Non-expansive Fuzzy Coalgebraic Logic

Stefan Gebhart, Lutz Schröder, Paul Wild

Chair of Theoretical Computer Science
Department of Computer Science
Friedrich-Alexander-Universität Erlangen-Nürnberg

July 9, 2025

Motivation

Motivation

- **Why Coalgebraic Logics?**

Motivation

- **Why Coalgebraic Logics?**
  - Provide a uniform, abstract framework for modeling and reasoning about state-based systems (e.g., automata, transition systems, probabilistic systems).

Motivation

- **Why Coalgebraic Logics?**
  - Provide a uniform, abstract framework for modeling and reasoning about state-based systems (e.g., automata, transition systems, probabilistic systems).
  - Use category-theoretic tools to generalize a wide range of modal and temporal logics.

Motivation

- **Why Coalgebraic Logics?**
  - Provide a uniform, abstract framework for modeling and reasoning about state-based systems (e.g., automata, transition systems, probabilistic systems).
  - Use category-theoretic tools to generalize a wide range of modal and temporal logics.
- **Why Fuzziness?**

Motivation

- **Why Coalgebraic Logics?**
  - Provide a uniform, abstract framework for modeling and reasoning about state-based systems (e.g., automata, transition systems, probabilistic systems).
  - Use category-theoretic tools to generalize a wide range of modal and temporal logics.
- **Why Fuzziness?**
  - Many real-world systems involve uncertainty, vagueness, or degrees of truth.

Motivation

- **Why Coalgebraic Logics?**
  - Provide a uniform, abstract framework for modeling and reasoning about state-based systems (e.g., automata, transition systems, probabilistic systems).
  - Use category-theoretic tools to generalize a wide range of modal and temporal logics.
- **Why Fuzziness?**
  - Many real-world systems involve uncertainty, vagueness, or degrees of truth.
  - Classical (crisp) logics are inadequate for modeling partial or approximate information.

Fuzzy Bases and Computational Hardness

Fuzzy Bases and Computational Hardness

- **Łukasiewicz logic:**

Fuzzy Bases and Computational Hardness

- **Łukasiewicz logic:**
  - Rich expressivity and strong logical properties.

Fuzzy Bases and Computational Hardness

- **Łukasiewicz logic:**
  - Rich expressivity and strong logical properties.
  - But **computationally hard**.

Fuzzy Bases and Computational Hardness

- **Łukasiewicz logic:**
  - Rich expressivity and strong logical properties.
  - But **computationally hard**.
- **Zadeh logic:**

Fuzzy Bases and Computational Hardness

- **Łukasiewicz logic:**
  - Rich expressivity and strong logical properties.
  - But **computationally hard**.
- **Zadeh logic:**
  - Simpler semantics: truth values interpreted via $\min/\max$.

Fuzzy Bases and Computational Hardness

- **Łukasiewicz logic:**
  - Rich expressivity and strong logical properties.
  - But **computationally hard**.
- **Zadeh logic:**
  - Simpler semantics: truth values interpreted via $\min/\max$.
  - **Efficient reasoning**, but entails little to no deviation from classical logic.

Fuzzy Bases and Computational Hardness

- **Łukasiewicz logic:**
  - Rich expressivity and strong logical properties.
  - But **computationally hard**.
- **Zadeh logic:**
  - Simpler semantics: truth values interpreted via $\min/\max$.
  - **Efficient reasoning**, but entails little to no deviation from classical logic.
- $\Rightarrow$ There is a trade-off between **expressivity** and **tractability**.

Non-Expansivity: A Computationally Friendly Middle Ground

Non-Expansivity: A Computationally Friendly Middle Ground

- **Non-expansivity** restricts operators and modalities to be 1-Lipschitz:

$$|f(x) - f(y)| \leq d(x, y)$$

Non-Expansivity: A Computationally Friendly Middle Ground

- **Non-expansivity** restricts operators and modalities to be 1-Lipschitz:
$$|f(x) - f(y)| \leq d(x, y)$$
- This captures a class of **computationally well-behaved** fuzzy logics.

Non-Expansivity: A Computationally Friendly Middle Ground

- **Non-expansivity** restricts operators and modalities to be 1-Lipschitz:
$$|f(x) - f(y)| \leq d(x, y)$$
- This captures a class of **computationally well-behaved** fuzzy logics.
- Non-expansive semantics often allow for **efficient model checking and reasoning.**

Non-Expansivity: A Computationally Friendly Middle Ground

- **Non-expansivity** restricts operators and modalities to be 1-Lipschitz:
$$|f(x) - f(y)| \leq d(x, y)$$
- This captures a class of **computationally well-behaved** fuzzy logics.
- Non-expansive semantics often allow for **efficient model checking and reasoning.**
- $\Rightarrow$ **Non-expansive fuzzy coalgebraic logic** offers a principled bridge:

Non-Expansivity: A Computationally Friendly Middle Ground

- **Non-expansivity** restricts operators and modalities to be 1-Lipschitz:
$$|f(x) - f(y)| \leq d(x, y)$$
- This captures a class of **computationally well-behaved** fuzzy logics.
- Non-expansive semantics often allow for **efficient model checking and reasoning.**
- $\Rightarrow$ **Non-expansive fuzzy coalgebraic logic** offers a principled bridge:
  - Retains useful structure from Łukasiewicz.

Non-Expansivity: A Computationally Friendly Middle Ground

- **Non-expansivity** restricts operators and modalities to be 1-Lipschitz:
$$|f(x) - f(y)| \leq d(x, y)$$
- This captures a class of **computationally well-behaved** fuzzy logics.
- Non-expansive semantics often allow for **efficient model checking and reasoning.**
- $\Rightarrow$ **Non-expansive fuzzy coalgebraic logic** offers a principled bridge:
  - Retains useful structure from Łukasiewicz.
  - Avoids worst-case complexity; closer to Zadeh in tractability.

Non-expansive Fuzzy Coalgebraic Logic

Non-expansive Fuzzy Coalgebraic Logic

- Formulas over signature A, Λ are given by:

$$\phi, \psi ::= 0 \mid p \mid \neg\phi \mid \phi \ominus c \mid \phi \sqcap \psi \mid \heartsuit\phi$$

with $p \in A$, $c \in [0,1]$, $\heartsuit \in \Lambda$.

Non-expansive Fuzzy Coalgebraic Logic

- Formulas over signature A, Λ are given by:

$$\phi, \psi ::= 0 \mid p \mid \neg\phi \mid \phi \ominus c \mid \phi \sqcap \psi \mid \heartsuit\phi$$

  with $p \in A$, $c \in [0, 1]$, $\heartsuit \in \Lambda$.
- A *predicate lifting* of $\heartsuit \in \Lambda$ given $T : \mathsf{Set} \to \mathsf{Set}$ is a natural transformation
  $[\![\heartsuit]\!] : \mathrm{Hom}_{\mathsf{Set}}(-, [0, 1]) \Rightarrow \mathrm{Hom}_{\mathsf{Set}}(T^{\mathrm{op}}(-), [0, 1])$.

Non-expansive Fuzzy Coalgebraic Logic

- Formulas over signature A, Λ are given by:

$$\phi, \psi ::= 0 \mid p \mid \neg\phi \mid \phi \ominus c \mid \phi \sqcap \psi \mid \heartsuit\phi$$

  with $p \in A$, $c \in [0, 1]$, $\heartsuit \in \Lambda$.
- A *predicate lifting* of $\heartsuit \in \Lambda$ given $T : \mathsf{Set} \to \mathsf{Set}$ is a natural transformation
  $[\![\heartsuit]\!] : \mathrm{Hom}_{\mathsf{Set}}(-, [0, 1]) \Rightarrow \mathrm{Hom}_{\mathsf{Set}}(T^{\mathrm{op}}(-), [0, 1])$.
- A *T-model* is a coalgebra $M = (X \in \mathsf{Set}, \xi : X \to TX)$.

Non-expansive Fuzzy Coalgebraic Logic

- The *extension* $[\![\phi]\!]_M : X \to [0,1]$ for a formula is given by:

$$[\![0]\!]_M = 0 \qquad [\![\neg\phi]\!]_M = 1 - [\![\phi]\!]_M$$

$$[\![\phi \ominus c]\!]_M = [\![\phi]\!]_M \ominus c \qquad [\![\phi \sqcap \psi]\!]_M = \min([\![\phi]\!]_M, [\![\psi]\!]_M)$$

$$[\![\heartsuit\phi]\!]_M = [\![\heartsuit]\!]_X([\![\phi]\!]_M) \circ \xi$$

Examples

Examples

- Fix $T = \mathcal{D}$ as the distribution functor.

Examples

- Fix $T = \mathcal{D}$ as the distribution functor.
- Given piecewise linear monotonic $h : [0,1] \to [0,1]$ the logic non-expansive fuzzy $\mathcal{L}^h_{\text{gen}}$ is defined by: $\Lambda = \{\boldsymbol{G}\}$ with

$$([\![\boldsymbol{G}]\!]_X(\nu))\mu := \sup_{\alpha \in [0,1]} \{\min(\alpha, h(\mu(\{x \in X \mid \nu(x) \geq \alpha\})))\}$$

Examples

- Fix $T = \mathcal{D}$ as the distribution functor.
- Given piecewise linear monotonic $h : [0,1] \to [0,1]$ the logic non-expansive fuzzy $\mathcal{L}^h_{\text{gen}}$ is defined by: $\Lambda = \{\boldsymbol{G}\}$ with

$$([\![\boldsymbol{G}]\!]_X(\nu))\mu := \sup_{\alpha \in [0,1]} \{\min(\alpha, h(\mu(\{x \in X \mid \nu(x) \geq \alpha\})))\}$$

- For $h = \text{id}$ write non-expansive fuzzy $\mathcal{L}_{\text{gen}}$.

Examples

- Fix $T = \mathcal{D}$ as the distribution functor.
- Given piecewise linear monotonic $h : [0,1] \to [0,1]$ the logic non-expansive fuzzy $\mathcal{L}^h_{\text{gen}}$ is defined by: $\Lambda = \{\boldsymbol{G}\}$ with

$$([\![\boldsymbol{G}]\!]_X(\nu))\mu := \sup_{\alpha \in [0,1]} \{\min(\alpha, h(\mu(\{x \in X \mid \nu(x) \geq \alpha\})))\}$$

- For $h = \text{id}$ write non-expansive fuzzy $\mathcal{L}_{\text{gen}}$.
- Define non-expansive quantitative fuzzy $\mathcal{ALC}$ by: $\Lambda = \{\mathrm{M}_p \mid p \in [0,1]\}$ with

$$([\![\mathrm{M}_p]\!]_X(\nu))\mu := \sup\{\alpha \mid \sum_{x \in X, \nu(x) \geq \alpha} \mu(x) > p\}$$

Labelled Interval Systems

Labelled Interval Systems

- A *labelled interval system* (LIS) over a set L is a function $\mathscr{I} : L \to Z$, where Z is the set of all intervals in $[0, 1]$ (including the empty interval).

Labelled Interval Systems

- A *labelled interval system* (LIS) over a set L is a function $\mathscr{I} : L \to Z$, where Z is the set of all intervals in $[0, 1]$ (including the empty interval).
- $\mathscr{J}$ is a *sub-LIS* of $\mathscr{I}$ if $\mathbb{D}(\mathscr{J}) = \mathbb{D}(\mathscr{I})$ and for all $l \in \mathbb{D}(\mathscr{I})$ we have $\mathscr{J}(l) \subseteq \mathscr{I}(l)$.

Labelled Interval Systems

- A *labelled interval system* (LIS) over a set L is a function $\mathscr{I} : L \to Z$, where Z is the set of all intervals in $[0, 1]$ (including the empty interval).
- $\mathscr{J}$ is a *sub-LIS* of $\mathscr{I}$ if $\mathbb{D}(\mathscr{J}) = \mathbb{D}(\mathscr{I})$ and for all $I \in \mathbb{D}(\mathscr{I})$ we have $\mathscr{J}(I) \subseteq \mathscr{I}(I)$.
- Can write $\mathscr{I}$ as a set of assertions of the form $\phi \in I$ with $\phi \in L$, $\mathscr{I}(\phi) = I$.

Labelled Interval Systems

- A *labelled interval system* (LIS) over a set L is a function $\mathscr{I} : L \to Z$, where Z is the set of all intervals in $[0, 1]$ (including the empty interval).
- $\mathscr{J}$ is a *sub-LIS* of $\mathscr{I}$ if $\mathbb{D}(\mathscr{J}) = \mathbb{D}(\mathscr{I})$ and for all $I \in \mathbb{D}(\mathscr{I})$ we have $\mathscr{J}(I) \subseteq \mathscr{I}(I)$.
- Can write $\mathscr{I}$ as a set of assertions of the form $\phi \in I$ with $\phi \in L$, $\mathscr{I}(\phi) = I$.
- LIS $\mathscr{I}$ over formulas L is *satisfied* by state x in model M if for every $\phi \in L$ we have $[\![\phi]\!]_M(x) \in \mathscr{I}(\phi)$ and we write $M, x \models \mathscr{I}$.

One-step logics

One-step logics

- For a set V write $\Lambda(V) := \{\heartsuit v \mid v \in V, \heartsuit \in \Lambda\}$.

One-step logics

- For a set V write $\Lambda(V) := \{\heartsuit v \mid v \in V, \heartsuit \in \Lambda\}$.
- Define *one-step formulas* $\mathrm{Prop}(\Lambda(V))$ over Λ by:

$$\phi, \psi ::= 0 \mid \neg\phi \mid \phi \ominus c \mid \phi \sqcap \psi \mid \heartsuit v$$

One-step logics

- For a set V write $\Lambda(V) := \{\heartsuit v \mid v \in V, \heartsuit \in \Lambda\}$.
- Define *one-step formulas* $\mathrm{Prop}(\Lambda(V))$ over Λ by:

$$\phi, \psi ::= 0 \mid \neg\phi \mid \phi \ominus c \mid \phi \sqcap \psi \mid \heartsuit v$$

- Define *T-one-step model* as tuple $M = (X, \tau, t)$ with $X \in \mathsf{Set}$, $t \in TX$ and $\tau : V \to (X \to [0, 1])$.

One-step logics

- Define extension by:

$$[\![0]\!]_M = 0 \qquad [\![\neg\phi]\!]_M = 1 - [\![\phi]\!]_M$$

$$[\![\phi \ominus c]\!]_M = [\![\phi]\!]_M \ominus c \qquad [\![\phi \sqcap \psi]\!]_M = \min([\![\phi]\!]_M, [\![\psi]\!]_M)$$

$$[\![\heartsuit v]\!]_M = [\![\heartsuit]\!]_X(\tau(v))(t)$$

One-step logics

- Define extension by:

$$[\![0]\!]_M = 0 \qquad [\![\neg\phi]\!]_M = 1 - [\![\phi]\!]_M$$

$$[\![\phi \ominus c]\!]_M = [\![\phi]\!]_M \ominus c \qquad [\![\phi \sqcap \psi]\!]_M = \min([\![\phi]\!]_M, [\![\psi]\!]_M)$$

$$[\![\heartsuit v]\!]_M = [\![\heartsuit]\!]_X(\tau(v))(t)$$

- LIS $\mathscr{I}$ over $L \subseteq \mathsf{Prop}(\Lambda(V))$ is *one-step satisfiable* if there exists a T-one-step model M such that we have $[\![l]\!]_M \in \mathscr{I}(l)$ for each $l \in L$.

One-step logics

- Define extension by:

$$[\![0]\!]_M = 0 \qquad [\![\neg\phi]\!]_M = 1 - [\![\phi]\!]_M$$

$$[\![\phi \ominus c]\!]_M = [\![\phi]\!]_M \ominus c \qquad [\![\phi \sqcap \psi]\!]_M = \min([\![\phi]\!]_M, [\![\psi]\!]_M)$$

$$[\![\heartsuit v]\!]_M = [\![\heartsuit]\!]_X(\tau(v))(t)$$

- LIS $\mathscr{I}$ over $L \subseteq \mathsf{Prop}(\Lambda(V))$ is *one-step satisfiable* if there exists a T-one-step model M such that we have $[\![l]\!]_M \in \mathscr{I}(l)$ for each $l \in L$.
- We then write $M \models \mathscr{I}$.

From full logic to one-step logic

From full logic to one-step logic

- A *top-level decomposition* of a LIS $\mathscr{I}$ over formulas L is $\mathscr{I}^\flat : V \to \mathcal{F}(\Lambda)$ and a LIS $\mathscr{I}^\sharp$ over one-step formulas such that each $v \in V$ occurs exactly once in $\mathbb{D}(\mathscr{I}^\sharp)$ and replacing each v by $\mathscr{I}^\flat(v)$ in $\mathscr{I}^\sharp$ gives us back $\mathscr{I}$.

From full logic to one-step logic

- A *top-level decomposition* of a LIS $\mathscr{I}$ over formulas L is $\mathscr{I}^\flat : V \to \mathcal{F}(\Lambda)$ and a LIS $\mathscr{I}^\sharp$ over one-step formulas such that each $v \in V$ occurs exactly once in $\mathbb{D}(\mathscr{I}^\sharp)$ and replacing each v by $\mathscr{I}^\flat(v)$ in $\mathscr{I}^\sharp$ gives us back $\mathscr{I}$.

Lemma

A LIS over formulas $L \subseteq \mathcal{F}(\Lambda)$ is satisfiable in a logic $\mathcal{L}$ iff its top-level decomposition $(V, \mathscr{I}^\flat, \mathscr{I}^\sharp)$ has the following property: $\mathscr{I}^\sharp$ is one-step satisfiable in a one-step model $M = (X, \tau, t)$ where for each $x \in X$ we have a satisfiable LIS $\mathscr{J}_x$ over the image of $\mathscr{I}^\flat$ such that for all $v \in V$ we have $\tau(v)(x) \in \mathscr{J}_x(\mathscr{I}^\flat(v))$.

A Tableau Calculus

Tableau Rules

$$(\mathrm{Ax})\ \frac{S, \phi \in \emptyset}{\bot} \qquad (\mathrm{Ax}\ 0)\ \frac{S, 0 \in \llparenthesis a, b \rrparenthesis}{\bot} \quad (\text{if } 0 \notin \llparenthesis a, b \rrparenthesis)$$

$$(\neg)\ \frac{S, \neg\phi \in \llparenthesis a, b \rrparenthesis, \phi \in I}{S, \phi \in I \cap \rrparenthesis^{-1} 1-b, 1-a \llparenthesis^{-1}}$$

$$(\ominus)\ \frac{S, \phi \ominus c \in \llparenthesis a, b \rrparenthesis, \phi \in I}{S, \phi \in I \cap \llparenthesis a+c, b+c \rrparenthesis} \quad (\text{if } 0 \notin \llparenthesis a, b \rrparenthesis)$$

$$(\ominus')\ \frac{S, \phi \ominus c \in \llparenthesis a, b \rrparenthesis, \phi \in I}{S, \phi \in I \cap [0, b+c \rrparenthesis} \quad (\text{if } 0 \in \llparenthesis a, b \rrparenthesis)$$

$$(\sqcap)\ \frac{S, \phi \sqcap \psi \in \llparenthesis a, b \rrparenthesis, \phi \in I_1, \psi \in I_2}{S, \phi \in I_1 \cap \llparenthesis a, b \rrparenthesis, \psi \in I_2 \cap \llparenthesis a, 1] \qquad S, \phi \in I_1 \cap \llparenthesis a, 1], \psi \in I_2 \cap \llparenthesis a, b \rrparenthesis}$$

A Tableau Calculus

Lemma

LIS $\mathscr{I}$ over one-step formulas L is one-step satisfiable if and only if there exists a tableau graph with leaf with label $Y \neq \bot$ and the LIS $\mathscr{I}^Y$ (over formulas of the form $\heartsuit v$) is one-step satisfiable.

A Tableau Calculus

Lemma

LIS $\mathscr{I}$ over one-step formulas L is one-step satisfiable if and only if there exists a tableau graph with leaf with label $Y \neq \bot$ and the LIS $\mathscr{I}^Y$ (over formulas of the form $\heartsuit v$) is one-step satisfiable.

Lemma

Deciding if LIS $\mathscr{I}$ over one-step formulas L has a tableau graph with leaf with label $Y \neq \bot$ is in NP (with respect to the syntactic size of formulas in L). Furthemore if such a tableau graph exists, the LIS $\mathscr{I}^Y$ can be computed in non-deterministic polynomial time.

Polynomially Space Bounded Logics

Polynomially Space Bounded Logics

- Logic $\mathcal{L}$ is *one-step exponentially bounded* if any LIS $\mathscr{I}$ over one-step formulas L is one-step satisfiable iff it is one-step satisfiable in a one-step model with at most exponentially many states $X_{\mathscr{I}}$ in $|L|$.

Polynomially Space Bounded Logics

- Logic $\mathcal{L}$ is *one-step exponentially bounded* if any LIS $\mathscr{I}$ over one-step formulas L is one-step satisfiable iff it is one-step satisfiable in a one-step model with at most exponentially many states $X_{\mathscr{I}}$ in $|L|$.
- One-step exponentially bounded logic $\mathcal{L}$ is *exponentially branching* if for any LIS $\mathscr{I}$ over one-step formulas L there exists a *satisfying set* $Y_{\mathscr{I}}$ of at most exponentially many LIS over $X_{\mathscr{I}} \times V$ such that for $(X_{\mathscr{I}}, \tau)$ there exists $t \in TX_{\mathscr{I}}$ with $(X_{\mathscr{I}}, \tau, t) \models \mathscr{I}$ if and only if there exists $Q \in Y_{\mathscr{I}}$ with $\tau(v)(x) \in Q(x, v)$ for all $v \in V, x \in X_{\mathscr{I}}$.

Polynomially Space Bounded Logics

- Exponentially branching logic $\mathcal{L}$ *polynomial space bounded* if for any LIS $\mathscr{I}$ over one-step formulas L we have the following properties:
  - Fixing a satisfying set $Y_{\mathscr{I}}$ as $\{Q_1, \ldots, Q_m\}$ and computing some Q_i can be done in polynomial space.
  - Deciding whether a LIS Q over $V \times X_{\mathscr{I}}$ is a sub-LIS of some Q_i is in PSPACE.

  Here these bounds refer to the combined syntactic size of L.

Polynomially Space Bounded Logics

- Exponentially branching logic $\mathcal{L}$ *polynomial space bounded* if for any LIS $\mathscr{I}$ over one-step formulas L we have the following properties:
  - Fixing a satisfying set $Y_{\mathscr{I}}$ as $\{Q_1, \ldots, Q_m\}$ and computing some Q_i can be done in polynomial space.
  - Deciding whether a LIS Q over $V \times X_{\mathscr{I}}$ is a sub-LIS of some Q_i is in PSPACE.

  Here these bounds refer to the combined syntactic size of L.

Theorem

Satisfiability of a LIS $\mathscr{I}$ over formulas L in a polynomial space bounded logic $\mathcal{L}$ is decidable in PSPACE *(bounded in the combined syntactic size of L).*

The Logic $\mathcal{L}_{\text{gen}}$

The Logic $\mathcal{L}_{\text{gen}}$

Lemma

The logic non-expansive fuzzy $\mathcal{L}_{\text{gen}}$ is one-step exponentially bounded.

The Logic $\mathcal{L}_{\text{gen}}$

Lemma

The logic non-expansive fuzzy $\mathcal{L}_{\text{gen}}$ is one-step exponentially bounded.

Lemma

The logic non-expansive fuzzy $\mathcal{L}_{\text{gen}}$ is exponentially branching.

The Logic $\mathcal{L}_{\text{gen}}$

Lemma

The logic non-expansive fuzzy $\mathcal{L}_{\text{gen}}$ is one-step exponentially bounded.

Lemma

The logic non-expansive fuzzy $\mathcal{L}_{\text{gen}}$ is exponentially branching.

Theorem

The logic non-expansive fuzzy $\mathcal{L}_{\text{gen}}$ is polynomial space bounded.

Conclusion & Future Work

Conclusion

Conclusion & Future Work

Conclusion

- We introduced non-expansive fuzzy coalgebraic logic.

Conclusion & Future Work

Conclusion

- We introduced non-expansive fuzzy coalgebraic logic.
- We reduced satisfiability to that of one-step logics.

Conclusion & Future Work

Conclusion

- We introduced non-expansive fuzzy coalgebraic logic.
- We reduced satisfiability to that of one-step logics.
- We introduced conditions under which satisfiability in such a logic is decidable in PSPACE and proved that this is actually the case.

Conclusion & Future Work

Conclusion

- We introduced non-expansive fuzzy coalgebraic logic.
- We reduced satisfiability to that of one-step logics.
- We introduced conditions under which satisfiability in such a logic is decidable in PSPACE and proved that this is actually the case.
- We proved this for the logic $\mathcal{L}_{\text{gen}}$.

Future work

Conclusion & Future Work

Conclusion

- We introduced non-expansive fuzzy coalgebraic logic.
- We reduced satisfiability to that of one-step logics.
- We introduced conditions under which satisfiability in such a logic is decidable in PSPACE and proved that this is actually the case.
- We proved this for the logic $\mathcal{L}_{\text{gen}}$.

Future work

- Cover more logics (partially done).