 3.** Example of Intuitionistic Reversal Fuzzy Switch Graph.

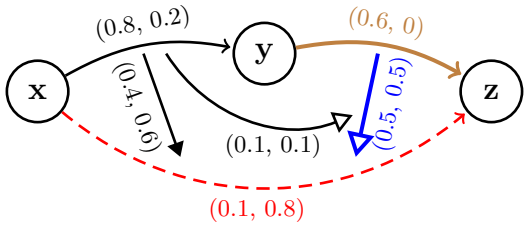



**Fig. 4.** Reconfiguration of the Intuitionistic Reversal Fuzzy Switch Graph shown in Figure 3.

## 5 Discussion and future work

This paper introduces a first approach to relation-changing formalisms in the context of intuitionistic fuzzy graphs. To do this, this work introduces relation-changing intuitionistic fuzzy graphs, which are obtained by enriching intuitionistic fuzzy graphs with higher-level edges. While these structures are designed taking into account the concept of switch graph, another step is even taken further, by connecting these structures to reversal fuzzy switch graphs.

The paper shows that both structures introduced are well-defined and compatible with the relation-changing updates, highlighting the basic differences resulting from the different definitions, and providing some examples.

In the future, we intend to theoretically study these structures and explore their dynamics with more detail. One goal would be to understand conditions for obtaining behavioral equivalences. For that we can make use of the notion of path-equivalence, developed for other relation-changing structures [16]. A formal modal language can also be explored in order to reason about systems modeled through such structures.

### Acknowledgment

This work is supported by FCT – Fundação para a Ciência e a Tecnologia through projects UIDB/04106/2025 at CIDMA and by National and European Funds through SACCCT- IC&DT - Sistema de Apoio à Criação de Conhecimento Científico e Tecnológico, as part of COMPETE2030, within the project BANKSY with reference COMPETE2030-FEDER-00892000.

## References

1. R. Parvathi and M. G. Karunambigai. Intuitionistic fuzzy graphs. In Bernd Reusch, editor, *Computational Intelligence, Theory and Applications*, pages 139–150, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.
2. Krassimir T. Atanassov. Intuitionistic fuzzy sets. *Fuzzy Sets and Systems*, 20(1):87–96, 1986.

3. John N Mordeson and Premchand S Nair. Fuzzy graphs. In *Fuzzy Graphs and Fuzzy Hypergraphs*, pages 19–81. Springer, 2000.
4. Suene Campos, Regivan Santiago, Manuel A. Martins, and Daniel Figueiredo. Reversal fuzzy switch graphs. In Gustavo Carvalho and Volker Stolz, editors, *Formal Methods: Foundations and Applications*, pages 137–154, Cham, 2020. Springer International Publishing.
5. Suene Campos, Regivan Santiago, Manuel A Martins, and Daniel Figueiredo. Introduction to reversal fuzzy switch graph. *Science of Computer Programming*, 216:102776, 2022.
6. Alexandre Madeira, José M Faria, Manuel A Martins, and Luís S Barbosa. Hybrid specification of reactive systems: An institutional approach. In *International Conference on Software Engineering and Formal Methods*, pages 269–285. Springer, 2011.
7. Daniel Figueiredo, Manuel A Martins, and Madalena Chaves. Applying differential dynamic logic to reconfigurable biological networks. *Mathematical Biosciences*, 291:10–20, 2017.
8. Carlos Areces, Facundo Carreiro, Santiago Figueira, and Sergio Mera. Basic model theory for memory logics. In *International Workshop on Logic, Language, Information, and Computation*, pages 20–34. Springer, 2011.
9. Daniel Figueiredo and Luís Soares Barbosa. Reactive models for biological regulatory networks. In Madalena Chaves and Manuel A. Martins, editors, *Molecular Logic and Computational Synthetic Biology*, pages 74–88, Cham, 2019. Springer International Publishing.
10. Mário Benevides, Alexandre Madeira, and Manuel A Martins. Graded epistemic logic with public announcement. *Journal of Logical and Algebraic Methods in Programming*, 125:100732, 2022.
11. Raul Fervari, Daniel Figueiredo, and Manuel A. Martins. Graded relation updates in modal logic. In Dexter Kozen and Ruy de Queiroz, editors, *Logic, Language, Information, and Computation*, pages 278–292, Cham, 2026. Springer Nature Switzerland.
12. Diana Costa, Daniel Figueiredo, and Manuel A Martins. Relation-changing models meet paraconsistency. *Journal of Logical and Algebraic Methods in Programming*, 133:100870, 2023.
13. Dov Gabbay and Sérgio Marcelino. Global view on reactivity: switch graphs and their logics. *Annals of Mathematics and Artificial Intelligence*, 66(1-4):131–162, 2012.
14. Erich Peter Klement, Radko Mesiar, and Endre Pap. *Triangular norms*, volume 8. Springer Science & Business Media, 2013.
15. Gleb Beliakov, Ana Pradera, Tomasa Calvo, et al. *Aggregation functions: A guide for practitioners*, volume 221. Springer, 2007.
16. Dov M Gabbay and Sérgio Marcelino. Modal logics of reactive frames. *Studia Logica*, 93(2):405–446, 2009.

# Weak algebraizability for logics of indicative conditionals

Miguel Muñoz Pérez[0009−0001−3468−8208]

Departamento de Lógica, Historia y Filosofía de la Ciencia, UNED
Madrid, Spain
mupemiguel99@gmail.com

**Abstract.** We consider weak versions of some trivalent logics of indicative conditionals, obtained by restricting atomic assignments to the classical truth-values, in the context of the formal framework recently developed by Nakov and Quadrellaro. We analyze how the original structural logics can be recovered from their weak counterparts and, additionally, study which of the three algebraizability notions available for weak logics hold for our new systems.

**Keywords:** Indicative conditionals · Three-valued logics · Algebraizability · Twist-structures · Algebraic semantics

## 1 Introduction

In this note we wish to present some ongoing work continuing the study from the series of papers [13, 21, 17, 14, 16, 18], which have been concerned with the study of several algebraic and axiomatic features of some three-valued logics of indicative conditionals, namely, those attributed in the literature to De Finetti (see, e.g., [11, 6, 7]), Cooper (see, e.g., [5, 21]), Farrell (see, e.g., [10, 9, 17]) and Cantwell (see, e.g., [3, 15, 20]). In particular, our main aim is to analyze how several generalizations of Tarskian logics and algebraizability, introduced in [19], apply to the aforementioned systems.

Three-valued logics of indicative conditionals have been vindicated in [6, 7, 22, 8, 23] as a very appealing approach for studying the semantics of conditional statements, both philosophically and mathematically. The trivalent approach not only provides truth-functional truth conditions for these statements, but also seems to smoothly fit with the probabilistic considerations behind the traditional analysis assertability conditions (see, e.g., [8]). On the mathematical side, since these logics are defined as induced by logical matrices, the usual tools of algebraic logic can be deployed in order to settle their behavior. Define $A_3 := \{\mathbf{0}, \top, \mathbf{1}\}$ and $D := \{\top, \mathbf{1}\}$; the aforementioned logics are:

1. *De Finetti's logic* DF, the logic induced by the matrix $(\mathbf{DF_3}, D)$, where $\mathbf{DF_3} := (A_3; \neg, \wedge_{\mathsf{K}}, \vee_{\mathsf{K}}, \rightarrow_{\mathsf{DF}})$;
2. The structural version of *Cooper's logic of ordinary language*, the logic sOL induced by the matrix $(\mathbf{sOL_3}, D)$, where $\mathbf{sOL_3} := (A_3; \neg, \wedge_{\mathsf{OL}}, \vee_{\mathsf{OL}}, \rightarrow_{\mathsf{OL}})$;
3. *Farrell's logic* F, the logic induced by the matrix $(\mathbf{F_3}, D)$, where one sets $\mathbf{F_3} := (A_3; \neg, \wedge_{\mathsf{K}}, \vee_{\mathsf{K}}, \rightarrow_{\mathsf{F}})$;

4. *Cantwell's logic of conditional negation* CN, the logic induced by the matrix $(\mathbf{CN_3}, D)$, where $\mathbf{CN_3} := (A_3; \neg, \wedge_{\mathsf{K}}, \vee_{\mathsf{K}}, \rightarrow_{\mathsf{OL}})$;

where the operations are defined in Fig. 1. We already know that DF fails to be protoalgebraic (and hence algebraizable) while being truth-equational (see, e.g., [17]), that sOL is algebraizable with respect to the class of OL-algebras (see, e.g., [21, 17]) and that both F and CN are algebraizable with respect to the class of, respectively, F and CN-algebras (see, e.g., [17, 18]).

| $\wedge_{\mathsf{OL}}$ | **0** | $\top$ | **1** |
|---|---|---|---|
| **0** | **0** | **0** | **0** |
| $\top$ | **0** | $\top$ | **1** |
| **1** | **0** | **1** | **1** |

| $\vee_{\mathsf{OL}}$ | **0** | $\top$ | **1** |
|---|---|---|---|
| **0** | **0** | **0** | **1** |
| $\top$ | **0** | $\top$ | **1** |
| **1** | **1** | **1** | **1** |

| $\wedge_{\mathsf{K}}$ | **0** | $\top$ | **1** |
|---|---|---|---|
| **0** | **0** | **0** | **0** |
| $\top$ | **0** | $\top$ | $\top$ |
| **1** | **0** | $\top$ | **1** |

| $\vee_{\mathsf{K}}$ | **0** | $\top$ | **1** |
|---|---|---|---|
| **0** | **0** | $\top$ | **1** |
| $\top$ | $\top$ | $\top$ | **1** |
| **1** | **1** | **1** | **1** |

| | $\neg$ |
|---|---|
| **0** | **1** |
| $\top$ | $\top$ |
| **1** | **0** |

| $\rightarrow_{\mathsf{OL}}$ | **0** | $\top$ | **1** |
|---|---|---|---|
| **0** | $\top$ | $\top$ | $\top$ |
| $\top$ | **0** | $\top$ | **1** |
| **1** | **0** | $\top$ | **1** |

| $\rightarrow_{\mathsf{DF}}$ | **0** | $\top$ | **1** |
|---|---|---|---|
| **0** | $\top$ | $\top$ | $\top$ |
| $\top$ | $\top$ | $\top$ | $\top$ |
| **1** | **0** | $\top$ | **1** |

| $\rightarrow_{\mathsf{F}}$ | **0** | $\top$ | **1** |
|---|---|---|---|
| **0** | $\top$ | $\top$ | $\top$ |
| $\top$ | **0** | $\top$ | $\top$ |
| **1** | **0** | $\top$ | **1** |

**Fig. 1.** Three-valued connectives.

These logics are instances of the more general notion of logic usually studied in algebraic logic (see, e.g., [1, 12]). Such notion is commonly taken to include the condition of *structurality*: in case we have a valid inference $\Gamma \vdash \varphi$, we immediately obtain $\sigma(\Gamma) \vdash \sigma(\varphi)$, for any substitution $\sigma$. However, Cooper's original proposal was defined as induced by the matrix above but restricting the evaluation of atomic formulas to the classical values **0** and **1** (see, e.g., [5, 6, 21]); the resulting valuations have been called *Cooper's evaluations* in [6, 8]. The idea behind this rationale is that the truth-value $\top$ codifying semantic gaps should enter the picture exclusively by the application of, say, conditional statements in which the antecedent is false. As a consequence, full structurality fails in OL.

This seems to be also consistent with our remaining logics. For example, against Reichenbach, De Finetti defends that the extra truth-value $\top$ only belongs to propositions *subjectively*, that a proposition is objectively either true or false (see [11, §7,8]). In a similar fashion, Farrell's intended interpretation of $\top$ as *being inappropriate* explicitly implies 'that serious discussion should eschew consideration of atomic propositions which are known to be inappropriate, and hence [$\top$] does not figure into the combinations of values for the atomic propositions' (see [10, §1]). In general, the philosophical analysis of indicative conditionals, be it or not through trivalent semantics, has traditionally distinguished the problem of providing an account of indicative conditionals of atomic propositions from that of *nested* conditionals; both tasks seem to be generally taken as radically different in nature (see, e.g., [4]).

## 2 Technical preliminaries

Let us present all the notions from [19] that we will use later on. We will assume some familiarity with the standard setting of algebraizable logics. For all the unexplained notions or more details see, e.g., [1, 12, 2, 19]. As a matter of convenience, we will use different names for some concepts from [19].

### 2.1 Weak logics

Given an algebraic language $\mathcal{L}$, we will denote the set of atomic formulas by Var and the term-algebra by $\mathrm{Fm}_{\mathcal{L}}$, dropping the reference to the language when the context is clear. An *atomic substitution* is a substitution sending atomic formulas to atomic formulas (see [19, Def.2.1]). A *weak logic* is a finitary standard logic in which the structurality condition is required to hold only for atomic substitutions (see [19, Defs.1.2.,2.2])[1]. In general, a given substitution $\sigma$ will be *admissible* for a weak logic $\vdash$ if $\Gamma \vdash \varphi$ implies that $\sigma[\Gamma] \vdash \sigma(\varphi)$; by definition, all atomic substitutions are admissible (see [19, Def.2.6]). Alternatively, given a weak logic $\vdash$ one can look at the terms $\psi$ such that $\Gamma \vdash \varphi$ implies that $\Gamma[x/\psi] \vdash \varphi[x/\psi]$; the set of all these is called the *core* of $\vdash$ and, by definition, it contains Var (see [19, Def.2.8]).

Since we are concerned with matrix-induced logics, it seems natural to ask how these can be generalized in the setting of weak logics. By *expanded algebra* we will understand a structure $(\mathbf{A}, C)$ formed by an algebra $\mathbf{A}$ and a unary predicate symbol $C$; the interpretation of $C$ will be called the *core* of the expanded algebra and we will denote it by $core(\mathbf{A})$ (see [19, Def.2.10]). The usual model-theoretic definitions apply to this case (see [19, Def.1.8]): given two expanded algebras $(\mathbf{A}, C)$ and $(\mathbf{B}, C)$, a *homomorphism* $h : (\mathbf{A}, C) \to (\mathbf{B}, C)$ is an algebraic homomorphism $h : \mathbf{A} \to \mathbf{B}$ such that $h(core(\mathbf{A})) \subseteq core(B)$; such homomorphism is an *embedding* if it is injective and $h(a) \in core(\mathbf{B})$ implies $a \in core(\mathbf{A})$; $(\mathbf{A}, C)$ is a *substructure* of $(\mathbf{B}, C)$ if the identity map $id : A \to B$ is an embedding; finally, when $(\mathbf{A}, C)$ and $(\mathbf{B}, C)$ are embeddable into each other we will say that they are *isomorphic*. In a similar way, one can define the usual class-operators $\mathbf{S}, \mathbf{H}, \mathbf{I}$ and $\mathbf{P}$ (see [19, 1.10])[2] and, consequently, the *(expanded) variety* and *quasivariety* generated by a class of structures $\mathbf{K}$, denoted, respectively, by $\mathbf{V}(\mathbf{K})$ and $\mathbf{Q}(\mathbf{K})$ (see [19, Def.1.11]).

Let us see how we can define the weak counterpart of matrix-induced logics. The *expanded term-algebra* has algebraic reduct $\mathrm{Fm}_{\mathcal{L}}$ and $core(\mathrm{Fm}_{\mathcal{L}}) := \mathrm{Var}$; hence, an $(\mathbf{A}, C)$-*valuation* is an $\mathbf{A}$-valuation $h : \mathrm{Fm}_{\mathcal{L}} \to \mathbf{A}$ such that $h(\mathrm{Var}) \subseteq core(\mathbf{A})$ (see [19, Def.2.15]). By *expanded matrix* we will simply understand a tuple $(\mathbf{A}, C, F)$, where $(\mathbf{A}, C)$ is an expanded algebra and $F \subseteq A$ (see [19, Def.6.4]). Given a class $\mathbf{K}$ of similar expanded matrices, we define the *consequence* $\vdash$ *induced by* $\mathbf{K}$ as follows: $\Gamma \vdash \varphi$ if and only if for every $(\mathbf{A}, C, F) \in \mathbf{K}$ and every $(\mathbf{A}, C)$-valuation $h$ it holds that $h[\Gamma] \subseteq F$

[1] Note that, since our target logics are all finitary by definition, we do not have to explicitly deal with the restriction to the finitary case.

[2] Here we have to observe that, in [19], $\mathbf{H}$ is taken to mean the closure under *strong* homomorphisms, that is, those verifying that $h[core(\mathbf{A})] = core(\mathbf{B})$. On the reasons for this choice, see [19, Rmk.1.9].

implies $h(\varphi) \in F$ (see [19, Def.6.6]). It is proven in [19, Prop.6.8] that the finitary companion of the consequence relation induced by a class of expanded matrices is a weak logic.

Finally, given a weak logic $\vdash$, we can recover the largest standard logic contained in it; we will call the *schematic fragment* of $\vdash$ the standard logic $\vdash_s$ defined by: $\Gamma \vdash_s \varphi$ if and only if, for every substitution $\sigma$, $\sigma[\Gamma] \vdash \sigma(\varphi)$ (see [19, Def.4.1]). Given a set $X$ of formulas, denote by $\mathrm{At}[X]$ its closure under atomic substitutions; we will say that a weak logic $\vdash$ is *(resp. finitely) representable* if there is (resp. finite) a set of formulas in one variable $\Sigma(x)$ such that $\Gamma \vdash \varphi$ if and only if $\Gamma \cup \mathrm{At}[\Sigma(x)] \vdash_s \varphi$ (see [19, Def.4.3]).

### 2.2 Weak algebraizability

Recall that a standard logic $\vdash$ is *algebraizable* in case there is a tuple $(\mathbf{Q}, \tau, \Delta)$, where $\mathbf{Q}$ is a quasivariety and $\tau, \Delta$ are *structural transformers*, i.e. substitution-commuting maps from formulas to equations and from equations to formulas, respectively (see, e.g., [19, Def.3.1]), verifying: (i) $\Gamma \vdash \varphi$ iff $\tau(\Gamma) \vdash_{\mathbf{Q}} \tau(\varphi)$ and (ii) $e \dashv\vdash_{\mathbf{Q}} \tau[\Delta(e)]$, where $\vdash_{\mathbf{Q}}$ is the equational consequence of $\mathbf{Q}$ (see, e.g., [19, Def.3.2; 3.4]). Let us briefly introduce the three counterparts for standard algebraizability presented in [19]. In what follows, we will denote equations by $e, f, \ldots$ and their corresponding components (seen as pairs of terms) as $e_0, e_1, f_0, f_1$ and so on.

To begin with, we need to define the equational consequence relation in this context. Given a class $\mathbf{K}$ of expanded $\mathcal{L}$-algebras and $E \cup \{e\}$ a set of $\mathcal{L}$-equations, the *equational consequence of* $\mathbf{K}$ is defined as: $E \vdash_{\mathbf{K}} e$ if and only if for every $(\mathbf{A}, C) \in \mathbf{K}$ and every $(\mathbf{A}, C)$-valuation $h$, if $h(f_0) = h(f_1)$ for every $f \in E$ then $h(e_0) = h(e_1)$ (see [19, Def.2.17]). We will also need to restrict ourselves to a special kind of expanded algebras: an expanded algebra $(\mathbf{A}, C)$ is said to be *core-generated* if $\mathbf{A}$ is spanned by $core(\mathbf{A})$. Given a class $\mathbf{K}$ of expanded algebras, denote by $\mathbf{K}_{CG}$ the family of its core-generated elements; if $\mathbf{K}$ is a variety (resp. a quasivariety) then it is said to be *core-generated* if $\mathbf{K} = \mathbf{V}(\mathbf{K}_{CG})$ (resp. if $\mathbf{K} = \mathbf{Q}(\mathbf{K}_{CG})$) (see [19, Def.2.19]). We can then define (see [19, Def.3.6]):

**Definition 1.** *A weak logic $\vdash$ is* loosely algebraizable *if there is a core-generated expanded quasivariety $\mathbf{Q}$ and structural transformers $\tau, \Delta$ such that*

1. *$\Gamma \vdash \varphi$ if and only if $\tau(\Gamma) \vdash_{\mathbf{Q}} \tau(\varphi)$,*
2. *$\Delta(E) \vdash \Delta(e)$ if and only if $E \vdash_{\mathbf{Q}} e$,*
3. *$\varphi \dashv\vdash \Delta(\tau(\varphi))$,*
4. *$e \dashv\vdash_{\mathbf{Q}} \tau(\Delta(e))$.*

*In such case, we say that $\vdash$ is* loosely algebraized *by* $(\mathbf{Q}, \tau, \Delta)$.

Loose algebraizability enjoys the usual uniqueness properties (see [19, Thm.3.7]) and, in order to establish it, it is enough to do so with either conditions (1) and (4) or (2) and (3) (see [19, Prop.3.8]). The second generalized concept of algebraizability is the following (see [19, 3.16]):

**Definition 2.** *A weak logic $\vdash$ is* strictly algebraizable *if it is loosely algebraizable by $(\mathbf{Q}, \tau, \Delta)$ and, moreover, there is a finite set $\Sigma(x)$ of equations in one variable such that,*

*for every* $(\mathbf{A}, C) \in \mathbf{Q}$*, it holds that* $core(\mathbf{A}) = \Sigma[A]$*, where* $\Sigma[A]$ *is the set of all the* $a \in A$ *such that* $e_0(a) = e_1(a)$*, for every* $e(x) \in \Sigma(x)$*. In such case, we say that* $\vdash$ *is* strictly algebraized *by* $(\mathbf{Q}, \tau, \Delta, \Sigma(x))$.

Both loose and strict algebraizability satisfy the usual uniqueness theorem (see [19, Thms.3.7,3.17]). The last definition of weak algebraizability that we will present is far more restrictive than the previous ones. Given a quasivariety of expanded $\mathcal{L}$-algebras $\mathbf{Q}$, any unary $\mathcal{L}$-term in one variable $\delta(x)$ is called a *selector term for* $\mathbf{Q}$ if $\vdash_{\mathbf{Q}} \delta(x) \approx \delta^2(x)$ (see [19, 3.20]). Then, define (see [19, Def.3.22]):

**Definition 3.** *A weak logic* $\vdash$ *is* fixpoint algebraizable *if it is strictly algebraized by* $(\mathbf{Q}, \tau, \Delta, \Sigma(x))$ *and, moreover,* $\Sigma(x) = \{x \approx \delta(x)\}$ *for some selector term* $\delta(x)$.

## 3 Weak logics of indicative conditionals

### 3.1 Generalizing the original logics

Recall that our intuitions behind the introduction of weak logics had to do with the fact that we wanted to restrict the domain to which atomic formulas are mapped, namely, the classical truth-values. Hence, for the algebras $\mathbf{DF_3}$, $\mathbf{OL_3}$, $\mathbf{F_3}$, $\mathbf{CN_3}$, their core should be $C := \{\mathbf{0}, \mathbf{1}\}$. Thus, define:

1. wDF as the weak logic induced by the expanded matrix $(\mathbf{DF_3}, C, D)$;
2. OL as the weak logic induced by the expanded matrix $(\mathbf{OL_3}, C, D)$;
3. wF as the weak logic induced by the expanded matrix $(\mathbf{F_3}, C, D)$;
4. wCN as the weak logic induced by the expanded matrix $(\mathbf{CN_3}, C, D)$.

It is immediate, by inspection of the table in Fig. 1, that the expanded algebras with algebraic reducts $\mathbf{DF_3}$, $\mathbf{OL_3}$, $\mathbf{F_3}$, $\mathbf{CN_3}$ and core $C$ are core-generated, for this amounts to being able to obtain $\top$ by means of $\mathbf{0}$ and $\mathbf{1}$ and the corresponding connectives. It is natural to ask for the exact relationship between the original standard logics and their weak counterparts. We can start by proving the following:

**Lemma 4.** *Let* $\vdash \in \{\text{wDF}, \text{OL}, \text{wF}, \text{wCN}\}$ *and denote by* $(\mathbf{A}, C, D)$ *its generating expanded matrix. Then, given an* $\mathbf{A}$*-valuation* $v$*, there is an* $(\mathbf{A}, C)$*-valuation* $h$ *and a substitution* $\sigma$ *such that* $v = h \circ \sigma$.

*Proof.* This is an instance of [19, Lem.2.20], which holds because each $(\mathbf{A}, C)$ is core-generated. ∎

**Theorem 5.** *The schematic fragments of* wDF*,* OL*,* wF *and* wCN *are, respectively,* DF*,* sOL*,* F *and* CN.

*Proof.* Let us denote $\vdash \in \{\text{wDF}, \text{OL}, \text{wF}, \text{wCN}\}$ and the corresponding standard logic by $\vdash' \in \{\text{DF}, \text{sOL}, \text{F}, \text{CN}\}$. Denote the expanded matrix $(\mathbf{A}, C, D)$.

Let us check that $\vdash' \subseteq \vdash$. Note that, if $\Gamma \vdash' \varphi$, we then have that for every $\mathbf{A}$-valuation $v$ it holds that $v[\Gamma] \subseteq D$ implies that $v(\varphi) \in D$. Then, $\Gamma \vdash \varphi$ holds if and only if for every

$(\mathbf{A}, C)$-valuation $h$ it holds that $h[\Gamma] \subseteq D$ implies $h(\varphi) \in D$. However, note that every $(\mathbf{A}, C)$-valuation is, by definition, an $\mathbf{A}$-valuation, so the result follows. Now, since $\vdash_s$ is, by definition, the largest structural logic contained in $\vdash$, we get that $\vdash' \subseteq \vdash_s$.

On the other hand, let us see that $\vdash_s \subseteq \vdash'$. Given an inference $\Gamma \vdash_s \varphi$, this means that, for every substitution $\sigma$, it holds that $\sigma(\Gamma) \vdash \sigma(\varphi)$. This, in turn, means that, for every $(\mathbf{A}, C)$-valuation $h$ and every substitution $\sigma$, $h(\sigma(\Gamma)) \subseteq D$ implies that $h(\sigma(\varphi)) \in D$. Now, we want to see that $\Gamma \vdash' \varphi$. Let $v$ be an arbitrary $\mathbf{A}$-valuation such that $v(\Gamma) \subseteq D$. By the Lemma above, there are an $(\mathbf{A}, C)$-valuation $h$ and a substitution $\sigma$ with $v = h \circ \sigma$. Hence, we have that $v(\Gamma) = h(\sigma(\Gamma)) \subseteq D$. By the preceding assumption, we deduce that $h(\sigma(\varphi)) = v(\varphi) \in D$, as desired.

Hence, we have seen that $\vdash_s = \vdash'$. ■

From here, we can already deduce the following negative result:

**Corollary 6.** wDF *is not loosely algebraizable. Hence, it is neither strictly nor fixpoint algebraizable.*

*Proof.* It is known [19, Prop.4.4] that if a weak logic $\vdash$ is loosely algebraized by $(\mathbf{Q}, \tau, \Delta)$ then $\vdash_s$ is algebraized by $(\mathbf{Q} \uparrow \mathcal{L}, \tau, \Delta)$, where the elements of $\mathbf{Q} \uparrow \mathcal{L}$ are the algebraic reducts of those of $\mathbf{Q}$. Hence, if wDF were loosely algebraizable, $\mathrm{wDF}_s$ would be algebraizable. However, since the previous Theorem tells us that $\mathrm{wDF}_s$ = DF, this would imply that DF is algebraizable, which we know that cannot be the case (see above and [17, Prop.1]). ■

It then remains to see what happens with the weak logics whose schematic fragment is algebraizable.

### 3.2 Core-definability

It seems sensible to first attack strict algebraizability since, in case we obtained a positive result for our remaining logics, we would get loose algebraizability for free. Our strategy will consist in using [19, Cor.4.6], which tells us that an $\mathcal{L}$-weak logic $\vdash$ is strictly algebraized by $(\mathbf{Q}, \Sigma, \tau, \Delta)$ if and only if $\vdash_s$ is algebraized by $(\mathbf{Q} \uparrow \mathcal{L}, \Delta, \tau)$ and $\vdash$ is representable by $\Delta(\Sigma)$. Since we know that sOL, F and CN are algebraizable with respect to the classes of OL, F and CN-algebras, respectively (see above and [21, 17]), these are the candidates for $\mathbf{Q} \uparrow \mathcal{L}$. However, these classes contain, in each case, the algebraic reduct of the corresponding generating matrix, i.e., $\mathbf{OL_3}$, $\mathbf{F_3}$ and $\mathbf{CN_3}$. The next observation shows that these algebras do not admit a definable core as required by strict algebraizability and that, as a consequence, the corresponding quasivarieties cannot be witnesses of such property:

**Lemma 7.** *Let* $\vdash \in \{\mathrm{wDF}, \mathrm{OL}, \mathrm{wF}, \mathrm{wCN}\}$ *and denote each corresponding generating matrix by* $(\mathbf{A}, C, D)$. *Then, $C$ is not definable in* $\mathbf{A}$ *by any finite set of equations in one variable.*

*Proof.* Otherwise, we would be able to find such a set $\Sigma(x)$ with only solutions $\mathbf{0}$ and $\mathbf{1}$. However, by structural induction plus checking the tables of the connectives one can see that every unary term $t(x)$ in the original signature $(\neg, \wedge, \vee, \rightarrow)$ verifies that $t(\top) = \top$

(indeed, the basic terms $\neg x$, $x \rightarrow x$, $x \wedge x$ and $x \vee x$ all yield $\top$ when evaluated in $\top$; the remaining cases follow easily from the induction hypothesis). Hence, for every $e(x) \in \Sigma(x)$ we would have that $e_0(\top) = \top = e_1(\top)$, so that $\top$ would also be a solution of $\Sigma(x)$, which is impossible. ■

Note that the preceding reasoning applies to selector terms as well.

**Theorem 8.** OL, wF *and* wCN *fail to be strictly algebraizable and, as a consequence, fixpoint algebraizable.*

*Proof.* Let us prove this for OL, the remaining cases are analogous. Suppose that OL is strictly algebraized by $(\mathbf{Q}, \Sigma(x), \tau, \Delta)$. Then, in particular, we know that $(\mathbf{Q} \upharpoonright \mathcal{L}, \tau, \Delta)$ is an algebraization of $\mathrm{OL}_s$, which coincides with sOL by the last Theorem. By the uniqueness of the algebraization of sOL (see, e.g., [19, 3.5]), we must have that $\mathbf{Q} \upharpoonright \mathcal{L}$ is the class of OL-algebras, which includes $\mathbf{OL_3}$. Hence, $\mathbf{Q}$ contains, in particular, the expanded algebra $(\mathbf{OL_3}, C)$. By assumption, we obtain that $C$ must be definable in $\mathbf{OL_3}$ by $\Sigma(x)$, which contradicts the preceding Lemma. Therefore, OL cannot be strictly algebraizable. ■

Hence, we cannot hope to obtain the loose algebraizability of OL, wF and wCN for free. However, we can still prove it straightforwardly:

**Proposition 9.** OL, wF *and* wCN *are loosely algebraizable.*

*Proof.* Let us check the case of $\vdash :=$ OL, since the others are similar. Consider the same transformers $\tau, \Delta$ as in the standard case (see above). Define $\mathbf{Q} := Q(\mathbf{OL_3}, C, D)$. Then, since $(\mathbf{OL_3}, C, D)$ is core-generated, we have that $\mathbf{Q}$ is core-generated[3]. On the other hand, note that $\vdash_{\{(\mathbf{OL_3}, C, D)\}} = \vdash_{\mathbf{Q}}$ is clear by definition. It only remains to check loose algebraizability and, moreover, it is enough to check conditions (1) and (4) above. The proof is essentially that of [17, Thm.6] taking under account that we are now dealing with $(\mathbf{OL_3}, C)$-valuations. Therefore, OL is loosely algebraizable with respect to $(\mathbf{Q}, \tau, \Delta)$. ■

### 3.3 Expanding the language

The failure of strict algebraizability seems to indicate that we should change the language: indeed, if we add the constant $\mathbf{1}$ to the original signature $(\neg, \wedge, \vee, \rightarrow)$, the set of equations $\Sigma(x) := \{x \vee \neg x \approx \mathbf{1}\}$ defines the core for the preceding three-element algebras[4]. Hence, it seems natural to define:

[3] This is a general fact: note that $\mathbf{Q} = Q(\mathbf{OL_3}, C, D) \subseteq Q(\mathbf{Q}_{CG}) \subseteq \mathbf{Q}$.

[4] Note that, in general, equational definability of the core can be obtained by appropriately expanding the language. Given an algebra $\mathbf{A}$ and a given element $b \in A$, we can expand its signature with the unary operation $f$ sending each $a$ to itself if $a \in core(A)$ and to $b$ otherwise, so that $\{x \approx f(x)\}$ defines $core(A)$. Despite this, one would still need to see what happens with algebraizability issues. Moreover, here we will expand the language minimally, i.e. by only adding a constant symbol.

1. sOL$^{\mathbf{1}}$, the logic induced by $(\mathbf{OL_3}^{\mathbf{1}}, D)$, where $\mathbf{OL_3}^{\mathbf{1}} := (A_3; \mathbf{1}, \neg, \wedge_{\mathsf{OL}}, \vee_{\mathsf{OL}}, \rightarrow_{\mathsf{OL}})$;
2. F$^{\mathbf{1}}$, the logic induced by $(\mathbf{F_3}^{\mathbf{1}}, D)$, where $\mathbf{F_3}^{\mathbf{1}} := (A_3; \mathbf{1}, \neg, \wedge_{\mathsf{K}}, \vee_{\mathsf{K}}, \rightarrow_{\mathsf{F}})$;
3. CN$^{\mathbf{1}}$, the logic induced by $(\mathbf{CN_3}^{\mathbf{1}}, D)$, where $\mathbf{CN_3}^{\mathbf{1}} := (A_3; \mathbf{1}, \neg, \wedge_{\mathsf{K}}, \vee_{\mathsf{K}}, \rightarrow_{\mathsf{OL}})$;

and, letting again $C := \{\mathbf{0}, \mathbf{1}\}$:

1. OL$^{\mathbf{1}}$, the weak logic induced by the expanded matrix $(\mathbf{OL_3}^{\mathbf{1}}, C, D)$;
2. wF$^{\mathbf{1}}$, the weak logic induced by the expanded matrix $(\mathbf{F_3}^{\mathbf{1}}, C, D)$;
3. wCN$^{\mathbf{1}}$, the weak logic induced by the expanded matrix $(\mathbf{CN_3}^{\mathbf{1}}, C, D)$.

Now, the general idea is the following. First, we have to check that each standard logic expanded with $\mathbf{1}$ is algebraizable relative to some tuple $(\mathbf{Q}, \tau, \Delta)$. Then, we need to prove that each of these logics coincides with the schematic fragment of the corresponding weak ones. With this, [19, Prop.3.12] tells us that, taking $\mathbf{Q}$ and $\Sigma(x) := \{x \vee \neg x \approx \mathbf{1}\}$, the class $\{(\mathbf{A}, \Sigma[A]) \mid \mathbf{A} \in \mathbf{Q}\}$ is an expanded quasivariety. Once we also had the core-definability of such quasivariety, it would remain to check that the corresponding weak logic is representable by $\Delta[\Sigma(x)]$; this, by [19, Cor.4.6], would imply the strict algebraizability of such logic.

Let us see what happens in each case separately.

**Cooper.** Let us restrict ourselves to the case of sOL$^{\mathbf{1}}$. We can prove the following result in a similar way as [17, Thm.6]:

**Lemma 10.** sOL$^{\mathbf{1}}$ *is algebraizable with respect to* $(\mathbf{Q}(\mathbf{OL_3}^{\mathbf{1}}), \tau, \Delta)$*, where*

$$\tau : x \mapsto x \approx x \rightarrow x,$$

$$\Delta : e \mapsto \{e_0 \rightarrow e_1, e_1 \rightarrow e_0, \neg e_0 \rightarrow \neg e_1, \neg e_1 \rightarrow \neg e_0\}.$$

*Proof.* Since the witnessing quasivariety is $\mathbf{Q}(\mathbf{OL_3}^{\mathbf{1}})$ and $\models_{\mathbf{Q}(\mathbf{OL_3}^{\mathbf{1}})} = \models_{\{\mathbf{OL_3}^{\mathbf{1}}\}}$, we only need to check that happens with the conditions (i) and (ii) of standard algebraizability in the three-element case, which are easy computations. ∎

Note that $(\mathbf{OL_3}^{\mathbf{1}}, C)$ is core-generated; this allows us to prove the following proposition as we did before:

**Lemma 11.** *The schematic fragment of* OL$^{\mathbf{1}}$ *is precisely* sOL$^{\mathbf{1}}$.

Now, we wish to prove that the quasivariety

$$\mathbf{OL}^{\mathbf{1}} := \{(\mathbf{A}, \Sigma[A]) \mid \mathbf{A} \in \mathbf{Q}(\mathbf{OL_3}^{\mathbf{1}})\},$$

where $\Sigma(x) := \{x \vee \neg x \approx \mathbf{1}\}$, is core-generated. It turns out that:

**Lemma 12.** $\mathbf{OL}^{\mathbf{1}} = \mathbf{Q}((\mathbf{OL_3}^{\mathbf{1}}, C))$. *Therefore,* $\mathbf{OL}^{\mathbf{1}}$ *is generated by a core-generated element, hence core-generated.*

*Proof.* Clearly, $\mathbf{Q}((\mathbf{OL_3^1}, \{\mathbf{0}, \mathbf{1}\})) \subseteq \mathbf{OL^1}$. On the other hand, given $(\mathbf{A}, \Sigma[A]) \in \mathbf{OL^1}$, we know that $\mathbf{A} \in \mathbf{Q}(\mathbf{OL_3^1}) = \mathbf{ISPP}_U(\mathbf{OL_3^1})$. We have to show that $(\mathbf{A}, \Sigma[A]) \in \mathbf{ISPP}_U((\mathbf{OL_3^1}, C))$. In order to do so, it is enough to check that $\Sigma(x)$ commutes with $\mathbf{I}$, $\mathbf{S}$, $\mathbf{P}$ and $\mathbf{P}_U$, which is clear since equations have this property in general. ■

Now we are in conditions of checking whether strict algebraizability holds:

**Theorem 13.** *The weak logic* $\mathrm{OL^1}$ *is strictly algebraized by* $(\mathbf{OL^1}, \Sigma(x), \tau, \Delta)$*, where* $\Sigma(x) := \{x \vee \neg x \approx \mathbf{1}\}$*. In consequence,* $\mathrm{OL^1}$ *is loosely algebraizable.*

*Proof.* We have seen before that $\mathrm{sOL^1}$ is algebraized by $(\mathbf{Q}(\mathbf{OL_3^1}), \tau, \Delta)$. Note that, indeed, $\mathbf{OL^1} \upharpoonright \mathcal{L} = \mathbf{Q}(\mathbf{OL_3^1})$. Hence, it is enough to check that it is representable by $\Delta(\{x \vee \neg x \approx \mathbf{1}\})$ by [19, Cor.4.6]. Observe that

$$\Delta(\{x \vee \neg x \approx \mathbf{1}\}) = \{x \vee \neg x \to \mathbf{1}, \neg(x \vee \neg x) \to \neg\mathbf{1}, \mathbf{1} \to x \vee \neg x, \neg\mathbf{1} \to \neg(x \vee \neg x)\}.$$

Denote $E := \mathrm{At}(\Delta(\Sigma(x)))$, $\vdash := \mathrm{OL^1}$ and $\vdash_s := \mathrm{sOL^1}$. We have to see that:

$$\Gamma \vdash \varphi \text{ iff } \Gamma \cup E \vdash_s \varphi.$$

($\leftarrow$) If $\Gamma \cup E \vdash_s \varphi$ then, for every $\mathbf{OL_3^1}$-valuation $h$ it will hold that $h(\Gamma \cup E) \subseteq D$ implies that $h(\varphi) \in D$. Given an arbitrary $(\mathbf{OL_3^1}, C)$-valuation $g$ such that $g(\Gamma) \subseteq D$, note that $g(E) \subseteq D$ by computation (since we only have the cases $\mathbf{1} \leftrightarrow \mathbf{1}$ or $\mathbf{0} \leftrightarrow \mathbf{0}$, for $x$ can only receive values $\mathbf{0}, \mathbf{1}$), so $g(\Gamma \cup E) \subseteq g(\Gamma) \cup g(E) \subseteq D$. Since, in particular, $g$ is an $\mathbf{OL_3^1}$ valuation, it holds that $g(\varphi) \in D$ by assumption.

($\rightarrow$) If $\Gamma \vdash \varphi$, this means that for every $(\mathbf{OL_3^1}, C)$-valuation $h$ it holds that $h(\Gamma) \subseteq D$ implies $h(\varphi) \in D$. Now, let $g$ be an arbitrary $\mathbf{OL_3^1}$-valuation such that $g(\Gamma \cup E) \subseteq D$. In particular, $g(\Gamma) \subseteq D$. Let us see that $g$ is an $(\mathbf{OL_3^1}, C)$-valuation. Suppose otherwise. Then, there is an atomic formula $p$ such that $g(p) = \top$. Hence, since $g(\Delta(\{p \vee \neg p \approx \mathbf{1}\})) \subseteq E$ (we have applied the atomic substitution $\sigma : x \mapsto p$), we must have that $g(\Delta(\{p \vee \neg p \approx \mathbf{1}\})) \subseteq D$. However, note that $g(p \vee \neg p) = \top$, so we have that $\top \to \mathbf{0} = \mathbf{0} \in D$, which is impossible. Hence, $g$ is an $(\mathbf{OL_3^1}, C)$-valuation such that $g(\Gamma) \subseteq D$ and, by assumption, $g(\varphi) \in D$. ■

**Farrell.** Since CN is essentially connected to F (see [17, Thm.8]), we restrict ourselves to the latter. We begin by checking that the corresponding standard logic is algebraizable:

**Lemma 14.** $\mathrm{F^1}$ *is algebraizable with respect to* $(\mathbf{Q}(\mathbf{F_3^1}), \tau, \Delta)$*, where*

$$\tau : x \mapsto x \approx x \to x,$$

$$\Delta : e \mapsto \{e_0 \to e_1, e_1 \to e_0, \neg e_0 \to \neg e_1, \neg e_1 \to \neg e_0\}.$$

And that it coincides with the schematic fragment of the weak logic (note that $(\mathbf{F_3}, C)$ is core-generated):

**Lemma 15.** *The schematic fragment of* $\mathrm{wF^1}$ *is precisely* $\mathrm{F^1}$.

Now, as before, we want to check that the expanded quasivariety

$$\mathbf{F^1} := \{(\mathbf{A}, \Sigma[A]) \mid \mathbf{A} \in \mathbf{Q}(\mathbf{F_3}^{\mathbf{1}})\},$$

where $\Sigma(x) := \{x \vee \neg x \approx \mathbf{1}\}$, is core-generated. As before, it is enough to see that:

**Lemma 16.** $\mathbf{F^1} = \mathbf{Q}((\mathbf{F_3}^{\mathbf{1}}, C))$. *Therefore,* $\mathbf{F^1}$ *is generated by a core-generated element, hence core-generated.*

It remains to observe that the proof of the last theorem works for this case as well, since $\top \to_{\mathsf{F}} \mathbf{0} = \mathbf{0}$:

**Theorem 17.** *The weak logic* $\mathrm{F}^{\mathbf{1}}$ *is strictly algebraized by* $(\mathbf{F^1}, \Sigma(x), \tau, \Delta)$*, where* $\Sigma(x) := \{x \vee \neg x \approx \mathbf{1}\}$*. In consequence,* $\mathrm{F}^{\mathbf{1}}$ *is loosely algebraizable.*

### 3.4 Twist-representations

It is natural to ask how the expansion of the language interacts with the twist-representations of OL, F and CN-algebras provided in [21, 17, 18]. In particular, we wish to relate core-definability with some basic definability property of twist-algebras.

As before, we proceed with each case separately.

**Cooper.** The candidate of twist-construction for $\mathrm{sOL}^{\mathbf{1}}$ looks as follows (cf. [18, Def.7]):

**Definition 18.** *Let* $\mathbf{B} := (B; 0, 1, \neg, \wedge, \to)$ *be a Boolean algebra. The* full $\mathrm{sOL}^{\mathbf{1}}$-twist-algebra over $\mathbf{B}$ *is the algebra* $\mathbf{B}^{\bowtie}$ *having as universe* $\{(a_1, a_2) \in B^2 \mid a_1 \to a_2 = a_2\}$ *(the condition can be equivalently stated as* $a_1 \vee a_2 = 1$*) and, as operations:*

*i.* $(x_1, x_2) * (y_1, y_2) := (x_1 \wedge y_1, (x_1 \to y_2) \wedge (y_1 \to x_2))$,
*ii.* $(x_1, x_2) \to (y_1, y_2) := (x_1 \to y_1, x_1 \to y_2)$,
*iii.* $\neg(x_1, x_2) := (x_2, x_1)$,
*iv.* $\mathbf{1} := (1, 0)$.

*The operation* $+$ *is obtained as the De Morgan dual of* $*$*, namely,* $x + y := \neg(\neg x * \neg y)$.

**Definition 19.** *An* $\mathrm{sOL}^{\mathbf{1}}$-twist-algebra over $\mathbf{B}$ *is any subalgebra* $\mathbf{A} \leqslant \mathbf{B}^{\bowtie}$ *such that* $\pi_1[A] = B$ *(where* $\pi_1$ *is the projection on the first coordinate).*

**Lemma 20.** *Every* $\mathrm{sOL}^{\mathbf{1}}$*-twist-algebra over* $\mathbf{B}$ *is full.*

*Proof.* Let $\mathbf{A} \leqslant \mathbf{B}^{\bowtie}$ with $\pi_1[A] = B$. Let $(x, y) \in \mathbf{B}^{\bowtie}$. Let us see that $(x, y)$ belongs to $\mathbf{A}$. Note that, by definition, we must have that $(x, u) \in \mathbf{A}$ for some $u \in B$. We then have:

$$\neg(x, u) \to (x, u) = (u, x) \to (x, u) = (u \to x, u \to u) = (x, 1) \in \mathbf{A}.$$

Similarly, since $\neg(x, y) = (y, x) \in \mathbf{B}^{\bowtie}$, we also have that $(y, v) \in \mathbf{A}$ for some $v \in B$ and, as before, $(y, 1) \in \mathbf{A}$. Then,

$$(x, 1) * (1, y) = (x \wedge 1, (x \to y) \wedge (1 \to 1)) = (x, (x \to y)) = (x, y) \in \mathbf{A},$$

as desired. ∎

On the other hand, we need a class of algebras to which $\mathrm{sOL}^{\mathbf{1}}$-twist-algebras belong (cf. [18, Def.6]):

**Definition 21.** *An* sOL$^{\mathbf{1}}$-algebra *is an algebra* $(A; \mathbf{1}, \neg, *, \rightarrow)$ *such that* $(A; \neg, *, \rightarrow)$ *is an OL-algebra and, moreover, satisfies*[5]*:*

$$\mathbf{1} \rightarrow \neg\mathbf{1} \leqslant \neg x \rightarrow x \leqslant \neg\mathbf{1} \rightarrow \mathbf{1},$$

*for every* $x \in A$.

The idea now is to generalize the results from [18, §2.1] for the case of sOL:

**Lemma 22.** *Given a* sOL$^{\mathbf{1}}$*-algebra* $\mathbf{A}$*, define* $\Diamond x := \neg x \rightarrow x$*. Then,* $\Diamond(\mathbf{A}) := (\Diamond[A]; \mathbf{1} \rightarrow \neg\mathbf{1}, \neg\mathbf{1} \rightarrow \mathbf{1}, \neg, \wedge, \rightarrow)$ *is a Boolean algebra.*

*Proof.* We already know that $(\Diamond[A]; \wedge, \rightarrow)$ is a generalized Boolean algebra (see [18, §2.1]). Note that, by the definition of sOL$^{\mathbf{1}}$-algebras, $\Diamond x = \neg x \rightarrow x \leqslant \neg\mathbf{1} \rightarrow \mathbf{1}$ and $\mathbf{1} \rightarrow \neg\mathbf{1} \leqslant \neg x \rightarrow x = \Diamond x$. Hence, $(\Diamond[A]; \wedge, \rightarrow)$ is a bounded generalized Boolean algebra, i.e. a Boolean algebra. ■

**Proposition 23.** *Every* sOL$^{\mathbf{1}}$*-twist-algebra is a* sOL$^{\mathbf{1}}$*-algebra. Conversely, every* sOL$^{\mathbf{1}}$*-algebra* $\mathbf{A}$ *is isomorphic to a* sOL$^{\mathbf{1}}$*-twist-algebra over* $\Diamond(\mathbf{A})$ *as witnessed by* $\iota : \mathbf{A} \rightarrow \Diamond(\mathbf{A}) \times \Diamond(\mathbf{A}) : a \mapsto (\Diamond a, \Diamond\neg a)$.

*Proof.* In order to see that every sOL$^{\mathbf{1}}$-twist-algebra is a sOL$^{\mathbf{1}}$-algebra one only needs to check that the additional condition from the definition of sOL$^{\mathbf{1}}$-algebras holds, as it is indeed the case. It remains to see that $\iota$ preserves the constant operation $\mathbf{1}$ (see [21, Thm.2.5], [18, Thm.8]):

$$\iota(\mathbf{1}) = (\Diamond\mathbf{1}, \Diamond\neg\mathbf{1}) = (\neg\mathbf{1} \rightarrow \mathbf{1}, \neg\neg\mathbf{1} \rightarrow \neg\mathbf{1}) = (\neg\mathbf{1} \rightarrow \mathbf{1}, \mathbf{1} \rightarrow \neg\mathbf{1}),$$

where the first and second components are the top and bottom elements of $\Diamond\mathbf{A}$, respectively. ■

Hence, in particular, every element of $\mathbf{Q}(\mathbf{OL_3}^{\mathbf{1}})$ is twist-representable in the sense above. We have, in fact:

**Proposition 24.** *Every expanded algebra formed by a* sOL$^{\mathbf{1}}$*-twist-algebra and the core defined by* $\{x \vee \neg x \approx \mathbf{1}\}$ *is core-generated.*

*Proof.* Let us denote the core by $C$. If $(x, y) \in C$ then

$$(x, y) + \neg(x, y) = (x, y) + (y, x) = ((y \rightarrow y) \wedge (x \rightarrow x), y \wedge x) = (1, y \wedge x) = (1, 0)$$

and this, together with the fact that we assume in general that $x \vee y = 1$[6], gives us that $x$ and $y$ are Boolean complements. Suppose that we have an arbitrary element of our twist-algebra $(x, y)$. We hence can define $(x, -x)$ and $(y, -y)$ as elements in the core, where $-a$ is the Boolean complement of $a$. Then, $(-x, x) = \neg(x, -x) \in \langle C \rangle$ and

$$(-x, x) \rightarrow (-x, x) = (-x \rightarrow -x, -x \rightarrow x) = (1, x)$$

[5] Note that this condition can be stated as the two equations $(\mathbf{1} \rightarrow \neg\mathbf{1}) \wedge (\neg x \rightarrow x) \approx \mathbf{1} \rightarrow \neg\mathbf{1}$ and $(\neg x \rightarrow x) \wedge (\neg\mathbf{1} \rightarrow \mathbf{1}) \approx \neg x \rightarrow x$, respectively.

[6] Indeed, this condition is equivalent to the one imposed in the universe of our algebra.

must be in $\langle C\rangle$, as well as $(x,1) = \neg(1,x)$. But note that

$$(x,1) * (1,y) = (x \wedge 1, (x \to y) \wedge (1 \to 1)) = (x, (x \to y) \wedge 1) = (x, x \to y) = (x,y),$$

so we also have $(x,y) \in \langle C\rangle$. Thus, $C$ spans the whole universe of our twist-algebra. ■

**Farrell.** The twist construction here is the following (cf. [18, Def.30]):

**Definition 25.** *Consider a Boolean algebra* $\mathbf{B}$*. We define the* full $\mathrm{F}^{\mathbf{1}}$ twist-algebra over $\mathbf{B}$ *as the algebra* $\mathbf{B}^{\bowtie}$ *with universe* $\{(a_1,a_2) \in B \times B \mid a_1 \vee a_2 = 1\}$ *and operations*

*i.* $(x_1,x_2) \wedge (y_1,y_2) := (x_1 \wedge y_1, x_2 \vee y_2)$,
*ii.* $\neg(x_1,x_2) := (x_2,x_1)$,
*iii.* $(x_1,x_2) \to (y_1,y_2) := (x_1 \to y_1, x_2 \vee y_2)$,
*iv.* $\mathbf{1} := (1,0)$.

*As before, one defines* $\vee$ *as the De Morgan dual of* $\wedge$ *and, moreover, we obtain* $\top := \neg\mathbf{1} \to \mathbf{1}$.

**Definition 26.** *An* $\mathrm{F}^{\mathbf{1}}$-twist-algebra over $\mathbf{B}$ *is any subalgebra* $\mathbf{A} \leqslant \mathbf{B}^{\bowtie}$ *such that* $\pi_1[A] = B$.

It is known that not every F-twist-algebra is full; the counterexample also works in this case (see [18, Fact 31]).

On the other hand, the corresponding class of algebras looks as follows (cf. [18, Def.32]):

**Definition 27.** *An* $\mathrm{F}^{\mathbf{1}}$-algebra *is an algebra* $(A; \mathbf{1}, \top, \wedge, \vee, \to, \neg)$ *where* $(A; \top, \wedge, \vee, \to, \neg)$ *is a F-algebra and where it holds that:*

$$\neg\mathbf{1} \leqslant x \wedge \top \leqslant \mathbf{1},$$

*for every* $x \in A$.

Defining $\Diamond x := x \wedge \top$, it follows that $(\Diamond[A]; \top, \wedge, \to)$ is a bounded generalized Boolean algebra (see [18, Lem.33]). Therefore, we get:

**Lemma 28.** *Given a* $\mathrm{F}^{\mathbf{1}}$*-algebra* $\mathbf{A}$*, define* $\Diamond x := x \wedge \top$*. Then,* $\Diamond(\mathbf{A}) := (\Diamond[A]; \neg\mathbf{1}, \top, \neg, \wedge, \to)$ *is a Boolean algebra.*

Additionally, by adapting the proof of [18, Thm.34]:

**Proposition 29.** *Every* $\mathrm{F}^{\mathbf{1}}$*-twist-algebra is a* $\mathrm{F}^{\mathbf{1}}$*-algebra. Conversely, every* $\mathrm{F}^{\mathbf{1}}$*-algebra* $\mathbf{A}$ *is isomorphic to a* $\mathrm{F}^{\mathbf{1}}$*-twist-algebra over* $\Diamond(\mathbf{A})$ *as witnessed by* $\iota : \mathbf{A} \to \Diamond(\mathbf{A}) \times \Diamond(\mathbf{A}) : a \mapsto (\Diamond a, \Diamond\neg a)$.

*Proof.* As before, the first claim is a matter of computation. For the second one, we need to check that the map $\iota$ preserves $\mathbf{1}$:

$$\iota(\mathbf{1}) = (\Diamond\mathbf{1}, \Diamond\neg\mathbf{1}) = (\mathbf{1} \wedge \top, \neg\mathbf{1} \wedge \top) = (\top, \neg\mathbf{1}),$$

and these coincide with, respectively, the top and bottom elements of $\Diamond(\mathbf{A})$, as desired. ■

In particular, every element of $\mathbf{Q}(\mathbf{F_3}^{\mathbf{1}})$ is twist-representable in the sense above.

As before, we are now able to directly check core-definability, with the notable restriction to *full* twist-algebras. The reason for this is that precisely the example of a non-full F-twist-algebra (see above and [18, Fact 31]) indicates that a non-full $\mathrm{F}^{\mathbf{1}}$-twist-algebra $\mathbf{A}$ does not contain, in general, elements of the form $(a, -a)$, where $a \in \pi_1[A]$; in these cases, the following trick does not work anymore:

**Proposition 30.** *Every expanded algebra formed by a full* $\mathrm{F}^{\mathbf{1}}$*-twist-algebra and the core defined by* $\{x \vee \neg x \approx \mathbf{1}\}$ *is core-generated.*

*Proof.* Let us denote again the core by $C$. Note that, given $(x, y) \in C$,

$$(x, y) \vee \neg(x, y) = (x, y) \vee (y, x) = (x \vee y, x \wedge y) = (1, x \wedge y) = (1, 0),$$

so again we have in our core elements of the form $(a, -a)$, where $-a$ is the Boolean complement of $a$. Note that, for every $(a, b) \in \langle C\rangle$ we have that

$$(a, b) \to (1, 0) = (a \to 1, b \vee 0) = (1, b) \in \langle C\rangle.$$

Now, given an arbitrary element $(x, y)$ from our algebra, consider the elements $(x, -x), (-y, y) \in \langle C\rangle$. By the previous observation, we also have that $(1, y) \in \langle C\rangle$. Therefore,

$$(x, -x) \wedge (1, y) = (x \wedge 1, -x \vee y) = (x, -x \vee y) \in \langle C\rangle.$$

We only need to check that

$$-x \vee y = (-x \vee y) \wedge 1 = (-x \vee y) \wedge (x \vee y) = (-x \wedge x) \vee y = 0 \vee y = y.$$

Hence, we obtain that $(x, y) \in \langle C\rangle$, as desired. ■

## 4 Future work

It would be nice to find a solution for the following problems in the future:

1. *Axiomatizability.* It is known that OL can be axiomatized by taking the axiomatization of sOL and adding the axiom $\vdash (p \wedge \neg p) \to q$, which holds for the atomic case (see [13]). As happens with DF, F and CN, sOL admits an analytic axiomatization that can be translated into a single-conclusion one (see [14]). However, for the case of OL, a multiple-conclusion axiomatization for OL can be obtained by taking the preceding one of sOL and adding the rules (see [13, Thm.3]):

$$\{\{p, \neg p\}/\varnothing[p] \mid p \in \mathrm{Var}\}.$$

Similarly, sOL admits a disjunction (in the sense of [13]), so this multiple-conclusion calculus can be converted into a single-conclusion one. Since the disjunction of DF, F and CN is, by definition, also a disjunction of wDF, wF and wCN, it would remain to check how to obtain a multiple-conclusion axiomatization of wDF, wF and wCN: can the preceding observation on OL work for these cases as well?

2. *Philosophical motivation.* Above, we have argued that considering the weak versions of our logics sits well with the philosophical content intended to be captured by the original logics. Later, we have expanded the language with $\mathbf{1}$ so, in some sense, the resulting logics are generalizations of the previous ones where one can explicitly talk about truth (and falsity). Is this also consistent with the motivation behind a logic of indicative conditionals?

**Acknowledgments.** I would like to thank Umberto Rivieccio for providing me with some crucial insights and introducing me to the notion of weak algebraizability in the context of the three-valued logics of indicative conditionals presented in the paper. I also wish to thank the anonymous referees of ReacTS 2026 for their exhaustive comments and suggestions.

## References


1. Blok, W., Pigozzi, D.: Algebraizable Logics. Memoirs of the AMS Series, American Mathematical Society (1989)
2. Burris, S., Sankappanavar, H.P.: A Course in Universal Algebra. Springer (2011)
3. Cantwell, J.: The Logic of Conditional Negation. Notre Dame Journal of Formal Logic 49(3): 245-260 (2008). https://doi.org/10.1215/00294527-2008-010
4. Cooper, W.S.: Foundations of Logico-Linguistics. A Unified Theory of Information, Language, and Logic. Studies in Linguistics and Philosophy, Springer Dordrecht (1978)
5. Cooper, W.S.: The Propositional Logic of Ordinary Discourse. Inquiry: An Interdisciplinary Journal of Philosophy **11**(1-4), 295–320 (1968). https://doi.org/10.1080/00201746808601531
6. Égré, P., Rossi, L., Sprenger, J.: De Finettian logics of indicative conditionals part I: trivalent semantics and validity. Journal of Philosophical Logic **50**(2), 187–213 (Apr 2021). https://doi.org/10.1007/s10992-020-09549-6
7. Egré, P., Rossi, L., Sprenger, J.: De Finettian logics of indicative conditionals part II: proof theory and algebraic semantics. Journal of Philosophical Logic **50**(2), 215–247 (Apr 2021). https://doi.org/10.1007/s10992-020-09572-7
8. Égré, P. and Rossi, L. and Sprenger, J.: Probability for trivalent conditionals. Preprint (2024)
9. Farrell, R.J.: Implication and Presupposition. Notre Dame Journal of Formal Logic, Volume 27, Number 1 (January 1986)
10. Farrell, R.J.: Material implication, confirmation, and counterfactuals. Notre Dame Journal of Formal Logic 20 (2), pp. 383-394 (1979)
11. de Finetti, B.: La logique de la probabilité. Actes du Congrés International de Philosophie Scientifique (1936)
12. Font, J.M.: Abstract Algebraic Logic: An introductory textbook. College Publications (04 2016)
13. Greati, V., Marcelino, S., Rivieccio, U.: Axiomatizing the Logic of Ordinary Discourse. Proceeding of IPMU24 (to appear) (May 5 2024)
14. Greati, V., Marcelino, S., Muñoz Pérez, M., Rivieccio, U.: Analytic calculi for logics of indicative conditionals. In: Pozzato, G. L., Uustalu, T. (eds) Automated Reasoning with Analytic Tableaux and Related Methods. TABLEAUX 2025. Lecture Notes in Computer Science(), vol 15980. Springer, Cham. (2026)
15. Mortensen, C.: Aristotle's thesis in consistent and inconsistent logics. Studia Logica 43, 107–116 (1984)
16. Muñoz Pérez, M.: Four-valued logics of indicative conditionals. In: Bianchini, F.; Pavonne A.; Proença, J.; Requeno Jarabo, J. I. & Rivieccio, U. (eds.) Software Engineering and Formal Methods. SEFM 2025 Collocated Workshops, Lecture Notes in Computer Science, Springer (to appear) (2026)
17. Muñoz Pérez, M., Rivieccio, U.: Indicative Conditionals: Some Algebraic Considerations. In Kozen, D. & de Queiroz, R. (eds) Logic, Language, Information, and Computation, WoLLIC 2025, Lecture Notes in Computer Science, vol 15942, Springer (2025)
18. Muñoz Pérez, M., Rivieccio, U.: Indicative conditionals: from three to four values. (submitted) Preprint available at https://arxiv.org/abs/2510.21882 (2026)
19. Nakov, G., Quadrellaro, D.E.: Algebraizable weak logics. The Journal of Symbolic Logic. Published online:1-50. doi:10.1017/jsl.2026.10180. Available at https://arxiv.org/abs/2210.06047 (2026)
20. Olkhovikov, G.K.: On a new three-valued paraconsistent logic. IfCoLog Journal of Logics and their Applications 3, 317–334 (2016)

21. Rivieccio, U.: The algebra of ordinary discourse. On the semantics of Cooper's logic. Archive for Mathematical Logic (2025). https://doi.org/{10.1007/s00153-024-00961-2}
22. Sprenger, J.: The Conditional in Three-Valued Logic. In Paul Egre & Lorenzo Rossi, Handbook of Three-Valued Logic. Cambridge, Massachusetts: The MIT Press (forthcoming) (2024)
23. Égré, P., Rossi, L., Sprenger, J.: Gibbardian Collapse and Trivalent Conditionals. In: Kaufmann, S., Over, D.E., Sharma, G. (eds) Conditionals. Palgrave Studies in Pragmatics, Language and Cognition. Palgrave Macmillan, Cham. (2023)

# Opening the Box to a Probabilistic Dynamic Logic

Leandro Gomes[1] and Alexandre Madeira[2]

[1] Université de Lille, CNRS, Inria, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France
[2] CIDMA, Dep. Mathematics, Aveiro University, Aveiro, Portugal

**Abstract.** Dynamic logic is a well-known framework for reasoning about imperative programs, enabling the verification of possibility and necessity properties, such as liveness and safety. In its standard form (cf. [13]), dynamic logic focuses on the satisfaction of properties after program execution. However, this 'closed-box' view offers no means to reason about internal behaviour - that is, property satisfaction at each step during execution.
To address this limitation, a system called Dynamic Logic with Trace Modalities was introduced [6], an extension of dynamic logic with two new modalities *throughout* and *at least once*, designed to reason about properties throughout program execution, such as safety and invariants.
We propose *Open-box probabilistic dynamic logic* (`OBPDL`), a while-free probabilistic dynamic logic, which includes the probabilistic choice operator, extended with two modalities to reason throughout program execution with with a minimum probability value. We present this as the 'open-box' view.
We present an operational semantics for program execution, the satisfaction semantics, and several valid properties of this logic, establishing, in particular, connections between open-box and closed-box modalities.

## 1 Introduction

Formal methods have become central to modern software development, as an orthogonal approach to testing, offering rigorous techniques for specification, verification, and analysis of programs. Among these, *dynamic logic* (`DL`) has proven especially effective in reasoning about imperative programs, enabling the formal verification of possibility (e.g., liveness) and necessity (e.g., safety) properties.

`DL` is based on modal logic and offers two modalities: $\langle c\rangle\varphi$ asserts the possibility that $\varphi$ holds after some execution of program $c$, while $[c]\varphi$ ensures that $\varphi$ holds after all executions. Moreover, its axiomatic system supports decompositional reasoning, as illustrated by the equivalence between $\varphi \to [c_1; c_2]\psi$ and $\varphi \to [c_1][c_2]\psi$. While powerful, classical `DL` approach programs in a 'closed-box' perspective: it focuses on pre and postconditions, ignoring the internal steps of program execution. As a result, it falls short in expressing or verifying properties that depend on intermediate states or behaviours during execution. This

is particularly relevant when we are interested in runtime properties, such as invariants or stepwise guarantees.

*Dynamic logic with trace modalities* was introduced in [6] precisely to address such a limitation. It extends DL with two new modalities $[\![c]\!]\varphi$ , meaning that "$\varphi$ is true throughout the execution of program $c$" and $\langle\!\langle\ c\ \rangle\!\rangle\varphi$ , meaning that "there is an intermediate step in the execution of $c$ where $\varphi$ holds".

However, the uncertain behaviour is common in nowadays systems, coming with a lot of challenges for designing and formal verification of software. This uncertainty is commonly modelled in the form of probabilities, and present in frameworks such as the probabilistic weakest-precondition calculus [17], the verification of probabilistic network programs [19], or the reasoning about probabilistic bounds of program termination [3].

In this work, we introduce *Open-Box Probabilistic Dynamic Logic (OBPDL)*, a probabilistic dynamic logic for a while free language, extended with two modalities to reason about internal program execution with with a minimum probability value. In order to handle probabilities, we adopt the grammar of programs from [7], which does not include nondeterminism, but is endowed with guarded sum to model conditionals. This deterministic version of classic DL is first extended with a probabilistic choice operator $c_1 \oplus_r c_2$, which represents the coin flipping that obtains $c_1$ with probability $r$ and $c_2$ with probability $1 - r$. Then it is extended with two modalities $[\![_]\!]_p$ and $\langle\!\langle\ _\ \rangle\!\rangle_p$ , which enables to express stepwise properties that hold, respectively, at at every step of execution, or at specific intermediate points, with a minimum probability $p$. This is what mean by "open-box modalities". We define an operational semantics for program execution, the satisfaction semantics, and several valid properties of this logic, establishing, in particular, connections between open-box and closed-box modalities.

**Outline of the paper** The remaining of the paper is organised as follows. Section 2 introduces OBPDL, discussing its syntax and semantics. Section 3 discuss the soundness of some properties, paving the first steps for the definition of an axiomatisation for OBPDL. Finally, Section 4 sumarizes and points some directions for future work.

## 2 Open-box dynamic logic

### Syntax

Let Var be a set of variables, $\mathbb{Z}$ the set of integers and $\mathbb{B} = \{\texttt{true}, \texttt{false}\}$. Formulas of OBPDL extends the formulas do $DL$ with open box modalities.

$$\mathrm{Fm}^{\texttt{OBPDL}} \ni \varphi ::= b \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \to \varphi \mid \langle\!\langle\ c\ \rangle\!\rangle_{\mathbf{p}}\varphi \mid \langle c\rangle_{\mathbf{p}}\varphi \mid [\![c]\!]_{\mathbf{p}}\varphi \mid [c]_{\mathbf{p}}\varphi$$

where,

$$\texttt{AExp} \ni a ::= z \mid x \mid a + a \mid a - a \mid a * a,\ z \in \mathbb{Z}, x \in \texttt{Var} \tag{1}$$

$$\texttt{BExp} \ni b ::= a \mid a = a \mid a > a \mid \neg b \mid b \wedge b \mid b \vee b \tag{2}$$

$$\texttt{Cmd} \ni c ::= \texttt{skip} \mid x := a \mid c; c \mid c +_b c \mid c \oplus_r c \mid ?b\ , r \in [0, 1] \tag{3}$$

**Semantics**

The interpretation of programs for open modalities is given by small-steps semantics; the interpretation of programs for closed modalities is given by big steps semantics:

- *States* are elements of $\Sigma = \mathbb{Z}^{\mathtt{Var}}$, i.e. $\Sigma$ is the set of functions of form $\sigma : \mathtt{Var} \to \mathbb{Z}$; in other words, states are evaluations of variables to the integers;
- *Configurations* are elements of $Conf = \mathtt{Cmd} \times \Sigma$, i.e. are pairs $\langle c, \sigma \rangle$, where $c \in \mathtt{Cmd}$ and $\sigma \in \Sigma$.

The structural operation rules for our while-free language are defined in fig.1.

We write

$$\langle c, \sigma \rangle \underset{q}{\Rightarrow} \langle c', \sigma' \rangle$$

to denote the execution of an intermediate step of the program $c$ with probability $q$. It represents then the configuration relation between configuration $\langle c, \sigma \rangle$ and configuration $\langle c', \sigma' \rangle$, where $c'$ is a subprogram of $c$ and $\sigma'$ is the state after executing $c'$.

We write

$$\langle c, \sigma \rangle \underset{q}{\to} \sigma'$$

to denote the execution of the full program $c$ with probability $q$. It represents the configuration relation between configuration $\langle c, \sigma \rangle$ and configuration $\sigma'$, which is is the state after executing $c$.

- The *open-box interpretation of a program* $c \in \mathtt{Cmd}$ is a configurations relation that includes all the intermediate steps of the execution of the program $c$. According to this we write

  $$\langle c, \sigma \rangle \overset{1 \ldots k}{\underset{q}{\Longrightarrow}} \gamma$$

  whenever we have

  $$\langle c, \sigma \rangle \underset{q_1}{\Longrightarrow} \langle c_1, \sigma_1 \rangle \underset{q_2}{\Longrightarrow} \cdots \underset{q_k}{\Longrightarrow} \gamma \text{ with } q = q_1 * \cdots * q_k.$$

  For the case $k = 1$, we simply write $\langle c, \sigma \rangle \underset{q}{\Rightarrow} \gamma$.

  $$\mathtt{OB}(c) = \{(\sigma, \sigma')_q \mid \langle c, \sigma \rangle \overset{1 \ldots k}{\underset{q}{\Longrightarrow}} \gamma, \text{ for some } k \geq 1\}$$

  The *open-box interpretation of a program* $c \in \mathtt{Cmd}$ *from a state* $\sigma \in \Sigma$ is defined by

  $$\mathtt{OB}(c)(\sigma) : \Sigma \to [0, 1]$$
  $$\sigma' \mapsto \begin{cases} q, & (\sigma, \sigma')_q \in \mathtt{OB}(c) \\ 0, & \text{otherwise} \end{cases}$$

$$(skip)\ \frac{}{\langle \texttt{skip},\sigma\rangle \underset{1}{\rightarrow} \sigma} \quad (\text{assign})\ \frac{\langle a,\sigma\rangle \rightarrow z}{\langle\, x := a,\sigma\rangle \underset{1}{\rightarrow} \sigma[x \leftarrow z]} \quad \text{where } \sigma[x \leftarrow z](y) = \begin{cases} z, & y = x \\ \sigma(y), & \text{otherwise} \end{cases}$$

$$(\text{seq})\ \frac{\langle c_1,\sigma\rangle \xrightarrow[q_1]{} \sigma' \quad \langle c_2,\sigma'\rangle \xrightarrow[q_2]{} \sigma''}{\langle c_1;c_2,\sigma\rangle \xrightarrow[q_1 * q_2]{} \sigma''}$$

$$(\text{choice-1})\ \frac{\langle c_1,\sigma\rangle \underset{q}{\rightarrow} \sigma' \quad \langle b,\sigma\rangle \rightarrow \texttt{true}}{\langle c_1 +_b c_2,\sigma\rangle \underset{q}{\rightarrow} \sigma'} \qquad (\text{choice-2})\ \frac{\langle c_2,\sigma\rangle \underset{q}{\rightarrow} \sigma' \quad \langle b,\sigma\rangle \rightarrow \texttt{false}}{\langle c_1 +_b c_2,\sigma\rangle \xrightarrow[1-q]{} \sigma'}$$

$$(\text{test})\ \frac{\langle b,\sigma\rangle \rightarrow \texttt{true}}{\langle ?b,\sigma\rangle \underset{1}{\rightarrow} \sigma}$$

$$(\text{p-choice})\ \frac{\langle c_1,\sigma\rangle \xrightarrow[q_1]{} \sigma' \quad \langle c_2,\sigma\rangle \xrightarrow[q_2]{} \sigma'}{\langle c_1 \oplus_r c_2,\sigma\rangle \xrightarrow[r*q_1+(1-r)*q_2]{} \sigma'}$$

(a) Big-steps Semantics (BSS)

$$(skip)\ \frac{}{\langle \texttt{skip},\sigma\rangle \underset{1}{\Rightarrow} \sigma}$$

$$(\text{assign})\ \frac{\langle a,\sigma\rangle \rightarrow z}{\langle\, x := a,\sigma\rangle \underset{1}{\Rightarrow} \sigma[x \leftarrow z]}, \text{ where } \sigma[x \leftarrow z](y) = \begin{cases} z, & y = x \\ \sigma(y), & \text{otherwise} \end{cases}$$

$$(\text{seq-1})\ \frac{\langle c_1,\sigma_1\rangle \underset{q}{\Rightarrow} \sigma_2}{\langle c_1;c_2,\sigma_1\rangle \underset{q}{\Rightarrow} \langle c_2,\sigma_2\rangle} \qquad (\text{seq-2})\ \frac{\langle c_1,\sigma_1\rangle \underset{q}{\Rightarrow} \langle c_1',\sigma_2\rangle}{\langle c_1;c_2,\sigma_1\rangle \underset{q}{\Rightarrow} \langle c_1';c_2,\sigma_2\rangle}$$

$$(\text{choice-1})\ \frac{\langle b,\sigma\rangle \rightarrow \texttt{true}}{\langle c_1 +_b c_2,\sigma\rangle \underset{1}{\Rightarrow} \langle c_1,\sigma\rangle} \qquad (\text{choice-2})\ \frac{\langle b,\sigma\rangle \rightarrow \texttt{false}}{\langle c_1 +_b c_2,\sigma\rangle \underset{1}{\Rightarrow} \langle c_2,\sigma\rangle}$$

$$(\text{test})\ \frac{\langle b,\sigma\rangle \rightarrow \texttt{true}}{\langle ?b,\sigma\rangle \underset{1}{\Rightarrow} \sigma}$$

$$(\text{p-choice-1})\ \frac{}{\langle c_1 \oplus_r c_2,\sigma\rangle \underset{r}{\Rightarrow} \langle c_1,\sigma\rangle} \qquad (\text{p-choice-2})\ \frac{}{\langle c_1 \oplus_r c_2,\sigma\rangle \underset{(1-r)}{\Longrightarrow} \langle c_2,\sigma\rangle}$$

(b) Small Steps Semantics (SSS)

$$\frac{}{\langle t,\sigma\rangle \rightarrow t}$$

$$\frac{\langle a_1,\sigma\rangle \rightarrow z_1 \quad \langle a_2,\sigma\rangle \rightarrow z_2}{\langle a_1 = a_2,\sigma\rangle \rightarrow \texttt{true}} \text{ if } z_1 = z_2 \qquad \frac{\langle a_1,\sigma\rangle \rightarrow z_1 \quad \langle a_2,\sigma\rangle \rightarrow z_2}{\langle a_1 = a_2,\sigma\rangle \rightarrow \texttt{false}} \text{ if } z_1 \neq z_2$$

$$\frac{\langle a_1,\sigma\rangle \rightarrow z_1 \quad \langle a_2,\sigma\rangle \rightarrow z_2}{\langle a_1 > a_2,\sigma\rangle \rightarrow \texttt{true}} \text{ if } z_1 > z_2 \qquad \frac{\langle a_1,\sigma\rangle \rightarrow z_1 \quad \langle a_2,\sigma\rangle \rightarrow z_2}{\langle a_1 > a_2,\sigma\rangle \rightarrow \texttt{false}} \text{ se } z_1 \leq z_2$$

$$\frac{\langle b,\sigma\rangle \rightarrow \texttt{false}}{\langle \neg b,\sigma\rangle \rightarrow \texttt{true}} \qquad \frac{\langle b,\sigma\rangle \rightarrow \texttt{true}}{\langle \neg b,\sigma\rangle \rightarrow \texttt{false}}$$

$$\frac{\langle b_1,\sigma\rangle \rightarrow \texttt{true} \quad \langle b_2,\sigma\rangle \rightarrow \texttt{true}}{\langle b_1 \wedge b_2,\sigma\rangle \rightarrow \texttt{true}} \qquad \frac{\langle b_1,\sigma\rangle \rightarrow \texttt{true} \quad \langle b_2,\sigma\rangle \rightarrow \texttt{false}}{\langle b_1 \wedge b_2,\sigma\rangle \rightarrow \texttt{false}}$$

$$\frac{\langle b_1,\sigma\rangle \rightarrow \texttt{false} \quad \langle b_2,\sigma\rangle \rightarrow \texttt{true}}{\langle b_1 \wedge b_2,\sigma\rangle \rightarrow \texttt{false}} \qquad \frac{\langle b_1,\sigma\rangle \rightarrow \texttt{false} \quad \langle b_2,\sigma\rangle \rightarrow \texttt{false}}{\langle b_1 \wedge b_2,\sigma\rangle \rightarrow \texttt{false}}$$

$$\frac{\langle b_1,\sigma\rangle \rightarrow \texttt{true} \quad \langle b_2,\sigma\rangle \rightarrow \texttt{true}}{\langle b_1 \vee b_2,\sigma\rangle \rightarrow \texttt{true}} \qquad \frac{\langle b_1,\sigma\rangle \rightarrow \texttt{true} \quad \langle b_2,\sigma\rangle \rightarrow \texttt{false}}{\langle b_1 \vee b_2,\sigma\rangle \rightarrow \texttt{true}}$$

$$\frac{\langle b_1,\sigma\rangle \rightarrow \texttt{false} \quad \langle b_2,\sigma\rangle \rightarrow \texttt{true}}{\langle b_1 \vee b_2,\sigma\rangle \rightarrow \texttt{true}} \qquad \frac{\langle b_1,\sigma\rangle \rightarrow \texttt{false} \quad \langle b_2,\sigma\rangle \rightarrow \texttt{false}}{\langle b_1 \vee b_2,\sigma\rangle \rightarrow \texttt{false}}$$

(c) Interpretation of Boolean expressions


$$\frac{}{\langle t,\sigma\rangle \rightarrow t}$$

$$\frac{\langle a_1,\sigma\rangle \rightarrow z_1 \quad \langle a_2,\sigma\rangle \rightarrow z_2}{\langle a_1 = a_2,\sigma\rangle \rightarrow \texttt{true}} \text{ if } z_1 = z_2 \qquad \frac{\langle a_1,\sigma\rangle \rightarrow z_1 \quad \langle a_2,\sigma\rangle \rightarrow z_2}{\langle a_1 = a_2,\sigma\rangle \rightarrow \texttt{false}} \text{ if } z_1 \neq z_2$$

$$\frac{\langle a_1,\sigma\rangle \rightarrow z_1 \quad \langle a_2,\sigma\rangle \rightarrow z_2}{\langle a_1 > a_2,\sigma\rangle \rightarrow \texttt{true}} \text{ if } z_1 > z_2 \qquad \frac{\langle a_1,\sigma\rangle \rightarrow z_1 \quad \langle a_2,\sigma\rangle \rightarrow z_2}{\langle a_1 > a_2,\sigma\rangle \rightarrow \texttt{false}} \text{ se } z_1 \leq z_2$$

$$\frac{\langle b,\sigma\rangle \rightarrow \texttt{false}}{\langle \neg b,\sigma\rangle \rightarrow \texttt{true}} \qquad \frac{\langle b,\sigma\rangle \rightarrow \texttt{true}}{\langle \neg b,\sigma\rangle \rightarrow \texttt{false}}$$

$$\frac{\langle b_1,\sigma\rangle \rightarrow \texttt{true} \quad \langle b_2,\sigma\rangle \rightarrow \texttt{true}}{\langle b_1 \wedge b_2,\sigma\rangle \rightarrow \texttt{true}} \qquad \frac{\langle b_1,\sigma\rangle \rightarrow \texttt{true} \quad \langle b_2,\sigma\rangle \rightarrow \texttt{false}}{\langle b_1 \wedge b_2,\sigma\rangle \rightarrow \texttt{false}}$$

$$\frac{\langle b_1,\sigma\rangle \rightarrow \texttt{false} \quad \langle b_2,\sigma\rangle \rightarrow \texttt{true}}{\langle b_1 \wedge b_2,\sigma\rangle \rightarrow \texttt{false}} \qquad \frac{\langle b_1,\sigma\rangle \rightarrow \texttt{false} \quad \langle b_2,\sigma\rangle \rightarrow \texttt{false}}{\langle b_1 \wedge b_2,\sigma\rangle \rightarrow \texttt{false}}$$

$$\frac{\langle b_1,\sigma\rangle \rightarrow \texttt{true} \quad \langle b_2,\sigma\rangle \rightarrow \texttt{true}}{\langle b_1 \vee b_2,\sigma\rangle \rightarrow \texttt{true}} \qquad \frac{\langle b_1,\sigma\rangle \rightarrow \texttt{true} \quad \langle b_2,\sigma\rangle \rightarrow \texttt{false}}{\langle b_1 \vee b_2,\sigma\rangle \rightarrow \texttt{true}}$$

$$\frac{\langle b_1,\sigma\rangle \rightarrow \texttt{false} \quad \langle b_2,\sigma\rangle \rightarrow \texttt{true}}{\langle b_1 \vee b_2,\sigma\rangle \rightarrow \texttt{true}} \qquad \frac{\langle b_1,\sigma\rangle \rightarrow \texttt{false} \quad \langle b_2,\sigma\rangle \rightarrow \texttt{false}}{\langle b_1 \vee b_2,\sigma\rangle \rightarrow \texttt{false}}$$


(d) Interpretatic ithmetic expressions



Fig. 1: Operational Semantics for `Cmd`

– The *closed-box interpretation of a program* $c \in \mathit{Cmd}$ is a states-relation that considers the final result of the full execution of the program $c$, i.e.,

$$\mathtt{CB}(c) = \{(\sigma, \sigma')_q \mid \sigma \underset{q}{\rightarrow} \sigma'\}$$

The *closed-box interpretation of a program* $c \in \mathit{Cmd}$ from a state $\sigma \in \Sigma$ is defined by

$$\mathtt{CB}(c)(\sigma) : \Sigma \rightarrow [0,1]$$
$$\mathtt{CB}(c)(\sigma)(\sigma') = \sum_{(\sigma,\sigma')_q \in \mathtt{CB}(c)} q$$

where the relations $\Rightarrow \subseteq \ Conf \times Conf$ and $\rightarrow \subseteq \ \Sigma \times \Sigma$ are the standard Big-Steps and Small-steps semantics of programs, given by the inference rules depicted in Fig. 1a and in Fig. 1b, respectively.
The big steps and small steps structural operational semantics was adapted from the imperative programming language **IMP** in [20] to our abstract language `Cmd`.

As expected, relations `OB` and `CB` inherit the well know relation between Big and Small semantics:

**Lemma 1.** *For any program* $c \in \mathit{Cmd}$*,*

*i) for any state* $\sigma$*,* $\mathit{CB}(c)(\sigma) \subseteq \mathit{OB}(c)(\sigma)$*, and*
*ii)* $\mathit{CB}(c) \subseteq \mathit{OB}(c)$

*Proof.* In appendix

**Proposition 1.** *For any* $c \in \mathit{Cmd}$ *and for any states* $\sigma, \sigma' \in \Sigma$*. Then,*

$$\textit{for some } k > 0 \ \langle c, \sigma\rangle \underset{q}{\overset{1\ldots k}{\Longrightarrow}} \sigma' \textit{ iff } \langle c, \sigma\rangle \underset{q}{\rightarrow} \sigma'.$$

*Proof.* In appendix

Next result establishes an useful relation between closed and open box interpretations of programs:

**Lemma 2.** *For any program* $c \in \mathit{Cmd}$ *and for any states* $\sigma, \sigma' \in \Sigma$*,*

$$\mathit{OB}(c_1; c_2) \ = \ \mathit{OB}(c_1) \cup (\mathit{CB}(c_1) \cdot \mathit{OB}(c_2)) \tag{4}$$

**Satisfaction**

The satisfaction relation of `OBPDL` is a relation $\models \subseteq \ \Sigma \times \mathrm{Fm}^{\mathtt{OBPDL}}$ recursively defined as follows:

– $\sigma \models b$ iff $\langle b, \sigma\rangle \rightarrow \mathtt{true}$

– $\sigma \models \neg\varphi$ iff is false $\sigma \models \varphi$
– $\sigma \models \varphi \wedge \varphi'$ iff $\sigma \models \varphi$ and $\sigma \models \varphi'$
– $\sigma \models \langle\!\langle\ c\ \rangle\!\rangle_p \varphi'$ iff there is a $(\sigma, \sigma')_q \in \mathtt{OB}(c)$ such that $p \leq q$ and $\sigma' \models \varphi$
– $\sigma \models [\![c]\!]_p \varphi'$ iff for any $(\sigma, \sigma')_q \in \mathtt{OB}(c)$ with $p \leq q$, we have $\sigma' \models \varphi$
– $\sigma \models \langle c \rangle_p \varphi'$ iff there is a $(\sigma, \sigma')_q \in \mathtt{CB}(c)$ such that $p \leq q$ and $\sigma' \models \varphi$
– $\sigma \models [c]_p \varphi'$ iff for any $(\sigma, \sigma')_q \in \mathtt{CB}(c)$ with $p \leq q$, we have $\sigma' \models \varphi$

As usual we say that a *formula is valid*, in symbols $\models \varphi$, if for any state $\sigma \in \Sigma$, $\sigma \models \varphi$.

## 3 First steps for an axiomatisation of OBPDL

In this section we explore some validities in this logic. The standard axiomatisation of $DL$ [13] can be used to axiomatise closed modalities:

**Proposition 2.** *The following formulas are valid in OBPDL:*

$$\langle c \rangle_p \varphi \ \leftrightarrow \neg[c]_p \neg\varphi \tag{5}$$
$$\langle c_1 \rangle_{p_1} \langle c_2 \rangle_{p_2} \varphi \ \rightarrow \langle c_1; c_2 \rangle_{p_1 * p_2} \varphi \tag{6}$$
$$[c_1; c_2]_{p_1 * p_2} \varphi \ \rightarrow [c_1]_{p_2} [c_2]_{p_1} \varphi \tag{7}$$
$$[c_1 +_b c_2]_p \varphi \ \leftrightarrow ((b \rightarrow [c_1]_p \varphi\ ) \wedge (\neg b \rightarrow [c_2]_p \varphi\ )) \tag{8}$$
$$(\langle c_1 \rangle_{p_1} \varphi \ \wedge \langle c_2 \rangle_{p_2} \varphi\ ) \rightarrow \langle c_1 \ \oplus_r c_2 \rangle_p \varphi\ , \textit{where } p = r * p_1 + (1 - r) * p_2 \tag{9}$$
$$[c_1 \ \oplus_r c_2]_p \varphi \ \rightarrow ([c_1]_{p_1} \varphi \ \wedge [c_2]_{p_2} \varphi\ ), \textit{where } p = r * p_1 + (1 - r) * p_2 \tag{10}$$
$$[b?]_1 \varphi \ \leftrightarrow (b \rightarrow \varphi) \tag{11}$$
$$[\ x \ := t]_1 \varphi \ \leftrightarrow \varphi[x/t] \tag{12}$$

*Proof.* **(5):** Let $\sigma \in \Sigma$. The, we observe that:

$\sigma \models \langle c \rangle_p \varphi$
$\Leftrightarrow$ { $\models$ defn}
exists $(\sigma, \sigma')_q \in \mathtt{CB}(c)$ with $p \leq q$ and $\sigma' \models \varphi$
$\Leftrightarrow$ { quantification duality}
it is false that for any $(\sigma, \sigma')_q \in \mathtt{CB}(c)$ with $p \leq q$ we have $\sigma' \models \neg\varphi$
$\Leftrightarrow$ { $\models$ defn}
$\sigma \models \neg[c]_p \neg\varphi$

and, therefore, $\sigma \models \langle c_1 \rangle_p \varphi \ \leftrightarrow \neg[c]_p \neg\varphi$ .

**(6):** Let $\sigma \in \Sigma$. The, we observe that:

$\sigma \models \langle c_1 \rangle_{p_1} \langle c_2 \rangle_{p_2} \varphi$
$\Leftrightarrow$ { $\models$ defn}

exists $(\sigma,\sigma')_{q_1} \in \mathtt{CB}(c_1)$ with $p_1 \leq q_1$ and
exists $(\sigma',\sigma'')_{q_2} \in \mathtt{CB}(c_2)$ with $p_2 \leq q_2$ and $\sigma'' \models \varphi$

$\Rightarrow$ { composition defn + monotonicity of $*$}

exists $(\sigma,\sigma'')_{q_1*q_2} \in \mathtt{CB}(c_1;c_2)$ with $p_1 * p_2 \leq q_1 * q_2$ and $\sigma' \models \langle c_2\rangle_{p_2}\varphi$

$\Leftrightarrow$ { $\models$ defn}

$\sigma \models \langle c_1;c_2\rangle_{p_1*p_2}\varphi$

and, therefore, $\sigma \models \langle c_1\rangle_{p_1}\langle c_2\rangle_{p_2}\varphi \ \rightarrow \langle c_1;c_2\rangle_{p_1*p_2}\varphi$ .

**(7):** By instantiation of (6) we have

$$\begin{array}{ll} & \sigma \models \langle c_1\rangle_{p_1}\langle c_2\rangle_{p_2}\neg\varphi \ \rightarrow \langle c_1;c_2\rangle_{p_1*p_2}\neg\varphi \\ \Leftrightarrow & \{ \text{ by contraposition}\} \\ & \sigma \models \neg\langle c_1;c_2\rangle_{p_1*p_2}\neg\varphi \ \rightarrow \neg\langle c_1\rangle_{p_1}\langle c_2\rangle_{p_2}\neg\varphi \\ \Leftrightarrow & \{ (5)\} \\ & \sigma \models [c_1;c_2]_{p_1*p_2}\varphi \ \rightarrow [c_1]_{p_1}[c_2]_{p_2}\varphi \end{array}$$

**(8):** We just need to observe that:

$\sigma \models [c_1 +_b c_2]_p\varphi$

$\Leftrightarrow$ { $\models$ defn}

for any $(\sigma,\sigma')_q$ s.t. $p \leq q$ and $(\sigma,\sigma')_q \in \mathtt{CB}(c_1 +_b c_2)$, we have $\sigma' \models \varphi$

$\Leftrightarrow$ { closed box semantics}

for any $(\sigma,\sigma')_q$ s.t. $p \leq q$ and
$(\sigma,\sigma')_q \in \{(\sigma,\sigma')_q \mid$ if $\langle\sigma,b\rangle \rightarrow \mathtt{true}$ then $(\sigma,\sigma')_q \in \mathtt{CB}(c_1)\}$ and
$(\sigma,\sigma')_q \in \{(\sigma,\sigma')_q \mid$ if $\langle\sigma,b\rangle \rightarrow \mathtt{false}$ then $(\sigma,\sigma')_q \in \mathtt{CB}(c_2)\}$
we have $\sigma' \models \varphi$

$\Leftrightarrow$ { simplification}

for any $(\sigma,\sigma')_q$ s.t. $p \leq q$ and if $\langle\sigma,b\rangle \rightarrow \mathtt{true}$ then $(\sigma,\sigma')_q \in \mathtt{CB}(c_1)$
we have $\sigma' \models \varphi$ and
for any $(\sigma,\sigma')_q$ s.t. $p \leq q$ and if $\langle\sigma,b\rangle \rightarrow \mathtt{false}$ then $(\sigma,\sigma')_q \in \mathtt{CB}(c_2)$
we have $\sigma' \models \varphi$

$\Leftrightarrow$ { $\models$ defn}

$\sigma \models (b \rightarrow [c_1]_p\varphi\ ) \wedge (\neg b \rightarrow [c_2]_p\varphi\ )$

**(9):** Let $\sigma \in \Sigma$, and let us assume $\sigma \models \langle c_1\rangle_{p_1}\varphi \ \wedge \langle c_2\rangle_{p_2}\varphi$ i.e.

– $\exists\sigma' \cdot (\sigma,\sigma')_{q_1} \in \mathtt{CB}(c_1)$, with $p_1 \leq q_1$ and $\sigma' \models \varphi$

– $\exists\sigma' \cdot (\sigma,\sigma')_{q_2} \in \mathtt{CB}(c_2)$, with $p_2 \leq q_2$ and $\sigma' \models \varphi$

The goal is to prove $\sigma \models \langle c_1 \oplus_r c_2\rangle_p\varphi$ , i.e. $\exists\sigma \cdot (\sigma,\sigma')_q \in \mathtt{CB}(c_1 \oplus_r c_2)$, with $p \leq q$ and $\sigma' \models \varphi$, where $q = r * q_1 + (1-r) * q_2$. $p = r * p_1 + (1-r) * p_2$.

By definition of closed-box semantics, $\mathtt{CB}(c_1 \oplus_r c_2) = \{(\sigma,\sigma')_q \mid (\sigma,\sigma')_{q_1} \in \mathtt{CB}(c_1)$ and $(\sigma,\sigma')_{q_2} \in CB(c_2)$ and $q = r * q_1 + (1-r) * q_2\}$. By the assumptions, we already know that $\exists\sigma'\cdot(\sigma,\sigma')_{q_1}$ and $\exists\sigma'\cdot(\sigma,\sigma')_{q_2} \in \mathtt{CB}(c_2)$. It remains to prove that $p \leq q = r * q_1 + (1-r) * q_2$. But, by monotonicity and by the assumptions, $p_1 \leq q_1 \wedge p_2 \leq q_2 \Rightarrow p = r * p_1 + (1-r) * p_2 \leq r * (q_1 + (1-r) * q_2) = q$, which proves that $\exists\sigma' \cdot (\sigma,\sigma')_q \in \mathtt{CB}(c_1 \oplus_r c_2)$, with $p \leq q$ qnd $\sigma' \models \varphi$.

**(10):** By instantiation of (9),

$$
\begin{array}{ll}
 & \sigma \models \langle c_1\rangle_{p_1}\neg\varphi \ \wedge \langle c_2\rangle_{p_2}\neg\varphi \ \rightarrow \langle c_1 \oplus_r c_2\rangle_p\neg\varphi \\
\Leftrightarrow & \quad \{ \text{contraposition}\} \\
 & \sigma \models \neg\langle c_1 \oplus_r c_2\rangle_p\neg\varphi \ \rightarrow \neg(\langle c_1\rangle_{p_1}\neg\varphi \ \wedge \langle c_2\rangle_{p_2}\neg\varphi \ ) \\
\Leftrightarrow & \quad \{ (5)\} \\
 & \sigma \models [c_1 \oplus_r c_2]_p\varphi \ \rightarrow \neg\langle c_1\rangle_{p_1}\neg\varphi \ \vee \neg\langle c_2\rangle_{p_2}\neg\varphi \\
\Leftrightarrow & \quad \{ (6)\} \\
 & \sigma \models [c_1 \oplus_r c_2]_p\varphi \ \rightarrow [c_1]_{p_1}\varphi \ \vee [c_2]_{p_2}\varphi
\end{array}
$$

**(11) and (12):** This can be taken from [13], since the semantics of `OBPDL` is an extension of $DL$ with open modalities.

We discuss now the relation between open box and closed box modalities.

**Proposition 3.** *For any $p \leq p'$:*

$$[c]_p\varphi \ \rightarrow [c]_{p'}\varphi \tag{13}$$
$$[\![c]\!]_p\varphi \ \rightarrow [\![c]\!]_{p'}\varphi \tag{14}$$
$$\langle c\rangle_p\varphi \ \rightarrow \langle c\rangle_{p'}\varphi \tag{15}$$
$$\langle\!\langle\ c\ \rangle\!\rangle_p\varphi \ \rightarrow \langle\!\langle\ c\ \rangle\!\rangle_{p'}\varphi \tag{16}$$

**Proposition 4.** *The following formulas are valid in `OBPDL`:*

$$[\![c]\!]_p\varphi \ \rightarrow [c]_p\varphi \tag{17}$$
$$\langle c\rangle_p\varphi \ \rightarrow \langle\!\langle\ c\ \rangle\!\rangle_p\varphi \tag{18}$$

*Proof.* In order to prove (17) let us assume $\sigma \models [\![c]\!]_p\varphi$ . By the satisfaction relation, we have that for any $(\sigma,\sigma')_q \in \mathtt{OB}(c)$ with $p \leq q$, we have $\sigma' \models \varphi$. By Lemma 2, $\mathtt{CB}(c) \subseteq \mathtt{OB}(c)$. So, by set inclusion, for any $\sigma' \in \Sigma$, if $(\sigma,\sigma')_q \in \mathtt{CB}(c)$ and $p \leq q$ then $\sigma' \models \varphi$. Hence, by the satisfaction relation, $\sigma' \models [c]_p\varphi$ .

In order to prove (18) let us assume $\sigma \models \langle c\rangle_p\varphi$ . By the satisfaction relation, we have that exists $(\sigma,\sigma')_q \in \mathtt{CB}(c)$ with $p \leq q$ and $\sigma' \models \varphi$. By Lemma 2, $\mathtt{CB}(c) \subseteq \mathtt{OB}(c)$. So, by set inclusion, exists $(\sigma,\sigma')_q \in \mathtt{OB}(c)$ with $p \leq q$ and $\sigma' \models \varphi$. Hence $\sigma' \models \langle\!\langle\ c\ \rangle\!\rangle_p\varphi$ .

We observe that $\not\models [c]_p\varphi \rightarrow [\![c]\!]_p\varphi$ and $\not\models \langle c\rangle_p\varphi \rightarrow \langle\!\langle\ c\ \rangle\!\rangle_p\varphi$ are consequences of Lemma 2. Open box and diamond are also dual. In analogy with ([15]), we have:

**Proposition 5.** *The following formula is valid in OBPDL:*

$$\langle\!\langle\ c\ \rangle\!\rangle_p\varphi \leftrightarrow \neg[\![c]\!]_p\neg\varphi \tag{19}$$

*Proof.* Let $\sigma \in \Sigma$. The, we observe that:

$\sigma \models \langle\!\langle\ c\ \rangle\!\rangle_p\varphi$

$\Leftrightarrow$ $\{ \models$ defn$\}$

exists $(\sigma,\sigma')_q \in \mathtt{OB}(c)$ with $p \leq q$ and $\sigma' \models \varphi$

$\Leftrightarrow$ $\{$ quantification duality$\}$

it is false that for any $(\sigma,\sigma')_q \in \mathtt{OB}(c)$ with $p \leq q$ we have $\sigma' \models \neg\varphi$

$\Leftrightarrow$ $\{ \models$ defn$\}$

$\sigma \models \neg[\![c]\!]_p\neg\varphi$

and, therefore, $\sigma \models \langle\!\langle\ c_1\ \rangle\!\rangle_p\varphi \leftrightarrow \neg[\![c]\!]_p\neg\varphi$ .

Now, we want to examine how the sequential composition of programs interacts with open-box modalities.

In particular, we observe that the equivalence $\langle\!\langle\ c_1\ \rangle\!\rangle_{p_1}\langle\!\langle\ c_2\ \rangle\!\rangle_{p_2}\varphi \leftrightarrow \langle\!\langle\ c_1; c_2\ \rangle\!\rangle_{p_1*p_2}\varphi$ , which which we can think of as the probabilistic open-box correspondant of the classic equivalence form Dynamic Logic ([15]) does not hold when replacing closed modalities with open-box modalities.

This discrepancy arises because the validity of formulas such as $[\![c_1; c_2]\!]_p\varphi$ and $[\![c_1]\!]_{p_1}[\![c_2]\!]_{p_2}\varphi$ , with $p, p_1, p_2 \in [0,1]$, depends on the semantics of internal transitions - specifically on the relations $\mathtt{OB}(c_1; c_2)(\sigma)$ and $\mathtt{OB}(c_1)(\sigma) \cdot \mathtt{OB}(c_2)(\sigma)$. While all transitions from $\mathtt{OB}(c_1)(\sigma)$ are included in $\mathtt{OB}(c_1; c_2)(\sigma)$, the converse does not hold in general: the transitions in $\mathtt{OB}(c_1; c_2)(\varphi)$ are not necessarily captured by the composed transitions in $\mathtt{OB}(c_1)(\sigma)\cdot\mathtt{OB}(c_2)(\sigma)$. As a consequence, the formulas $[\![c_1; c_2]\!]_p\varphi \rightarrow [\![c_1]\!]_{p_1}[\![c_2]\!]_{p_2}\varphi$ and $\langle\!\langle\ c_1; c_2\ \rangle\!\rangle_p\varphi \rightarrow \langle\!\langle\ c_1\ \rangle\!\rangle_{p_1}\langle\!\langle\ c_2\ \rangle\!\rangle_{p_2}\varphi$ , with $p = p_1 * p_2$, are not valid.

For example, consider the program $c \equiv\ x\ := 1;\ x\ := 2$. We have $\models \langle\!\langle\ c\ \rangle\!\rangle(x = 1)\ p$, for some $p \in [0,1]$, because during the execution, the intermediate state where $x = 1$ is indeed reached with probability at least $p$. However, for any $p_1, p_2 \in [0,1]$ such that $p = p_1 * p_2$, $\not\models \langle\!\langle\ x\ := 1\ \rangle\!\rangle_{p_1}\langle\!\langle\ x\ := 2\ \rangle\!\rangle_{p_2}(x = 1)$ .

Note that the limitation is not caused by probabilities, but rather by the nature of the open-box semantics. We cannot assure with any probability $p_1, p_2 \in [0,1]$ that the formula is true.

Furthermore, the composed set $\mathtt{OB}(c_1)(\sigma) \cdot \mathtt{OB}(c_2)(\sigma)$ includes compositions of intermediate transitions from $c_1$ with those of $c_2$ that are not valid internal transitions of $\mathtt{OB}(c_1; c_2)(\sigma)$. this implies that both directions of implication $[\![c_1]\!][\![c_2]\!]\varphi \rightarrow [\![c_1; c_2]\!]\varphi$ and $[\![c_1; c_2]\!]\varphi \rightarrow [\![c_1]\!][\![c_2]\!]\varphi$ are generally invalid.

As another example, consider $c \equiv \ y \ := 1;\ y \ := 2;\ x \ := y$. We have $\models \langle\!\langle \ y \ := 1;\ y \ := 2 \ \rangle\!\rangle_{p_1} \langle\!\langle \ x \ := y \ \rangle\!\rangle_{p_2} (x = 1)$ , since at an intermediate point $y = 1$ an then $x$ can be assigned 1. But $\not\models \langle\!\langle \ c \ \rangle\!\rangle_p (x = 1)$ , for any $p = p_1 * p_2$, since the final value of $y$ is 2, and $x$ is ultimately assigned 2.

These examples illustrate that the decompositional reasoning principles from classical dynamic logic fail in the probabilistic open-box setting. A finer-grained analysis is required to understand how program execution unfolds step-by-step.

Finally, we have the intended characterization:

**Proposition 6.** *The following formulas are valid in* `OBPDL`*:*

$$[c_1]_{p_1}[\![c_2]\!]_{p_2}\varphi \ \wedge [\![c_1]\!]_p\varphi \ \rightarrow [\![c_1; c_2]\!]_p\varphi \quad \textit{with } p = p_1 * p_2 \tag{20}$$

$$\langle\!\langle \ c_1; c_2 \ \rangle\!\rangle_p\varphi \ \rightarrow \langle\!\langle \ c_1 \ \rangle\!\rangle_p\varphi \ \vee \langle c_1\rangle_{p_1}\langle\!\langle \ c_2 \ \rangle\!\rangle_{p_2}\varphi \quad \textit{with } p = p_1 * p_2 \tag{21}$$

*Proof.* **(20)**: we just observe that:

$\sigma \models [\![c_1]\!]_p\varphi \ \wedge [c_1]_{p_1}[\![c_2]\!]_{p_2}\varphi \quad$ with $p = p_1 * p_2$

$\Leftrightarrow \quad \{ \models \text{defn}\}$

$\sigma \models [\![c_1]\!]_p\varphi \ \text{ and}$
$\sigma \models [c_1]_{p_1}[\![c_2]\!]_{p_2}\varphi \quad$ with $p = p_1 * p_2$

$\Leftrightarrow \quad \{ \models \text{defn}\}$

if $(\sigma, \sigma')_q \in \mathtt{OB}(c_1)$ and $p \leq q$ then $\sigma' \models \varphi$ and
if $(\sigma, \sigma_1)_{q_1} \in \mathtt{CB}(c_1)$ with $p_1 \leq q_1$ then $\sigma_1 \models [\![c_2]\!]_{p_2}\varphi$

$\Leftrightarrow \quad \{ \models \text{defn}\}$

if $(\sigma, \sigma')_q \in \mathtt{OB}(c_1)$ and $p \leq q$ then $\sigma' \models \varphi$ and
if $(\sigma, \sigma_1)_{q_1} \in \mathtt{CB}(c_1)$ with $p_1 \leq q_1$ then
  if $(\sigma_1, \sigma')_{q_2} \in \mathtt{OB}(c_2)$ with $p_2 \leq q_2$ then $\sigma' \models \varphi$

$\Leftrightarrow \quad \{ A \rightarrow (B \rightarrow C) \equiv (A \wedge B) \rightarrow C\}$

if $(\sigma, \sigma')_q \in \mathtt{OB}(c_1)$ and $p \leq q$ then $\sigma' \models \varphi$ and
if $(\sigma, \sigma_1)_{q_1} \in \mathtt{CB}(c_1)$ with $p_1 \leq q_1, (\sigma_1, \sigma')_{q_2} \in \mathtt{OB}(c_2)$ with $p_2 \leq q_2$ then $\sigma' \models \varphi$

$\Leftrightarrow \quad \{ \cdot \text{ defn}\}$

if $(\sigma, \sigma')_q \in \mathtt{OB}(c_1)$ and $p \leq q$ then $\sigma' \models \varphi$ and
if $(\sigma, \sigma')_{q_1 * q_2} \in \mathtt{CB}(c_1) \cdot \mathtt{OB}(c_2)$ with $p_1 * p_2 \leq q_1 * q_2,$ then $\sigma' \models \varphi$

$\Rightarrow \quad \{$ since $p = p_1 * p_2$ + particular case$\}$

if $(\sigma, \sigma')_{q_1 * q_2} \in \mathtt{OB}(c_1)$ and $p_1 * p_2 \leq q_1 * q_2$ then $\sigma' \models \varphi$ and
if $(\sigma, \sigma')_{q_1 * q_2} \in \mathtt{CB}(c_1) \cdot \mathtt{OB}(c_2)$ with $p_1 * p_2 \leq q_1 * q_2,$ then $\sigma' \models \varphi$

$\Rightarrow \quad \{$ since $(A \rightarrow C) \wedge (B \rightarrow C) \equiv (A \wedge B) \rightarrow C\}$

if $(\sigma, \sigma')_{q_1 * q_2} \in \mathtt{OB}(c_1), (\sigma, \sigma')_{q_1 * q_2} \in \mathtt{CB}(c_1) \cdot \mathtt{OB}(c_2)$ and $p_1 * p_2 \leq q_1 * q_2$

then $\sigma' \models \varphi$

$\Leftrightarrow$ { simplification}

if $(\sigma,\sigma')_{q_1*q_2} \in \mathtt{OB}(c_1) \cup (\mathtt{CB}(c_1) \cdot \mathtt{OB}(c_2))$ and $p_1 * p_2 \leq q_1 * q_2$ then $\sigma' \models \varphi$

$\Leftrightarrow$ { Lemma 2}

if $(\sigma,\sigma')_{q_1*q_2} \in \mathtt{OB}(c_1;c_2)$ and $p_1 * p_2 \leq q_1 * q_2$ then $\sigma' \models \varphi$

$\Leftrightarrow$ { $\models$ defn}

$\sigma \models [\![c_1;c_2]\!]_p\varphi$ with $p = p_1 * p_2$

**(20)**: we observe that

$$
\begin{array}{ll}
& ([c_1]_{p_1}[\![c_2]\!]_{p_2}\neg\varphi \ \wedge [\![c_1]\!]_p\varphi\ ) \rightarrow [\![c_1;c_2]\!]_p\neg\varphi \\
\Leftrightarrow & \quad \{ \text{ by contraposition}\} \\
& \neg[\![c_1;c_2]\!]_p\neg\varphi \ \rightarrow \neg([\![c_1]\!]_{p_1}\neg\varphi \ \wedge [c_1]_{p_1}[\![c_2]\!]_{p_2}\neg\varphi \ ) \\
\Leftrightarrow & \quad \{ \text{ by Morgan Law}\} \\
& \neg[\![c_1;c_2]\!]_p\neg\varphi \ \rightarrow \neg[\![c_1]\!]_{p_1}\neg\varphi \ \vee \neg[c_1]_{p_1}[\![c_2]\!]_{p_2}\neg\varphi \ ) \\
\Leftrightarrow & \quad \{ \text{ by (19)}\} \\
& \langle\!\langle\ c_1;c_2\ \rangle\!\rangle_p\varphi \ \leftrightarrow \langle\!\langle\ c_1\ \rangle\!\rangle_{p_1}\varphi \ \vee \langle c_1\rangle_{p_1}\langle\!\langle\ c_2\ \rangle\!\rangle_{p_2}\varphi
\end{array}
$$

**Proposition 7.** *The following formula is valid in* `OBPDL`*:*

$$[\![c_1 +_b c_2]\!]_p\varphi \ \leftrightarrow ((b \rightarrow [\![c_1]\!]_p\varphi\ ) \wedge (\neg b \rightarrow [\![c_2]\!]_p\varphi\ )) \tag{22}$$

*Proof.* The proof is analogous to (8).

**Proposition 8.** *The following formulas are valid in* `OBPDL`*:*

$$(\langle\!\langle\ c_1\ \rangle\!\rangle_{p_1}\varphi \ \wedge \langle\!\langle\ c_2\ \rangle\!\rangle_{p_2}\varphi\ ) \rightarrow \langle\!\langle\ c_1\ \oplus_r c_2\ \rangle\!\rangle_p\varphi\ , \textit{where } p = r * p_1 + (1-r) * p_2 \tag{23}$$

$$[\![c_1\ \oplus_r c_2]\!]_p\varphi \ \rightarrow ([\![c_1]\!]_{p_1}\varphi \ \wedge [\![c_2]\!]_{p_2}\varphi\ ), \textit{where } p = r * p_1 + (1-r) * p_2 \tag{24}$$

*Proof. The proof is analogous to (9) and (10).*

**Proposition 9.** *The following formula is valid in* `OBPDL`*:*

$$[\![b?]\!]\psi \ \leftrightarrow (b \rightarrow \psi) \tag{25}$$

*Proof.* In order to prove this is just need to observe that

$$\mathtt{OB}(b?)(\sigma) = \begin{cases} \{(\sigma,\sigma)\} & \text{if } \sigma \models b \\ \emptyset & \text{otherwise} \end{cases}.$$

Finally, for basic assignments:

**Proposition 10.** *The following formulæare valid in* OBPDL*:*

$$[\![ x := t ]\!]\varphi \leftrightarrow \varphi[x/t] \tag{26}$$

$$\langle\!\langle x := t \rangle\!\rangle\varphi \leftrightarrow \varphi[x/t] \tag{27}$$

*Proof.* In appendix

## 4 Discussion and future work

This work takes a first step towards a formal framework for reasoning about *evolution* of a program enriched with probabilistic choice, called *Open-box dynamic logic* (OBPDL). Unlike classical dynamic logic, which treats programs as atomic units with only observable input-output behaviour, OBPDL allows programs to be decomposed into smaller atomic steps, enabling the analysis of properties that hold during the execution of a program with a minimum probability.

In this sense, OBPDL can be compared in some sense to Probabilistic Computational tree logic (PCTL) [12], since both can express properties over the execution paths of programs with probabilistic behaviour. However, unlike PCTL, OBPDL - inherited from DL - uses program terms inside modalities, allowing for more expressive reasoning about imperative programs enriched with probabilistic choice.

Although this preliminary work presents only the initial ideas of OBPDL, it opens the door to a broad range of possible extensions. The first obvious next step is to discuss (guarded) iteration axioms. Those contain obviously three important components that make them diverge from the classical DL $^*$ axioms: one is the use of guarded loop, which we can reffer to [7] as a starting point. Another one is the inclusion of probabilities in the modalities. The third one is the adoption of the open-box modalities.

Another natural next step is to adapt the logic to a relational setting, in order to reason about pairs of probabilistic programs and compare their behaviours. This is motivated by existing work in relational logics [14, 1, 9, 2, 4], and would support reasoning about program equivalence or refinement. Moreover, the inclusion of *probabilistic reasoning* over a relational setting could be especially relevant for *differential privacy*, where relational properties between probabilistic executions are crucial [5].

Further generalisations could also include *weighted*, non probabilistic versions of OBPDL, along the lines of [16, 11, 10, 18], to model systems with uncertainty or cost, such as *fuzzy programs* [8].

A natural long-term goal is to investigate completeness results for these new variants of the logic. This would provide formal foundations for tools and proof systems based on OBPDL.

In conclusion, our long-term objective is pursue this agenda with OBPDL, in order to express and verify such *evolution-based properties* over probabilistic programs, bridging the gap between classical input-output reasoning and fine-grained behavioural verification, with a focus on applications where an uncertain behaviour of systems is present.

## References


1. Timos Antonopoulos, Eric Koskinen, Ton Chanh Le, Ramana Nagasamudram, David A. Naumann, and Minh Ngo. An algebra of alignment for relational verification. *CoRR*, abs/2202.04278, 2022.
2. Gilles Barthe, François Dupressoir, Benjamin Grégoire, César Kunz, Benedikt Schmidt, and Pierre-Yves Strub. Easycrypt: A tutorial. In Alessandro Aldini, Javier López, and Fabio Martinelli, editors, *Foundations of Security Analysis and Design VII - FOSAD 2012/2013 Tutorial Lectures*, volume 8604 of *Lecture Notes in Computer Science*, pages 146–166. Springer, 2013.
3. Gilles Barthe, Marco Gaboardi, Benjamin Grégoire, Justin Hsu, and Pierre-Yves Strub. A program logic for union bounds. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 107:1–107:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.
4. Gilles Barthe, Benjamin Grégoire, and Santiago Zanella Béguelin. Formal certification of code-based cryptographic proofs. In *Proceedings of the 36th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*, 2009.
5. Gilles Barthe, Boris Köpf, Federico Olmedo, and Santiago Zanella Béguelin. Probabilistic relational reasoning for differential privacy. *ACM Trans. Program. Lang. Syst.*, 35(3):9:1–9:49, 2013.
6. Bernhard Beckert and Steffen Schlager. A sequent calculus for first-order dynamic logic with trace modalities. In Rajeev Goré, Alexander Leitsch, and Tobias Nipkow, editors, *Automated Reasoning*, pages 626–641, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.
7. Mario Benevides, Leandro Gomes, and Bruno Lopes. Towards determinism in pdl: relations and proof theory. *Journal of Logic and Computation*, 35(5):exae022, 07 2025.
8. Jeroen S. de Bruin, Christian J. Schuh, Andrea Rappelsberger, and Klaus-Peter Adlassnig. Medical fuzzy control systems with fuzzy arden syntax. In Janusz Kacprzyk, Eulalia Szmidt, Slawomir Zadrozny, Krassimir T. Atanassov, and Maciej Krawczak, editors, *Advances in Fuzzy Logic and Technology 2017 - Proceedings of: EUSFLAT-2017 - The 10th Conference of the European Society for Fuzzy Logic and Technology, 2017, Warsaw, Poland IWIFSGN'2017 - The Sixteenth International Workshop on Intuitionistic Fuzzy Sets and Generalized Nets, 2017, Warsaw, Poland, Volume 1*, volume 641 of *Advances in Intelligent Systems and Computing*, pages 574–584. Springer, Cham, 2018.
9. Leandro Gomes, Patrick Baillot, and Marco Gaboardi. Bigkat: An algebraic framework for relational verification of probabilistic programs. In Parosh Aziz Abdulla and Delia Kesner, editors, *Foundations of Software Science and Computation Structures - 28th International Conference, FoSSaCS 2025, Held as Part of the International Joint Conferences on Theory and Practice of Software, ETAPS 2025, Hamilton, ON, Canada, May 3-8, 2025, Proceedings*, volume 15691 of *Lecture Notes in Computer Science*, pages 243–264. Springer, 2025.
10. Leandro Gomes, Alexandre Madeira, and Luís Soares Barbosa. A semantics and a logic for fuzzy arden syntax. *Soft Comput.*, 25(9):6789–6805, 2021.
11. Leandro Gomes, Alexandre Madeira, Manisha Jain, and Luís Soares Barbosa. On the generation of equational dynamic logics for weighted imperative programs. In

Yamine Aït Ameur and Shengchao Qin, editors, *ICFEM 2019, Shenzhen, China, Proc.*, volume 11852 of *LNCS*, pages 154–169. Springer, 2019.
12. Hans Hansson and Bengt Jonsson. A logic for reasoning about time and reliability. *Formal Aspects Comput.*, 6(5):512–535, 1994.
13. David Harel, Dexter Kozen, and Jerzy Tiuryn. *Dynamic logic*. MIT Press, 2000.
14. Juraj Kolcák, Jérémy Dubut, Ichiro Hasuo, Shin-ya Katsumata, David Sprunger, and Akihisa Yamada. Relational differential dynamic logic. In Armin Biere and David Parker, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 26th International Conference, TACAS 2020, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2020, Dublin, Ireland, April 25-30, 2020, Proceedings, Part I*, volume 12078 of *Lecture Notes in Computer Science*, pages 191–208. Springer, 2020.
15. D. Kozen. On Hoare logic and Kleene algebra with tests. *ACM Trans. on Comp. Logic*, 1(212):1–14, 2000.
16. Alexandre Madeira, Renato Neves, and Manuel A. Martins. An exercise on the generation of many-valued dynamic logics. *Journal of Logical and Algebraic Methods in Programming*, 1:1–29, 2016.
17. Carroll Morgan, Annabelle McIver, and Karen Seidel. Probabilistic predicate transformers. *ACM Trans. Program. Lang. Syst.*, 18(3):325–353, May 1996.
18. Igor Sedlár. Kleene algebra of weighted programs with domain. In Nina Gierasimczuk and Fernando R. Velázquez-Quesada, editors, *Dynamic Logic. New Trends and Applications - 5th International Workshop, DaLí 2023, Tbilisi, Georgia, September 15-16, 2023, Revised Selected Papers*, volume 14401 of *Lecture Notes in Computer Science*, pages 52–67. Springer, 2023.
19. Steffen Smolka, Praveen Kumar, David M. Kahn, Nate Foster, Justin Hsu, Dexter Kozen, and Alexandra Silva. Scalable verification of probabilistic networks. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2019, page 190–203, New York, NY, USA, 2019. Association for Computing Machinery.
20. Glynn Winskel. *The Formal Semantics of Programming Languages: An Introduction*. MIT Press, Cambridge, MA, USA, 1993.

# Appendix

## A Proof of Lemma 1

The proof of *i)* is given by induction on the structure of the transition derivation trees. Then *ii)* is a direct corollary of *i)*.

Let $(\sigma, \sigma')_q \in \mathtt{CB}(c)(\sigma)$, i.e., $\sigma \underset{q}{\rightarrow} \sigma'$. Then:

**Atomic programs $c \equiv \mathtt{skip}$ and $c \equiv\ x\ := a$:** If $c \equiv \mathtt{skip}$ by the rules of $BSS$, we know that $\sigma \underset{1}{\rightarrow} \sigma'$ was proved using the rule $(skip)$ and that $\sigma' = \sigma$. Hence, by (skip) of $SSS$ we have also that $\sigma \underset{1}{\Rightarrow} \sigma'$ and hence $(\sigma, \sigma')_q \in \mathtt{OB}(c)(\sigma)$.

Analogously, If $c \equiv\ x\ := a$ we know that $\sigma \underset{1}{\rightarrow} \sigma'$ was proved using the rule (assign) and that $\sigma' = \sigma[x \leftarrow a]$. By (assign) of $SSS$ we have also that $\sigma \underset{1}{\Rightarrow} \sigma'$ and hence $(\sigma, \sigma')_q \in \mathtt{OB}(c)(\sigma)$.

**Composed programs:**

Let us consider that $c \equiv c_1; c_2$. By the the rules of $BSS$, we know that $\sigma \underset{q}{\rightarrow} \sigma'$ was proved using the rule (Seq), and that there exists a state $\sigma''$ and derivations $A$ and $B$ such that $\dfrac{\overset{A}{\langle c_1,\sigma\rangle \underset{q_1}{\longrightarrow} \sigma''} \quad \overset{B}{\langle c_2,\sigma''\rangle \underset{q_2}{\longrightarrow} \sigma'}}{\langle c_1;c_2,\sigma\rangle \underset{q_1 * q_2}{\longrightarrow} \sigma'}$, with $q = q_1 * q_2$. Hence, using $SSS$ we have $\dfrac{A}{\langle c_1;c_2,\sigma\rangle \underset{q_1}{\Longrightarrow} \langle c_2,\sigma''\rangle}$ and $\dfrac{B}{\langle c_2,\sigma''\rangle \underset{q_2}{\Longrightarrow} \sigma'}$. Therefore $(\sigma, \sigma')_q \in \mathtt{OB}(c)(\sigma)$.

Analogously, if $c \equiv c_1 + c_2$, by the the rules of $BSS$, we know that $\sigma \underset{q}{\rightarrow} \sigma'$ was proved using the rule (choice-1) or (choice-2), i.e., that there is a derivation $A$ such that $\dfrac{\overset{A}{\langle c_1,\sigma\rangle \underset{q}{\longrightarrow} \sigma'}}{\langle c_1+c_2,\sigma\rangle \underset{q}{\longrightarrow} \sigma'}$, or a derivation $B$ such that $\dfrac{\overset{B}{\langle c_2,\sigma\rangle \underset{q}{\longrightarrow} \sigma'}}{\langle c_1+c_2,\sigma\rangle \underset{q}{\longrightarrow} \sigma'}$. Hence, using $SSS$ we have $\dfrac{A}{\langle c_1+c_2,\sigma\rangle \underset{q}{\Longrightarrow} \langle c_1,\sigma'\rangle}$ or $\dfrac{B}{\langle c_1+c_2,\sigma\rangle \underset{q}{\Longrightarrow} \langle c_2,\sigma'\rangle}$. Therefore $(\sigma, \sigma')_q \in \mathtt{OB}(c)(\sigma)$.

The proof for $c$ of form $b?$ is direct since the rules (test) coincide in both $BSS$ and $SSS$.

## B Proof of Lemma 2

For the $\subseteq$ inclusion, let us suppose that $(\sigma, \sigma')_q \in \mathtt{OB}(c_1; c_2)$. Then, there is a $k > 0$ such that

**(A)** $\langle c_1; c_2, \sigma\rangle \underset{q}{\overset{1\ldots k}{\Longrightarrow}} \langle c', \sigma'\rangle$.

**(B)** $\langle c_1; c_2, \sigma\rangle \underset{q}{\overset{1\ldots k}{\Longrightarrow}} \sigma'$, or

In case **(A)**, we have also two cases to be considered:

**(A1)** $\langle c_1; c_2, \sigma\rangle \underset{q}{\overset{1\ldots k}{\Longrightarrow}} \langle c_1'; c_2, \sigma'\rangle$ (before the consumption of $c_1$).

**(A2)** $\langle c_1; c_2, \sigma\rangle \overset{1\ldots k}{\underset{q}{\Longrightarrow}} \langle c_2', \sigma'\rangle$ (after the consumption of $c_1$).

We prove **(A1)** by induction on $k$: <u>if $k = 1$</u>, by the structure of SSS rules, we know that the transition was given by an application of (Seq-2) with $\langle c_1, \sigma\rangle \underset{q}{\Rightarrow} \langle c_1', \sigma'\rangle$. Hence $(\sigma, \sigma') \in \mathtt{OB}(c_1) \subseteq \mathtt{OB}(c_1) \cup (\mathtt{CB}(c_1) \cdot \mathtt{OB}(c_2))$.

<u>If $k > 1$</u> we have $\langle c_1; c_2, \sigma\rangle \overset{1\ldots k-1}{\underset{q_1}{\Longrightarrow}} \langle c_1''; c_2, \sigma'\rangle \underset{q_2}{\Longrightarrow} \langle c_1'; c_2, \sigma'\rangle$ with $q = q_1 * q_2$. By induction hypothesis we have $(\sigma, \sigma'')_{q_1} \in \mathtt{OB}(c_1)$; moreover, we have $(\sigma'', \sigma')_{q_2} \in \mathtt{OB}(c_1)$ and hence that $(\sigma, \sigma')_q \in \mathtt{OB}(c_1)$.

In order to prove **(A2)** let us assume $\langle c_1; c_2, \sigma\rangle \overset{1\ldots k}{\underset{q}{\Longrightarrow}} \langle c_2', \sigma'\rangle$. By the SSS structure, we know that

$$\langle c_1; c_2, \sigma\rangle \overset{1\ldots k_1}{\underset{q_1}{\Longrightarrow}} \langle c_1'; c_2, \sigma_1'\rangle \underset{1}{\Rightarrow}_{(Seq-1)} \langle c_2, \sigma_2'\rangle \overset{1\ldots k_2}{\underset{q_2}{\Longrightarrow}} \langle c_2', \sigma'\rangle,$$

with $k = k_1 + 1 + k_2$ and $q = q_1 * 1 * q_2$ Due the application of (Seq-1) we know that $\langle c_1', \sigma_1'\rangle \underset{1}{\Rightarrow} \sigma_2'$.

Moreover, $\langle c_1; c_2, \sigma\rangle \overset{1\ldots k_1}{\underset{q_1}{\Longrightarrow}} \langle c_1'; c_2, \sigma_1'\rangle$ was derived from a sequence of applications of (Seq-2) $\langle c_1^i; c_2, \sigma^i\rangle \underset{q_1^i}{\Longrightarrow} \langle c_1^{i+1}; c_2, \sigma^{i+1}\rangle$ with $i \leq k_1$, $q_1 = q_1^1 * \cdots * q_1^{k_1}$, $c_1^0 = c_1$ and $c_1^{k_1} = c_1'$, each one justified by the transitions $\langle c_1^i, \sigma^i\rangle \underset{q_1^i}{\Longrightarrow} \langle c_1^{i+1}, \sigma^{i+1}\rangle$, respectively. Therefore

$$\langle c_1, \sigma\rangle \overset{1\ldots k_1}{\underset{q_1}{\Longrightarrow}} \langle c_1', \sigma_1'\rangle \underset{1}{\Rightarrow} \sigma_2'$$

i.e. $\langle c_1, \sigma\rangle \overset{1\ldots k_1+1}{\underset{q_1}{\Longrightarrow}} \sigma_2'$ and hence $(\sigma, \sigma_2')_{q_1} \in \mathtt{CB}(c_1)$ (by Proposition 1). Moreover, since $\langle c_2, \sigma_2'\rangle \overset{1\ldots k_2}{\underset{q_2}{\Longrightarrow}} \langle c_2', \sigma'\rangle$, we have $(\sigma_2', \sigma')_{q_2} \in \mathtt{OB}(c_2)$, and hence $(\sigma, \sigma')_{q_1 * q_2} = (\sigma, \sigma')_q \in \mathtt{CB}(c_1) \cdot \mathtt{OB}(c_2)$.

In order to prove **(B)** we note that if $\langle c_1; c_2, \sigma\rangle \overset{1\ldots k}{\underset{q}{\Longrightarrow}} \sigma'$, we have

$$\langle c_1; c_2, \sigma\rangle \overset{1\ldots k_1}{\underset{q_1}{\Longrightarrow}} \langle c_1'; c_2, \sigma_1'\rangle \underset{1}{\Rightarrow}_{(Seq-1)} \langle c_2, \sigma_2'\rangle \overset{1\ldots k_2}{\underset{q_2}{\Longrightarrow}} \sigma'$$

with $q = q_1 * 1 * q_2$. Hence, $(\sigma, \sigma_2')_{q_1} \in \mathtt{CB}(c_1)$, $(\sigma_2', \sigma')_{q_2} \in \mathtt{OB}(c_2)$ and, therefore $(\sigma, \sigma')_{q_1 * 1 * q_2} = (\sigma, \sigma')_q \in \mathtt{CB}(c_1) \cdot \mathtt{OB}(c_2)$.

For the $\supseteq$ inclusion, let us suppose that $(\sigma, \sigma')_q \in \mathtt{OB}(c_1) \cup (\mathtt{CB}(c_1) \cdot \mathtt{OB}(c_2))$. Obviously, if $(\sigma, \sigma')_q \in \mathtt{OB}(c_1)$, then $(\sigma, \sigma')_q \in \mathtt{OB}(c_1; c_2)$ (Seq-2) of SSS. If $(\sigma, \sigma')_q \in \mathtt{CB}(c_1) \cdot \mathtt{OB}(c_2)$, there exists a state $\sigma'' \in \Sigma$ such that $(\sigma, \sigma'')_{q_1} \in \mathtt{CB}(c_1)$ and $(\sigma'', \sigma')_{q_2} \in \mathtt{OB}(c_2)$ and $q = q_1 * q_2$, i.e. $\langle c_1, \sigma\rangle \underset{q_1}{\longrightarrow} \sigma''$ and $\langle c_2, \sigma''\rangle \overset{1\ldots k_2}{\underset{q_2}{\Longrightarrow}} \sigma'$,

for some $k_2 \geq 1$. By Proposition 1, we have also $\langle c_1, \sigma\rangle \xRightarrow[q_1]{1\ldots k_1} \sigma''$ for some $k_1 \geq 1$, i.e., we have a chain

$$\langle c_1, \sigma\rangle \xRightarrow[q_1^1]{} \langle c_1^1, \sigma^1\rangle \xRightarrow[q_1^2]{} \cdots \xRightarrow[q_1^{k_1}]{} \sigma''$$

Applying (Seq-2) $k_1 - 1$ times, we get

$$\langle c_1; c_2, \sigma\rangle \xRightarrow[q_1^1]{} \langle c_1^1; c_2, \sigma^1\rangle \xRightarrow[q_1^2]{} \cdots \xRightarrow[q_1^{k_1}]{} \langle c_2, \sigma''\rangle$$

Hence, we have

$$\langle c_1; c_2, \sigma\rangle \xRightarrow[q_1^1]{} \langle c_1^1; c_2, \sigma^1\rangle \xRightarrow[q_1^2]{} \cdots \xRightarrow[q_1^{k_1}]{} \langle c_2, \sigma''\rangle \xRightarrow[q_2]{1\ldots k_2} \sigma'$$

and hence $\langle c_1; c_2, \sigma\rangle \xRightarrow[q]{1\ldots k} \sigma'$, where $q = q_1 * q_2$ and $k = k_1 + k_2$. Therefore $(\sigma, \sigma') \in \mathtt{OB}(c_1; c_2)$.

## C Proof of Proposition 11

First, we observe that, for any $b \in \mathtt{AExp}$,

$$\text{If } \langle b, \sigma\rangle \to z, \text{ then } \langle a, \sigma[x \leftarrow z]\rangle \to n \text{ iff } \langle a[x/b], \sigma\rangle \to n \tag{28}$$

This is proved by induction on the structure of $\mathtt{AExp}$:

- For $a = z' \in \mathbb{Z}$, $\langle z', \sigma[x \leftarrow z]\rangle \to z'$ and $\langle z'[x/b], \sigma\rangle = \langle z', \sigma\rangle \to z'$.
- For $a = y$, we observe, on the one hand that

  $$\langle y, \sigma[x \leftarrow z]\rangle \to \sigma[x \leftarrow z](y) = \begin{cases} z, & z = x \\ \sigma(y), & x \neq y \end{cases}$$

  and, on the other hand, that

  $$\begin{cases} \text{If } y = x, & \langle y[x/b], \sigma\rangle = \langle a, \sigma\rangle \to z \\ \text{If } y \neq x, & \langle y, \sigma\rangle = \langle b, \sigma\rangle \to \sigma(y) \end{cases}$$

- For $a = a_1 + a_2$: we observe, that $\langle a_1 + a_2, \sigma[x \leftarrow z]\rangle \to n$ iff $\langle a_1, \sigma[x \leftarrow z]\rangle \to n_1, \langle a_2, \sigma[x \leftarrow z]\rangle \to n_2$ and $n = n_1 + n_2$. Assuming $\langle b, \sigma\rangle \to z$, we have by induction hypothesis that $\langle a_1[x/b], \sigma\rangle \to n_1$ and $\langle a_2[x/b], \sigma\rangle \to n_2$ with $n = n_1 + n_2$. Hence $\langle a_1[x/b] + a_2[x/b], \sigma\rangle = \langle a_1 + a_2[x/b], \sigma\rangle \to n$. Proofs for terms $a = a_1 - a_2$ and $a = a_1 * a_2$ are analogous.

  Now we prove (26) by induction on the structure of $\varphi$.

  $$\sigma \models [\![ x := t]\!]\varphi$$

$$
\begin{array}{ll}
\Leftrightarrow & \{ \text{ defn } \models \} \\
& \text{for any } \sigma', (\sigma, \sigma') \in \texttt{OB}(\ x\ := t)(\varphi), \sigma' \models \varphi \\
\Leftrightarrow & \{ \text{ defn } \texttt{OB}(\ x\ := t)(\varphi) \} \\
& \sigma[x \leftarrow z] \models \varphi, \text{ where } \langle t, \sigma \rangle \to z
\end{array}
$$

- For $\varphi$ of form $b \in \texttt{BExp}$, for $a_1 = a_2$,

$$
\begin{array}{ll}
& \sigma[x \leftarrow z] \models a_1 = a_2 \\
\Leftrightarrow & \{ \text{ defn } \models \} \\
& \begin{cases} \langle a_1, \sigma[x \leftarrow z] \rangle \to z_1 \\ \langle a_2, \sigma[x \leftarrow z] \rangle \to z_2 \end{cases} \quad \text{and } z_1 = z_2 \\
\Leftrightarrow & \{ \text{ I.H. twice} \} \\
& \begin{cases} \langle a_1[x/b], \sigma \rangle \to z_1 & \text{with } \langle b, \sigma \rangle \to z_1 \\ \langle a_2[x/b], \sigma \rangle \to z_2 & \text{with } \langle b, \sigma \rangle \to z_2 \end{cases} \text{and } z_1 = z_2, \\
\Leftrightarrow & \{ \text{ defn } \models \} \\
& \sigma \models a_1[x/b] + a_2[x/b] \\
\Leftrightarrow & \{ \text{ substition defn} \} \\
& \sigma \models (a_1 + a_2)[x/b]
\end{array}
$$

- Proof for $\varphi$ of form $a_1 \leq a_2$, and we can get other Boolean expressions by direct application of I.H.

The proof of (27) follows directly from (26) since, as the interpretation of assignments is deterministic and total, we have that $\sigma \models [\![\ x\ := t]\!]\varphi$ iff $\sigma \models \langle\!\langle\ x\ := t\ \rangle\!\rangle \varphi$ .

# A Reusable Formalisation of Hennessy–Milner Logic for Arbitrary Transition Systems in Lean

Fabrizio Montesi[1 [0000−0003−4666−901X]], Marco Peressotti[1 [0000−0002−0243−0480]], and Alexandre Rademaker[2 [0000−0002−7583−0792]]

[1] FORM, University of Southern Denmark, Odense, Denmark
{fmontesi,peressotti}@imada.sdu.dk
[2] School for Applied Mathematics, Getulio Vargas Foundation, Rio de Janeiro, Brazil
alexandre.rademaker@fgv.br

**Abstract.** Labelled Transition Systems (LTSs) are a common substrate for modelling reactive and reconfigurable systems, while Hennessy–Milner Logic (HML) provides a standard modal language for reasoning about their behaviour. We present a library-level formalisation of HML in Lean, developed as part of the Lean Computer Science Library (CSLib), a community project providing a general foundation for formalised computer science research and verified software development. The development covers the syntax, satisfaction relation, and denotational semantics of HML, along with logical equivalence of propositions and its fundamental metatheory: correctness of the denotational semantics, bisimulation invariance, and the Hennessy–Milner theorem, which establishes that over image-finite LTSs logical equivalence coincides with bisimilarity. A distinguishing feature of the development is its genericity: definitions and theorems are parametric over CSLib's LTS API, allowing systems expressed through this interface to directly inherit the established metatheory. We demonstrate this on deterministic, nondeterministic, and $\omega$-automata, as well as the Calculus of Communicating Systems.

**Keywords:** Hennessy–Milner Logic · Labelled Transition Systems · Bisimulation · Interactive Theorem Proving · Lean · CSLib.

## 1 Introduction

Labelled Transition Systems (LTSs) are the go-to framework for defining the operational behaviour of many computational systems, including communication protocols, automata, and reactive systems [18]. Hennessy–Milner Logic (HML) [10] is a fundamental modal logic for reasoning about such systems. Its distinguishing feature is the *Hennessy–Milner theorem*: over image-finite LTSs, two states satisfy exactly the same HML propositions if and only if they are strongly bisimilar. This tight correspondence makes HML a very useful logical foundation in many contexts, from concurrency theory to model checking.

Despite its foundational status, mechanised treatments of HML at library-grade generality are scarce. Existing formalisations tend to treat variations of HML or specific kinds of LTSs. In this paper we report on a new formalisation of HML developed in Lean

as part of the Lean Computer Science Library (CSLib) – a rapidly-growing community project aimed at providing a universal basis for formalised computer science research and verified software development [2].

Our main contributions are: (1) universe-polymorphic definitions of the syntax, satisfaction relation, and denotational semantics of HML; (2) its fundamental metatheory, including correctness of the denotational semantics, bisimulation invariance, and the Hennessy–Milner theorem; (3) an instantiation of CSLib's generic logical-equivalence framework, giving congruence reasoning for HML propositions under contexts; (4) integration into CSLib with documentation and APIs for downstream use, exercised on checked instantiations to deterministic, nondeterministic and $\omega$-automata and to CCS [14]. Section 7 reports on how the development is structured around proof automation, the aspect we expect to transfer most readily to other formalisations.

The two halves of the Hennessy–Milner theorem are not comparable in difficulty: that bisimilar states are logically equivalent holds for arbitrary LTSs and is a structural induction, whereas the converse is what requires image-finiteness and is the only result whose proof is not essentially automatic, needing a noncomputable enumeration of a state's image and an appeal to choice.

We do not model reconfiguration: the LTSs considered here are fixed, and HML has no vocabulary for a transition structure that changes. What a library-level formalisation offers is that a reconfigurable model whose operational semantics is presented as a CSLib `LTS` – for instance, with states pairing a configuration and a process, and reconfiguration steps as labelled transitions – obtains the HML semantics, bisimulation invariance, and, under image-finiteness, the Hennessy–Milner theorem without further proof. Section 6 demonstrates this reuse across independently developed models, without adapting either the HML development or the models themselves. Logics that speak about reconfiguration directly, such as the relation-changing modal logics whose bisimulation theory has been mechanised in Coq [9], are a natural next target for the same infrastructure.

In the remainder, we assume some familiarity with theorem provers.

*Code and reproducibility.* Our development is publicly available in the official CSLib repository.[3] All listings in this paper are taken from commit `990e65a` and were checked with Lean 4.34.0.

## 2 Background: Transition Systems and Bisimilarity in CSLib

This section recalls relevant pre-existing CSLib infrastructure [11]. Our definitions and theorems are stated entirely in its terms, and the precise shape of these APIs is what makes the reuse described in Section 6 possible.

A *labelled transition system* is a triple $(S, L, \rightarrow)$ of states, labels, and a transition relation $\rightarrow \subseteq S \times L \times S$; we write $s \xrightarrow{\mu} s'$ for $(s, \mu, s') \in \rightarrow$. In CSLib this is a structure `LTS State Label` parametric in the state and label types, holding the relation as `Tr : State → Label → State → Prop`. (We write Lean code in `fixed width`.) The

[3] github.com/leanprover/cslib. The code lives in the modules `Cslib.Logics.HML.Basic` and `Cslib.Logics.HML.LogicalEquivalence` (note that the namespace they declare is `Cslib.Logic.HML`, in the singular), with examples in `CslibTests.HML`.

*image* of $s$ under $\mu$ is $\mathsf{image}(s, \mu) = \{s' \mid s \xrightarrow{\mu} s'\}$, defined identically in CSLib using Mathlib's set notation [6], and an LTS is *image-finite* when every such set is finite. The latter is a typeclass abbreviation, `LTS.ImageFinite := ∀ s μ, Finite (lts.image s μ)`, and this matters more than it may appear: because it is a class, it is discharged by instance search rather than by hand, and CSLib supplies instances covering the common cases – every deterministic LTS is image-finite (`deterministic_imageFinite`), as is every LTS over a finite state type (`finiteState_imageFinite`). Image-finiteness is the side condition of the Hennessy–Milner theorem, so wherever such an instance applies the theorem applies with no proof obligation on the user (Section 6).

**Definition 1 (Bisimulation and Bisimilarity).** *A relation $R \subseteq S \times S$ is a* bisimulation *if, whenever $s_1\ R\ s_2$, for every label $\mu$: (i) if $s_1 \xrightarrow{\mu} s_1'$ then $s_2 \xrightarrow{\mu} s_2'$ for some $s_2'$ with $s_1'\ R\ s_2'$; and (ii) if $s_2 \xrightarrow{\mu} s_2'$ then $s_1 \xrightarrow{\mu} s_1'$ for some $s_1'$ with $s_1'\ R\ s_2'$. States $s_1$ and $s_2$ are* bisimilar*, written $s_1 \sim s_2$, if $s_1\ R\ s_2$ for some bisimulation $R$.*

CSLib formalises this slightly more generally, for a relation between the states of two possibly *different* LTSs over different state types (which is what lets one state that a process and an automaton are bisimilar). Relations are encoded as binary predicates, and clauses (i) and (ii) appear literally as the two conjuncts.

```
def IsBisimulation (lts₁ : LTS State₁ Label) (lts₂ : LTS State₂ Label)
    (r : State₁ → State₂ → Prop) : Prop :=
  ∀ {{s₁ s₂}}, r s₁ s₂ → ∀ μ, (
    (∀ s₁', lts₁.Tr s₁ μ s₁' → ∃ s₂', lts₂.Tr s₂ μ s₂' ∧ r s₁' s₂') ∧
    (∀ s₂', lts₂.Tr s₂ μ s₂' → ∃ s₁', lts₁.Tr s₁ μ s₁' ∧ r s₁' s₂'))

def Bisimilarity (lts₁ : LTS State₁ Label) (lts₂ : LTS State₂ Label) :
    State₁ → State₂ → Prop :=
  fun s₁ s₂ => ∃ r, r s₁ s₂ ∧ IsBisimulation lts₁ lts₂ r
```

Definition 1 is the homogeneous special case, which CSLib names `IsHomBisimulation lts := IsBisimulation lts lts` and `HomBisimilarity lts := Bisimilarity lts lts`. Our results concern one LTS at a time and so are stated with the `Hom` variants throughout; we write $\sim$ for `HomBisimilarity`. Lifting the metatheory to the heterogeneous setting is a natural next step that we leave to future work, since standard HML speaks about a single transition system.

## 3 HML Formalisation in Lean

We now describe the core components of our formalisation of HML: syntax of propositions, satisfaction (semantics in terms of transitions), denotational semantics, and theory equivalence.

### 3.1 Propositions

Given a set of transition labels $L$, the language of HML propositions is given by the grammar:

$$\varphi ::= \top \mid \varphi \wedge \varphi \mid \neg\varphi \mid \langle\mu\rangle\varphi \qquad \text{where } \mu \in L.$$

Propositions state properties about a given state. The modality $\langle\mu\rangle\varphi$ ('diamond') asserts the existence of a transition with label $\mu$ from the given state to a state that satisfies $\varphi$. We formalise this in Lean as the next inductive type

```
inductive Proposition (Label : Type u) : Type u where
  | true | and (φ₁ φ₂ : Proposition Label) | not (φ : Proposition Label)
  | diamond (μ : Label) (φ : Proposition Label)
```

and instantiate the appropriate Mathlib and CSLib typeclasses (e.g., `HasAnd`) to offer the usual notation, e.g., $\varphi_1 \wedge \varphi_2$ stands for `and` $\varphi_1\ \varphi_2$. At the time of writing, CSLib expresses $\langle\mu\rangle\varphi$ as $\mathtt{d}\langle\mu\rangle\varphi$, where the 'd' stands for 'dynamic'.

There are two popular approaches to the language of propositions in HML. The original includes a negation connective [10], while some later developments drop negation and instead require both $\top$ and $\bot$ as primitives [1]. We follow the original, and push the design further: we keep the set of *primitive* connectives minimal and recover every other connective as a derived construct in the standard way for modal logics [5]. Each remaining connective is thus a one-line definition – `Proposition.false` $:= \neg\top$, `or` $\varphi_1\ \varphi_2 := \neg(\neg\varphi_1 \wedge \neg\varphi_2)$, `box` $\mu\ \varphi := \neg\mathtt{d}\langle\mu\rangle\neg\varphi$, and similarly for implication and bi-implication – each carrying `@[match_pattern]` so it remains usable in pattern matches, and each accompanied by a characterisation lemma (Section 3.2) proved once and reused everywhere afterwards. Keeping `Proposition` minimal in this way has the advantage that every proof by induction on propositions – including correctness of the denotational semantics (Theorem 3) and bisimulation invariance (Theorem 4) – has just four cases.

The detail of what is primitive or derived is otherwise largely immaterial. Both primitive and derived connectives are registered with CSLib's notation classes, so propositions are written with standard symbols: $\top$, $\bot$, $\wedge$, $\vee$, $\neg$, $\rightarrow$, $\leftrightarrow$, $\mathtt{d}\langle\mu\rangle\varphi$, and $\mathtt{d}[\mu]\varphi$. All theorems use notation instead of the raw Lean definitions.

The definition of `Proposition` is universe-polymorphic in the label type (`Type u`). Propositions can thus be formed over label types in any universe, which lets the development compose with CSLib's other universe-polymorphic structures without universe constraint clashes during elaboration.

### 3.2 Satisfaction Relation

The satisfaction relation $s \models \varphi$ ('state $s$ satisfies proposition $\varphi$') is defined by the following clauses.

| | | | |
|---|---|---|---|
| $s \models \top$ | always | $s \models \neg\varphi$ | if $s \not\models \varphi$ |
| $s \models \varphi_1 \wedge \varphi_2$ | if $s \models \varphi_1$ and $s \models \varphi_2$ | $s \models \langle\mu\rangle\varphi$ | if $\exists s'.\ s \xrightarrow{\mu} s' \wedge s' \models \varphi$ |

Because negation is primitive, satisfaction cannot be given as an inductive predicate: the clause for $\neg\varphi$ is a negative occurrence, which Lean's positivity check rejects. We define it instead by structural recursion into `Prop`.

```
def Satisfies (lts : LTS State Label) (s : State) :
    Proposition Label → Prop
  | .true => True
```

```
  | .and φ₁ φ₂ => Satisfies lts s φ₁ ∧ Satisfies lts s φ₂
  | .not φ => ¬Satisfies lts s φ
  | .diamond μ φ => ∃ s', lts.Tr s μ s' ∧ Satisfies lts s' φ
```

CSLib provides a general interface for inference systems, which we instantiate for HML. A `Judgement` bundles an LTS, a state and a proposition. We establish the notation `HML[lts,s ⊨ φ]` for such judgements. We then instantiate CSLib's `InferenceSystem` typeclass with these judgements and our `Satisfies`, obtaining the notation `⇓HML[lts,s ⊨ φ]` for `Satisfies lts s φ` – ⇓ stands for 'derivation'. We use the two interchangeably, and write $s \models \varphi$ when the LTS is clear from context. Going through this interface is what lets HML inherit CSLib's generic machinery for logical judgements, in particular logical equivalence (Section 5).

A good consequence of our setup is that recursive definition unfolds definitionally, so the characterisation lemmas for the derived connectives hold by `rfl` or by a single automated step, whereas an inductive predicate would require inversion lemmas. For instance, the correctness of negation, which in a negation-free presentation is a theorem proved by induction, is here literally reflexivity.

```
theorem Satisfies.not_iff_not :
    ⇓HML[lts,s ⊨ ¬φ] ↔ ¬⇓HML[lts,s ⊨ φ] := by rfl
```

The derived connectives are characterised by results of similar shape, and it is these – not the definitions – that the rest of the development uses.

```
theorem Satisfies.or_iff_or :
    ⇓HML[lts,s ⊨ φ₁ ∨ φ₂] ↔ ⇓HML[lts,s ⊨ φ₁] ∨ ⇓HML[lts,s ⊨ φ₂]
theorem Satisfies.box_iff_forall :
    ⇓HML[lts,s ⊨ d[μ]φ] ↔ ∀ s', lts.Tr s μ s' → ⇓HML[lts,s' ⊨ φ]
```

### 3.3 Denotational Semantics

HML comes with a compositional *denotational semantics*, mapping each proposition to the set of states that satisfy it.

$$[\![\top]\!] = S \quad \text{(the set of all states)} \qquad [\![\neg\varphi]\!] = \overline{[\![\varphi]\!]} \quad \text{(complement)}$$

$$[\![\varphi_1 \wedge \varphi_2]\!] = [\![\varphi_1]\!] \cap [\![\varphi_2]\!] \qquad [\![\langle\mu\rangle\varphi]\!] = \{s \mid \exists s'.\, s \xrightarrow{\mu} s' \wedge s' \in [\![\varphi]\!]\}$$

The translation to Lean mirrors the four clauses.

```
def Proposition.denotation (lts : LTS State Label) :
    Proposition Label → Set State
  | .true => Set.univ
  | .and φ₁ φ₂ => φ₁.denotation lts ∩ φ₂.denotation lts
  | .not φ => (φ.denotation lts)ᶜ
  | .diamond μ φ => {s | ∃ s', lts.Tr s μ s' ∧ s' ∈ φ.denotation lts}
```

The derived connectives then denote the expected sets – $[\![\bot]\!] = \emptyset$, $[\![\varphi_1 \vee \varphi_2]\!] = [\![\varphi_1]\!] \cup [\![\varphi_2]\!]$, and $[\![[\mu]\varphi]\!] = \{s \mid \forall s'.\, s \xrightarrow{\mu} s' \Rightarrow s' \in [\![\varphi]\!]\}$ – but we prove these as consequences rather than hardcode them as clauses.

### 3.4 Theories and Theory Equivalence

Satisfaction yields the concepts of theory, theory equivalence (sometimes called 'logical equivalence' in previous work), and distinguishing propositions.

**Definition 2 (Theory and Theory Equivalence).** *The* theory *of a state $s$ is the set of all propositions it satisfies:* $\mathsf{theory}(s) = \{\varphi \mid s \models \varphi\}$. *Two states are* theory equivalent, *written* $\mathsf{TheoryEq}(s_1, s_2)$, *if* $\mathsf{theory}(s_1) = \mathsf{theory}(s_2)$.

We translate this as:

```
abbrev theory (lts : LTS State Label) (s : State) :
    Set (Proposition Label) := {φ | ⇓HML[lts,s ⊨ φ]}
abbrev TheoryEq (lts : LTS State Label) (s1 s2 : State) :=
  theory lts s1 = theory lts s2
```

The key lemma of the whole development is that two states that are not theory equivalent admit a distinguishing proposition.

```
lemma not_theoryEq_satisfies (h : ¬TheoryEq lts s1 s2) :
    ∃ φ, (⇓HML[lts,s1 ⊨ φ] ∧ ¬⇓HML[lts,s2 ⊨ φ])
```

Having a notion of negation is important here. Two states with different theories differ on some proposition, but *a priori* the difference could go either way: without negation the conclusion would have to read `∃ φ, (⇓HML[lts,s1 ⊨ φ] ∧ ¬⇓HML[lts,s2 ⊨ φ]) ∨ (¬⇓HML[lts,s1 ⊨ φ] ∧ ⇓HML[lts,s2 ⊨ φ])`, and every use site would have to dispatch both disjuncts. With negation available, the second case is reduced to the first by replacing $\varphi$ with $\neg\varphi$, and the proof is a single automated step. Section 4.3 shows where this pays off: the proof of the Hennessy–Milner theorem invokes this lemma once per state in the image of a state, and a two-sided version would double that case analysis at each invocation.

## 4 Metatheory

We move to our key results, which formalise the metatheory of HML.

### 4.1 Correctness of the Denotational Semantics

The inductive satisfaction relation and the denotational semantics coincide.

**Theorem 3 (Semantic Equivalence).** *$s \in [\![\varphi]\!]$ if and only if $s \models \varphi$.*

The Lean proof follows the same principle found in the literature – we proceed by induction on the structure of the proposition. (The curly brackets denote an implicit parameter, which Lean tries to find in the context during elaboration.)

```
theorem mem_denotation_iff_satisfies {φ : Proposition Label} :
    s ∈ φ.denotation lts ↔ ⇓HML[lts,s ⊨ φ] := by
  induction φ generalizing s <;> grind [=_ derivation_def]
```

A companion result, `theoryEq_denotation_eq`, lifts this from single propositions to theories: theory equivalence coincides with denotational equivalence.

### 4.2 Bisimulation Invariance

A key property of HML is that bisimilar states cannot be logically distinguished.

**Theorem 4 (Bisimulation Invariance).** *If $R$ is a bisimulation and $s_1\ R\ s_2$, then $s_1 \models \varphi$ if and only if $s_2 \models \varphi$.*

The theorem statement in Lean reads as follows.

```
lemma bisimulation_satisfies {hrb : lts.IsHomBisimulation r}
    (hr : r s1 s2) (φ : Proposition Label) :
    ⇓HML[lts,s1 ⊨ φ] ↔ ⇓HML[lts,s2 ⊨ φ] := by
  induction φ generalizing s1 s2 <;> grind [IsBisimulation]
```

The proof is by induction on $\varphi$, but the biconditional form is not a cosmetic strengthening: it is forced by the choice of primitive negation. In the `.not` case the induction hypothesis must be used contrapositively, so an implication-shaped statement does not go through, and one would have to prove the two directions by simultaneous induction instead. Stating the lemma as an equivalence makes the induction hypothesis strong enough for all four cases and lets a single `grind` invocation, hinted with the definition of bisimulation, close them.

With this theorem and the fact that bisimilarity is symmetric, we can prove that bisimulation implies theory equivalence.

**Corollary 5 (Bisimulation implies Theory Equivalence).** *If $R$ is a bisimulation and $s_1\ R\ s_2$, then* $\mathsf{TheoryEq}(s_1, s_2)$.

```
lemma bisimulation_theoryEq {hrb : lts.IsHomBisimulation r}
    (hr : r s1 s2) : TheoryEq lts s1 s2 := by grind
```

### 4.3 Characterisation of Bisimilarity (Hennessy–Milner Theorem)

The converse of Corollary 5 – theory equivalence implies bisimilarity – holds only for image-finite LTSs. This is because HML propositions can completely specify only a finite number of behaviours in the image of a state. The proof proceeds by showing that theory equivalence is itself a bisimulation.

**Lemma 6.** *In any image-finite LTS,* $\mathsf{TheoryEq}$ *is a bisimulation.*

Using CSLib's API, we formulate the lemma as follows.

```
theorem theoryEq_isBisimulation (lts : LTS State Label)
    [image_finite : ∀ s μ, Finite (lts.image s μ)] :
    lts.IsHomBisimulation (TheoryEq lts)
```

The argument is the textbook one, and we claim no novelty for it; what is worth reporting is how it decomposes in a proof assistant, since this is where a mechanised account differs from a pen-and-paper one. Assume $s_1$ and $s_2$ are theory equivalent and consider the case $s_1 \xrightarrow{\mu} s_1'$. We show that $s_2$ failing to match this transition leads to

a contradiction: assume every $\mu$-derivative of $s_2$ is distinguishable from $s_1'$. Image-finiteness lets us enumerate the $\mu$-derivatives of $s_2$, obtain a distinguishing proposition $\varphi_i$ for each, and assemble $\langle\mu\rangle \bigwedge_{i=1}^{n} \varphi_i$, which $s_1$ satisfies and $s_2$ does not.

Three steps of this argument are not visible on paper but dominate the formal proof. First, the enumeration is noncomputable: image-finiteness gives `Finite`, and we must pass to `Fintype.ofFinite` to obtain an actual list of derivatives. Second, choosing one distinguishing proposition per derivative is an appeal to choice, performed by Lean's `choose` tactic, which turns a family of existentials into a function `dist_formula` together with its specification. Third, the conjunction $\bigwedge_i \varphi_i$ is not a primitive: it is `finiteAnd`, a fold of binary conjunction over a list, and reasoning about it requires the characterisation lemma `Satisfies.finiteAnd_iff_forall` of Section 3.2.

Rather than inline these steps, we factor them into a reusable section whose lemmas are stated for an arbitrary assignment `stateMap` of propositions to the elements of an image: `propositions` builds the list of propositions over the (now finite) image, `propositions_complete` states that every derivative's proposition occurs in it, and `propositions_satisfies_conjunction` assembles the diamond formula $\langle\mu\rangle \bigwedge_i \varphi_i$ from a witnessing transition. With these in place each direction of Lemma 6 is about fifteen lines, and the two are genuinely symmetric rather than symmetric-modulo-bookkeeping. The factoring is deliberate: turning 'collect the image into a distinguishing conjunction' into named, independently usable lemmas is what makes the development library-grade rather than a self-contained script, and we expect them to be reused by extensions that repeat this style of argument.

The Hennessy–Milner theorem follows from Corollary 5 and Lemma 6.

**Theorem 7 (Hennessy–Milner).** *For any image-finite LTS,* $\mathsf{TheoryEq} = \sim$.

The Lean translation is immediate.

```
theorem theoryEq_eq_bisimilarity {lts : LTS State Label}
    [image_finite : ∀ s μ, Finite (lts.image s μ)] :
    TheoryEq lts = HomBisimilarity lts
```

Note that only this direction needs image-finiteness; Corollary 5 holds for arbitrary LTSs.

## 5 Logical Equivalence

So far we have focused on whether a *state* satisfies a proposition; another interesting question is when two *propositions* mean the same thing, and here integrating satisfaction with CSLib's judgement interface gives a considerable pay off, since the library has a generic notion of logical equivalence that we need only instantiate.

We define two propositions to be equivalent with respect to an LTS when every state agrees on them, and *universally* equivalent when this holds for every LTS over every state type.

```
def Proposition.Equiv (lts : LTS State Label)
    (φ₁ φ₂ : Proposition Label) := ∀ (s : State), ⇓HML[lts,s ⊨ φ₁ ↔ φ₂]
```

```
def Proposition.UEquiv.{u, v} {Label : Type v}
    (φ₁ φ₂ : Proposition Label) :=
  ∀ {{State : Type u}} (lts : LTS State Label), φ₁ ≡[Equiv lts] φ₂
```

Note that `UEquiv` quantifies over state types in a universe independent of the label type's, which is expressible only because `Proposition` is polymorphic (Section 3.1).

Having these definitions is not enough to reason with; what one wants is to replace a subproposition by an equivalent one *inside* a larger proposition. Therefore, we prove that these relations are instances of CSLib's `LogicalEquivalence` typeclass, giving access to the standard notation ≡ for them. This requires proving a series of properties. First, they are equivalence relations. Second, they are lawful congruences, i.e., they are closed under any syntactic context for propositions and thus allow for local rewriting of propositions. Third, they preserve derivability (`Satisfies` is preserved by logical equivalence). Again, these proofs benefit from the low number of cases given by `Proposition`.

We prove a small database of expected logical equivalences, for example the standard dual axiom of modal logic given next.

```
theorem dual (μ : Label) (φ : Proposition Label) : (d⟨μ⟩φ) ≡ (¬d[μ]¬φ)
```

These results can then be combined with CSLib's machinery for logical equivalence and derivability. The following example uses `dual` in combination with closure under propositional contexts to derive `(¬d[μ]¬φ₁) ∧ φ₂` from a proof of `(d⟨μ⟩φ₁) ∧ φ₂`.

```
example (h : ⇓HML[lts,s ⊨ (d⟨μ⟩φ₁) ∧ φ₂]) :
    ⇓HML[lts,s ⊨ (¬d[μ]¬φ₁) ∧ φ₂]
```

## 6 Applicability

Because every definition and theorem above is parametric in an `LTS`, they apply verbatim to any CSLib development – or downstream project – that presents its semantics through that API. We illustrate this aspect in this section through a few examples. First, we demonstrate how our formalisation of the Hennessy–Milner theorem applies to deterministic automata, nondeterministic and $\omega$-automata, and CCS. Then, we discuss how HML specifications are checked against such models. The mathematical content of these instances is classical; what we report is that obtaining them mechanically costs nothing beyond image-finiteness, which is often discharged by instance search.

*Deterministic automata.* A deterministic automaton (`DA`) in CSLib is built on the concept of *functional* LTS, `FLTS`, which carries a transition function rather than a relation; `FLTS .toLTS` converts it to an LTS. Because the obtained transition relation is deterministic, image-finiteness is discharged by instance search, and the Hennessy–Milner theorem applies with no hypothesis at all.

```
example (da : DA State Symbol) :
    TheoryEq da.toFLTS.toLTS = HomBisimilarity da.toFLTS.toLTS :=
  theoryEq_eq_bisimilarity ..
```

*Nondeterministic and $\omega$-automata.* A nondeterministic automaton (`NA`) in CSLib extends `LTS` directly, and a nondeterministic Büchi automaton `NA.Buchi` extends `NA`, so our results apply to both directly. Here image-finiteness is a genuine assumption, as it must be, and it is the only one.

```
example {State Symbol : Type} (b : NA.Buchi State Symbol)
    [∀ s μ, Finite (b.toNA.toLTS.image s μ)] :
    TheoryEq b.toNA.toLTS = HomBisimilarity b.toNA.toLTS :=
  theoryEq_eq_bisimilarity ..
```

When the state type is finite – the usual setting for automata – even that assumption is discharged automatically, by the `finiteState_imageFinite` instance of Section 2.

*CCS.* The same holds of CSLib's Calculus of Communicating Systems [14], for any set `defs` of recursive procedure definitions respecting image-finiteness (e.g., under guarded recursion [18]).

```
example [∀ p μ, Finite ((CCS.lts (defs := defs)).image p μ)] :
    TheoryEq (CCS.lts (defs := defs))
      = HomBisimilarity (CCS.lts (defs := defs)) :=
  theoryEq_eq_bisimilarity ..
```

None of these instantiations required any change to the HML development or to the automata and CCS formalisations, nor any adapter code. Moreover, an HML specification is meaningful against all of them at once, so one can prove that a CCS process and an automaton satisfy the same proposition despite being described in completely different terms.

As an example of the latter point, consider the typical HML proposition distinguishing the two versions of Milner's vending machine: `d[Coin](d⟨Tea⟩⊤ ∧ d⟨Coffee⟩⊤)` (after inserting a coin, one can choose between tea and coffee). Proving that a system satisfies such a proposition reduces, via the characterisation lemmas of Section 3.2, to exhibiting transitions for each diamond and enumerating them for each box; and by Theorem 4, a system that branches after the coin and one that branches nondeterministically on it are thereby proved *not* bisimilar, whether each is given in CCS or as an automaton.

## 7 Proof Automation and Engineering

The development is short – around 560 lines across two modules – and the reason is worth reporting, because it is the part of this work that transfers to other formalisations rather than being specific to HML.

*Structuring definitions for `grind`.* Most of our semantic results carry carefully chosen `@[scoped grind]` annotations, registering it with Lean's `grind` tactic, a proof-search procedure combining congruence closure, rewriting, and instantiation of annotated facts [15]. Choosing these annotations is a main design activity. Our rule of thumb, arrived at by trial and error, is that *characterisation* lemmas become rewrite rules (`grind =`) and *structural* facts forward rules (`grind ⇒`): `Satisfies.and_iff_and` and `Satisfies.box_iff_forall` are rewrites, so any goal about a derived connective reduces to one

about primitives, whereas `theoryEq_isBisimulation` is a forward rule, so a `TheoryEq` hypothesis makes the bisimulation property available. The effect is that most proofs involving satisfaction in HML can be automatically synthesised by grind.

*What automation does and does not close.* Four results in Section 4 are closed by a single `grind` call, in two cases after an `induction` $\varphi$: correctness of the denotational semantics, its extension to theories, bisimulation invariance, and the existence of one-sided distinguishing propositions. A second group needs a one- or two-line skeleton – e.g., extensionality or a case split – but no domain reasoning. An interesting exception is Lemma 6: automation could not find the proof by contradiction, the passage from `Finite` to `Fintype`, or the appeal to choice; these are supplied by hand. What automation can handle, once that skeleton is in place, is the bookkeeping between the steps: `grind` removed the tedium but none of the mathematics.

*Maintenance.* Automation plays an important role for maintenance. After our first version of HML was integrated into CSLib, bisimulation got generalised to relate two LTSs, renaming `IsBisimulation` to `IsHomBisimulation`. Our development remained practically unaffected, modulo applying this renaming to some signatures. No proof changed, because a proof reading `induction` $\varphi$ `<;> grind` does not name the lemmas it uses. The grind tactic could latch on the new bisimulation lemmas without manual intervention.

## 8 Related Work

Variations and extensions of HML have been formalised in several provers [13,7,19,12], and fully verified model-checking procedures have been developed in Isabelle. These target particular logics or particular transition systems; none aims at the general metatheory of HML over an arbitrary LTS, which is our concern here. Modal logic over Kripke models has also been mechanised, in some respects more generally than here. De Wind [21] formalised modal logic in Coq; Benzmüller et al. [4] verified the modal logic cube in Isabelle/HOL via a semantic embedding in higher-order logic; D'Abrera and Goré [8] synthesised verified Sahlqvist correspondents in Coq; and Fervari et al. [9] mechanised in Coq bisimulation theorems for relation-changing modal logics, whose operators modify the model during evaluation and thus extend the static setting considered here. These works generalise along the *logic* – richer axiomatics, frame correspondence, model-changing operators – whereas ours generalises along the *model*: the metatheory is stated once over the transition-system interface of a shared library and instantiated, with checked proofs, to formalisations developed independently of it. The two directions are compatible, and bringing relation-changing operators to CSLib's `LTS` API is how we would expect to obtain an HML able to speak about reconfiguration. Within CSLib itself, HML was the first modal logic to be formalised, and it has since been joined by a general modal logic development (`Cslib.Logics.Modal`) covering Kripke semantics and the frame properties corresponding to the standard axioms. The two are complementary and deliberately separate: the modal development is parameterised by Kripke frames with state-labelling by atomic propositions, whereas HML is parameterised by

CSLib's LTS, where transitions rather than states carry the labels. Relating the two – in effect, presenting an LTS as a multi-modal frame – is an obvious and worthwhile piece of future work that would let each inherit results from the other.

The closest prior work is the Isabelle development of Parrow et al. [17,16], formalised in the Archive of Formal Proofs [20]. It introduces nominal transition systems and an extension of HML for them, and proves the Hennessy–Milner theorem in that setting. The extension is motivated by process calculi with name mobility, such as the $\pi$-calculus, where the derivative of a transition may depend on names bound by the transition label; reasoning about these requires propositions that can depend on the bound names appearing in modalities. Their propositional language is thus strictly richer than standard HML, as in our case.

The two *formalisations* are nonetheless incomparable rather than nested. The AFP entry contains no denotational semantics: the journal version defines one for the $\mu$-calculus extension but excludes it from the Isabelle development for lack of support for transfinite induction, whereas ours is machine-checked together with its correctness theorem. Nor does the AFP entry instantiate its framework to any concrete system – the journal version calls CCS and the $\pi$-calculus trivial instances but leaves this unproved – whereas Section 6 checks instantiations to automata and CCS formalisations developed independently of this work.

Why a different path rather than a port of [17]? The decisive factor was not expressiveness but the existence of a library to build on: CSLib already had an LTS API with a substantial population of clients, so the design constraint was to state HML in exactly those terms. A nominal framework introduces its own type signatures, and previous Isabelle developments of CCS [3] would require integration work to connect to it – the very problem our design exists to avoid. We believe that exploring extensions of our development to nominal transition systems as in [17] and, in particular, their bridging to existing APIs represents interesting future work.

## 9 Conclusion

We have presented a library-level formalisation of HML in CSLib, covering syntax, satisfaction, denotational semantics, logical equivalence, and the metatheory up to the Hennessy–Milner theorem, and we have shown that it readily applies to all developments which present their semantics through CSLib's LTS API with no extra work.

Two extensions strike us as both tractable and on-theme. The first is to lift the metatheory to CSLib's heterogeneous bisimulation (Section 2), letting HML compare states of two *different* transition systems – the natural formulation of 'the reconfigured system refines the original'. The second is to add fixpoint operators – HML with recursion [1] or, more generally, the modal $\mu$-calculus [13] – which is the form of the logic most relevant to verification, as it expresses the recurrence and reachability properties reconfigurable models are usually specified with. We kept to finitary HML here because it is the setting of the Hennessy–Milner theorem; fixpoints require the lattice of state sets as semantic domain, and the image-to-conjunction lemmas of Section 4.3 and the congruence infrastructure of Section 5 were built with such extensions in mind.

**Acknowledgments.** We thank Christopher Henson for comments on our code. Alexandre Rademaker gratefully acknowledges support from Schmidt Sciences. Co-funded by the European Union (ERC, CHORDS, 101124225). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

## References

1. Aceto, L., Ingólfsdóttir, A.: Testing Hennessy-Milner logic with recursion. In: Thomas, W. (ed.) Foundations of Software Science and Computation Structure, Second International Conference, FoSSaCS'99, Held as Part of the European Joint Conferences on the Theory and Practice of Software, ETAPS'99, Amsterdam, The Netherlands, March 22-28, 1999, Proceedings. Lecture Notes in Computer Science, vol. 1578, pp. 41–55. Springer (1999). https://doi.org/10.1007/3-540-49019-1_4, https://doi.org/10.1007/3-540-49019-1_4
2. Barrett, C., Chaudhuri, S., Montesi, F., Grundy, J., Kohli, P., de Moura, L., Rademaker, A., Yingchareonthawornchai, S.: CSLib: The Lean Computer Science Library (2026), https://arxiv.org/abs/2602.04846
3. Bengtson, J.: Ccs in nominal logic. Archive of Formal Proofs (May 2012), https://isa-afp.org/entries/CCS.html, Formal proof development
4. Benzmüller, C., Claus, M., Sultana, N.: Systematic verification of the modal logic cube in Isabelle/HOL. In: Proceedings Fourth Workshop on Proof eXchange for Theorem Proving, PxTP 2015. EPTCS, vol. 186, pp. 27–41 (2015). https://doi.org/10.4204/EPTCS.186.5
5. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic, Cambridge Tracts in Theoretical Computer Science, vol. 53. Cambridge University Press (2001). https://doi.org/10.1017/CBO9781107050884, https://doi.org/10.1017/CBO9781107050884
6. mathlib Community, T.: The Lean mathematical library. In: Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs. p. 367–381. CPP 2020, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3372885.3373824, https://doi.org/10.1145/3372885.3373824
7. Coupet-Grimal, S.: An axiomatization of linear temporal logic in the Calculus of Inductive Constructions. J. Log. Comput. **13**(6), 801–813 (2003). https://doi.org/10.1093/LOGCOM/13.6.801, https://doi.org/10.1093/logcom/13.6.801
8. D'Abrera, C., Goré, R.: Verified synthesis of (very simple) Sahlqvist correspondents via Coq. In: Advances in Modal Logic (AiML 2018), Short Presentations. pp. 26–30. College Publications (2018)
9. Fervari, R., Trucco, F., Ziliani, B.: Verification of dynamic bisimulation theorems in Coq. J. Log. Algebraic Methods Program. **120**, 100642 (2021). https://doi.org/10.1016/j.jlamp.2021.100642
10. Hennessy, M., Milner, R.: Algebraic laws for nondeterminism and concurrency. Journal of the ACM **32**(1), 137–161 (1985)
11. Henson, C., Montesi, F.: Computer science as infrastructure: the spine of the lean computer science library (cslib). CoRR **abs/2602.15078** (2026). https://doi.org/10.48550/ARXIV.2602.15078, https://doi.org/10.48550/arXiv.2602.15078
12. van Hulst, A.C., Reniers, M.A., Fokkink, W.J.: Maximal synthesis for Hennessy-Milner logic. ACM Trans. Embed. Comput. Syst. **14**(1), 10:1–10:21 (2015). https://doi.org/10.1145/2680540, https://doi.org/10.1145/2680540
13. Miculan, M.: On the formalization of the modal $\mu$-Calculus in the Calculus of Inductive Constructions. Inf. Comput. **164**(1), 199–231 (2001). https://doi.org/10.1006/INCO.2000.2902, https://doi.org/10.1006/inco.2000.2902

14. Milner, R.: A Calculus of Communicating Systems, Lecture Notes in Computer Science, vol. 92. Springer (1980). https://doi.org/10.1007/3-540-10235-3, https://doi.org/10.1007/3-540-10235-3
15. Morrison, K., de Moura, L.: grind: An smt-inspired tactic for lean 4 (short paper - system description). In: Biere, A., Lutz, C., Negri, S. (eds.) Automated Reasoning - 13th International Joint Conference, IJCAR 2026, Lisbon, Portugal, July 26-29, 2026, Proceedings, Part I. Lecture Notes in Computer Science, vol. 16688, pp. 106–114. Springer (2026). https://doi.org/10.1007/978-3-032-32589-1_7, https://doi.org/10.1007/978-3-032-32589-1_7
16. Parrow, J., Borgström, J., Eriksson, L., Gutkovas, R., Weber, T.: Modal logics for Nominal Transition Systems. In: 26th International Conference on Concurrency Theory (CONCUR 2015). LIPIcs, vol. 42, pp. 198–211. Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2015). https://doi.org/10.4230/LIPIcs.CONCUR.2015.198
17. Parrow, J., Borgström, J., Eriksson, L., Gutkovas, R., Weber, T.: Modal logics for Nominal Transition Systems. Log. Methods Comput. Sci. **17**(1) (2021), https://lmcs.episciences.org/7137
18. Sangiorgi, D.: Introduction to Bisimulation and Coinduction. Cambridge University Press, Cambridge (2011)
19. Tsai, M.H., Wang, B.Y.: Formalization of CTL* in the Calculus of Inductive Constructions. In: Okada, M., Satoh, I. (eds.) Advances in Computer Science - ASIAN 2006. Secure Software and Related Issues. pp. 316–330. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)
20. Weber, T., Eriksson, L.H., Parrow, J., Borgström, J., Gutkovas, R.: Modal logics for Nominal Transition Systems. Archive of Formal Proofs (2016), https://isa-afp.org/entries/Modal_Logics_for_NTS.html, Formal proof development
21. de Wind, P.: Modal Logic in Coq. Master's thesis, Vrije Universiteit Amsterdam (2002)

# Typed Modalities and Friends

Ionuţ Ţuţu[1] and Claudia Elena Chiriţă[2]

[1] Simion Stoilow Institute of Mathematics
of the Romanian Academy, Romania
ittutu@gmail.com
[2] University of Bucharest, Romania
claudia.elena.chirita@gmail.com

**Abstract.** There is a growing domain of modal-logic approaches to modelling and reasoning about reconfigurable systems. Notable examples include hybrid logics, relation-changing logics, and reactive frames. Each extends basic modal logic in some significant way, such as naming and referring to specific states, or supporting various updates to relational structures during sentence evaluation. In this paper, we study the kindred framework of dynamic networks of interactions (DNI). Its distinguishing feature is the use of typed modalities, which, unlike ordinary modalities, enable us to track reconfigurations, identifying which actants trigger them and what are their outcomes. Our focus is on exploring connections between DNI and other modal logics. To that end, we formalize DNI as a stratified institution and show how it can embed or encode other modal formalisms by means of comorphisms. We then prove that, despite allowing for more complex system dynamics, DNI can in turn be translated in a conservative manner into first-order modal logic.

**Keywords:** Reconfigurable systems · Modal logic · Typed modalities · Stratified institutions · Comorphisms

## 1 Introduction

For the longest time, modal operators have been a quintessential part of logical toolkits for exploring graph structures. As advances in system development have led to increasingly complex graphs, modal operators and the logics they belong to have evolved also. Recently, there has been a shift in interest towards graphs whose structure, or configuration, may change, for example, with the passing of time, due to external events, or as a result of certain interactions between their components. Relation-changing logics, including Johan van Benthem's sabotage logics [3], Dov Gabbay's reactive frames [13], and towers of hybridization [8,18] are representative logics for this new class of structures. Some of these logics employ dynamic operators to update accessibility relations in some predetermined way, while others consider the full space of possible configurations together with suitable transitions between them, leading to a higher-level structure of configurations. But all of these frameworks address the intuitive notion of reconfiguration.

The logic of dynamic networks of interactions (DNI) [28] adheres to the second approach. In many ways, it is similar to a hybridized logic: it defers to a base logic on capturing configurations, and uses modal and hybrid operators to model changes. But it diverges from the hybridization approach by using a new kind of modality with inner and outer types to indicate which components are involved in a reconfiguration. The inner types match triggers, while the outer ones match outcomes. When coupled with quantifiers, typed modalities allow for a finer-grained record of reconfigurations and for tracking interactions.

Previously, we have shown that the standard translation of modal logic into first-order logic can be extended to typed modalities, but the actual placement of DNI in the universe of modal logics remained open. This paper is an attempt to address that question by showing how the typed-modal operators of DNI are related to their conventional, untyped counterparts. We do so through the use of stratified institutions [1,6], which refine Goguen and Burstall's original notion [14] in order to accommodate models with states (configurations).

The structure and contributions of the paper are as follows: In Section 2, we recall a trimmed down version of DNI that retains typed modalities, while dropping orthogonal features already present in other modal logics. We illustrate the main elements of the logical framework through a simple game. In Section 3, we show how DNI can be formalized as a stratified institution and explain how other modal logics, in particular first-order modal logic, can be presented similarly, as fragments of DNI. In the last section, we introduce stratified comorphisms to capture mappings between stratified institutions. We present embedding comorphisms of first-order modal logics into DNI, and suggest encoding comorphisms from relation-changing logics. More significantly, we then show that there also exists a comorphism from DNI to first-order modal logic, thus encoding typed modalities in an untyped setting. Finally, we develop an unravelling technique for DNI to prove that the encoding into first-order modal logic is conservative.

## 2 Dynamic networks of interactions

In its original formulation, DNI is a complex logical system that brings together aspects such as many-sortedness, partiality, varying-domain semantics, hybrid operators, and both possibilist and actualist quantifications. But the main, distinctive feature of the logic is the use of typed modalities. Since, in this paper, we focus almost exclusively on typed modalities, we present a simplified version of DNI without the aforementioned aspects. We base this version of DNI on single-sorted first-order logic. We write first-order signatures as pairs $\langle F, P\rangle$, where $F$ and $P$ are arity-indexed families of sets of function and predicate symbols. Terms and sentences are defined as usual – see, e.g., [7, Chapter 3]. For any first-order structure, or model, $M$ for a signature $\langle F, P\rangle$, we denote the interpretation of a function symbol, term, or predicate symbol $\varsigma$ in $M$ by $M_\varsigma$.

**Definition 1.** *A non-hybrid single-sorted* DNI-signature *is a tuple* $\Xi = \langle \Sigma, \Omega, \Lambda\rangle$ *where* $\Sigma$ *is a first-order signature of* rigid *symbols,* $\Omega \supseteq \Sigma$ *is an extension of* $\Sigma$

*with* flexible *symbols, and* $\Lambda$ *is a set of* typed modalities. *Each such modality is a triple denoted* $\lambda\colon u \times v$*, where* $\lambda$ *is the name of the modality,* $u \in \mathbb{N}$ *is its* inner type*, and* $v \in \mathbb{N}$ *its* outer type.

Intuitively, the interpretations of the symbols in $\Omega$ form structures whose flexible parts (corresponding to symbols in $\Omega \setminus \Sigma$) change as a result of interactions that may occur between their elements. We model those interactions using modalities: the inner types of modalities indicate how many elements interact, while their outer types capture outcomes. To simplify notations, unless there is some risk of confusion (as in overloading), we may refer to modalities only by their names; in all those cases, the types are implied.

*Example 1.* We illustrate how typed modalities can be used in practice to model interactions using a simple abstract strategy game. *Paper soccer* [21], a turn based pencil-and-paper game, is played by drawing lines on a grid representing a football pitch. The two players, H (home) and A (away), are trying to move the ball – represented as a dot on the grid – from the centre of the field to the opponent's goal. The ball's path forms a segmented line on the grid. To move the ball, players extend that line with a new horizontal, vertical, or diagonal segment for each 'shot' they take. The line can only be continued within the field and cannot overlap its margins. If it is extended to a position in the grid that has already been visited, or if the newly added segment is tangent to the margins, the player must shoot again. Otherwise, they lose their turn. The player who first scores by moving the ball into the opponent's goal wins. If a player scores an own goal or cannot extend the path of the ball any more, they lose the match.

To formalize the game, we consider two graphs: a graph for modelling the pitch and one for tracking the path of the ball. The nodes of the pitch graph are grid positions, while the edges are all the possible moves. The path graph is a subgraph of the pitch graph; it has the same nodes as the pitch graph, and its edges are either margins or moves that have already been played. Whenever a player shoots, we extend the path graph with a suitable edge.

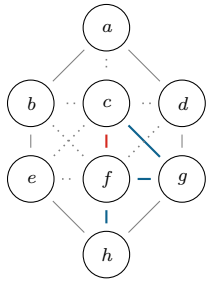



**Fig. 1.** Paper soccer

We represent matches as in Figure 1. We draw the shots of the away player with red, and the shots of the home player with blue. The nodes defining the pitch are labelled with letters from $a$ to $h$, with $a$ marking the goal of the home player, $h$ marking the goal of the away player, and nodes $b$ to $g$ indicating valid pitch positions. Shots are logged in order, as tuples comprising the name of the active player (H or A), the start position, and the end position: A$fc$, H$cg$, H$gf$, H$fh$.

We thus define a DNI-signature containing rigid predicates for the pitch (two unary predicates for marking the goals, and a binary predicate $\mathsf{neigh}$ for indicating neighbouring nodes), flexible predicates for recording matches (a binary predicate $\mathsf{edge}$ for margins and for logging shots, and two nullary predicates $\mathsf{H}$ and $\mathsf{A}$ for tracking turns), and a modality $\mathsf{shoot}\colon 1 \times 1$ for capturing shooting actions.

The semantics of DNI is defined in terms of networks.

**Definition 2.** *For every* DNI*-signature* $\Xi$*, a* network $\langle D, W, M\rangle$ *consists of:*

- *a first-order model* $D$ *of* $\Sigma$ *defining the* domain*, or rigid part, of the network;*
- *a* Kripke frame $W$*; we denote by* $|W|$ *its underlying set of* possible worlds*, which we also call* configurations*, and by* $W_\lambda \subseteq |W| \times |W|$ *the* accessibility relations *interpreting modalities* $\lambda\colon u \rightarrowtail v$ *in* $\Lambda$*;*
- *for every configuration* $w \in |W|$*, a* local $\Omega$*-model* $M_w$ *such that* $M_w{\restriction_\Sigma} = D$*, meaning that* $M_w$ *has the same underlying carrier set as* $D$ *and interprets all the (rigid) symbols in* $\Sigma$ *the same as* $D$*;*
- *for every transition, or* reconfiguration*,* $(w, w') \in W_\lambda$*, an* interaction relation $M_{w\lambda w'} \subseteq M^u_w \times M^v_{w'}$ *between* $u$ *elements in* $M_w$ *(seen as triggering actants for* $\lambda$*) and* $v$ *elements in* $M_{w'}$ *(outcomes of* $\lambda$*).*

*Example 2.* In Figure 2, we depict a possible network $\langle D, W, M\rangle$ for the paper-soccer signature, highlighting the reconfiguration process. Following the notations in Figure 1, $D$ comprises elements $a$–$h$. The network has five configurations, $w_0$–$w_4$, and four transitions describing a complete match. Each configuration is annotated with either letter H or A indicating which of the two predicates holds. For each transition, an arrow is drawn between the start and end position of the ball. For instance, for the first move, we have $M_{w_0\mathsf{shoot}w_1} = \{(f, c)\}$.

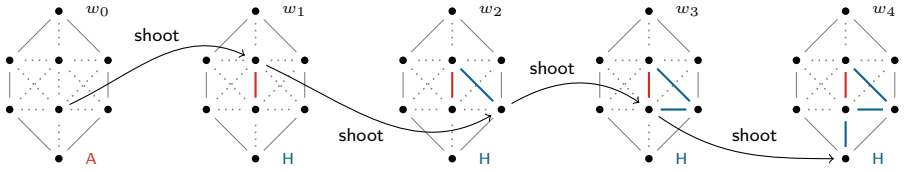



**Fig. 2.** A paper-soccer network

DNI-sentences are built from first-order atoms using Boolean connectives, first-order quantifiers, and typed modal operators. Variables are introduced and bound both by quantifiers and by modal operators. In keeping with the institution-theoretic approach [7], we deal with variables by means of signature extensions with constants. This allows us to avoid defining additional notions of valuation and open formula – which get derived, when needed, from working with models and sentences across signature extensions. For every signature $\Xi = \langle \Sigma, \Omega, \Lambda\rangle$, with $\Omega = \langle F, P\rangle$, and every set (or tuple) of variables $X$, we let $\Xi(X)$ be $\langle \Sigma(X), \Omega(X), \Lambda\rangle$, where $\Sigma(X)$ and $\Omega(X)$ are extensions of $\Sigma$ and $\Omega$ with new constants of the form $(x, F_0)$, for each $x \in X$. Through this annotation, we ensure that none of the newly added constants clashes with those already declared.

**Definition 3.** *We write* $\Xi$-sentences *according to the following grammar:*

$$\rho \ ::= \ e \mid \neg\rho \mid \rho \wedge \rho \mid [t\, \lambda\, z]\rho' \mid \forall X \cdot \rho^\dagger$$

*where: $e$ is either an equational or a relational atom over $\Omega$; $\lambda\colon u \rightharpoonup v$ is a modality, $t$ is a tuple of size $u$ of $\Omega$-terms, $z$ is a tuple of size $v$ of first-order variables, and $\rho'$ is a sentence over the extended signature $\Xi(z)$; and $X$ is a finite set of first-order variables and $\rho^\dagger$ is a sentence over $\Xi(X)$.*

Other, derived logical connectives can be defined in the usual manner. We denote disjunctions by $\rho_1 \vee \rho_2$, implications by $\rho_1 \Rightarrow \rho_2$, equivalences by $\rho_1 \Leftrightarrow \rho_2$, possibility statements by $\langle t\, \lambda\, z\rangle \rho$, which stands for $\neg[t\, \lambda\, z]\neg\rho$, and existentially quantified sentences by $\exists X \cdot \rho$, which stands for $\neg\forall X \cdot \neg\rho$.

Extending signatures allows us to view sentences over extensions $\Xi(X)$ as open formulæ over the original signature $\Xi$ with free variables from $X$. It also allows us to bundle $\Xi$-networks and valuations of $X$ as networks over the extended signature: for every $\Xi(X)$-network $\langle D', W, M'\rangle$, we let $\langle D', W, M'\rangle\restriction_\Xi$ be the $\Xi$-network $\langle D, W, M\rangle$ with the same underlying Kripke frame, and with $D = D'\restriction_\Sigma$ and $M_w = M'_w\restriction_\Omega$ for all $w \in |W|$. We say that $\langle D, W, M\rangle$ is a *$\Xi$-reduct* of $\langle D', W, M'\rangle$, and that $\langle D', W, M'\rangle$ is a *$\Xi(X)$-expansion* of $\langle D, W, M\rangle$ – what is 'added' is precisely a valuation of $X$ in $\langle D, W, M\rangle$. This relation between reducts and expansions is key to defining the satisfaction of sentences.

**Definition 4.** *The* local-satisfaction relations $\langle D, W, M\rangle \models^w_\Xi \rho$ *between $\Xi$-networks $\langle D, W, M\rangle$ and $\Xi$-sentences $\rho$ at configurations $w \in |W|$ are defined by structural induction on sentences, as follows:*

- $\langle D, W, M\rangle \models^w_\Xi t_1 = t_2$ *if* $M_{w,t_1} = M_{w,t_2}$*, for equational atoms;*
- $\langle D, W, M\rangle \models^w_\Xi \pi(t)$ *if* $M_{w,t} \in M_{w,\pi}$*, for relational atoms;*
- $\langle D, W, M\rangle \models^w_\Xi \neg\rho$ *if* $\langle D, W, M\rangle \not\models^w_\Xi \rho$*;*
- $\langle D, W, M\rangle \models^w_\Xi \rho_1 \wedge \rho_2$ *if* $\langle D, W, M\rangle \models^w_\Xi \rho_1$ *and* $\langle D, W, M\rangle \models^w_\Xi \rho_2$*;*
- $\langle D, W, M\rangle \models^w_\Xi [t\, \lambda\, z]\rho'$ *if* $\langle D', W, M'\rangle \models^{w'}_{\Xi(z)} \rho'$ *for all* $(w, w') \in W_\lambda$ *and all $\Xi(z)$-expansions $\langle D', W, M'\rangle$ of $\langle D, W, M\rangle$ such that* $(M_{w,t}, M'_{w',z}) \in M_{w\lambda w'}$*;*
- $\langle D, W, M\rangle \models^w_\Xi \forall X \cdot \rho^\dagger$ *if* $\langle D^\dagger, W, M^\dagger\rangle \models^w_{\Xi(X)} \rho^\dagger$ *for all $\Xi(X)$-expansions $\langle D^\dagger, W, M^\dagger\rangle$ of $\langle D, W, M\rangle$.*

*Example 3.* Back to the paper field, we outline the specification of our game:

$$\mathsf{A} \Leftrightarrow \neg\mathsf{H} \tag{1}$$

$$\forall\{x, y\} \cdot \big(\mathsf{neigh}(x, y) \wedge \neg\mathsf{edge}(x, y)\big) \Leftrightarrow \langle x\ \mathsf{shoot}\ z\rangle z = y \tag{2}$$

$$\forall\{x\} \cdot [x\ \mathsf{shoot}\ y]\mathsf{edge}(x, y) \tag{3}$$

$$\forall\{x\} \cdot \mathsf{A} \Rightarrow [x\ \mathsf{shoot}\ y]\big(\mathsf{A} \Leftrightarrow (\exists\{z\} \cdot \mathsf{edge}(z, y) \wedge x \neq z)\big) \tag{4}$$

The first sentence captures the fact that players take turns. Sentence (2) states that the ball can be moved with a single shot from $x$ to $y$ if and only if the two positions are adjacent and the grid segment between them is not already included in the path graph. The outcome of shooting from $x$ to $y$ is defined in (3) as the addition of an edge in the path graph. The last sentence is a continuation

condition for the away player: if A shoots from $x$ to $y$, they have to shoot again when $y$ is incident to an edge defining the margins of the pitch or has already been visited during the match. The continuation condition for H is similar.

## 3 DNI as a stratified institution

From an institution-theoretic perspective, DNI belongs to a broad family of logics whose models are endowed with states that act as parameters for local satisfaction relations as seen in Definition 4. These logics can be formalized categorically as stratified institutions, which we recall below from [6] in their *strict* form – a form simpler than the original, lax one yet adequate for our purpose. In what follows, apart from the composition of morphisms, which we write in diagrammatic order and denote by a blackboard bold semicolon, most of the terminology and the notations that we use are based on [17]. We denote by $\mathbb{S}\mathrm{et}$ the category of sets and functions, by $\mathbb{C}\mathrm{at}$ the higher category of categories and functors, and by $\mathrm{Set}$ the constant functor $\mathbb{C} \to \mathbb{C}\mathrm{at}$ (for any category $\mathbb{C}$) that maps every object and morphism of $\mathbb{C}$ to $\mathbb{S}\mathrm{et}$ and the identity functor $id_{\mathbb{S}\mathrm{et}}$, respectively.

**Definition 5.** *A* strict stratified institution $\mathcal{S}$ *consists of:*

- *a category* $\mathbb{S}\mathrm{ig}^{\mathcal{S}}$ *of* signatures *and* signature morphisms,
- *a* sentence functor $\mathrm{Sen}^{\mathcal{S}} \colon \mathbb{S}\mathrm{ig}^{\mathcal{S}} \to \mathbb{S}\mathrm{et}$ *defining, for every signature* $\Sigma$*, a set* $\mathrm{Sen}^{\mathcal{S}}(\Sigma)$ *of* $\Sigma$-sentences *and, for every signature morphism* $\varphi \colon \Sigma \to \Sigma'$*, a* sentence-translation map $\mathrm{Sen}^{\mathcal{S}}(\varphi) \colon \mathrm{Sen}^{\mathcal{S}}(\Sigma) \to \mathrm{Sen}^{\mathcal{S}}(\Sigma')$,
- *a* model functor $\mathrm{Mod}^{\mathcal{S}} \colon (\mathbb{S}\mathrm{ig}^{\mathcal{S}})^{\mathrm{op}} \to \mathbb{C}\mathrm{at}$ *defining, for every signature* $\Sigma$*, a category* $\mathrm{Mod}^{\mathcal{S}}(\Sigma)$ *of* $\Sigma$-models *and, for every morphism* $\varphi \colon \Sigma \to \Sigma'$*, a* reduct functor $\mathrm{Mod}^{\mathcal{S}}(\varphi) \colon \mathrm{Mod}^{\mathcal{S}}(\Sigma') \to \mathrm{Mod}^{\mathcal{S}}(\Sigma)$,
- *a* stratification *natural transformation* $[\![_]\!]^{\mathcal{S}} \colon \mathrm{Mod}^{\mathcal{S}} \Rightarrow \mathrm{Set}$ *defining, for every signature* $\Sigma$*, a* state-space functor $[\![_]\!]^{\mathcal{S}}_{\Sigma} \colon \mathrm{Mod}^{\mathcal{S}}(\Sigma) \to \mathbb{S}\mathrm{et}$*, and*
- *a signature-indexed family of model-indexed families of* local satisfaction relations *between model states and sentences; we write* $M \models^{\mathcal{S},w}_{\Sigma} \rho$ *to indicate that a* $\Sigma$*-model* $M$ locally satisfies *a* $\Sigma$*-sentence* $\rho$ at *a state* $w \in [\![M]\!]^{\mathcal{S}}_{\Sigma}$*,*

*such that the following* satisfaction condition *holds for every signature morphism* $\varphi \colon \Sigma \to \Sigma'$*, every* $\Sigma'$*-model* $M'$*, state* $w \in [\![M']\!]^{\mathcal{S}}_{\Sigma'}$*, and* $\Sigma$*-sentence* $\rho$*:*

$$M' \models^{\mathcal{S},w}_{\Sigma'} \mathrm{Sen}^{\mathcal{S}}(\varphi)(\rho) \quad \textit{if and only if} \quad \mathrm{Mod}^{\mathcal{S}}(\varphi)(M') \models^{\mathcal{S},w}_{\Sigma} \rho.$$

We may omit superscripts or subscripts from the notations introduced above when the institution and/or signature that we are referring to can be easily inferred. Moreover, in line with well-established institution-theoretic conventions, we usually denote sentence-translation functions $\mathrm{Sen}(\varphi)$ by $\varphi(_)$ and model-reduct functors $\mathrm{Mod}(\varphi)$ by $_{\restriction_\varphi}$. When $M = M'{\restriction_\varphi}$, we may say that $M$ is the $\varphi$*-reduct* of $M'$, or that $M'$ is a $\varphi$*-expansion* of $M$, thus generalizing the notions of reduct and expansion along a signature extension introduced in Section 2.

We already have a nearly complete picture of DNI as a stratified institution: signatures, sentences, networks in the role of models, a stratification given by

$[\![\langle D, W, M\rangle]\!] = |W|$ for every network $\langle D, W, M\rangle$, and local satisfaction relations. The only major building blocks that are missing are a notion of signature morphism together with its corresponding sentence translations and model reductions. More model-theoretic studies of DNI would also require model homomorphisms; but the results that we are presenting in this paper can also be established without taking homomorphisms into account, hence we omit them for now and opt to work only with discrete categories of models (i.e., categories with no homomorphisms except for the identities).

A *morphism* between DNI signatures $\Xi = \langle \Sigma, \Omega, \Lambda\rangle$ and $\Xi' = \langle \Sigma', \Omega', \Lambda'\rangle$ consists of a pair $\langle \varphi, \psi\rangle$ such that $\varphi\colon \Omega \to \Omega'$ is an ordinary first-order signature morphism that preserves rigid symbols, meaning that $\varphi(\Sigma) \subseteq \Sigma'$, and $\psi\colon \Lambda \to \Lambda'$ is a type-preserving mapping of modalities, meaning that $\psi(\lambda)\colon u \rtimes v \in \Lambda'$ for all $\lambda\colon u \rtimes v \in \Lambda$. The translation of $\Xi$-sentences along $\langle \varphi, \psi\rangle$ is defined inductively:

- $\langle \varphi, \psi\rangle(e) = \varphi(e)$ for equational or relational atoms;
- $\langle \varphi, \psi\rangle(\neg\rho) = \neg\langle \varphi, \psi\rangle(\rho)$;
- $\langle \varphi, \psi\rangle(\rho_1 \wedge \rho_2) = \langle \varphi, \psi\rangle(\rho_1) \wedge \langle \varphi, \psi\rangle(\rho_2)$;
- $\langle \varphi, \psi\rangle([t\ \lambda\ z]\rho') = [\varphi(t)\ \psi(\lambda)\ z]\langle \varphi, \psi\rangle^z(\rho')$, where $\langle \varphi, \psi\rangle^z\colon \Xi(z) \to \Xi'(z)$ is the canonical extension of $\langle \varphi, \psi\rangle$ that updates the annotation of the added constants in $\Xi(z)$; that is, $\langle \varphi, \psi\rangle^z(x, F_0) = (x, F_0')$, for each variable $x$ in $z$, where $F_0$ and $F_0'$ are the sets of constants in $\Omega$ and $\Omega'$, respectively;
- $\langle \varphi, \psi\rangle(\forall X \cdot \rho^\dagger) = \forall X \cdot \langle \varphi, \psi\rangle^X(\rho^\dagger)$, where $\langle \varphi, \psi\rangle^X\colon \Xi(X) \to \Xi'(X)$ is another extension of $\langle \varphi, \psi\rangle$ defined similarly to $\langle \varphi, \psi\rangle^z$ above.

With regard to model reducts, for every $\Xi'$-network $\langle D', W', M'\rangle$, we let $\langle D', W', M'\rangle{\restriction_{\langle \varphi, \psi\rangle}}$ be the $\Xi$-network $\langle D, W, M\rangle$ with the same set of possible worlds $|W| = |W'|$, the same carrier sets as $\langle D', W', M'\rangle$, and with:

- $D_\varsigma = D'_{\varphi(\varsigma)}$ for all operation or relation symbols $\varsigma$ in $\Sigma$;
- $W_\lambda = W'_{\psi(\lambda)}$ for all modalities $\lambda\colon u \rtimes v$ in $\Lambda$;
- $M_{w,\varsigma} = M'_{w,\varphi(\varsigma)}$ for all $w \in |W|$ and all operation or relation symbols $\varsigma$ in $\Omega$;
- $M_{w\lambda w'} = M'_{w\psi(\lambda)w'}$ for all $\lambda$-reconfigurations $(w, w') \in W_\lambda$.

**Fact 1.** *The following properties can be checked easily, by mere calculations, based on the definitions above:*

a) *Morphisms of* DNI*-signatures together with their obvious componentwise composition and identities form a category* $\mathbb{S}\mathrm{ig}^{\mathrm{DNI}}$.
b) *Both the (covariant) translation of sentences and (contravariant) reduction of networks preserve that componentwise composition and the identities, leading to functors* $\mathrm{Sen}^{\mathrm{DNI}}\colon \mathbb{S}\mathrm{ig}^{\mathrm{DNI}} \to \mathbb{S}\mathrm{et}$ *and* $\mathrm{Mod}^{\mathrm{DNI}}\colon (\mathbb{S}\mathrm{ig}^{\mathrm{DNI}})^{\mathrm{op}} \to \mathbb{C}\mathrm{at}$.
c) *The stratification* $[\![_]\!]^{\mathrm{DNI}}$ *mapping networks to their underlying sets of configurations is a natural transformation* $\mathrm{Mod}^{\mathrm{DNI}} \Rightarrow \mathrm{Set}$.

**Proposition 1.** DNI *is a strict stratified institution.* □

Besides DNI, there is an abundance of examples of strict stratified institutions that originate from the broader field of modal logic. We mention just a few, which can be presented as fragments of DNI or as variations of such fragments.

Multi-modal *first-order modal logic* [11], abbreviated FOML, can be regarded as the fragment of DNI obtained by taking into account only modalities whose inner and outer types are both 0 – which implies that the modal operators available in that case are conventional, e.g., there is no variable binding involved. We can further restrict signatures to allow for a single modality symbol, thus arriving at the classic modal operators $\Box$ (for necessity) and $\Diamond$ (for possibility). It is worth noting that first-order modal logic comes in several different flavours. Some consider constant domains (as we do here), while others consider varying domains; for the latter, some only allow actualist quantification, while others support possibilist quantification as well; and some are single-sorted (as in this paper), while others are many-sorted. All the flavours that we have just mentioned can be seen as fragments of suitably enhanced variants (see [28]) of the simplified framework of DNI presented in Section 2.

*Term-modal logics* [12], abbreviated TML, can also be defined as fragments of DNI by considering signatures with only one modality whose inner type is 1 and outer type is 0. Those fragments are conceptually much closer to full DNI than to FOML. Their models are constrained to satisfy a monotonicity condition, and there is still no way for modal operators to bind variables; but they do make it possible to quantify separately over variables that occur in (the terms used within) modal operators – which is outside the scope of FOML.

Many *hybrid logics* [4], except for example some that arise through the hybridization [19,8] of complex base logics, form fragments of DNI as well. However, in this case, the simplified framework of DNI that we consider herein no longer suffices, meaning that we need to switch to a variant that supports nominals and local-satisfaction operators as in [28].

Last but not least, some *relation-changing logics* [2] as in Johan van Benthem's *sabotage game* [3], can be presented as 'fragments' of DNI. The quotes indicate that these logics are not proper fragments of DNI. Instead, we need to consider the stratified institution $\mathrm{DNI}^{th}$ of DNI theories, whose signatures are pairs $\langle \Xi, E \rangle$ of DNI-signatures $\Xi$ and sets $E$ of $\Xi$-sentences, and whose models are the $\Xi$-networks that globally satisfy, at all states, the sentences in $E$. For instance, for *sabotage logics*, we use DNI-signatures with no function symbols, no predicate symbols except for flexible monadic ones and for a flexible binary predicate $e$ (for "edge"), and a single modality $sb\colon 1 \times 1$ with the following properties:

$$\forall \{x, y\} \cdot e(x, y) \Rightarrow \langle x \; \mathsf{sb} \; z \rangle z = y \tag{5}$$

$$\forall \{x, x', y'\} \cdot \big( \langle x \; \mathsf{sb} \; y \rangle e(x', y') \big) \Rightarrow e(x', y') \tag{6}$$

$$\forall \{x\} \cdot [x \; \mathsf{sb} \; y] \neg e(x, y) \tag{7}$$

$$\forall \{x, x', y'\} \cdot e(x', y') \wedge x \neq x' \Rightarrow [x \; \mathsf{sb} \; z] e(x', y') \tag{8}$$

$$\forall \{x, x', y'\} \cdot e(x', y') \Rightarrow [x \; \mathsf{sb} \; z] z \neq y' \Rightarrow e(x', y') \tag{9}$$

According to (5), for every edge $(a, b)$ in a configuration $w$, there is a reconfiguration $(w, w')$ such that $(a, b) \in M_{w \mathsf{sb} w'}$. Moreover, by (6) and (7), it follows that $M_{w',e} \subseteq M_{w,e} \setminus \{(a, b)\}$; and by (8), (9) we get the right-to-left inclusion also.

## 4 "Conversations with Friends"

There are several ways to link institutions, depending on what kind of maps we consider between their components. Comorphisms are one of the most common, and are generally regarded as fundamental tools for moving theories and their consequences between logical systems [5]. In what follows, we adapt the original definition (cf. [20,25]) to the stratified setting using ideas from [27] and [7, Chapter 12]. Unlike Definition 5, the notion of comorphism that we consider here is more general, oplax [16]; we make this choice because, in the final part of the paper, we work with both strict and non-strict examples of comorphisms.

**Definition 6.** *An* oplax stratified comorphism $\mu\colon \mathcal{S} \to \mathcal{T}$ *consists of:*

- *a functor* $\mu^{\mathrm{sig}}\colon \mathbb{S}\mathrm{ig}^{\mathcal{S}} \to \mathbb{S}\mathrm{ig}^{\mathcal{T}}$ *encoding* $\mathcal{S}$*-signatures as* $\mathcal{T}$*-signatures,*
- *a natural transformation* $\mu^{\mathrm{sen}}\colon \mathrm{Sen}^{\mathcal{S}} \Rightarrow \mu^{\mathrm{sig}} \fatsemi \mathrm{Sen}^{\mathcal{T}}$ *translating* $\mathcal{S}$*-sentences over some signature* $\Sigma$ *to* $\mathcal{T}$*-sentences over* $\mu^{\mathrm{sig}}(\Sigma)$*,*
- *a natural transformation* $\mu^{\mathrm{mod}}\colon (\mu^{\mathrm{sig}})^{\mathrm{op}} \fatsemi \mathrm{Mod}^{\mathcal{T}} \Rightarrow \mathrm{Mod}^{\mathcal{S}}$ *reducing* $\mathcal{T}$*-models for* $\mu^{\mathrm{sig}}(\Sigma)$ *to* $\mathcal{S}$*-models for* $\Sigma$*, and*
- *a modification* $\mu^{\mathrm{st}}\colon \mu^{\mathrm{mod}} \fatsemi [\![_]\!]^{\mathcal{S}} \Rightarrow (\mu^{\mathrm{sig}})^{\mathrm{op}} [\![_]\!]^{\mathcal{T}}$ *mapping the* $\mathcal{S}$*-states of reduced* $\Sigma$*-models* $\mu^{\mathrm{mod}}(M)$ *to* $\mathcal{T}$*-states of the original* $\mu^{\mathrm{sig}}(\Sigma)$*-models* $M$*,*

*such that the following* satisfaction condition *holds for every* $\mathcal{S}$*-signature* $\Sigma$*, every* $\mu^{\mathrm{sig}}(\Sigma)$*-model* $M$*, state* $w \in [\![\mu^{\mathrm{mod}}(M)]\!]^{\mathcal{S}}_{\Sigma}$*, and every* $\Sigma$*-sentence* $\rho$*:*

$$M \models^{\mathcal{T},\mu^{\mathrm{st}}(w)}_{\mu^{\mathrm{sig}}(\Sigma')} \mu^{\mathrm{sen}}_{\Sigma}(\rho) \quad \textit{if and only if} \quad \mu^{\mathrm{mod}}_{\Sigma}(M) \models^{\mathcal{S},w}_{\Sigma} \rho.$$

When the context is clear, to streamline notations and avoid clutter, we may denote the encoding of an $\mathcal{S}$-signature $\Sigma$ along a comorphism $\mu$ simply by $\mu(\Sigma)$, the translation of a $\Sigma$-sentence $\rho$ by $\mu_{\Sigma}(\rho)$ or even $\mu(\rho)$, the reduction of a $\mu(\Sigma)$-model $M$ by $M{\restriction_{\mu_{\Sigma}}}$ or $M{\restriction_{\mu}}$, and the modification of a state $w$ by $\mu(w)$.

*Lax stratified comorphisms* can be defined in a similar manner, by reversing the direction of the modification $\mu^{\mathrm{st}}$. We say that an (op)lax comorphism is *strict* when all the components of $\mu^{\mathrm{st}}$ are identities. In other words, for every $\mathcal{S}$-signature $\Sigma$ and every $\mu(\Sigma)$-model $M$, $[\![M{\restriction_{\mu}}]\!]^{\mathcal{S}}_{\Sigma} = [\![M]\!]^{\mathcal{T}}_{\mu(\Sigma)}$. It follows that strict comorphisms are both lax and oplax. With one exception (see Proposition 3), all the comorphisms that we consider in this paper are strict.

*Example 4.* In practice, comorphisms capture both embeddings and encodings of one logical system into another. For instance, the constant-domain FOML and TML fragments of DNI presented in Section 3 can both be formalized as strict embedding comorphisms into DNI. The underlying signature functors of those embeddings are just inclusions of categories, while the sentence-translation maps and model-reduction functors are all identities. The same applies to many hybrid logics and the extended version of DNI presented in [28].

For varying-domain TML, one needs to consider a slightly different kind of comorphism that includes an axiomatization of the monotonicity condition constraining TML-models. Therefore, that comorphism can be seen more as

an encoding between logics. Similarly, but outside the scope of this paper, the links between relation-changing logics and DNI suggested in Section 3 can be formalized as encoding comorphisms by adding axioms such as (5)–(9).

**Proposition 2.** *There is a strict comorphism* $\epsilon\colon \mathrm{DNI} \to \mathrm{FOML}$ *such that:*

- *on signatures,* $\epsilon(\Sigma, \Omega, \Lambda) = \langle \Sigma, \widehat{\Omega}, \Lambda \rangle$*, where* $\widehat{\Omega}$ *extends* $\Omega$ *with fresh flexible predicates* $i_\lambda\colon u$ *and* $o_\lambda\colon v$ *for each typed modality* $\lambda\colon u \times v$ *in* $\Lambda$*;*
- *on sentences, for every* DNI*-signature* $\Xi = \langle \Sigma, \Omega, \Lambda \rangle$*,*
  - $\epsilon_\Xi(e) = e$ *for all equational or relational atoms,*
  - $\epsilon_\Xi(\neg\rho) = \neg\epsilon_\Xi(\rho)$*,*
  - $\epsilon_\Xi(\rho_1 \wedge \rho_2) = \epsilon_\Xi(\rho_1) \wedge \epsilon_\Xi(\rho_2)$*,*
  - $\epsilon_\Xi([t\,\lambda\,z]\rho') = \forall\{x\} \cdot (x = t)$[3] $\Rightarrow [\lambda]\forall\{z\} \cdot i_\lambda(x) \wedge o_\lambda(z) \Rightarrow \epsilon_{\Xi(z)}(\rho')$*, and*
  - $\epsilon_\Xi(\forall X \cdot \rho^\dagger) = \forall X \cdot \epsilon_{\Xi(X)}(\rho^\dagger)$*;*
- *on models, for every* $\epsilon(\Xi)$*-model* $\langle D, W, \widehat{M} \rangle$*,* $\langle D, W, \widehat{M} \rangle\restriction_\epsilon$ *is the* $\Xi$*-network* $\langle D, W, M \rangle$ *with the same domain and frame as the original* FOML*-model, and with* $M_w = \widehat{M}_w\restriction_\Omega$ *– i.e., forgetting the interpretation of the added predicates* $i_\lambda$ *and* $o_\lambda$ *– for all configurations* $w \in |W|$*, and* $M_{w\lambda w'} = \widehat{M}_{w',i_\lambda} \times \widehat{M}_{w',o_\lambda}$ *for all reconfigurations* $(w, w') \in W_\lambda$*.* □

For the last comorphism that we discuss, let TDNI be the subinstitution of DNI with the same signatures and sentences as DNI, but without signature morphisms, and with networks constrained by the next property.

**Definition 7.** *A* $\Xi$*-network* $\langle D, W, M \rangle$ *is* tree-like *when for every configuration* $w \in |W|$ *there exists a unique path in* $\langle D, W, M \rangle$ *ending in* $w$ *from a root of the network, where by* path *and* root *we mean the following:*

- *given a configuration* $w_0 \in |W|$*, a* $w_0$-rooted path *is a sequence of the form* $\vec{w} = (w_0, a_1, \lambda_1, b_1, w_1, \ldots, a_n, \lambda_n, b_n, w_n)$ *where, for every* $0 < i \leq n$*,* $w_i \in |W|$*,* $\lambda_i\colon u_i \times v_i$ *is a modality, and* $a_i \in M_{w_{i-1}}^{u_i}$*,* $b_i \in M_{w_i}^{v_i}$ *such that* $(w_{i-1}, w_i) \in W_{\lambda_i}$ *and* $(a_i, b_i) \in M_{w_{i-1}\lambda_i w_i}$*; we say that* $w_n = e(\vec{w})$ *is the* end *of* $\vec{w}$*,* $n$ *is its* length*, and we denote the* $w_0$*-rooted path of length* 0 *by* $(w_0)$*;*
- *a configuration* $w \in |W|$ *is a* root *of* $\langle D, W, M \rangle$ *when* $(w)$ *is the only path in* $\langle D, W, M \rangle$ *that ends in* $w$ *– i.e., there are no incoming transitions into* $w$*.*

**Proposition 3.** *There is a comorphism* $\upsilon\colon \mathrm{TDNI} \to \mathrm{DNI}$ *such that:*

- *on signatures,* $\upsilon^{\mathrm{sig}}$ *is just the wide inclusion of categories* $\mathbb{S}\mathrm{ig}^{\mathrm{TDNI}} \subseteq \mathbb{S}\mathrm{ig}^{\mathrm{DNI}}$*;*
- *on sentences, for every signature* $\Xi$*,* $\upsilon_\Xi^{\mathrm{sen}}$ *is an identity;*
- *on models, for every* $\Xi$*-network* $\langle D, W, M \rangle$*,* $\upsilon_\Xi^{\mathrm{mod}}(D, W, M)$ *is the* unravelling $\langle D, \overrightarrow{W}, \overrightarrow{M} \rangle$*, where* $|\overrightarrow{W}|$ *is the set of all paths in* $\langle D, W, M \rangle$ *and*
  - $\overrightarrow{W}_\lambda = \{(\vec{w}, (\vec{w}, a, \lambda, b, w')) \mid (e(\vec{w}), w') \in W_\lambda \text{ and } (a, b) \in M_{e(\vec{w})\lambda w'}\}$ *for every typed modality* $\lambda\colon u \times v$*,*
  - $\overrightarrow{M}_{\vec{w}} = M_{e(\vec{w})}$ *for every path* $\vec{w}$ *in* $\langle D, W, M \rangle$*, and*

[3] We use an additional variable (thus introducing a new rigid constant) because $t$ may be flexible, which means that its interpretation may change after a $\lambda$-transition.

  - $\overrightarrow{M}_{\vec{w}\lambda(\vec{w},a,\lambda,b,w')} = \{(a,b)\}$ *for every* $\lambda$*-reconfiguration from* $\vec{w}$*;*
- *on states, for every* $\Xi$*-network* $\langle D, W, M\rangle$*,* $\upsilon^{\mathrm{st}}_{\Xi}$ *is nothing but the function* $e$ *that maps every path* $\vec{w}$ *in* $\langle D, W, M\rangle$ *to its end,* $e(\vec{w})$*.* □

The unravelling technique used in Proposition 3 originates from [23] and is widely used in modal logic [15]. By spelling out the satisfaction condition for the unravelling comorphism $\upsilon$, we get another important property that has been studied extensively in the modal-logic literature (e.g., [29]).

**Corollary 1.** DNI *has the* tree-model property*: a sentence is satisfiable if and only if it is satisfiable in a tree-like network.* □

The remaining part of the paper is concerned with the preservation and reflection of semantic consequences along comorphisms. To that end, we first recall from [7] some basic institution-theoretic notions:

- a model $M$ for a signature $\Sigma$ *globally satisfies* a $\Sigma$-sentence $\rho$, in which case we write $M \models^{*}_{\Sigma} \rho$, when $M \models^{w}_{\Sigma} \rho$ for all states $w \in [\![M]\!]_{\Sigma}$;
- $\rho$ is a *semantic consequence* of a set $E$ of $\Sigma$-sentences, denoted $E \models^{*}_{\Sigma} \rho$, when it is satisfied globally by all models that globally satisfy all sentences in $E$;
- a comorphism $\mu\colon \mathcal{S} \to \mathcal{T}$ *preserves semantic consequences* when, for every $\mathcal{S}$-signature $\Sigma$, $E \models^{*}_{\Sigma} \rho$ implies $\mu(E) \models^{*}_{\mu(\Sigma)} \mu(\rho)$; and *reflects consequences*, or is *conservative*, when the opposite implication holds.

In addition, we say that a comorphism $\mu$ is *sentence-forming*, *model-expansive*, or *state-surjective* when all the sentence-translation maps in $\mu^{\mathrm{sen}}$, model-reduction functors in $\mu^{\mathrm{mod}}$, or state modifications in $\mu^{\mathrm{st}}$, respectively, are surjective.

These notions enable us to obtain several simple yet useful institution-independent results about the preservation and reflection of semantic consequences. The state-surjectivity property is the most important of them, as it allows us to establish a satisfaction condition for stratified comorphisms with respect to global satisfaction relations.

**Proposition 4.** *If a comorphism* $\mu$ *is state-surjective, then for every* $\mathcal{S}$*-signature* $\Sigma$*,* $\mu(\Sigma)$*-model* $M$*, and* $\Sigma$*-sentence* $\rho$*,* $M \models^{*}_{\mu(\Sigma)} \mu(\rho)$ *if and only if* $M{\restriction_{\mu}} \models^{*}_{\Sigma} \rho$*.* □

**Corollary 2.** *State-surjective comorphisms preserve semantic consequences.* □

Hence all strict comorphisms preserve consequences. In particular, the preservation property holds for the encoding of DNI into FOML presented in Proposition 2. The same applies for the unravelling comorphism from Proposition 3, since every configuration $w$ is the end of the path $(w)$ of length 0.

For state-surjective comorphisms, conservativity can be derived by taking model expansions, according to the following proposition.

**Proposition 5.** *If a comorphism* $\mu$ *is both state-surjective and model-expansive, then it is conservative.* □

It is not difficult to see, however, that the encoding comorphism $\epsilon$ presented in Proposition 2 is not model-expansive: take, for instance, the network described in Example 2 and add an extra interaction $(c, f)$ to $M_{w_0 \mathsf{shoot} w_1}$; then the resulting network does not admit an $\epsilon$-expansion. Therefore, we cannot apply Proposition 5 directly. Instead, we notice that the model-expansion property of $\epsilon$ holds for tree-like networks, which implies that the composed comorphism $\upsilon ; \epsilon \colon \mathrm{TDNI} \to \mathrm{FOML}$ (where composition is defined componentwise) is conservative. This prompts us to examine factorizations of conservative comorphisms.

**Proposition 6.** *Let $\mu$ and $\nu$ be comorphisms such that $\mu ; \nu$ is conservative.*

- *If $\mu$ is surjective on signatures, sentences, and states, then $\nu$ is conservative.*
- *If $\nu$ is state-surjective, then $\mu$ is conservative.* □

The unravelling comorphism $\upsilon$ is an identity on signatures; and its sentence-translation maps are also identities, hence $\upsilon$ is trivially sentence-forming. Moreover, we have already established that the comorphisms $\upsilon$ and $\epsilon$ are both state-surjective. Thus we arrive, by Proposition 6, at our last main result.

**Corollary 3.** *For every set $E$ of* DNI*-sentences over some signature $\Xi$, and every $\Xi$-sentence $\rho$, the following statements are equivalent:*

a) $E \models^{\mathrm{TDNI},*}_{\Xi} \rho$ b) $E \models^{\mathrm{DNI},*}_{\Xi} \rho$ c) $\epsilon(E) \models^{\mathrm{FOML},*}_{\epsilon(\Xi)} \epsilon(\rho)$. □

## 5 Conclusions

We have revisited DNI, a modal-logic framework aimed at modelling and reasoning about dynamic systems where configuration changes are governed by interactions between system components. This return to the foundations of DNI has been motivated by the need to place the core feature of the logic, namely the use of typed modalities, in a wider modal-logic context. To achieve that goal, we have resorted to a branch of institution theory that is specialized in dealing with logics whose models – akin to Kripke structures – are equipped with states. We have shown that DNI can be formalized as a stratified institution that embeds or encodes and, in turn, can be encoded in other modal logics by means of comorphisms. In this process, we have also proved that, despite allowing for finer-grained system reconfigurations, typed modalities can be conservatively encoded into first-order modal logic. That opens the possibility to employ first-order modal tools such as MleanCoP [22] to check DNI properties – e.g., the existence of winning strategies in the paper-soccer game.

There are multiple other benefits of formalizing DNI as an institution. For instance, it allows us to apply the rich institutional theory of formal system development [24] to DNI in order to take advantage of modularization techniques [10] for specifications and proofs alike. It also enables us to view typed modalities as a feature that can be added to other logical systems (not necessarily first-order) in a manner similar to modalities [9] or nominals [8]; or to describe typed modal operators as semantic connectives [26] that may be present (perhaps implicitly) in other logics as well. This suggests that DNI is representative of a family of logics with typed modalities that has yet to be explored.

## References


1. Aiguier, M., Diaconescu, R.: Stratified institutions and elementary homomorphisms. Information Processing Letters **103**(1), 5–13 (2007)
2. Areces, C.: A Survey on Relation-Changing Modal Logics, pp. 309–338. Springer Nature Switzerland (2026)
3. van Benthem, J.: An essay on sabotage and obstruction. In: Hutter, D., Stephan, W. (eds.) Mechanizing Mathematical Reasoning, Essays in Honor of Jörg H. Siekmann on the Occasion of His 60th Birthday. Lecture Notes in Computer Science, vol. 2605, pp. 268–276. Springer (2005)
4. Blackburn, P.: Representation, reasoning, and relational structures: A hybrid logic manifesto. Logic Journal of IGPL **8**(3), 339–365 (2000)
5. Cerioli, M., Meseguer, J.: May I borrow your logic? (Transporting logical structures along maps). Theoretical Computer Science **173**(2), 311–347 (1997)
6. Diaconescu, R.: Implicit Kripke semantics and ultraproducts in stratified institutions. Journal of Logic and Computation **27**(5), 1577–1606 (2017)
7. Diaconescu, R.: Institution-Independent Model Theory. Studies in Universal Logic, Birkhäuser, 2 edn. (2025)
8. Diaconescu, R., Madeira, A.: Encoding hybridized institutions into first-order logic. Mathematical Structures in Computer Science **26**(5), 745–788 (2016)
9. Diaconescu, R., Stefaneas, P.S.: Ultraproducts and possible worlds semantics in institutions. Theoretical Computer Science **379**(1-2), 210–230 (2007)
10. Diaconescu, R., Ţuţu, I.: On the algebra of structured specifications. Theoretical Computer Science **412**(28), 3145–3174 (2011)
11. Fitting, M., Mendelsohn, R.L.: First-Order Modal Logic. Kluwer Academic Publishers (1998)
12. Fitting, M., Thalmann, L., Voronkov, A.: Term-modal logics. Studia Logica **69**(1), 133–169 (2001)
13. Gabbay, D.M., Marcelino, S.: Modal logics of reactive frames. Studia Logica **93**(2-3), 405–446 (2009)
14. Goguen, J.A., Burstall, R.M.: Institutions: abstract model theory for specification and programming. Journal of the ACM **39**(1), 95–146 (1992)
15. Goranko, V., Otto, M.: Model theory of modal logic. In: Blackburn, P., van Benthem, J., Wolter, F. (eds.) Handbook of Modal Logic, Studies in logic and practical reasoning, vol. 3, pp. 249–329. North-Holland (2007)
16. Johnson-Freyd, T., Scheimbauer, C.: (Op)lax natural transformations, twisted quantum field theories, and "even higher" Morita categories. Advances in Mathematics **307**, 147–223 (2017)
17. Mac Lane, S.: Categories for the Working Mathematician. Graduate texts in mathematics, Springer (1998)
18. Madeira, A., Neves, R., Martins, M.A., Barbosa, L.S.: Hierarchical hybrid logic. In: Logical and Semantic Frameworks, with Applications. Electronic Notes in Theoretical Computer Science, vol. 338, pp. 167–184. Elsevier (2017)
19. Martins, M.A., Madeira, A., Diaconescu, R., Barbosa, L.S.: Hybridization of institutions. In: Algebra and Coalgebra in Computer Science – 4th International Conference, CALCO 2011. Lecture Notes in Computer Science, vol. 6859, pp. 283–297. Springer (2011)
20. Meseguer, J.: General logics. In: Logic Colloquium '87. Studies in Logic and the Foundations of Mathematics Series, vol. 129, pp. 275–329. Elsevier (1989)
21. O'Sullivan, J.: Paper Soccer, pp. 212–213. Lark Books (2008)

22. Otten, J.: Mleancop: A connection prover for first-order modal logic. In: Demri, S., Kapur, D., Weidenbach, C. (eds.) Automated Reasoning – 7th International Joint Conference. Lecture Notes in Computer Science, vol. 8562, pp. 269–276. Springer (2014)
23. Sahlqvist, H.: Completeness and correspondence in the first and second order semantics for modal logic. In: Kanger, S. (ed.) Proceedings of the Third Scandinavian Logic Symposium, Studies in Logic and the Foundations of Mathematics, vol. 82, pp. 110–143. Elsevier (1975)
24. Sannella, D., Tarlecki, A.: Foundations of Algebraic Specification and Formal Software Development. Monographs in Theoretical Computer Science. An EATCS Series, Springer (2011)
25. Tarlecki, A.: Moving between logical systems. In: Haveraaen, M., Owe, O., Dahl, O. (eds.) Recent Trends in Data Type Specification. Lecture Notes in Computer Science, vol. 1130, pp. 478–502. Springer (1995)
26. Ţuţu, I.: Bisimulations in an arbitrary stratified institution (2023), http://imar.ro/~itutu/papers/basi.pdf, manuscript submitted for publication
27. Ţuţu, I.: Transporting connectives along parchment addenda (2023), https://imar.ro/~itutu/papers/tcpa.pdf, manuscript submitted for publication
28. Ţuţu, I., Chiriţă, C.E., Fiadeiro, J.L.: Dynamic reconfiguration via typed modalities. In: Formal Methods – 24th International Symposium. Lecture Notes in Computer Science, vol. 13047, pp. 599–615. Springer (2021)
29. Vardi, M.Y.: Why is modal logic so robustly decidable? In: Immerman, N., Kolaitis, P.G. (eds.) Descriptive Complexity and Finite Models. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 31, pp. 149–183. DIMACS/AMS (1996)

# Bisimulation in Fuzzy Topological Semantics

Muhammad Afaq Khan[0009−0005−3738−1016], Manuel A. Martins[0000−0002−5109−8066], and Alexandre Madeira[0000−0002−0646−2017]

CIDMA & Department of Mathematics, University of Aveiro, Portugal
{mafaqkhan, martins and madeira}@ua.pt

**Abstract.** Bisimulation is a fundamental tool in modal logic for comparing models in terms of their observable behaviour. It provides a notion of behavioural equivalence and plays an important role in studying the invariance and expressive power of modal languages. Bisimulation has been investigated in different semantic settings, including Kripke models, transition systems, and fuzzy relational structures. This motivates the study of an appropriate notion of bisimulation for fuzzy topological semantics. In this paper, we study bisimulation for fuzzy topological models, where propositional variables are interpreted by topological fuzzy sets and the modal operators are interpreted using fuzzy interior and closure. We introduce a notion of fuzzy topological bisimulation, inspired by classical topological bisimulation and adapted to the many-valued setting. The definition combines preservation of atomic truth degrees with suitable forth and back conditions formulated in terms of fuzzy open sets. Our main result establishes the invariance of modal formulas under fuzzy topological bisimulation, showing that related points have the same modal truth degrees. This work provides a model-theoretic basis for comparing fuzzy topological models and contributes to the further development of fuzzy topological semantics.

## 1 Introduction

Bisimulation is a key notion in the model theory of modal logic. It plays an important role in many areas such as hardware and software verification, control and dynamic systems, communication networks, and social network analysis (see [1, 2]). In classical modal logic, a model consists of a set of states (possible worlds), a binary relation describing how states are connected, and a valuation that tells us, for each atomic proposition, which states make it true. A bisimulation between two models is a relation that pairs states from one model with states from the other model so that related states satisfy exactly the same atomic propositions, and their possible transitions match in a precise way. Since a semantic model of modal logic can be viewed as an automaton or a labeled transition system, checking whether a modal formula holds at a state is similar to running this automaton or computing over the transition system. From this computational perspective, bisimulation captures the idea of observational equivalence of states. Since modal formulas describe the observable behavior of states, bisimilar states satisfy the same modal formulas and are therefore modally

indistinguishable. Conversely, the classical Hennessy–Milner theorem establishes that, under suitable conditions such as image-finiteness, modal equivalence also implies bisimilarity. Thus, for image-finite Kripke models, two states are bisimilar if and only if they satisfy the same modal formulas (see [3]).

Although relational semantics (see [4]) is the standard interpretation of modal logic, topological semantics provides an important alternative perspective on modal operators. The connection between topology and modal logic goes back to the classical work of Tarski and McKinsey in [5]. In a topological model, propositions are interpreted as subsets of a topological space, with necessity represented by the interior operator and possibility by the closure operator. This interpretation gives modal logic a natural neighbourhood-based meaning and allows topological notions such as openness, continuity, and connectedness to be studied within a modal framework. In particular, the modal logic $S4$ is closely connected with the class of topological spaces, see, for example, [6, 7, 2] for detailed treatments of topological semantics. Thus, topological semantics is not merely an alternative to Kripke semantics, it provides a different mathematical framework in which modal behaviour is described through the local and global structure of spaces rather than only through accessibility relations.

The change from relational to topological semantics also requires a corresponding notion of bisimulation. Aiello, van Benthem, and collaborators studied topological bisimulation, also known as topo-bisimulation, in the context of topological models (see [2, 6]). Instead of matching accessible successors, a topo-bisimulation matches open neighbourhoods around related points. Besides agreement on atomic propositions, its forth and back conditions require that suitable open neighbourhoods in one model can be matched by open neighbourhoods in the other model. This notion reflects the topological interpretation of the modal operators and preserves the truth of modal formulas. Moreover, for finite topological models, modal equivalence implies topo-bisimilarity. In this sense, topo-bisimulation plays for topological semantics a role analogous to that of ordinary bisimulation in Kripke semantics.

When uncertainty, vagueness, or graded information is involved, classical two-valued models may be too restrictive. Fuzzy sets introduced by Lotfi A. Zadeh in [8], provide a natural way to represent such information by assigning degrees of membership in the unit interval. This idea has been extended to several mathematical and computational settings. In topology, Chang introduced fuzzy topological spaces in [9], where open sets are represented by fuzzy sets, allowing notions such as interior, closure, and neighbourhoods to be studied in a graded form. In modal logic, the move from the classical to the many-valued setting means that formulas may take truth values in structures such as the real unit interval $[0, 1]$, while accessibility relations may also be graded. This has led to the development of fuzzy and many-valued modal logics, including modal extensions of Gödel, Product, and Łukasiewicz logics (see [10, 11, 12]). To handle uncertainty in states and transitions, researchers have introduced fuzzy automata and fuzzy transition systems. These models incorporate fuzziness into the internal states and outputs of computational systems, and they have been widely stud-

ied and applied (see[13, 14, 15]). Bisimulation has also been studied in these fuzzy settings. Cao, Chen and Kerre investigated bisimulations for fuzzy transition systems [16], while Ćirić et al.[17] developed several notions of bisimulation for fuzzy automata using fuzzy relations. Jain, Madeira and Martins further studied bisimulation and simulation in fuzzy transition systems equipped with a Gödel-based fuzzy modal logic in [18]. These developments provide an important background for studying bisimulation in many-valued and fuzzy modal semantics.

In our previous work (see [19, 20]), we introduced a fuzzy topological semantics for many valued modal logic in which formulas are interpreted as fuzzy subsets of a fuzzy topological space. The propositional connectives are interpreted using Gödel operations, while the modal operators are interpreted through fuzzy interior and fuzzy closure. This framework extends the classical topological interpretation of modal logic to graded truth values and provides a connection between fuzzy topological models and fuzzy Kripke models through fuzzy Alexandrov spaces. We also investigated the validity of modal principles such as those of $S4$ in this setting. The development of fuzzy topological semantics naturally raises a model-theoretic question, what is the appropriate notion of bisimulation for fuzzy topological models? Classical topo-bisimulation is based on ordinary open sets and a crisp relation between points, while existing fuzzy bisimulations are mainly formulated for relational structures such as fuzzy Kripke models, fuzzy automata, and fuzzy transition systems. Neither approach directly captures a semantics in which both truth values and the topological neighbourhood structure are fuzzy. The aim of the present work is therefore to develop a notion of fuzzy topological bisimulation adapted to fuzzy topological models. The proposed relation combines preservation of atomic truth degrees with forth and back conditions formulated in terms of fuzzy open sets. We then study its connection with modal equivalence and, in particular, the extent to which fuzzy topological bisimilarity guarantees invariance of modal formulas. In this way, the paper extends the model-theoretic role of bisimulation from classical topological semantics to the fuzzy topological setting introduced in [19].

## 2 Classical modal logic

The study of reasoning about possibility, necessity, and change has a long history in logic, going back to Aristotle. Modal logic has therefore become one of the classical areas of research in logic and computer science. Alongside its traditional interpretations, modal logic has also been used to reason about space and spatial structures. In this direction, topological semantics provides an important framework by interpreting modal operators through topological notions such as interior and closure. Although logic and geometry have long been closely connected in the foundations of mathematics, systematic logical treatments of spatial structures have been relatively limited, with some of the best-known early contributions due to Tarski. Within this spatial perspective, topological semantics has played an important role in the development of modal logic. In

a topological space, modal operators can be interpreted using topological constructions, especially interior and closure. These interpretations go back to the classical work of McKinsey and Tarski in 1944 in [5], which was among the earliest contributions to the topological semantics of modal logic.

**Definition 1 (Modal Language).** *For a given set of propositional symbols Prop, we define modal formulas, by the following grammar,*

$$\varphi := p \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \varphi \to \psi \mid \Box\varphi \mid \Diamond\varphi, \quad p \in Prop$$

Before introducing the topological semantics of modal logic, we recall the standard notion of a topological space. A topology specifies the collection of subsets regarded as open and thereby provides the underlying structure for notions such as neighbourhood, interior, and closure (see [21]).

**Definition 2 (Topological Space).** *Let $X$ be a non-empty set. A* topology *on $X$ is a collection $\tau \subseteq \mathcal{P}(X)$ satisfying:*

*(i)* $\varnothing, X \in \tau$;
*(ii)* *if* $\{U_i\}_{i\in I} \subseteq \tau$, *then* $\cup_{i\in I} U_i \in \tau$;
*(iii)* *if* $U_1, \ldots, U_n \in \tau$, *then* $\cap_{j=1}^{n} U_j \in \tau$.

*The pair $(X, \tau)$ is called a* topological space, *and the members of $\tau$ are called* open sets.

**Definition 3 (Topological Model [6]).** *A topological model or simply a topo-model is topological space $(X, \tau)$ equipped with evaluation function $v : Prop \to \mathcal{P}(X)$.*

**Definition 4 (Basic Topological Semantics).** *Truth of modal formulas is defined inductively at points $x$ in a topo-model $M = (X, \tau, v)$:*

*i)* $M, x \models p$ *iff* $x \in v(p)$, *(with* $p \in Prop$*),*
*ii)* $M, x \models \neg\varphi$ *iff* $M, x \not\models \varphi$,
*iii)* $M, x \models \varphi \wedge \psi$ *iff* $M, x \models \varphi$ *and* $M, x \models \psi$,
*iv)* $M, x \models \Box\varphi$ *iff* $\exists U \in \tau \left(x \in U \text{ and } \forall y \in U,\ M, y \models \varphi\right)$
*v)* $M, x \models \Diamond\varphi$ *iff* $\forall U \in \tau \left(x \in U \to \exists y \in U :\ M, y \models \varphi\right)$.

*As usual, we can economize by defining, for example, $\varphi \vee \psi$ as $\neg(\neg\varphi \wedge \neg\psi)$, and $\Diamond\varphi$ as $\neg\Box\neg\varphi$. We will do this whenever convenient.*

Let $M = (X, \tau, V)$ be a topological model. For any modal formula $\varphi$, we denote by $[\varphi]_M$ its truth set in $M$, defined as $[\varphi]_M = \{x \in X : M, x \models \varphi\}$. Under the above semantics, the Boolean connectives have their usual set-theoretic interpretation. In particular, negation corresponds to complementation and conjunction to intersection. The modal operators admit a natural topological interpretation. Indeed, $M, x \models \Box\varphi$ holds exactly when there exists an open set $U \in \tau$ such that $x \in U$ and $U \subseteq [\varphi]_M$. This means $x$ has an open neighborhood entirely

contained within the truth set of $\varphi$, which is precisely the condition for $x$ to belong to the interior of $[\varphi]_M$. Therefore,

$$[\Box\varphi]_M = \mathrm{Int}\big([\varphi]_M\big).$$

Similarly, $M, x \models \Diamond\varphi$ holds exactly when every open neighbourhood $U \in \tau$ of $x$ contains a point satisfying $\varphi$, meaning every open neighborhood of $x$ intersects $[\varphi]_M$. This is precisely the condition for $x$ to belong to the **closure** of the truth set of $\varphi$. Thus,

$$[\Diamond\varphi]_M = \mathrm{Cl}\big([\varphi]_M\big).$$

Consequently, in topological semantics, $\Box$ is interpreted by the interior operator, whereas $\Diamond$ is interpreted by the closure operator.

To study the expressive power of a modal language, it is important to introduce an appropriate notion of bisimulation. The following definition is motivated by the semantics of the modal operators and involves two kinds of matching conditions, one concerning related points, and the other concerning the open sets that contain those points.

**Definition 5 (Topological Bisimulation [22]).** *Let $M_1 = (X_1, \tau_1, v_1)$ and $M_2 = (X_2, \tau_2, v_2)$ be two topo-models. A* topological bisimulation*, or simply a* topo-bisimulation*, between $M_1$ and $M_2$ is a non-empty relation*

$$Z \subseteq X_1 \times X_2$$

*such that whenever $x_1 Z x_2$, the following conditions hold*

*i)* (Atom)

$$x_1 \in v_1(p) \iff x_2 \in v_2(p), \qquad \text{for every } p \in Prop.$$

*ii)* (Forth) *For every $U_1 \in \tau_1$ with $x_1 \in U_1$, there exists $U_2 \in \tau_2$ such that $x_2 \in U_2$ and*

$$\forall y_2 \in U_2 \; \exists y_1 \in U_1 : y_1 Z y_2.$$

*iii)* (Back) *For every $U_2 \in \tau_2$ with $x_2 \in U_2$, there exists $U_1 \in \tau_1$ such that $x_1 \in U_1$ and*

$$\forall y_1 \in U_1 \; \exists y_2 \in U_2 : y_1 Z y_2.$$

A topo-bisimulation $Z$ is said to be *total* if its domain is $X_1$ and its range is $X_2$. If $Z$ satisfies only the *Atom* and *Forth* conditions, then we say that $M_2$ *simulates* $M_1$. Topo-bisimulation provides the appropriate notion of equivalence between the topo-models $M_1$ and $M_2$ with respect to the basic modal language (see [2, 22, 23]).

## 3 Fuzzy modal logic

Fuzzy modal logics extend classical modal logic by allowing modal notions such as necessity and possibility to be interpreted in a many-valued setting. Several fuzzy modal systems have been studied in the literature, depending on the many-valued logic adopted as their propositional basis. Among them, Gödel modal logic is obtained by combining the semantics of Gödel logic with modal reasoning and constitutes one of the well-studied approaches to many-valued modal logic. In the standard Gödel algebra over the unit interval ([0,1]), conjunction is interpreted by the Gödel (t)-norm, that is, the minimum operation, while implication is given by its residuum. In what follows, we take Gödel logic as the underlying fuzzy logic (see [19, 20]).

**Definition 6.** *Let Prop be a set of atomic propositions, the set of modal formulas* Form *is given by the following grammar:*

$$\varphi, \psi := p \mid \bot \mid \varphi \wedge \psi \mid \varphi \to \psi \mid \Diamond\varphi \mid \Box\varphi, \text{ where } p \in Prop.$$

*The negation of a formula $\varphi$ and $\top$ formula are defined by $\neg\varphi := \varphi \to \bot$ and $\top := \bot \to \bot$.*

**Fuzzy Topological Spaces.** Fuzzy set theory was first introduced by Lotfi Zadeh [8] in 1965, laying the foundation for fuzzy topology. Building on this, C.L. Chang [9] introduced the concept of fuzzy topological spaces in 1968, defining a fuzzy topology on a set $X$ as a family $\mathbf{T}$ of fuzzy sets in $X$[1] satisfying certain axioms, where each element of $\mathbf{T}$ is regarded as a fuzzy open set. J. A. Goguen [24] extended this framework to membership functions defined over arbitrary complete distributive lattices.

We consider the usual functions $0_X$ and $1_X$, where $0_X(x) = 0$ for all $x \in X$, known as the *fuzzy empty set* or the fuzzy set assigning zero membership to every element, and $1_X(x) = 1$ for all $x \in X$, which represents the fuzzy set with maximum (full) membership value for each element. The standard fuzzy set operations of union and intersection are defined pointwise using the maximum and minimum operators, corresponding to the operations of the Gödel algebra. The complement of a fuzzy set $A$ is defined by $A^c(x) = 1 - A(x)$. A fuzzy set is *fuzzy closed* if its complement is fuzzy open, the collection of all such sets is denoted $T^c = \{A : A^c \in T\}$.

**Definition 7 (Fuzzy Topological Space).** *A family $\mathbf{T}$ of fuzzy sets in a nonempty set $X$ is known as a* fuzzy topology *on $X$ if it satisfies the following conditions:*

*(T1)* $0_X, 1_X \in \mathbf{T}$
*(T2)* $G_1 \bigcap G_2 \in \mathbf{T}$, *for any* $G_1, G_2 \in \mathbf{T}$
*(T3)* $\bigcup G_j \in \mathbf{T}$, *for any arbitrary family* $G_j, j \in J$, *of elements of* $T$

*The elements of $\mathbf{T}$ are called* fuzzy open sets *and the pair $(X, \mathbf{T})$ is a* topological space.

[1] Recall that a fuzzy set in $X$ is a mapping $\mu : X \to [0, 1]$, i.e., an element of $I^X$.

The degree based framework defines topological concepts in terms of membership values, allowing us to study spatial relationships without assuming sharply defined boundaries. We recall below some fundamental fuzzy topological notions that will be used throughout this work. For any two fuzzy sets $\lambda, \mu \in [0,1]^X$, the pointwise partial order is defined by $\lambda \leq \mu$ if and only if $\lambda(x) \leq \mu(x)$ for every $x \in X$. The supremum and infimum of families of fuzzy sets are computed pointwise.

**Definition 8 (Fuzzy Interior and Fuzzy Closure).** *The union of all open subsets contained in a fuzzy set $\lambda$ is known as the* interior of the fuzzy set $\lambda$. *Formally, it is given by $Int(\lambda) = \sup\{P : P \leq \lambda, P \in \mathbf{T}\}$ and it is denoted as $Int(\lambda)$.*

*The* fuzzy closure *$Cl(\lambda)$ is the intersection of all closed sets which contain $\lambda$ and it is given by $Cl(\lambda) = \inf\{G : G^c \in \mathbf{T}, \lambda \leq G\}$.*

*Remark 1.* As an immediate consequence of Definition 8, the fuzzy interior and closure operators satisfy the De Morgan duality with respect to the standard pointwise complement $1_X - \lambda$:

$$\mathrm{Cl}(\lambda) = 1_X - \mathrm{Int}(1_X - \lambda) \qquad \text{and} \qquad \mathrm{Int}(\lambda) = 1_X - \mathrm{Cl}(1_X - \lambda).$$

**Definition 9 (Fuzzy Topological Model [19]).** *A* fuzzy topological model *is a tuple $\mathcal{M} = (X, \mathbf{T}, V)$ where $(X, \mathbf{T})$ is a fuzzy topological space and $V$ is a valuation, i.e., a map $V : Prop \to [0,1]^X$.*

**Definition 10.** *Let $\mathcal{M} = (X, \mathbf{T}, V)$ be a fuzzy topological model and $x \in X$. The interpretation of a formula $\varphi \in \mathsf{Form}$ at a point $x \in X$, in symbols $[\![\varphi]\!]^x_{\mathcal{M}} \in [0,1]$, is recursively defined by*

*i)* $[\![p]\!]^x_{\mathcal{M}} = V(p)(x), p \in Prop$
*ii)* $[\![\bot]\!]^x_{\mathcal{M}} = 0$
*iii)* $[\![\varphi \wedge \psi]\!]^x_{\mathcal{M}} = \min([\![\varphi]\!]^x_{\mathcal{M}}, [\![\psi]\!]^x_{\mathcal{M}})$ *(Gödel t-norm)*
*iv)* $[\![\varphi \to \psi]\!]^x_{\mathcal{M}} = \begin{cases} 1 & \text{if } [\![\varphi]\!]^x_{\mathcal{M}} \leq [\![\psi]\!]^x_{\mathcal{M}} \\ [\![\psi]\!]^x_{\mathcal{M}} & \text{otherwise} \end{cases}$ *(Gödel residuum)*
*v)* $[\![\Diamond\varphi]\!]^x_{\mathcal{M}} = \inf_{\mu^c \in \mathbf{T}}\{\mu(x) : [\![\varphi]\!]_{\mathcal{M}} \leq \mu\}$
*vi)* $[\![\Box\varphi]\!]^x_{\mathcal{M}} = \sup_{\mu \in \mathbf{T}}\{\mu(x) : \mu \leq [\![\varphi]\!]_{\mathcal{M}}\}$.

The value $[\![\varphi]\!]^x_{\mathcal{M}}$ is read as "*the degree of truth of the formula $\varphi$ in the model $\mathcal{M}$ at the point $x$*". A formula $\varphi$ is said to be *true in a model* $\mathcal{M} = (X, \mathbf{T}, V)$ if $[\![\varphi]\!]^x_{\mathcal{M}} = 1$ for every $x \in X$. We say that $\varphi$ is *valid in a fuzzy topological space* $(X, \mathbf{T})$, denoted as $(X, \mathbf{T}) \models \varphi$ or $\mathbf{T} \models \varphi$, if it is true in every model based on $(X, \mathbf{T})$. Finally, $\varphi$ is said to be *valid in a class $K$* of fuzzy topological spaces if it is valid in every member of $K$. Furthermore, as a direct consequence of the definition of the residuum and the abbreviation $\top := \bot \to \bot$, it is useful to note that $[\![\top]\!]^x_{\mathcal{M}} = 1$ for any $x \in X$. It is not difficult to see that $[\![\varphi]\!]_{\mathcal{M}}(x) = [\![\varphi]\!]^x_{\mathcal{M}}$. Hence, $\varphi$ is valid in $\mathcal{M}$ iff $[\![\varphi]\!]_{\mathcal{M}} = 1_X$. Consequently, from the properties of residuum, we have that $\varphi \to \psi$ is valid in $\mathcal{M}$ iff $[\![\varphi]\!]_{\mathcal{M}} \leq [\![\psi]\!]_{\mathcal{M}}$.

*Example 1.* Consider the fuzzy topological model $\mathcal{M} = (X, \tau, V)$, where $X = \{x_1, x_2\}$. A fuzzy subset of $X$ will be represented by the ordered pair $A = (A(x_1), A(x_2))$. Let $\mathbf{0}_X = (0,0)$, $\mathbf{1}_X = (1,1)$, and consider the fuzzy set $U = (0.2, 0.4)$. Define the fuzzy topology,

$$\tau = \{\mathbf{0}_X, U, \mathbf{1}_X\}.$$

Since $\mathbf{0}_X, \mathbf{1}_X \in \tau$, and $\tau$ is closed under arbitrary suprema and finite infima, $(X, \tau)$ forms a fuzzy topological space. Let $p$ be a propositional variable with valuation $V(p) = A = (0.5, 0.5)$. The modal operators are interpreted through the fuzzy interior and closure, respectively. For any formula $\varphi$, their interpretations in $\mathcal{M}$ are given by

$$[\![\Box\varphi]\!]_{\mathcal{M}} = \mathrm{Int}([\![\varphi]\!]_{\mathcal{M}}), \qquad [\![\Diamond\varphi]\!]_{\mathcal{M}} = \mathrm{Cl}([\![\varphi]\!]_{\mathcal{M}}).$$

Recall that the fuzzy interior is defined as, $\mathrm{Int}(A) = \sup\{O \in \tau : O \leq A\}$, where the order between fuzzy sets is pointwise. For $A = (0.5, 0.5)$, we have $\mathbf{0}_X \leq A$ and $U \leq A$, whereas $\mathbf{1}_X \not\leq A$. Hence, the largest fuzzy open set contained in $A$ is $U$. Therefore,

$$\mathrm{Int}(A) = U = (0.2, 0.4).$$

Consequently,$[\![\Box p]\!]_{\mathcal{M}} = \mathrm{Int}(V(p)) = (0.2, 0.4)$, and thus, $[\![\Box p]\!]_{\mathcal{M}}(x_1) = 0.2$ and $[\![\Box p]\!]_{\mathcal{M}}(x_2) = 0.4$. Now consider the diamond operator. Using the standard fuzzy-set complement $B^c(x) = 1 - B(x)$, the fuzzy closed sets associated with $\tau$ are $\mathbf{0}_X$, $U^c = (0.8, 0.6)$, and $\mathbf{1}_X$. The fuzzy closure is given by,

$$\mathrm{Cl}(A) = \inf\{C : C^c \in \tau, A \leq C\}.$$

Since $A = (0.5, 0.5) \leq (0.8, 0.6) = U^c$ and $A \leq \mathbf{1}_X$, the smallest closed fuzzy set containing $A$ is $U^c$. Hence, $\mathrm{Cl}(A) = (0.8, 0.6)$. Therefore, $[\![\Diamond p]\!]_{\mathcal{M}} = (0.8, 0.6)$, so that,

$$[\![\Diamond p]\!]_{\mathcal{M}}(x_1) = 0.8, \quad [\![\Diamond p]\!]_{\mathcal{M}}(x_2) = 0.6.$$

This example illustrates the topological meaning of the modal operators in the fuzzy setting. The formula $\Box p$ is evaluated by taking the greatest fuzzy open set contained in the truth-value fuzzy set of $p$, whereas $\Diamond p$ is evaluated by taking the least fuzzy closed set containing it.

## 4 Fuzzy topological bisimulation

In this section, we introduce notions of simulation and bisimulation for fuzzy topological models, extending the corresponding classical topological notions to the fuzzy setting. Our approach allows us to compare two models that may be based on different fuzzy topologies and different fuzzy valuation functions.

**Definition 11 (Fuzzy topological bisimulation).** *Let $\mathcal{M}_1 = (X_1, \mathbf{T}_1, V_1)$ and $\mathcal{M}_2 = (X_2, \mathbf{T}_2, V_2)$ be fuzzy topological models, and let $Z \subseteq X_1 \times X_2$ be*

*a non-empty relation. For a fuzzy set* $\mu \in [0,1]^{X_1}$*, define its relational image under* $Z$ *by*

$$Z[\mu](y) = \sup\{\mu(x) : x \in X_1,\ xZy\}, \qquad y \in X_2,$$

*and, for a fuzzy set* $\lambda \in [0,1]^{X_2}$*, the relational image under* $Z^{-1}$

$$Z^{-1}[\lambda](x) = \sup\{\lambda(y) : y \in X_2,\ xZy\}, \qquad x \in X_1,$$

*where the supremum of the empty set is taken to be* 0*. The relation* $Z$ *is called a* fuzzy topological bisimulation *between* $\mathcal{M}_1$ *and* $\mathcal{M}_2$ *if the following conditions hold:*

*i)* (Atom) *For every* $x \in X_1$*,* $y \in X_2$*, and propositional variable* $p$*, if* $xZy$*, then* $V_1(p)(x) = V_2(p)(y)$*.*
*ii)* (Zig) *For every* $\mu \in \mathbf{T}_1$*,* $Z[\mu] \in \mathbf{T}_2$*.*
*iii)* (Zag) *For every* $\lambda \in \mathbf{T}_2$*,* $Z^{-1}[\lambda] \in \mathbf{T}_1$*.*

*Two points* $x \in X_1$ *and* $y \in X_2$ *are said to be* fuzzy topologically bisimilar*, written*

$$(\mathcal{M}_1, x) \leftrightarrow_F (\mathcal{M}_2, y),$$

*if there exists a fuzzy topological bisimulation* $Z$ *between* $\mathcal{M}_1$ *and* $\mathcal{M}_2$ *such that* $xZy$*.*

A fuzzy topological bisimulation $Z \subseteq X_1 \times X_2$ is called *total* if domain is $X_1$ and range is $X_2$. If $Z$ satisfies only the Atom condition and the Zig condition, then $Z$ is called a *fuzzy topological simulation* from $\mathcal{M}_1$ to $\mathcal{M}_2$. In this case, we say that $\mathcal{M}_2$ simulates $\mathcal{M}_1$. We use a crisp relation $Z$ to ensure exact preservation of modal truth degrees at related points. A fuzzy-valued relation would require different bisimulation conditions, since related points could have different atomic truth degrees. Moreover, weighted relational images may not preserve fuzzy openness, making (Zig) and (Zag) more difficult to formulate. Therefore, we adopt a crisp relation with exact atomic agreement and openness preservation, following the classical notion of topological bisimulation. By Definition 11, the domain and range of a fuzzy topological bisimulation $Z \subseteq X_1 \times X_2$ have fuzzy open characteristic functions. Indeed, applying (Zig) and (Zag) to the constant-one fuzzy sets gives

$$Z[\mathbf{1}X_1] = \chi\mathrm{ran}(Z) \in \mathbf{T}2, \qquad Z^{-1}[\mathbf{1}X_2] = \chi_{\mathrm{dom}(Z)} \in \mathbf{T}_1,$$

where $\chi_A$ denotes the characteristic function of $A$. Thus, $Z$ need not relate every point of either model, but its domain and range must have fuzzy open characteristic functions.

Given a classical topological model $\mathcal{M} = (X, \tau, v)$, we define its corresponding crisp fuzzy topological model as $\mathcal{M}^F = (X, \mathbf{T}, V)$, where $\mathbf{T} = \{\chi_U : U \in \tau\}$ and $V(p) = \chi_{v(p)}$ for all $p \in \mathsf{Prop}$. The characteristic function $\chi_A \colon X \to \{0,1\}$ is defined by

$$\chi_A(x) = \begin{cases} 1, & \text{if } x \in A, \\ 0, & \text{if } x \notin A. \end{cases}$$

**Proposition 1.** *Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be classical topological models, and let $\mathcal{M}_1^F$ and $\mathcal{M}_2^F$ be their corresponding crisp fuzzy models. A non-empty relation $Z \subseteq X_1 \times X_2$ is a classical topological bisimulation between $\mathcal{M}_1$ and $\mathcal{M}_2$ if and only if it is a fuzzy topological bisimulation between $\mathcal{M}_1^F$ and $\mathcal{M}_2^F$.*

*Proof.* We first observe how the relational image behaves on crisp fuzzy sets. Let $U \in \tau_1$. By definition,

$$Z[\chi_U](y) = \sup\{\chi_U(x) : x \in X_1,\ xZy\},$$

Note that, $\sup \varnothing = 0$. Since $\chi_U$ takes only the values 0 and 1, we have $Z[\chi_U](y) = 1$ if and only if there exists $x \in U$ such that $xZy$; otherwise $Z[\chi_U](y) = 0$. Hence, $Z[\chi_U] = \chi_{Z[U]}$, where

$$Z[U] = \{y \in X_2 : \exists x \in U \text{ such that } xZy\}.$$

Similarly, for every $U' \in \tau_2$, $Z^{-1}[\chi_{U'}] = \chi_{Z^{-1}[U']}$, where

$$Z^{-1}[U'] = \{x \in X_1 : \exists y \in U' \text{ such that } xZy\}.$$

We now compare the corresponding bisimulation conditions. For the (Atom) condition, suppose $xZy$. Since

$$V_1(p) = \chi_{v_1(p)} \qquad \text{and} \qquad V_2(p) = \chi_{v_2(p)},$$

the equality $V_1(p)(x) = V_2(p)(y)$ holds if and only if

$$x \in v_1(p) \quad \Longleftrightarrow \quad y \in v_2(p).$$

Thus, the fuzzy (Atom) condition reduces exactly to the classical atomic condition.

Consider now the (Zig) condition. In the fuzzy setting, for every $\chi_U \in \mathbf{T}_1$, $Z[\chi_U] \in \mathbf{T}_2$. Using $Z[\chi_U] = \chi_{Z[U]}$, this is equivalent to $Z[U] \in \tau_2$. We show that this is equivalent to the classical (Zig) condition. Suppose first that $Z[U]$ is open for every $U \in \tau_1$. Let $xZy$ and let $U \in \tau_1$ with $x \in U$. Then $y \in Z[U]$, and since $Z[U] \in \tau_2$, we may take

$$U' = Z[U].$$

For every $y' \in U'$, by the definition of $Z[U]$, there exists $x' \in U$ such that $x'Zy'$. Therefore the classical (Zig) condition holds.

Conversely, suppose that the classical (Zig) condition holds. Let $U \in \tau_1$, and take any $y \in Z[U]$. Then there exists $x \in U$ such that $xZy$. By the classical (Zig) condition, there exists an open set $U'_y \in \tau_2$ such that

$$y \in U'_y$$

and every $y' \in U'_y$ is related by $Z$ to some point of $U$. Therefore $U'_y \subseteq Z[U]$. Thus every point of $Z[U]$ has an open neighbourhood contained in $Z[U]$, and consequently $Z[U] \in \tau_2$. Hence,

$$Z[\chi_U] = \chi_{Z[U]} \in \mathbf{T}_2,$$

which is precisely the fuzzy (Zig) condition.
The same argument, with the roles of the two models reversed, shows that

$$Z^{-1}[\chi_{U'}] \in \mathbf{T}_1$$

for every $U' \in \tau_2$ is equivalent to the classical (Zag) condition. Therefore the (Atom), (Zig), and (Zag) conditions of fuzzy topological bisimulation reduce exactly to the corresponding conditions of classical topological bisimulation whenever the fuzzy topologies and valuations are crisp. Hence $Z$ is a fuzzy topological bisimulation between $\mathcal{M}_1^F$ and $\mathcal{M}_2^F$ if and only if it is a classical topological bisimulation between $\mathcal{M}_1$ and $\mathcal{M}_2$.

**Theorem 1.** *Let $\mathcal{M}_1 = (X_1, \mathbf{T}_1, V_1)$ and $\mathcal{M}_2 = (X_2, \mathbf{T}_2, V_2)$ be fuzzy topological models, and let $Z \subseteq X_1 \times X_2$ be a fuzzy topological bisimulation between them. If $x \in X_1$, $y \in X_2$, and $xZy$, then, for every modal formula $\varphi$,*

$$[\![\varphi]\!]^x_{\mathcal{M}_1} = [\![\varphi]\!]^y_{\mathcal{M}_2}.$$

*Proof.* We prove, by induction on the structure of the formula $\varphi$, that for all $x \in X_1$ and $y \in X_2$,

$$xZy \quad \Longrightarrow \quad [\![\varphi]\!]^x_{\mathcal{M}_1} = [\![\varphi]\!]^y_{\mathcal{M}_2}.$$

Let $\varphi = p$, where $p \in Prop$. If $xZy$, then, since $Z$ is a fuzzy topological bisimulation, the Atom condition gives

$$V_1(p)(x) = V_2(p)(y).$$

Hence,

$$[\![p]\!]^x_{\mathcal{M}_1} = [\![p]\!]^y_{\mathcal{M}_2}.$$

The propositional connectives are evaluated pointwise. Therefore, their cases follow immediately from the induction hypothesis, since applying the same truth function to equal truth values yields equal results. We show for conjunction and implication, if $xZy$, then by the induction hypothesis, $[\![\varphi]\!]^x_{\mathcal{M}_1} = [\![\varphi]\!]^y_{\mathcal{M}_2}$ and $[\![\psi]\!]^x_{\mathcal{M}_1} = [\![\psi]\!]^y_{\mathcal{M}_2}$. Consequently,

$$\begin{aligned} [\![\varphi \wedge \psi]\!]^x_{\mathcal{M}_1} &= [\![\varphi]\!]^x_{\mathcal{M}_1} \wedge [\![\psi]\!]^x_{\mathcal{M}_1} \\ &= [\![\varphi]\!]^y_{\mathcal{M}_2} \wedge [\![\psi]\!]^y_{\mathcal{M}_2} \\ &= [\![\varphi \wedge \psi]\!]^y_{\mathcal{M}_2}. \end{aligned}$$

Since implication is evaluated pointwise by the same fuzzy implication operation in both models, it follows that

$$\begin{aligned} [\![\varphi \to \psi]\!]^x_{\mathcal{M}_1} &= [\![\varphi]\!]^x_{\mathcal{M}_1} \Rightarrow [\![\psi]\!]^x_{\mathcal{M}_1} \\ &= [\![\varphi]\!]^y_{\mathcal{M}_2} \Rightarrow [\![\psi]\!]^y_{\mathcal{M}_2} \\ &= [\![\varphi \to \psi]\!]^y_{\mathcal{M}_2}. \end{aligned}$$

Thus the invariance property is preserved under implication. The remaining propositional connectives are treated analogously. We now establish an auxiliary observation that will be used for the modal cases.

*Claim.* Let $B_1 : X_1 \to [0,1]$ and $B_2 : X_2 \to [0,1]$ be fuzzy sets such that $uZv \implies B_1(u) = B_2(v)$ for all $u \in X_1$ and $v \in X_2$. Then, for every $x \in X_1$ and $y \in X_2$ such that $xZy$,

$$\mathrm{Int}_{\mathbf{T}_1}(B_1)(x) = \mathrm{Int}_{\mathbf{T}_2}(B_2)(y).$$

To prove the claim, fix $xZy$. Let $\mu \in \mathbf{T}_1$ satisfy $\mu \leq B_1$. By the Zig condition, $Z[\mu] \in \mathbf{T}_2$, where

$$Z[\mu](v) = \sup\{\mu(u) : uZv\}.$$

We show that $Z[\mu] \leq B_2$. Fix $v \in X_2$. If $uZv$, then $\mu(u) \leq B_1(u) = B_2(v)$. Thus every value occurring in the supremum defining $Z[\mu](v)$ is bounded above by $B_2(v)$, and therefore

$$Z[\mu](v) \leq B_2(v).$$

If there is no $u \in X_1$ such that $uZv$, then $Z[\mu](v) = 0 \leq B_2(v)$, using the fact $\sup \varnothing = 0$. Hence, $Z[\mu] \leq B_2$. Since $xZy$, the value $\mu(x)$ occurs among the values whose supremum defines $Z[\mu](y)$. Hence,

$$\mu(x) \leq Z[\mu](y).$$

Since $Z[\mu] \in \mathbf{T}_2$ and $Z[\mu] \leq B_2$, the definition of fuzzy interior yields $Z[\mu](y) \leq \mathrm{Int}_{\mathbf{T}_2}(B_2)(y)$. Therefore, $\mu(x) \leq \mathrm{Int}_{\mathbf{T}_2}(B_2)(y)$. Taking the supremum over all $\mu \in \mathbf{T}_1$ such that $\mu \leq B_1$, we obtain

$$\mathrm{Int}_{\mathbf{T}_1}(B_1)(x) \leq \mathrm{Int}_{\mathbf{T}_2}(B_2)(y).$$

For the converse inequality, let $\nu \in \mathbf{T}_2$ satisfy $\nu \leq B_2$. By the Zag condition, $Z^{-1}[\nu] \in \mathbf{T}_1$, where

$$Z^{-1}[\nu](u) = \sup\{\nu(v) : uZv\}.$$

We claim that $Z^{-1}[\nu] \leq B_1$. Indeed, if $uZv$, then $\nu(v) \leq B_2(v) = B_1(u)$. Hence every value occurring in the supremum defining $Z^{-1}[\nu](u)$ is bounded above by $B_1(u)$. If there is no $v \in X_2$ such that $uZv$, then $Z^{-1}[\nu](u) = 0 \leq B_1(u)$. Thus,

$$Z^{-1}[\nu] \leq B_1.$$

Since $xZy$, we also have $\nu(y) \leq Z^{-1}[\nu](x)$. As $Z^{-1}[\nu] \in \mathbf{T}_1$ and $Z^{-1}[\nu] \leq B_1$, it follows that $Z^{-1}[\nu](x) \leq \mathrm{Int}_{\mathbf{T}_1}(B_1)(x)$. Hence,

$$\nu(y) \leq \mathrm{Int}_{\mathbf{T}_1}(B_1)(x).$$

Taking the supremum over all $\nu \in \mathbf{T}_2$ such that $\nu \leq B_2$, we obtain $\mathrm{Int}_{\mathbf{T}_2}(B_2)(y) \leq \mathrm{Int}_{\mathbf{T}_1}(B_1)(x)$. Therefore,

$$\mathrm{Int}_{\mathbf{T}_1}(B_1)(x) = \mathrm{Int}_{\mathbf{T}_2}(B_2)(y),$$

which proves the claim.
We now consider the modal operators. For the $\Box$-case, let $A_1 = [\![\varphi]\!]_{\mathcal{M}_1}$ and

$A_2 = [\![\varphi]\!]_{\mathcal{M}_2}$. By the induction hypothesis, $uZv \implies A_1(u) = A_2(v)$ for all $u \in X_1$ and $v \in X_2$. Therefore, for every $xZy$, the claim gives

$$\mathrm{Int}_{\mathbf{T}_1}(A_1)(x) = \mathrm{Int}_{\mathbf{T}_2}(A_2)(y).$$

By the semantics of $\Box$, $[\![\Box\varphi]\!]_{\mathcal{M}_i} = \mathrm{Int}_{\mathbf{T}_i}(A_i)$, and hence

$$[\![\Box\varphi]\!]^x_{\mathcal{M}_1} = [\![\Box\varphi]\!]^y_{\mathcal{M}_2}.$$

For the $\Diamond$-case, again let $A_1 = [\![\varphi]\!]_{\mathcal{M}_1}$ and $A_2 = [\![\varphi]\!]_{\mathcal{M}_2}$. By the induction hypothesis, $uZv \implies A_1(u) = A_2(v)$. Let

$$A_i^c = 1 - A_i$$

denote the standard fuzzy-set complement used in Chang fuzzy topology (see [9]). This complement is a topological operation and should not be confused with Gödel negation. From the induction hypothesis, it follows that

$$uZv \implies A_1^c(u) = A_2^c(v).$$

Applying the claim to $A_1^c$ and $A_2^c$, we obtain, whenever $xZy$, $\mathrm{Int}_{\mathbf{T}_1}(A_1^c)(x) = \mathrm{Int}_{\mathbf{T}_2}(A_2^c)(y)$. Since fuzzy closure is given by $\mathrm{Cl}_{\mathbf{T}_i}(A_i) = (\mathrm{Int}_{\mathbf{T}_i}(A_i^c))^c$, we have

$$\begin{aligned}\mathrm{Cl}_{\mathbf{T}_1}(A_1)(x) &= 1 - \mathrm{Int}_{\mathbf{T}_1}(A_1^c)(x)\\ &= 1 - \mathrm{Int}_{\mathbf{T}_2}(A_2^c)(y)\\ &= \mathrm{Cl}_{\mathbf{T}_2}(A_2)(y).\end{aligned}$$

By the semantics of $\Diamond$, $[\![\Diamond\varphi]\!]_{\mathcal{M}_i} = \mathrm{Cl}_{\mathbf{T}_i}(A_i)$, and therefore $[\![\Diamond\varphi]\!]^x_{\mathcal{M}_1} = [\![\Diamond\varphi]\!]^y_{\mathcal{M}_2}$. Thus every case of the induction has been established. Consequently, for all $x \in X_1$ and $y \in X_2$,

$$xZy \implies [\![\varphi]\!]^x_{\mathcal{M}_1} = [\![\varphi]\!]^y_{\mathcal{M}_2}.$$

Hence the truth degree of every modal formula is invariant under fuzzy topological bisimulation.

## 5 Conclusion and Future Work

In this article, fuzzy topological bisimulation, compatible with the underlying fuzzy topological semantics, is introduced between fuzzy topological models using crisp relations together with relational images of fuzzy open sets. The proposed framework provides an extension of the classical setting, as the classical topological bisimulation can be recovered through the crisp embedding of classical topological models. Modal invariance is also established, showing that fuzzy topologically bisimilar points have the same truth degree for every formula of the modal language.

The proposed work opens up several directions for future research, including a directly related problem to investigate Hennessy-Milner type converse by identifying suitable topological or structural conditions under which modal equivalence

implies fuzzy topological bisimilarity. Suitable finiteness assumptions and particular classes of fuzzy topological spaces, such as fuzzy Alexandrov spaces, are also open to be examined. The applications including decision procedures and algorithms for checking bisimilarity in finite fuzzy topological models can also be explored under the computational aspects of the proposed notion. These developments can lead to provide a basis for applying fuzzy topological bisimulation in real world applications such as automated reasoning and verification of fuzzy systems.

**Acknowledgement.** This work was supported by CIDMA (https://ror.org/05pm2mw36) through the Portuguese Foundation for Science and Technology (FCT, https://ror.org/00snfqn58), under grants UID/04106/2025 (https://doi.org/10.54499/UID/04106/2025) and UID/PRR/04106/2025 and by Project Banksy with reference COMPETE2030-FEDER-00892000.

## References

[1] Patrick Blackburn et al. *Modal logic*. Vol. 53. Cambridge University Press Cambridge, 2001. DOI: 10.1017/CBO9781107050884.

[2] Marco Aiello, Ian Pratt-Hartmann, Johan Van Benthem, et al. *Handbook of spatial logics*. Vol. 4. Springer, 2007.

[3] Matthew Hennessy and Robin Milner. "On observing nondeterminism and concurrency". In: *Automata, Languages and Programming*. Ed. by Jaco de Bakker and Jan van Leeuwen. Springer Berlin Heidelberg, 1980, pp. 299–309. DOI: 10.1007/3-540-10003-2_79.

[4] Saul A. Kripke. "Semantical considerations on modal logic". In: *Acta philosophica fennica* 16 (1963).

[5] J. C. C. McKinsey and Alfred Tarski. "The Algebra of Topology". In: *Annals of Mathematics* 45.1 (1944), pp. 141–191. DOI: 10.2307/1969080.

[6] Johan van Benthem and Guram Bezhanishvili. "Modal logics of space". In: *Handbook of spatial logics*. Springer, 2007, pp. 217–298. DOI: 10.1007/978-1-4020-5587-4_5.

[7] Robert Goldblatt. *Topoi: the categorial analysis of logic*. Vol. 98. Elsevier, 2014.

[8] Lotfi A. Zadeh. "Fuzzy sets". In: *Information and control* 8.3 (1965), pp. 338–353. DOI: 10.1016/S0019-9958(65)90241-X.

[9] Chin-Liang Chang. "Fuzzy topological spaces". In: *Journal of mathematical Analysis and Applications* 24.1 (1968), pp. 182–190. DOI: 10.1016/0022-247X(68)90057-7.

[10] Melvin C. Fitting. "Many-valued modal logics". In: *Fundamenta informaticae* 15.3-4 (1991), pp. 235–254. DOI: 10.3233/FI-1991-153-404.

[11] Félix Bou et al. "On the minimum many-valued modal logic over a finite residuated lattice". In: *Journal of Logic and computation* 21.5 (2011), pp. 739–790. DOI: 10.1093/logcom/exp062.

[12] Félix Bou et al. "Characterizing Fuzzy Modal Semantics by Fuzzy Multimodal Systems with Crisp Accessibility Relations." In: *IFSA/EUSFLAT Conf.* 2009, pp. 1541–1546.

[13] Alexandre Madeira, Renato Neves, and Manuel A. Martins. "An exercise on the generation of many-valued dynamic logics". In: *Journal of Logical and Algebraic Methods in Programming* 85.5 (2016), pp. 1011–1037. DOI: 10.1016/j.jlamp.2016.03.004.

[14] Tuan-Fang Fan. "Fuzzy Bisimulation for Gödel Modal Logic". In: *IEEE Transactions on Fuzzy Systems* 23.6 (2015), pp. 2387–2396. DOI: 10.1109/TFUZZ.2015.2426724.

[15] Mansoor Doostfatemeh and Stefan C Kremer. "New directions in fuzzy automata". In: *International Journal of Approximate Reasoning* 38.2 (2005), pp. 175–214. DOI: 10.1016/j.ijar.2004.08.001.

[16] Yongzhi Cao, Guoqing Chen, and Etienne E. Kerre. "Bisimulations for Fuzzy Transition Systems". In: *IEEE Transactions on Fuzzy Systems* 19.3 (2011), pp. 540–552. DOI: 10.1109/TFUZZ.2011.2117431.

[17] Miroslav Ćirić et al. "Bisimulations for fuzzy automata". In: *Fuzzy Sets and Systems* 186.1 (2012), pp. 100–139. DOI: 10.1016/j.fss.2011.07.003.

[18] Manisha Jain, Alexandre Madeira, and Manuel A. Martins. "A fuzzy modal logic for fuzzy transition systems". In: *Electronic Notes in Theoretical Computer Science* 348 (2020), pp. 85–103. DOI: 10.1016/j.entcs.2020.02.006.

[19] Muhammad Afaq Khan and Manuel A. Martins. "On Fuzzy Topological Semantics". In: *Dynamic Logic. New Trends and Applications*. Ed. by Jing Wang, Alexandre Madeira, and Lei Li. Cham: Springer Nature Switzerland, 2026, pp. 115–133. DOI: 10.1007/978-3-032-22626-6_7.

[20] Muhammad Afaq Khan and Manuel A. Martins. "Modal Definability and Preservation in Fuzzy Topological Semantics". In: *Journal of Logical and Algebraic Methods in Programming* (2026). Submitted.

[21] John L. Kelley. *General Topology*. Toronto-New York-London: D. Van Nostrand Company, Inc., 1955, pp. xiv+298.

[22] Marco Aiello and Johan Van Benthem. "Logical patterns in space". In: (2002).

[23] Marco Aiello and Johan van Benthem. "A Modal Walk Through Space". In: *Journal of Applied Non-Classical Logics* 12.3-4 (2002), pp. 319–363. DOI: 10.3166/jancl.12.319-363.

[24] Joseph A Goguen. "The fuzzy Tychonoff theorem". In: *Journal of Mathematical Analysis and Applications* 43.3 (1973), pp. 734–742.

# Timed Reconfigurable Systems: A Tool for Encoding and Analysis

Joshua Dourado[1][0009−0005−5360−6852] and Antonio César Castro Iglesias[1][0000−0002−8531−6535]

CIDMA - Research Center in Mathematics and Applications, Mathematics Department of University of Aveiro, Portugal

### Introduction and Motivation

Modern systems, such as dynamic recommendation algorithms, must satisfy strict timing constraints while continuously adapting their configurations based on user interactions. Classical Timed Automata (TA) [1] are extensively used to verify real-time systems but lack native support for such runtime reconfigurations. Encoding structural changes (e.g., enabling or disabling transitions) in classical TA relies on artificial workarounds, such as appending boolean flags to every guard. This obfuscates the model's semantics and worsen the state-space explosion problem.

To overcome these limitations, the Reconfigurable Timed Automata ReTA [6] formalism natively supports dynamic reconfigurations through the mathematical concept of *hyper-edges*. Since mainstream model checkers lack support for these evolving structures, we present RTAtool [1], a dedicated integrated environment designed specifically for the modeling, simulation, and formal analysis of ReTA 's and their derived structures.

### Architecture and Modeling Language

The RTAtool is a direct implementation of the RTA formalism. It processes specifications written in Re-Lang (a syntax introduced below), constructing the respective temporal reactive graph. The modeling pipeline follows these main stages:

1. **Parser and Model Generator:** Processes the Re-Lang source code to extract the formal sets $(L, Act, C)$ and the transition relations $(\rightarrow, \twoheadrightarrow, \rightarrowtail)$.
2. **Symbolic Simulator:** Generates an interactive and animated rendering for immediate feedback. This allows the user to manually trigger transitions and visually inspect the effects of reconfigurations.
3. **Verification Engine:** Checks finite $\mathrm{LTL}_f/\mathrm{CTL}_f$ [4,5,6]. properties directly over the constructed model.
4. **Multi-Format Export:** Supports several export options, including Uppaal [3] XML for formal analysis, LaTeX source code (generated via Mermaid) for inclusion in papers, and PNG images.

[1] Available at: `https://lcs.web.ua.pt/projects/RTAtool/`

**Language Syntax (Re-Lang).** The Re-Lang modeling language employs a simple and explicit syntax. A model begins with the definition of its initial state and primary transitions, followed by clocks, variables, and guards. The core keywords are summarized in Table 1.

Table 1: Global declaration syntax in Re-Lang

| Keyword | Example | Description |
|---|---|---|
| `name` | `name Like2` | Defines the identifier for the model. |
| `init` | `init FEED` | Sets the starting location of the system. |
| `int` | `int count = 0` | Declares an integer variable. |
| `clock` | `clock x` | Declares a clock variable (starts at 0). |
| `inv` | `inv FEED:x<=20` | Associates an invariant with a location. |
| `aut` | `aut Motor {...}` | Encapsulates logic into a modular component. |

**Edge and Hyper-edge Structure.** The Re-Lang distinguishes between ground edges (transitions between locations) and hyper-edges (reconfiguration rules acting on other transitions).

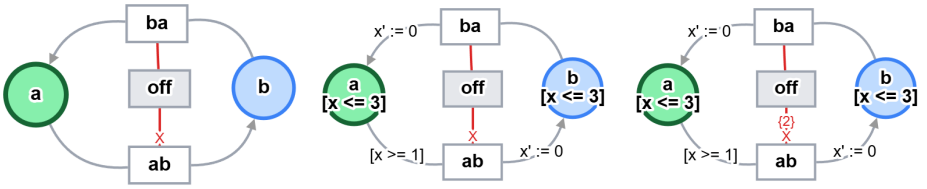
```
source <ARROW> target : label;
```

The behavior of the edge depends on the type of arrow used, which dictates how a transition affects the current configuration of the model: `-act->` denotes a standard discrete ground edge ($\ell \xrightarrow{act} \ell'$), `-»` denotes an activation hyper-edge that enables a target, and `-!` denotes a deactivation hyper-edge that disables it. **Guards, Updates, and Status** In order to represent reconfigurable dynamics, edges can be enriched with the following attributes: **Guards (`if`),** logical conditions over clocks or integers, where the transition can only be enabled if the condition evaluates to true (e.g., `if (x >= 1 AND x <= 9)`), **Updates (`then`),** updates that occur during the transition, using the prime symbol (`'`) to represent the next value of a variable (e.g., `then { x' := 0 }`) and **Initial Configuration,** where by default, the edges are active, but adding the keyword `disabled` at the end of an edge declaration forces it to start in an inactive state, requiring an activation arrow (`-»`) to be used.

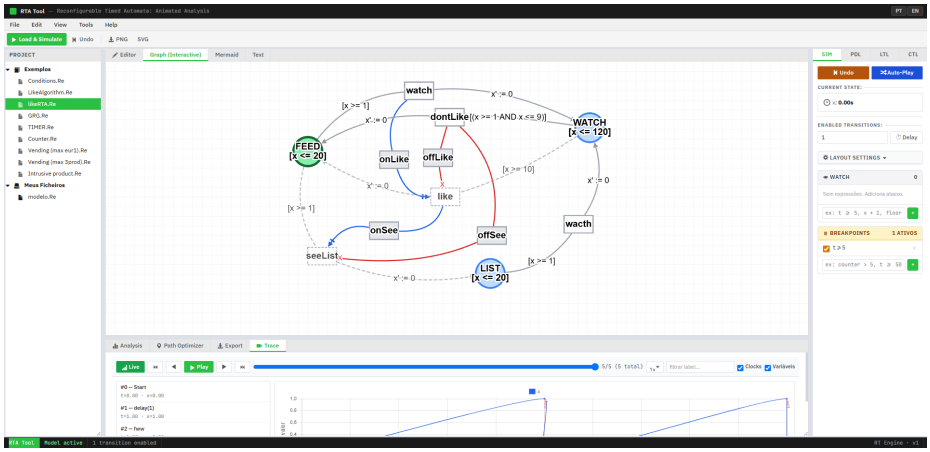
Beyond text-based coding, RTAtool offers an intuitive **Interactive Graphical Modeling** interface. Users can build and configure the entire system through right-click menus directly on the canvas, adding locations, variables, and drawing standard or hyper-edges making the modeling workflow entirely visual.

**Modeling and executing behaviors.**

The user interface (Fig. 1d) features an interactive visualizer that allows users to explore the possible behaviors of the model by executing the available transitions in each configuration. The current location is highlighted in green, while disabled transitions are represented by dotted lines. Executions can be simulated either by clicking on the enabled transitions originating from the current location or by using the simulation menu, where an action can be selected from the Enabled




(a) Simple reactive model (b) Guards and invariants (c) Pending delays




(d) Overview of the RTAtool integrated development environment, simulating the Like model.

Fig. 1: Graphical modeling examples in RTAtool: (a) simple reactive model, (b) timed model featuring guards and invariants, (c) model illustrating pending delays on a hyper-edge, and (d) full overview of the IDE.

Transitions panel. To illustrate the tool's capabilities in this context, we use the **Like** model (see Fig. 1d). The system represents a simplified reconfigurable content feed algorithm: a user starts in a `FEED` location and transitions to watching content. Depending on the time spent (measured by a clock `x`) and the user's reaction (e.g., liking a video via the `wlf` transition), hyper-edges dynamically reconfigure the system by activating or deactivating future possibilities, such as enabling the user to see a curated list.

### Verification and Analysis Engine

Beyond simulation, RTAtool embeds an automated verification engine that checks $\text{LTL}_f$ and $\text{CTL}_f$ properties directly over the constructed model, without requiring an external export to Uppaal. Two complementary evaluation modes are supported, selectable depending on the analysis being performed, namely an **Explicit (trace-based)** evaluation fitted for untimed models, where the tool

builds finite execution paths of the system and evaluates satisfaction structurally over the resulting sequence of configurations and also a **Symbolic (zone-based)** evaluation, suitable for timed models, where continuous clock valuations are represented symbolically as *timed zones* [2].

**Temporal Properties Syntax (LTL / CTL).** The verification engine parses temporal queries using a straightforward syntax that accommodates both Linear Temporal Logic ($\text{LTL}_f$) and Computation Tree Logic ($\text{CTL}_f$). A key feature of the query language is the clear distinction between structural atoms (evaluating the active status of edges or hyper-edges) and quantitative propositions (evaluating variables or clocks). The core operators are summarized in Table 2.

Table 2: Temporal logic syntax supported by the RTAtool verification engine.

| Category | Syntax Example | Description |
|---|---|---|
| **State Prop.** | `wlf` or `onLike` | Evaluates true if ground edge `wlf` or hyper-edge `onLike` is active. |
| **Data Prop.** | `[x >= 10]` | Enclosed in brackets. Evaluates conditions over variables or clocks. |
| **Boolean** | `!`, `&&`, `\|\|`, `->` | Standard propositional operators (NOT, AND, OR, IMPLIES). |
| **LTL/CTL** | `X`, `F`, `G`, `U`, `EX` | Standard linear and branching-time temporal operators. |

**Trace Generation and Statistical Sampling:** For explicit evaluation, the tool implements a randomized trace generator. Instead of constructing the entire exponential state space, the engine dynamically resolves the operational semantics at each step evaluating active hyper-edges, updates, and guards to compute enabled transitions. It selects a valid transition at random to produce a linear execution path. This enables highly efficient sample-based verification (e.g., batch-testing an $\text{LTL}_f$ property over $N$ random traces), serving as a powerful bug-hunting feature that provides immediate counterexamples.

### Conclusion and Future Work

The RTAtool provides a comprehensive pipeline to model, simulate, and verify Reconfigurable Timed Automata. By abstracting the complexity of structural reconfigurations, it offers a scalable and intuitive alternative to classical TA environments. Future work includes extending the verification engine to support stochastic behaviors and distributed synthesis. A comprehensive guide detailing the complete Re-Lang syntax, along with further examples and advanced modeling features, can be found in the tool's online documentation at `https://lcs.web.ua.pt/projects/RTAtool/help.html`.

## References

1. R. Alur and D. L. Dill. The theory of timed automata. In J. W. de Bakker, C. Huizing, W. P. de Roever, and G. Rozenberg, editors, *Real-Time: Theory in*

*Practice, REX Workshop, Mook, The Netherlands, June 3-7, 1991, Proceedings*, volume 600 of *Lecture Notes in Computer Science*, pages 45–73. Springer, 1991.
2. C. Baier and J. Katoen. *Principles of model checking.* MIT Press, 2008.
3. G. Behrmann, A. David, and K. G. Larsen. A tutorial on uppaal. In M. Bernardo and F. Corradini, editors, *Formal Methods for the Design of Real-Time Systems, International School on Formal Methods for the Design of Computer, Communication and Software Systems, SFM-RT 2004, Bertinoro, Italy, September 13-18, 2004, Revised Lectures*, volume 3185 of *Lecture Notes in Computer Science*, pages 200–236. Springer, 2004.
4. G. De Giacomo and M. Y. Vardi. Linear temporal logic and linear dynamic logic on finite traces. 2013.
5. V. Fionda and G. Greco. The complexity of ltl on finite traces: Hard and easy fragments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.
6. A. C. C. Iglesias, J. C. Dourado, A. Madeira, and M. A. Martins. A complete overview in reconfigurability: Semantics, logics and tools. Submitted for publication in Journal, 2026.

# A generalization of twist approach to sublogics

Alfredo Roque Freire

**Abstract.** In this presentation, we explore a strategy to generalize the twist structure technique developed by Manuel Fidel and Andrzej Sendlewski. Departing from the specific instantiation of the twist pair $\langle a, b\rangle$, where $a$ and $b$ are Boolean values representing the affirmative and negative aspects of a formula, respectively, we aim to understand what makes this technique successful in producing subsystems of classical logic. To achieve this, we observe that the twist values are based on the pair of pivot formulas $Val(\varphi) = \langle \varphi, \neg\varphi\rangle$. The operations are then defined by considering the operations over these pivot formulas: $Val(\varphi \vee \psi) = \langle \varphi \vee \psi, \neg(\varphi \vee \psi)\rangle$. The formulas in the pair are equivalent to $\langle \varphi \vee \psi, \neg\varphi \wedge \neg\psi\rangle$ in classical logic, connecting the values of $Val(\varphi \vee \psi)$ to the values in the pairs of $Val(\varphi)$ and $Val(\psi)$. We will explore the general behaviors of this kind of construction, considering generally the pivot pairs $Val(\varphi) = \langle G_1(\varphi), G_2(\varphi), \ldots, G_n(\varphi)\rangle$. We prove that all logics obtained in this way are subsystems of the original logic in a specific manner, and we will explore the extent of subsystems that can be obtained using this strategy.

# Reconfigurable Automata Meets Machine Learning: Modeling and Analizing Real Transit Scenarios

Antonio César Castro Iglesias[1][0000−0002−8531−6535], Joshua Dourado[2][0009−0005−5360−6852], and Mariana Ferreira[1][0009−0003−1519−7977]

[1] CIDMA - Research Center in Mathematics and Applications, Mathematics Department of University of Aveiro, Portugal
[2] University of Aveiro, Portugal

## 1 Introduction and objectives

Public transportation systems are inherently exposed to temporal variability arising from traffic conditions, passenger demand, and operational disturbances. Even small perturbations in travel time can propagate, affecting vehicle timings, arrivals and scheduling, degrading service reliability. In high-frequency bus services, where consecutive vehicles are strongly coupled through their headways, this dynamic often triggers the bus bunching phenomenon, also known as *accordion effect*: delayed buses accumulate more passengers at stops, extend dwell times and amplify delays, while following vehicles encounter fewer passengers and accelerate to catch up [6,12]. To model this dynamic behavior, reconfigurable (or reactive) structures, whose accessibility relations change dynamically during evaluation [7,5], and timed automata, which incorporate continuous temporal variables [1], emerge as ideal theoretical formalisms. Combining these frameworks yields Reconfigurable Timed Automata (ReTA), capable of capturing both dynamic structural adaptations and continuous temporal dynamics. To this end, this paper pursues **two main objective**s: first, to demonstrate the utility of Reconfigurable Timed Automata (ReTA) by showing how reconfigurable structures equipped with temporal dynamics can be effectively encoded and translated into systems interpretable by traditional model checkers by using the dedicated tool RTA-Tool (`https://github.com/LCS-Aveiro/RTATOOL`); second, to model and implement a transit system based on real-world data, perform statistical analysis, and apply ML algorithms to simulation-generated data from the reconfigurable timed model to predict transit fleet behaviors, including the accordion effect.

## 2 Theoretical Background

This section introduces ReTA [9], a natural extension of reconfigurable automata [13] that are additionally equipped with the characteristic features of timed automata, such as timed guards and invariants. These structures are later translated to timed automata with boolean guards, a type of automata that naturally

translates the reconfigurable elements of the ReTA into boolean variables, codifying the same behavior as the aforementioned [9].

On the other hand, the ML analysis is applied to simulation data generated by the proposed ReTA. The resulting dataset describes the temporal evolution of delays and accumulated delay, providing a controlled environment for identifying patterns associated with the transition towards the *accordion effect*.

### 2.1 Reconfigurable Timed Automata (ReTA)

In this section, we introduce the formal definition of ReTA, together with a simple illustrative example. Prior to the formal definition, we establish the notation used to reference the set of ground edges and hyper-edges within the system. We define such components in a recursive way, where: $HE = E \times E$, $HE_n = E \times HE_{n-1},\ n > 1$ and we set $H(E) = \cup_n HE_n$. Hyper-edges are a type of edge structure that has reconfigurable effects over other edges and hyper-edges of the automata. These can be deactivation hyper-edges, which means that, when their associated ground edge is traversed they turn their targeted edge (or hyper-edge) inactive, meaning it can not be traversed, or in the case of the hyper-edge, has no effects. In the same way, activation hyper-edges turn those inactive into active edges (or hyper-edges). Given a valuation function $\gamma$ of $Y$ into an ordered set $D$, we define the satisfaction relation $\gamma \models g$ as usual. As in timed automata, given a set of clock variables $C$, $C(C)$ is called the set of clock constraints that are interpreted in valuations $\eta : C \to \mathbb{R}_{\geq 0}$ (cf. [2]).

**Definition 1.** *A* Reconfigurable Timed Automata *for a set of nominal variables, Nom, is a structure*

$$RT = (L, CAct, T, \mathcal{N}, Inv, \alpha_0, \ell_0)$$

*where:*

- $L$ *is a* finite set of locations*;*
- $C$ *is a* finite set of clocks*;*
- $Act$ *is a* finite the set of actions*;*
- $T_t = (E, E_{\twoheadrightarrow}, E_{\rightarrowtail\!\!\!\!\!\times})$ *is an* edges structure *where:*
  - $E \subseteq L \times C(C) \times Act \times 2^C \times L$ *is the* set of ground edges.
  - $E_{\twoheadrightarrow} \subseteq H(E)$ *is the* set of activating hyper-edges.
  - $E_{\rightarrowtail\!\!\!\!\!\times} \subseteq H(E)$ *is the* set of deactivating hyper-edges.

  *and we define* $\hat{T}_t = E \cup E_{\twoheadrightarrow} \cup E_{\rightarrowtail\!\!\!\!\!\times}$ *where* $H(E) = E_{\twoheadrightarrow} \uplus E_{\rightarrowtail\!\!\!\!\!\times}$*;*
- $\mathcal{N} : Nom \to \hat{T}_t$ *is a* nominal assignment function *that maps a nominal identifier to each edge and hyper-edge;*
- $Inv : L \to C(C)$ *is an* invariant assignment function to locations*;*
- $\alpha_0 \subseteq E \cup E_{\twoheadrightarrow} \cup E_{\rightarrowtail\!\!\!\!\!\times}$ *are the* initial active edges *of the system;*
- $l_0 \in L$ *is the* initial location *of the system.*

ReTA can also be codified into timed automata with boolean variables acting as guards [9]. Such structure mapping allows for its direct implementation into well know solvers such as Uppaal. Fig.4 portrays such encoding in a simple way. The algorithmic process of codification is based in transforming the reconfigurable effects of the hyper-edges into boolean guards that get updated after every transition where the original ReTA model had associated hyper-edges. For further information related with semantics, logics or encodings on these and other related structures, see [9].

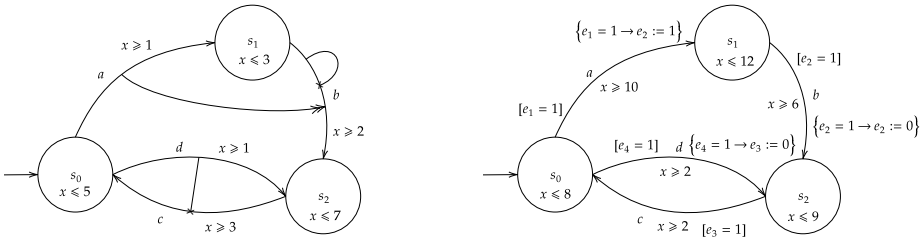



Fig. 1: ReTA and timed automata with boolean guards systems example

### 2.2 Machine Learning for Classification

Machine Learning (ML) encompasses computational methods that learn patterns from data to make predictions on unseen observations [8]. In this work, supervised learning is applied, as the objective is to predict the occurrence of the *accordion effect* from observed system variables. The task is framed as a binary classification problem, where the target classes denote the absence or presence of the *accordion effect*. To capture different modeling dynamics, three supervised techniques are evaluated: Logistic Regression, Random Forest, and Support Vector Machine (SVM). Logistic Regression is a statistical learning method for binary classification that models class probabilities via a linear combination of input features [8,11], serving as an interpretable baseline to analyze the influence of temporal and delay-related variables on the emergence of the *accordion effect*. Random Forest is an ensemble method that constructs multiple decision trees and aggregates their predictions [3]; by introducing randomness in feature selection and tree building, it effectively captures non-linear relationships and variable interactions as a non-linear alternative to Logistic Regression. Finally, Support Vector Machine (SVM) is a supervised method that constructs optimal decision boundaries to separate classes [4], utilizing kernel functions to project data into higher-dimensional spaces and resolve non-linear separation problems as a complementary classification framework. For comprehensive theoretical foundations and mathematical formulations of these methods, the reader is referred to [8] for general ML concepts, [3] for Random Forest, and [4] for Support Vector Machines.

## 3 Case Study

This case study analyzes bus travel times within a German transit network using real-world operational data to examine delay propagation and the resulting *accordion effect*—a sequential delay accumulation leading to critical service latency. The system is modeled in two phases: first as a ReTA automaton, and subsequently exported to Uppaal as a timed automaton with boolean guards. A statistical analysis evaluates system behavior, followed by ML algorithms trained on simulation data to predict transit fleet dynamics. Additionally, the reconfigurable structure resolves a key challenge in timed automata by enabling dynamic clocks that adapt location invariants and guards based on internal behavior and route-wide accumulated delays.

### Model Building and Behavior

The model represents a network connecting a central German bus terminal to an airport via interconnecting routes. Travel times were obtained directly from a real world dataset: location invariants represent maximum recorded travel times between adjacent stops, while transition guards use minimum recorded times. Reconfigurability captures the *accordion effect* across two operational modes: (1)**Baseline Mode**, where buses run standard routes within an acceptable delay margin (travel-time deviation from the mean plus a 10-minute tolerance buffer); and (2)**Reconfigured Accordion Mode**, triggered when consecutive buses suffer sequential delays. This mode opens an alternative route (Fig.2) where the accumulated delay ($\zeta$) dynamically updates location invariants and guards. The system remains in this state until operating fast enough to recover within acceptable thresholds, at which point it returns to baseline mode. Merging both systems in Fig.2, created using the RTA-Tool, yields the final ReTA model (Fig.9), which is later translated with the functionalities of the tool to Uppaal in Fig.10.

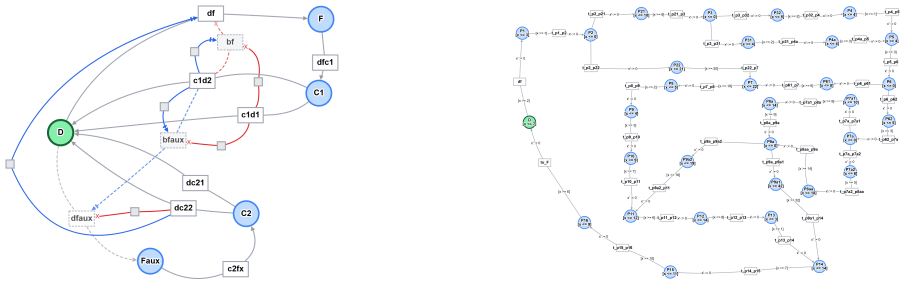



Fig. 2: Bus Transport System Components.

### System Analysis

Preliminary reachability and liveness (meaning that every stop is reached infinitely often) verification confirmed that all system locations are eventually reachable. Liveness was validated for state $F$ but disproved for $Faux$, indicating the system can cycle without triggering the *accordion effect*. Posterior probabilistic analysis using Uppaal-SMC established mean transit times over a 1.440 minutes (24 houra) window, computing expected bounds for maximum and minimum fleet delays ($\zeta$) alongside travel durations (Figs. 8b, 8a, and 8c). Model fidelity was confirmed against the available real world data: Uppaal produced a mean transit time of $105 \pm 2.83$ min vs. $107 \pm 3.27$ min in empirical data, and a peak $\zeta$ of $12.98 \pm 0.77$ min vs. 10.98 min empirically.

***Accordion Effect* Analysis** Evaluation followed a two-fold simulation scheme: long-term runs ($244,000$ min) to observe oscillatory behavior and generate ML datasets (Fig. 3), and 1,440-minute runs to query mode transitions and recovery capacity under varying $\zeta$:

- `simulate [<=1444; 10000] { Process.F } : Process.Faux` $\longrightarrow 0.68 \pm 0.009$: across 10,000 runs, the system returned from accordion mode (`Process.Faux`) to baseline (`Process.F`) 68% of the time within 1,440 minutes.
- `simulate [<=1444; 10000] { Process.F } : { Process.Faux && zeta>13 }` $\longrightarrow 0.37 \pm 0.009$: recovery probability dropped to 37% when $\zeta > 13$.
- `Pr[<=1440] (zeta>0 U zeta<=0)` $\longrightarrow 0.69 \pm 0.04$: evaluated system recovery using the until operator.

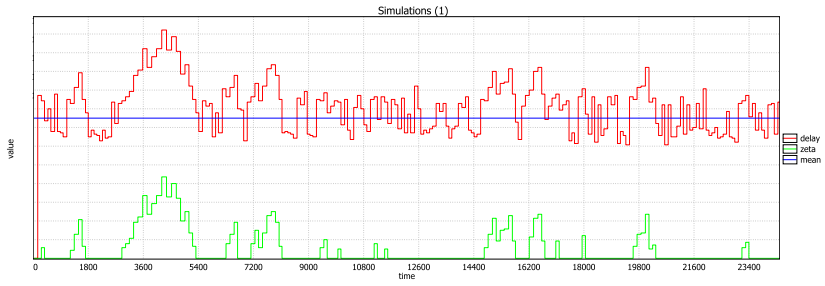



Fig. 3: Simulations of *zeta*, *mean*, and *delay* Values ($24,440$-Minute Window).

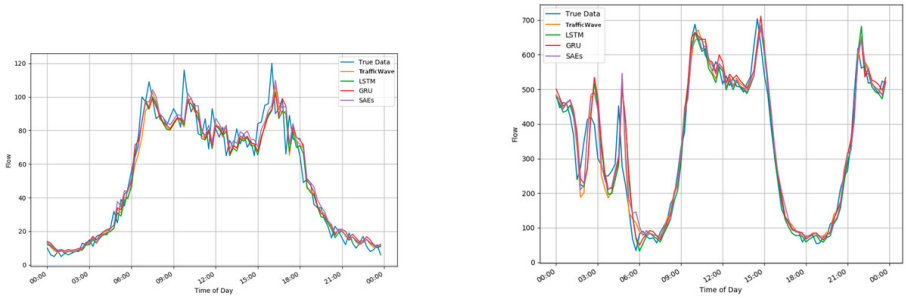



Fig. 4: *Accordion Effect* of Traffic Flows Over 24 Hours [10].

### Machine Learning Implementation

The ML component evaluates whether simulation-generated temporal features suffice to anticipate the *accordion effect*. The task is formulated as a one-step-ahead binary classification problem: predicting the presence of the *accordion effect* at step $t+1$ using timestamp, delay, and accumulated delay $\zeta$ observed at step $t$. This setup preserves temporal ordering without leakage.

Three supervised models, Logistic Regression, Random Forest, and SVM were tuned via cross-validation (hyperparameters in Table1). Feature normalization and class weighting were integrated into cross-validation pipelines to address class imbalance without data leakage.

Predictive performance was evaluated using the ROC curve analysis (Fig.5) and classification metrics (Table 2). All models outperformed random guessing: Logistic Regression achieved the highest AUC (0.937), closely followed by Random Forest (0.935), whereas SVM lagged at 0.862. Although Random Forest yielded slightly higher recall, Logistic Regression was selected as the optimal model due to its overall class balance, superior AUC, higher F1-score, and inherent interpretability.

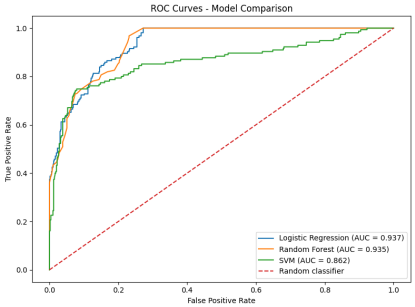
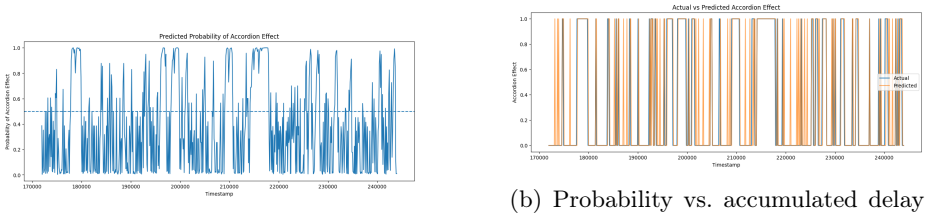



Fig. 5: ROC curves for LR, RF, and SVM.

As illustrated in Fig. 6, predicted probability increases with both delay and accumulated delay $\zeta$, with instantaneous delay having the strongest influence. These findings align with Logistic Regression coefficient analysis and validate the formal model: higher accumulated delay $\zeta$ reduces system recovery capacity and increases the likelihood of remaining trapped in reconfigured accordion mode.

## 4 Conclusion and Future Work

This work presented a methodology combining Reconfigurable Timed Automata, statistical model checking, and ML for the analysis and prediction of delay prop-




(a) Probability vs. delay.

(b) Probability vs. accumulated delay $\zeta$.

Fig. 6: Predicted probabilities of the *accordion effect* across delay indicators.

agation in public transit systems. The formal analysis positively reproduced the *accordion effect*, while ML enabled one-step-ahead prediction of this phenomenon. The results highlight the role of accumulated delay in system recovery and prediction, consistently reproducing the behavior observed in the real data. This work favorably bridged the gap between formal methods and ML, proving the complementarity of both fields of knowledge.

Future work will consider additional operational variables, longer prediction horizons and the integration of ML methods into the own structure of the reconfigurable control mechanism while preserving the structural integrity of the reconfigurable model.

**Acknowledgments.** This work was developed in the context of the scientific activities of CIDMA (https://ror.org/05pm2mw36), under the Portuguese Foundation for Science and Technology (FCT, https://ror.org/00snfqn58), Grants UID/04106/2025 (https://doi.org/10.54499/UID/04106/2025) and UID/PRR/04106/2025 (https://doi.org/10.54499/UID/PRR/04106/2025) and by Project Banksy with reference COMPETE2030-FEDER-00892000.

## References

1. R. Alur and D. L. Dill. The theory of timed automata. In J. W. de Bakker, C. Huizing, W. P. de Roever, and G. Rozenberg, editors, *Real-Time: Theory in Practice, REX Workshop, Mook, The Netherlands, June 3-7, 1991, Proceedings*, volume 600 of *Lecture Notes in Computer Science*, pages 45–73. Springer, 1991.
2. C. Baier and J. Katoen. *Principles of model checking*. MIT Press, 2008.
3. L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
4. C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.
5. M. Crochemore and D. M. Gabbay. Reactive automata. *Information and Computation*, 209(4):692–704, 2011.
6. C. F. Daganzo. A headway-based approach to eliminate bus bunching: Systematic analysis and comparisons. *Transportation Research Part B: Methodological*, 43(10):913–921, 2009.
7. D. M. Gabbay. Reactive kripke semantics and arc accessibility. 2007.
8. T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, New York, NY, USA, 2009.

9. A. C. C. Iglesias, J. C. Dourado, A. Madeira, and M. A. Martins. A complete overview in reconfigurability: Semantics, logics and tools. Submitted for publication in Journal, 2026.
10. D. Impedovo, V. Dentamaro, G. Pirlo, and L. Sarcinella. Trafficwave: Generative deep learning architecture for vehicular traffic flow prediction. *Applied Sciences*, 9(24), 2019.
11. G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning: with Applications in R*. Springer, New York, NY, USA, 2013.
12. G. F. Newell and R. B. Potts. Maintaining a bus schedule. *Proceedings of the 2nd Australian Road Research Board Conference*, 2:388–393, 1964.
13. D. Tinoco, A. Madeira, M. A. Martins, and J. Proença. Reactive graphs in action. In D. Marmsoler and M. Sun, editors, *Formal Aspects of Component Software - 20th International Conference, FACS*, volume 15189 of *Lecture Notes in Computer Science*, pages 97–105. Springer, 2024.

## A Supplementary Plots

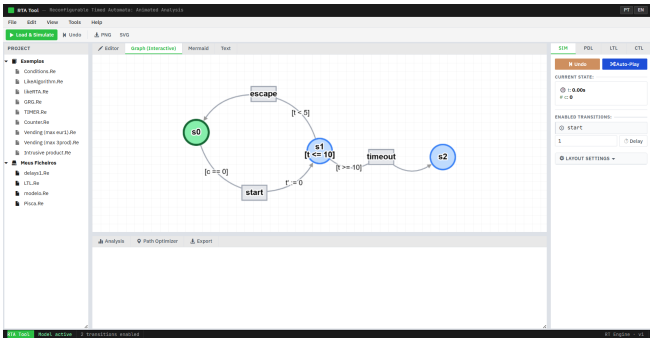
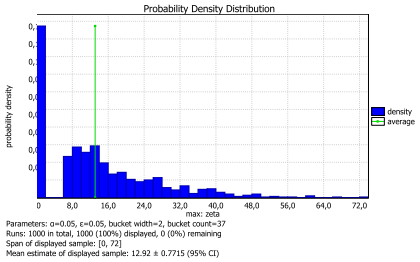
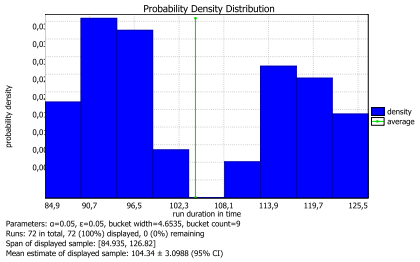


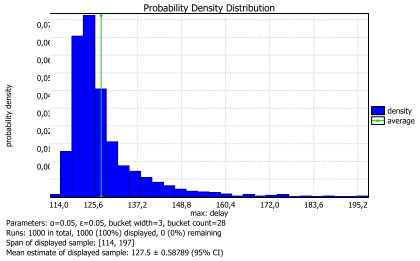

Fig. 7: RTAtool environment.



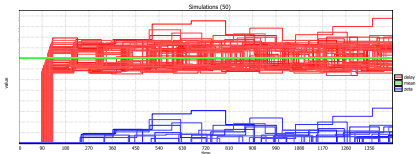

(a) Distribution of Delays in *Accordion Effect* Mode




(b) Distribution of Arrival Times at Airport in Baseline Mode




(c) Distribution of Arrival Times at Airport in Baseline and Accordion Mode




(d) Simulations of *Zeta*, *mean* and *delay* Values in a 1440 Minutes Time Span

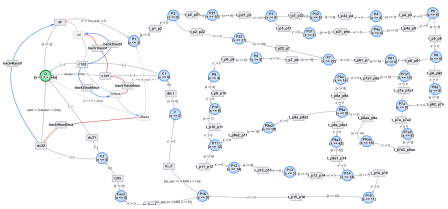


Fig. 9: Reconfigurable Bus Transport System

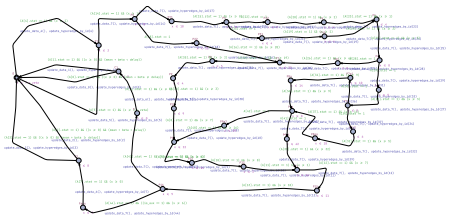
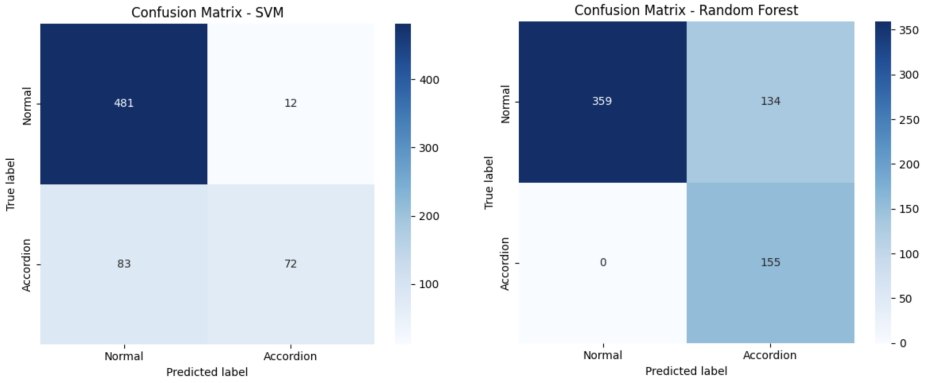


Fig. 10: Reconfigurable Bus Transport System Translated to Uppaal




(a) Logistic Regression (b) Random Forest

Fig. 11: Confusion matrices for the Logistic Regression and Random Forest models.

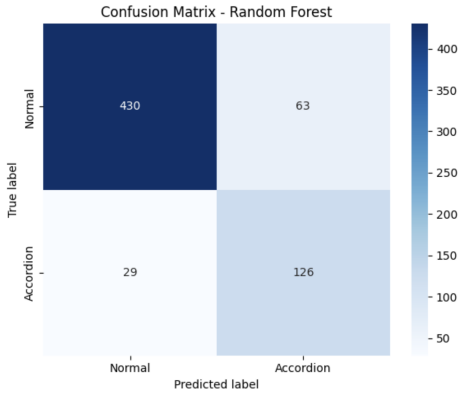



Fig. 12: Confusion matrix for the SVM model.

Table 1: Final hyperparameter configurations for the considered machine learning models.

| **Model** | **Hyperparameters** |
|---|---|
| Logistic Regression | $C = 0.1$, `penalty=l1`, `solver=liblinear` `class_weight=balanced`, `max_iter=2000` |
| Random Forest | `n_estimators=200`, `max_depth=5` `min_samples_leaf=2`, `min_samples_split=10` `class_weight=balanced` |
| SVM | $C = 1000$, $\gamma = 0.01$, `probability=True` |

Table 2: Classification performance of the considered machine learning models.

| **Model** | **Accuracy** | **Balanced Accuracy** | **Precision** | **Recall** | **F1-score** |
|---|---|---|---|---|---|
| Logistic Regression | 0.8580 | 0.8426 | 0.6667 | 0.8129 | 0.7326 |
| Random Forest | 0.7932 | 0.8641 | 0.5363 | 1.0000 | 0.6982 |
| SVM | 0.8534 | 0.7201 | 0.8571 | 0.4645 | 0.6025 |