Categorical Semantics for Counterpart-based Temporal Logics in Agda

Andrea Laretto[1], Fabio Gadducci[2], Davide Trotta[2]

1: Tallinn University of Technology, 2: University of Pisa

3rd ItaCa Workshop, Pisa

December 21st, 2022

Outline

In this work we present the categorical semantics of a counterpart-based temporal logic, and formalize it using the proof assistant Agda and the agda-categories library.

Outline

In this work we present the categorical semantics of a counterpart-based temporal logic, and formalize it using the proof assistant Agda and the agda-categories library.

1. Temporal logics and counterpart semantics

Outline

In this work we present the categorical semantics of a counterpart-based temporal logic, and formalize it using the proof assistant Agda and the agda-categories library.

1. Temporal logics and counterpart semantics
2. Categorical perspective

Outline

In this work we present the categorical semantics of a counterpart-based temporal logic, and formalize it using the proof assistant Agda and the agda-categories library.

1. Temporal logics and counterpart semantics
2. Categorical perspective
3. Agda formalization

Outline

In this work we present the categorical semantics of a counterpart-based temporal logic, and formalize it using the proof assistant Agda and the agda-categories library.

1. Temporal logics and counterpart semantics
2. Categorical perspective
3. Agda formalization
4. Conclusion and future work

Temporal logics

Well-known formalism for specifying and verifying complex systems

Temporal logics

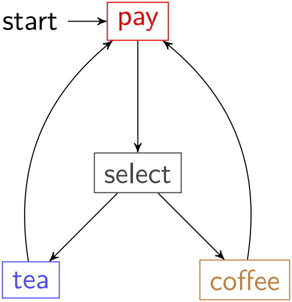
Well-known formalism for specifying and verifying complex systems

1. Represent the system as a **transition system**, called *model*

Temporal logics

Well-known formalism for specifying and verifying complex systems

1. Represent the system as a **transition system**, called *model*

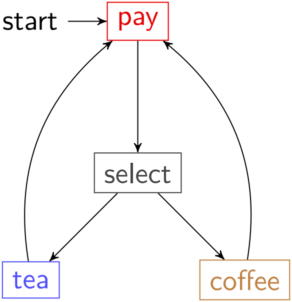


Transition system for a simple vending machine

Temporal logics

Well-known formalism for specifying and verifying complex systems

1. Represent the system as a **transition system**, called *model*



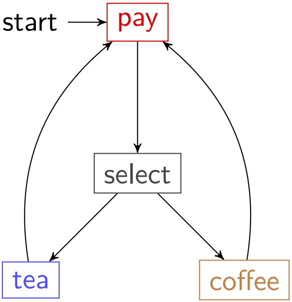
Transition system for a simple vending machine

2. Express desired properties as *formulas* in a **temporal logic**

Temporal logics

Well-known formalism for specifying and verifying complex systems

1. Represent the system as a **transition system**, called *model*



Transition system for a simple vending machine

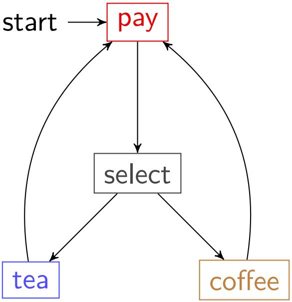
2. Express desired properties as *formulas* in a **temporal logic**

Always(Eventually(pay))

Temporal logics

Well-known formalism for specifying and verifying complex systems

1. Represent the system as a **transition system**, called *model*



Transition system for a simple vending machine

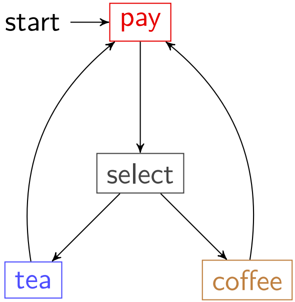
2. Express desired properties as *formulas* in a **temporal logic**

$$\text{Always}(\text{Eventually}(\text{pay})) \qquad \neg\,\text{Eventually}(\text{tea})$$

Temporal logics

Well-known formalism for specifying and verifying complex systems

1. Represent the system as a **transition system**, called *model*



Transition system for a simple vending machine

2. Express desired properties as *formulas* in a **temporal logic**

$$\text{Always}(\text{Eventually}(\text{pay})) \qquad \neg\, \text{Eventually}(\text{tea})$$

3. Use a program to *check* that the *model* **satisfies** the formula

Motivation: Multi-component models

- States are simply atomic points

Motivation: Multi-component models

- States are simply atomic points
- In practice, states often have structure that can change in time:

Motivation: Multi-component models

- States are simply atomic points
- In practice, states often have structure that can change in time:
  - Time evolution of *graph topologies*: merging nodes, deletion of edges

Motivation: Multi-component models

- States are simply atomic points
- In practice, states often have structure that can change in time:
  - Time evolution of *graph topologies*: merging nodes, deletion of edges
  - Managing *processes* in memory: forking, allocation and deallocation

Motivation: Multi-component models

- States are simply atomic points
- In practice, states often have structure that can change in time:
  - Time evolution of *graph topologies*: merging nodes, deletion of edges
  - Managing *processes* in memory: forking, allocation and deallocation
  - Dynamic behaviour of *election algorithms*: splitting and union of parties

Motivation: Multi-component models

- States are simply atomic points
- In practice, states often have structure that can change in time:
  - Time evolution of *graph topologies*: merging nodes, deletion of edges
  - Managing *processes* in memory: forking, allocation and deallocation
  - Dynamic behaviour of *election algorithms*: splitting and union of parties
- Objectives:

Can we enrich our models to express multi-component behaviour?

Motivation: Multi-component models

- States are simply atomic points
- In practice, states often have structure that can change in time:
  - Time evolution of *graph topologies*: merging nodes, deletion of edges
  - Managing *processes* in memory: forking, allocation and deallocation
  - Dynamic behaviour of *election algorithms*: splitting and union of parties
- Objectives:

Can we enrich our models to express multi-component behaviour?

Can we define logics that can reason on the fate of individual elements?

Motivation: Multi-component models

- States are simply atomic points
- In practice, states often have structure that can change in time:
  - Time evolution of *graph topologies*: merging nodes, deletion of edges
  - Managing *processes* in memory: forking, allocation and deallocation
  - Dynamic behaviour of *election algorithms*: splitting and union of parties
- Objectives:

  Can we enrich our models to express multi-component behaviour?
  Can we define logics that can reason on the fate of individual elements?

- *Yes!* Using **counterpart models** and quantified temporal logics

Counterpart paradigm

- Standard LTL traces: *sequences of states*

Counterpart paradigm

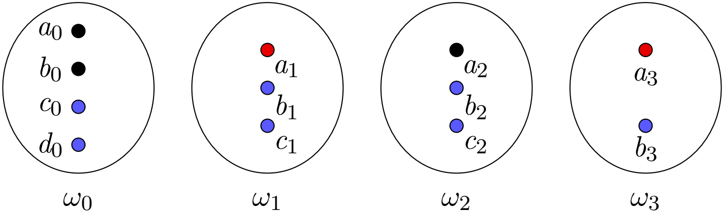
- Standard LTL traces: *sequences of states*

- Associate to each state a set of individuals, called **worlds**

Counterpart paradigm

- Standard LTL traces: *sequences of states*

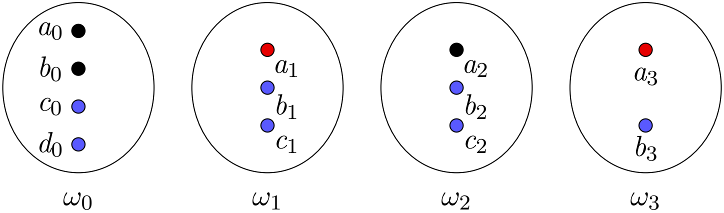
- Associate to each state a set of individuals, called **worlds**
- Our traces: *sequences of worlds*



Counterpart paradigm

- Standard LTL traces: *sequences of states*

- Associate to each state a set of individuals, called **worlds**
- Our traces: *sequences of worlds*

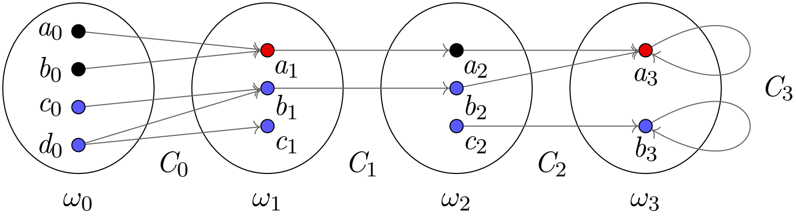


How do we represent transitions?

Counterpart paradigm

- Standard LTL traces: *sequences of states*

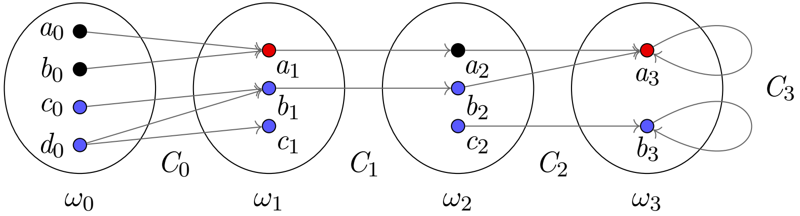
- Associate to each state a set of individuals, called **worlds**
- Our traces: *sequences of worlds, connected with* ***counterpart relations***



Counterpart paradigm

- Standard LTL traces: *sequences of states*

- Associate to each state a set of individuals, called **worlds**
- Our traces: *sequences of worlds, connected with* ***counterpart relations***

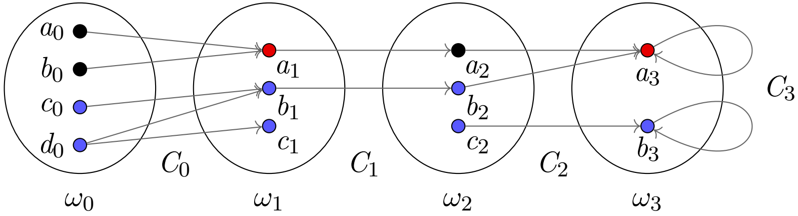


- Intuition: individuals connected by a relation *are the same after one step*

Counterpart paradigm

- Standard LTL traces: *sequences of states*

- Associate to each state a set of individuals, called **worlds**
- Our traces: *sequences of worlds, connected with* ***counterpart relations***



- Intuition: individuals connected by a relation *are the same after one step*
- We call these sequences of worlds and relations **counterpart traces**

QLTL

- QLTL: *(first-order) quantified linear temporal logic* using traces

QLTL

- QLTL: *(first-order) quantified linear temporal logic* using traces
- Syntax of QLTL formulas:

$$\phi := \mathsf{true} \mid \neg\phi \mid \phi_1 \vee \phi_2$$

QLTL

- QLTL: *(first-order) quantified linear temporal logic* using traces
- Syntax of QLTL formulas:

$$\phi := \mathsf{true} \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \mathsf{Next}(\phi) \mid \phi_1 \,\mathsf{Until}\, \phi_2$$

QLTL

- QLTL: *(first-order) quantified linear temporal logic* using traces
- Syntax of QLTL formulas:

$$\phi := \mathsf{true} \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \mathsf{Next}(\phi) \mid \phi_1 \,\mathsf{Until}\, \phi_2 \mid x = y \mid \exists x.\phi \mid P(x)$$

QLTL

- QLTL: *(first-order) quantified linear temporal logic* using traces
- Syntax of QLTL formulas:

$$\phi := \mathsf{true} \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \mathsf{Next}(\phi) \mid \phi_1 \,\mathsf{Until}\, \phi_2 \mid x = y \mid \exists x.\phi \mid P(x)$$

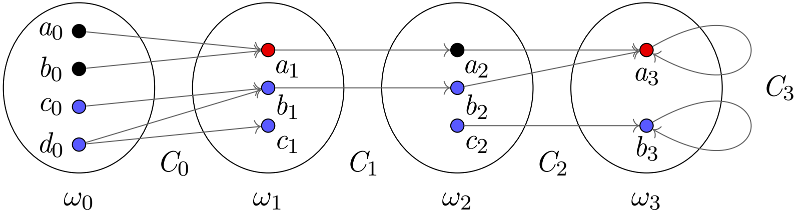
- Semantics: *for each world*, define the (tuples of) individuals satisfying ϕ

QLTL

- QLTL: *(first-order) quantified linear temporal logic* using traces
- Syntax of QLTL formulas:

$$\phi := \mathsf{true} \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \mathsf{Next}(\phi) \mid \phi_1 \,\mathsf{Until}\, \phi_2 \mid x = y \mid \exists x.\phi \mid P(x)$$

- Semantics: *for each world*, define the (tuples of) individuals satisfying ϕ

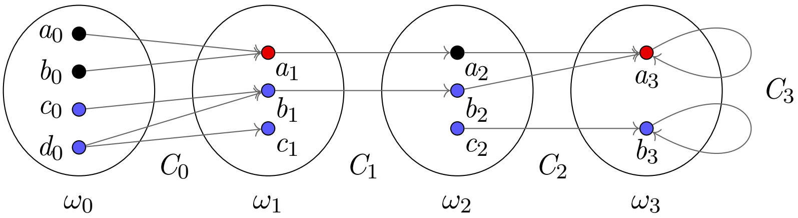


QLTL

- QLTL: *(first-order) quantified linear temporal logic* using traces
- Syntax of QLTL formulas:

$$\phi := \mathsf{true} \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \mathsf{Next}(\phi) \mid \phi_1 \,\mathsf{Until}\, \phi_2 \mid x = y \mid \exists x.\phi \mid P(x)$$

- Semantics: *for each world*, define the (tuples of) individuals satisfying ϕ



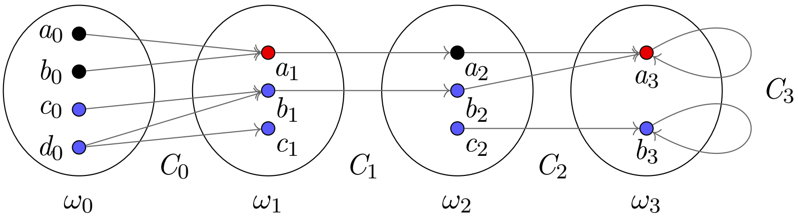
- $a_0 \vDash_{\omega_0} \mathsf{Next}(\mathsf{Red}(x))$

QLTL

- QLTL: *(first-order) quantified linear temporal logic* using traces
- Syntax of QLTL formulas:

$$\phi := \mathsf{true} \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \mathsf{Next}(\phi) \mid \phi_1 \,\mathsf{Until}\, \phi_2 \mid x = y \mid \exists x.\phi \mid P(x)$$

- Semantics: *for each world*, define the (tuples of) individuals satisfying ϕ



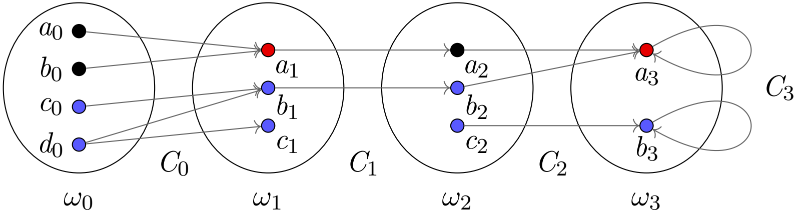
- $a_0 \vDash_{\omega_0} \mathsf{Next}(\mathsf{Red}(x))$
- $c_1 \vDash_{\omega_1} \neg\mathsf{Next}(\mathsf{Red}(x))$

QLTL

- QLTL: *(first-order) quantified linear temporal logic* using traces
- Syntax of QLTL formulas:

$$\phi := \mathsf{true} \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \mathsf{Next}(\phi) \mid \phi_1 \,\mathsf{Until}\, \phi_2 \mid x = y \mid \exists x.\phi \mid P(x)$$

- Semantics: *for each world*, define the (tuples of) individuals satisfying ϕ



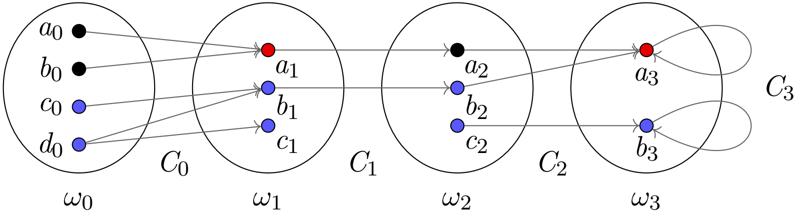
- $a_0 \vDash_{\omega_0} \mathsf{Next}(\mathsf{Red}(x))$
- $c_1 \vDash_{\omega_1} \neg\mathsf{Next}(\mathsf{Red}(x))$
- $c_0 \vDash_{\omega_0} \mathsf{Blue}(x)\,\mathsf{Until}\,\mathsf{Red}(x)$

QLTL

- QLTL: *(first-order) quantified linear temporal logic* using traces
- Syntax of QLTL formulas:

$$\phi := \mathsf{true} \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \mathsf{Next}(\phi) \mid \phi_1 \,\mathsf{Until}\, \phi_2 \mid x = y \mid \exists x.\phi \mid P(x)$$

- Semantics: *for each world*, define the (tuples of) individuals satisfying ϕ



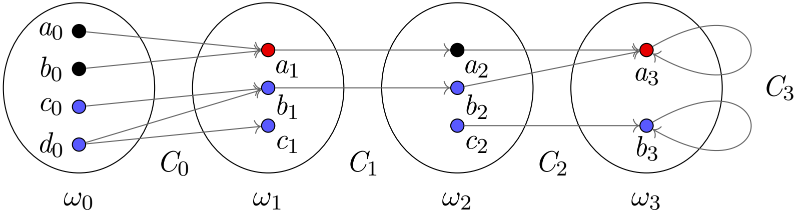
- $a_0 \vDash_{\omega_0} \mathsf{Next}(\mathsf{Red}(x))$
- $c_1 \vDash_{\omega_1} \neg\mathsf{Next}(\mathsf{Red}(x))$
- $c_0 \vDash_{\omega_0} \mathsf{Blue}(x) \,\mathsf{Until}\, \mathsf{Red}(x)$
- $(a_0, b_0) \vDash_{\omega_0} \mathsf{Next}(x = y)$

QLTL

- QLTL: *(first-order) quantified linear temporal logic* using traces
- Syntax of QLTL formulas:

$$\phi := \mathsf{true} \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \mathsf{Next}(\phi) \mid \phi_1 \,\mathsf{Until}\, \phi_2 \mid x = y \mid \exists x.\phi \mid P(x)$$

- Semantics: *for each world*, define the (tuples of) individuals satisfying ϕ



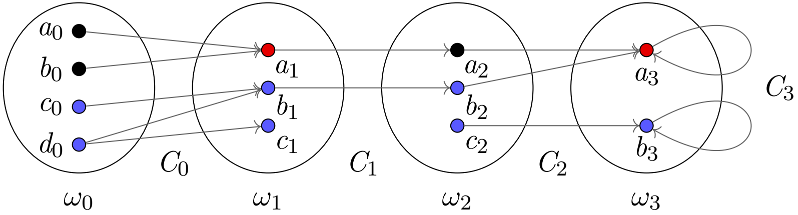
- $a_0 \vDash_{\omega_0} \mathsf{Next}(\mathsf{Red}(x))$
- $c_1 \vDash_{\omega_1} \neg\mathsf{Next}(\mathsf{Red}(x))$
- $c_0 \vDash_{\omega_0} \mathsf{Blue}(x)\,\mathsf{Until}\,\mathsf{Red}(x)$
- $(a_0, b_0) \vDash_{\omega_0} \mathsf{Next}(x = y)$
- $() \vDash_{w_2} \exists x.\mathsf{Next}(\mathsf{Blue}(x))$

QLTL

- QLTL: *(first-order) quantified linear temporal logic* using traces
- Syntax of QLTL formulas:

$$\phi := \mathsf{true} \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \mathsf{Next}(\phi) \mid \phi_1\, \mathsf{Until}\, \phi_2 \mid x = y \mid \exists x.\phi \mid P(x)$$

- Semantics: *for each world*, define the (tuples of) individuals satisfying ϕ



- $a_0 \vDash_{\omega_0} \mathsf{Next}(\mathsf{Red}(x))$
- $c_1 \vDash_{\omega_1} \neg\mathsf{Next}(\mathsf{Red}(x))$
- $c_0 \vDash_{\omega_0} \mathsf{Blue}(x)\, \mathsf{Until}\, \mathsf{Red}(x)$
- $(a_0, b_0) \vDash_{\omega_0} \mathsf{Next}(x = y)$
- $() \vDash_{w_2} \exists x.\mathsf{Next}(\mathsf{Blue}(x))$
- $(a_0, c_0) \vDash_{\omega_0} (\neg(x = y))\, \mathsf{Until}\, (x = y)$

Categorical semantics

- *Counterpart model*: a transition system enriched with worlds and counterpart relations between them

Categorical semantics

- *Counterpart model*: a transition system enriched with worlds and counterpart relations between them
- Our claim: counterpart models can be understood within the unifying perspective of *category theory and categorical logic:*

Categorical semantics

- *Counterpart model*: a transition system enriched with worlds and counterpart relations between them
- Our claim: counterpart models can be understood within the unifying perspective of *category theory and categorical logic:*

$$\textit{Counterpart model} \approx \textit{a category } \mathcal{W}$$

Categorical semantics

- *Counterpart model*: a transition system enriched with worlds and counterpart relations between them
- Our claim: counterpart models can be understood within the unifying perspective of *category theory and categorical logic:*

$$\begin{aligned} \textit{Counterpart model} \approx\ & \textit{a category } \mathcal{W} \\ +\ & \underbrace{\textit{a class } T \textit{ of selected morphisms of } \mathcal{W}}_{\textit{Temporal structure}} \end{aligned}$$

Categorical semantics

- *Counterpart model*: a transition system enriched with worlds and counterpart relations between them
- Our claim: counterpart models can be understood within the unifying perspective of *category theory and categorical logic:*

$$\begin{aligned} \textit{Counterpart model} \approx\ & \textit{a category } \mathcal{W} \\ +\ & \underbrace{\textit{a class } T \textit{ of selected morphisms of } \mathcal{W}}_{\textit{Temporal structure}} \\ +\ & \underbrace{\textit{a presheaf } D : \mathcal{W}^{op} \to \textbf{\textit{Rel}}}_{\textit{Relational presheaf}} \end{aligned}$$

Categorical semantics

- *Counterpart model*: a transition system enriched with worlds and counterpart relations between them
- Our claim: counterpart models can be understood within the unifying perspective of *category theory and categorical logic:*

$$\begin{aligned} \textit{Counterpart model} \approx\ & \textit{a category } \mathcal{W} \\ & + \underbrace{\textit{a class } T \textit{ of selected morphisms of } \mathcal{W}}_{\textit{Temporal structure}} \\ & + \underbrace{\textit{a presheaf } D : \mathcal{W}^{op} \to \textbf{\textit{Rel}}}_{\textit{Relational presheaf}} \end{aligned}$$

- Objects of $\mathcal{W}$ are the states of the underlying *transition system*

Categorical semantics

- *Counterpart model*: a transition system enriched with worlds and counterpart relations between them
- Our claim: counterpart models can be understood within the unifying perspective of *category theory and categorical logic:*

$$
\begin{aligned}
\textit{Counterpart model} \approx\ & \textit{a category } \mathcal{W} \\
+\ & \underbrace{\textit{a class } T \textit{ of selected morphisms of } \mathcal{W}}_{\textit{Temporal structure}} \\
+\ & \underbrace{\textit{a presheaf } D : \mathcal{W}^{op} \to \textbf{\textit{Rel}}}_{\textit{Relational presheaf}}
\end{aligned}
$$

- Objects of $\mathcal{W}$ are the states of the underlying *transition system*
- Morphisms of $\mathcal{W}$ represent *transitions* between states

Categorical semantics

- *Counterpart model*: a transition system enriched with worlds and counterpart relations between them
- Our claim: counterpart models can be understood within the unifying perspective of *category theory and categorical logic:*

$$\begin{aligned} \textit{Counterpart model} \approx\ & \textit{a category } \mathcal{W} \\ & + \underbrace{\textit{a class } T \textit{ of selected morphisms of } \mathcal{W}}_{\textit{Temporal structure}} \\ & + \underbrace{\textit{a presheaf } D : \mathcal{W}^{op} \to \textbf{\textit{Rel}}}_{\textit{Relational presheaf}} \end{aligned}$$

- Objects of $\mathcal{W}$ are the states of the underlying *transition system*
- Morphisms of $\mathcal{W}$ represent *transitions* between states
- The *temporal structure* identifies the one-step transitions of the model

Categorical semantics

- *Counterpart model*: a transition system enriched with worlds and counterpart relations between them
- Our claim: counterpart models can be understood within the unifying perspective of *category theory and categorical logic:*

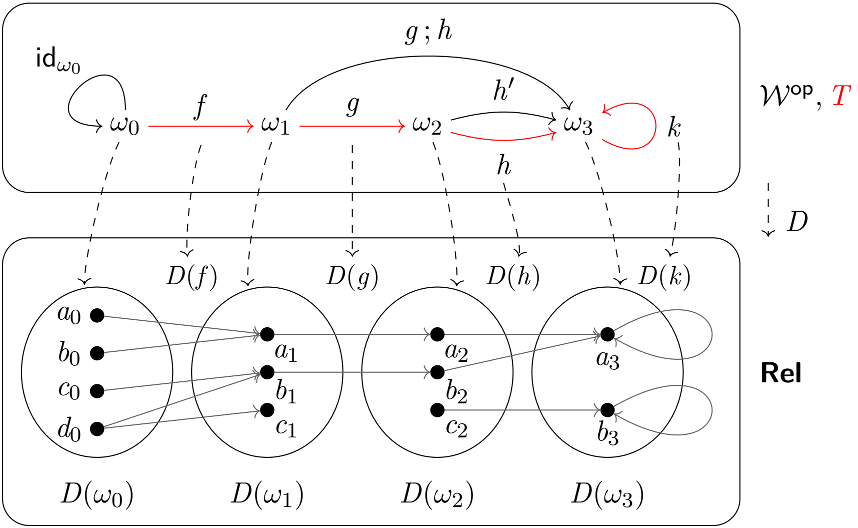
$$\begin{aligned} \textit{Counterpart model} \approx\ & \textit{a category } \mathcal{W} \\ +\ & \underbrace{\textit{a class } T \textit{ of selected morphisms of } \mathcal{W}}_{\textit{Temporal structure}} \\ +\ & \underbrace{\textit{a presheaf } D : \mathcal{W}^{op} \to \textbf{\textit{Rel}}}_{\textit{Relational presheaf}} \end{aligned}$$

- Objects of $\mathcal{W}$ are the states of the underlying *transition system*
- Morphisms of $\mathcal{W}$ represent *transitions* between states
- The *temporal structure* identifies the one-step transitions of the model
- The *relational presheaf* assigns worlds and counterpart relations to states

$$
\begin{tikzcd}
\omega_0 \arrow[loop left, "\mathsf{id}_{\omega_0}"] \arrow[r, "f"] & \omega_1 \arrow[r, "g"] \arrow[rr, bend left, "g\,;\,h"] & \omega_2 \arrow[r, bend left, "h'"] \arrow[r, bend right, "h"'] & \omega_3 \arrow[loop right, "k"]
\end{tikzcd}
$$

$\mathcal{W}^{\mathsf{op}}$, T

Relational presheaves – Example



Categorical and classical semantics

- *How is the semantics of our logic defined?*

Categorical and classical semantics

- *How is the semantics of our logic defined?*
- Semantics, the standard way: an *inductively defined relation* between formulae ϕ and (tuples of) individuals *in each world*

Categorical and classical semantics

- *How is the semantics of our logic defined?*
- Semantics, the standard way: an *inductively defined relation* between formulae ϕ and (tuples of) individuals *in each world*
- Semantics, the categorical way via *classical attributes*: *for each world*, define the *set* of (tuples of) individuals satisfying ϕ

Categorical and classical semantics

- *How is the semantics of our logic defined?*
- Semantics, the standard way: an *inductively defined relation* between formulae ϕ and (tuples of) individuals *in each world*
- Semantics, the categorical way via *classical attributes*: *for each world*, define the *set* of (tuples of) individuals satisfying ϕ

A **classical attribute** A **on** a relational presheaf $D : \mathcal{W}^{\mathsf{op}} \to \mathbf{Rel}$ is a family of sets $A := \{A_\omega\}_{\omega \in \mathcal{W}}$ such that $A_\omega \subseteq D(\omega)$ for every $\omega \in \mathcal{W}$.

Categorical and classical semantics

- *How is the semantics of our logic defined?*
- Semantics, the standard way: an *inductively defined relation* between formulae ϕ and (tuples of) individuals *in each world*
- Semantics, the categorical way via *classical attributes*: *for each world*, define the *set* of (tuples of) individuals satisfying ϕ

A **classical attribute** A **on** a relational presheaf $D : \mathcal{W}^{\mathsf{op}} \to \mathbf{Rel}$ is a family of sets $A := \{A_\omega\}_{\omega \in \mathcal{W}}$ such that $A_\omega \subseteq D(\omega)$ for every $\omega \in \mathcal{W}$.

- Idea: a relational presheaf provides a *common universe* of elements

Categorical and classical semantics

- *How is the semantics of our logic defined?*
- Semantics, the standard way: an *inductively defined relation* between formulae ϕ and (tuples of) individuals *in each world*
- Semantics, the categorical way via *classical attributes*: *for each world*, define the *set* of (tuples of) individuals satisfying ϕ

A **classical attribute** A **on** a relational presheaf $D : \mathcal{W}^{\mathsf{op}} \to \mathbf{Rel}$ is a family of sets $A := \{A_\omega\}_{\omega \in \mathcal{W}}$ such that $A_\omega \subseteq D(\omega)$ for every $\omega \in \mathcal{W}$.

- Idea: a relational presheaf provides a *common universe* of elements
- A classical attribute A identifies a subset of individuals which satisfy a given property A

Categorical and classical semantics

- Take a counterpart model $\langle \mathcal{W}, T, D \rangle$.

Categorical and classical semantics

- Take a counterpart model $\langle \mathcal{W}, T, D \rangle$.
- Assign relational presheaves to contexts $\Gamma = [x_1, \dots, x_n]$:

$$[\![\Gamma]\!] = \underbrace{D \times \cdots \times D}_{n \text{ times}}$$

Categorical and classical semantics

- Take a counterpart model $\langle \mathcal{W}, T, D \rangle$.
- Assign relational presheaves to contexts $\Gamma = [x_1, \ldots, x_n]$:

$$[\![\Gamma]\!] = \underbrace{D \times \cdots \times D}_{n \text{ times}}$$

- Take a QLTL formula in a context Γ: its semantics is a classical attribute on $[\![\Gamma]\!]$, defined as follows:

Categorical and classical semantics

- Take a counterpart model $\langle \mathcal{W}, T, D \rangle$.
- Assign relational presheaves to contexts $\Gamma = [x_1, \dots, x_n]$:

$$[\![\Gamma]\!] = \underbrace{D \times \cdots \times D}_{n \text{ times}}$$

- Take a QLTL formula in a context Γ: its semantics is a classical attribute on $[\![\Gamma]\!]$, defined as follows:
  - $[\![\mathsf{false}]\!]_\omega := \emptyset$;

Categorical and classical semantics

- Take a counterpart model $\langle \mathcal{W}, T, D \rangle$.
- Assign relational presheaves to contexts $\Gamma = [x_1, \ldots, x_n]$:
$$[\![\Gamma]\!] = \underbrace{D \times \cdots \times D}_{n \text{ times}}$$
- Take a QLTL formula in a context Γ: its semantics is a classical attribute on $[\![\Gamma]\!]$, defined as follows:
  - $[\![\mathsf{false}]\!]_\omega := \emptyset$;
  - $[\![\mathsf{true}]\!]_\omega := [\![\Gamma]\!](\omega)$;

Categorical and classical semantics

- Take a counterpart model $\langle \mathcal{W}, T, D \rangle$.
- Assign relational presheaves to contexts $\Gamma = [x_1, \ldots, x_n]$:

$$[\![\Gamma]\!] = \underbrace{D \times \cdots \times D}_{n \text{ times}}$$

- Take a QLTL formula in a context Γ: its semantics is a classical attribute on $[\![\Gamma]\!]$, defined as follows:
  - $[\![\mathsf{false}]\!]_\omega := \emptyset$;
  - $[\![\mathsf{true}]\!]_\omega := [\![\Gamma]\!](\omega)$;
  - $[\![\phi_1 \vee \phi_2]\!]_\omega := [\![\phi_1]\!]_\omega \cup [\![\phi_2]\!]_\omega$;

Categorical and classical semantics

- Take a counterpart model $\langle \mathcal{W}, T, D \rangle$.
- Assign relational presheaves to contexts $\Gamma = [x_1, \ldots, x_n]$:

$$[\![\Gamma]\!] = \underbrace{D \times \cdots \times D}_{n \text{ times}}$$

- Take a QLTL formula in a context Γ: its semantics is a classical attribute on $[\![\Gamma]\!]$, defined as follows:
  - $[\![\text{false}]\!]_\omega := \emptyset$;
  - $[\![\text{true}]\!]_\omega := [\![\Gamma]\!](\omega)$;
  - $[\![\phi_1 \vee \phi_2]\!]_\omega := [\![\phi_1]\!]_\omega \cup [\![\phi_2]\!]_\omega$;
  - $[\![\neg\psi]\!]_\omega := [\![\Gamma]\!](\omega) \setminus [\![[\Gamma]\psi]\!]_\omega$;

Categorical and classical semantics

- Take a counterpart model $\langle \mathcal{W}, T, D \rangle$.
- Assign relational presheaves to contexts $\Gamma = [x_1, \ldots, x_n]$:

$$[\![\Gamma]\!] = \underbrace{D \times \cdots \times D}_{n \text{ times}}$$

- Take a QLTL formula in a context Γ: its semantics is a classical attribute on $[\![\Gamma]\!]$, defined as follows:
  - $[\![\mathsf{false}]\!]_\omega := \emptyset$;
  - $[\![\mathsf{true}]\!]_\omega := [\![\Gamma]\!](\omega)$;
  - $[\![\phi_1 \vee \phi_2]\!]_\omega := [\![\phi_1]\!]_\omega \cup [\![\phi_2]\!]_\omega$;
  - $[\![\neg\psi]\!]_\omega := [\![\Gamma]\!](\omega) \setminus [\![[\Gamma]\psi]\!]_\omega$;
  - $[\![x = y]\!]_\omega := \{a \in [\![\Gamma]\!](\omega) \mid \pi_x(a) = \pi_y(a)\}$;

Categorical and classical semantics

- Take a counterpart model $\langle \mathcal{W}, T, D \rangle$.
- Assign relational presheaves to contexts $\Gamma = [x_1, \ldots, x_n]$:

$$\llbracket \Gamma \rrbracket = \underbrace{D \times \cdots \times D}_{n \text{ times}}$$

- Take a QLTL formula in a context Γ: its semantics is a classical attribute on $\llbracket \Gamma \rrbracket$, defined as follows:
  - $\llbracket \mathsf{false} \rrbracket_\omega := \emptyset$;
  - $\llbracket \mathsf{true} \rrbracket_\omega := \llbracket \Gamma \rrbracket(\omega)$;
  - $\llbracket \phi_1 \vee \phi_2 \rrbracket_\omega := \llbracket \phi_1 \rrbracket_\omega \cup \llbracket \phi_2 \rrbracket_\omega$;
  - $\llbracket \neg \psi \rrbracket_\omega := \llbracket \Gamma \rrbracket(\omega) \setminus \llbracket [\Gamma] \psi \rrbracket_\omega$;
  - $\llbracket x = y \rrbracket_\omega := \{ a \in \llbracket \Gamma \rrbracket(\omega) \mid \pi_x(a) = \pi_y(a) \}$;
  - $\llbracket \exists x.\phi \rrbracket_\omega := \{ a \in \llbracket \Gamma \rrbracket(\omega) \mid \exists b \in D(\omega). \langle a, b \rangle \in \llbracket \phi \rrbracket_\omega \}$;

Categorical and classical semantics

- Take a counterpart model $\langle \mathcal{W}, T, D \rangle$.
- Assign relational presheaves to contexts $\Gamma = [x_1, \dots, x_n]$:

$$[\![\Gamma]\!] = \underbrace{D \times \cdots \times D}_{n \text{ times}}$$

- Take a QLTL formula in a context Γ: its semantics is a classical attribute on $[\![\Gamma]\!]$, defined as follows:
  - $[\![\mathsf{false}]\!]_\omega := \emptyset$;
  - $[\![\mathsf{true}]\!]_\omega := [\![\Gamma]\!](\omega)$;
  - $[\![\phi_1 \vee \phi_2]\!]_\omega := [\![\phi_1]\!]_\omega \cup [\![\phi_2]\!]_\omega$;
  - $[\![\neg\psi]\!]_\omega := [\![\Gamma]\!](\omega) \setminus [\![[\Gamma]\psi]\!]_\omega$;
  - $[\![x = y]\!]_\omega := \{a \in [\![\Gamma]\!](\omega) \mid \pi_x(a) = \pi_y(a)\}$;
  - $[\![\exists x.\phi]\!]_\omega := \{a \in [\![\Gamma]\!](\omega) \mid \exists b \in D(\omega).\langle a, b\rangle \in [\![\phi]\!]_\omega\}$;
- *How is the semantics of temporal operators given?*

Temporal classical attributes

Take a presheaf $X : W^{\mathsf{op}} \to \mathbf{Rel}$ and classical attributes A, B on X.
Given $\omega \in \mathcal{W}$ and an element $s \in X(\omega)$, define the classical attributes:

Temporal classical attributes

Take a presheaf $X : W^{\mathsf{op}} \to \textbf{Rel}$ and classical attributes A, B on X. Given $\omega \in \mathcal{W}$ and an element $s \in X(\omega)$, define the classical attributes:

- $s \in (\mathsf{O}A)_\omega$ *iff* for every arrow $r : \omega \to \sigma$ of T there exists an element $z \in X(\sigma)$ such that $\langle z, s \rangle \in X(r)$ and $z \in A_\sigma$;

Temporal classical attributes

Take a presheaf $X : W^{\mathsf{op}} \to \mathbf{Rel}$ and classical attributes A, B on X.
Given $\omega \in \mathcal{W}$ and an element $s \in X(\omega)$, define the classical attributes:

- $s \in (\mathsf{O}A)_\omega$ *iff* for every arrow $r : \omega \to \sigma$ of T there exists an element $z \in X(\sigma)$ such that $\langle z, s \rangle \in X(r)$ and $z \in A_\sigma$;
- $s \in (A\mathsf{U}B)_\omega$ *iff* for every $t \in \mathsf{paths}(T, \omega)$ there exists $\bar{n} \geq 0$ such that
  1. for any $i < \bar{n}$, there exists $z_i \in X(\omega_i)$ such that $\langle z_i, s \rangle \in X(t_{\leq i})$ and $z_i \in A_{\omega_i}$;

Temporal classical attributes

Take a presheaf $X : W^{\mathsf{op}} \to \mathbf{Rel}$ and classical attributes A, B on X. Given $\omega \in \mathcal{W}$ and an element $s \in X(\omega)$, define the classical attributes:

- $s \in (\mathsf{O}A)_\omega$ *iff* for every arrow $r : \omega \to \sigma$ of T there exists an element $z \in X(\sigma)$ such that $\langle z, s\rangle \in X(r)$ and $z \in A_\sigma$;
- $s \in (A\mathsf{U}B)_\omega$ *iff* for every $t \in \mathsf{paths}(T, \omega)$ there exists $\bar{n} \geq 0$ such that
  1. for any $i < \bar{n}$, there exists $z_i \in X(\omega_i)$ such that $\langle z_i, s\rangle \in X(t_{\leq i})$ and $z_i \in A_{\omega_i}$;
  2. there exists $z_{\bar{n}} \in X(\omega_{\bar{n}})$ such that $\langle z_{\bar{n}}, s\rangle \in X(t_{\leq i})$ and $z_{\bar{n}} \in B_{\omega_{\bar{n}}}$.

Temporal classical attributes

Take a presheaf $X : W^{\mathrm{op}} \to \mathbf{Rel}$ and classical attributes A, B on X.
Given $\omega \in \mathcal{W}$ and an element $s \in X(\omega)$, define the classical attributes:

- $s \in (\mathsf{O}A)_\omega$ *iff* for every arrow $r : \omega \to \sigma$ of T there exists an element $z \in X(\sigma)$ such that $\langle z, s\rangle \in X(r)$ and $z \in A_\sigma$;
- $s \in (A\mathsf{U}B)_\omega$ *iff* for every $t \in \mathrm{paths}(T, \omega)$ there exists $\bar{n} \geq 0$ such that
  1. for any $i < \bar{n}$, there exists $z_i \in X(\omega_i)$ such that $\langle z_i, s\rangle \in X(t_{\leq i})$ and $z_i \in A_{\omega_i}$;
  2. there exists $z_{\bar{n}} \in X(\omega_{\bar{n}})$ such that $\langle z_{\bar{n}}, s\rangle \in X(t_{\leq i})$ and $z_{\bar{n}} \in B_{\omega_{\bar{n}}}$.

We can now define the semantics of temporal operators as:

- ...
- $[\![\mathsf{O}\phi]\!] := \mathsf{O}[\![\phi]\!]$;
- $[\![\phi_1 \mathsf{U} \phi_2]\!] := [\![\phi_1]\!]\mathsf{U}[\![\phi_2]\!]$.

Categorical and classical models

- *How are the two semantics and their models related?*

Categorical and classical models

- *How are the two semantics and their models related?*

Theorem (Classical models $\leftrightarrow$ categorical models)

($\rightarrow$) For any classical counterpart model M, there exists a categorical counterpart model W which satisfies exactly the same QLTL formulae ϕ.

Categorical and classical models

- *How are the two semantics and their models related?*

Theorem (Classical models $\leftrightarrow$ categorical models)

($\rightarrow$) For any classical counterpart model M, there exists a categorical counterpart model W which satisfies exactly the same QLTL formulae ϕ.

- Idea: construct the freely generated category $\mathcal{W}$ from the transition system; define a relational presheaf $D : \mathcal{W}^{\mathsf{op}} \rightarrow \mathbf{Rel}$ assigning worlds and relations; restrict the morphisms with the temporal structure T.

Categorical and classical models

- *How are the two semantics and their models related?*

Theorem (Classical models $\leftrightarrow$ categorical models)

($\rightarrow$) For any classical counterpart model M, there exists a categorical counterpart model W which satisfies exactly the same QLTL formulae ϕ.

- Idea: construct the freely generated category $\mathcal{W}$ from the transition system; define a relational presheaf $D : \mathcal{W}^{\mathsf{op}} \rightarrow \mathbf{Rel}$ assigning worlds and relations; restrict the morphisms with the temporal structure T.
- The *theorem* has not been formalized, but the *construction* is!

Algebraic QLTL

- *Worlds-as-algebras*: we want to generalize sets to *many-sorted algebras*

Algebraic QLTL

- *Worlds-as-algebras*: we want to generalize sets to *many-sorted algebras*

A **relational morphism** between two relational presheaves $X, Y : \mathcal{W}^{\mathsf{op}} \to \mathbf{Rel}$ is a family of *set functions* $\eta := \{\eta_\omega : X(\omega) \to Y(\omega)\}_{\omega \in W}$ such that for every $f : A \to B$ we have

$$\langle a, b\rangle \in X(f) \implies \langle \eta_A(a), \eta_B(b)\rangle \in Y(f).$$

Algebraic QLTL

- *Worlds-as-algebras*: we want to generalize sets to *many-sorted algebras*

A **relational morphism** between two relational presheaves $X, Y : \mathcal{W}^{\mathsf{op}} \to \mathbf{Rel}$ is a family of *set functions* $\eta := \{\eta_\omega : X(\omega) \to Y(\omega)\}_{\omega \in W}$ such that for every $f : A \to B$ we have

$$\langle a, b\rangle \in X(f) \implies \langle \eta_A(a), \eta_B(b)\rangle \in Y(f).$$

Let Σ be a many-sorted signature. An **algebraic counterpart model** on the signature Σ is a tuple $\langle \mathcal{W}, T, \mathcal{S}, \mathcal{F}\rangle$ such that:

- $\mathcal{W}$, T are the same of counterpart models;

Algebraic QLTL

- *Worlds-as-algebras*: we want to generalize sets to *many-sorted algebras*

A **relational morphism** between two relational presheaves $X, Y : \mathcal{W}^{\mathsf{op}} \to \mathbf{Rel}$ is a family of *set functions* $\eta := \{\eta_\omega : X(\omega) \to Y(\omega)\}_{\omega \in W}$ such that for every $f : A \to B$ we have

$$\langle a, b\rangle \in X(f) \implies \langle \eta_A(a), \eta_B(b)\rangle \in Y(f).$$

Let Σ be a many-sorted signature. An **algebraic counterpart model** on the signature Σ is a tuple $\langle \mathcal{W}, T, \mathcal{S}, \mathcal{F}\rangle$ such that:

- $\mathcal{W}$, T are the same of counterpart models;
- $\mathcal{S} = \{[\![\tau]\!] : \mathcal{W}^{\mathsf{op}} \to \mathbf{Rel}\}_{\tau \in \Sigma_S}$ is a set of *relational presheaves* on $\mathcal{W}$, assigning a relational presheaf to each sort in Σ_S;

Algebraic QLTL

- *Worlds-as-algebras*: we want to generalize sets to *many-sorted algebras*

A **relational morphism** between two relational presheaves $X, Y : \mathcal{W}^{\mathsf{op}} \to \mathbf{Rel}$ is a family of *set functions* $\eta := \{\eta_\omega : X(\omega) \to Y(\omega)\}_{\omega \in W}$ such that for every $f : A \to B$ we have

$$\langle a, b \rangle \in X(f) \implies \langle \eta_A(a), \eta_B(b) \rangle \in Y(f).$$

Let Σ be a many-sorted signature. An **algebraic counterpart model** on the signature Σ is a tuple $\langle \mathcal{W}, T, \mathcal{S}, \mathcal{F} \rangle$ such that:

- $\mathcal{W}$, T are the same of counterpart models;
- $\mathcal{S} = \{[\![\tau]\!] : \mathcal{W}^{\mathsf{op}} \to \mathbf{Rel}\}_{\tau \in \Sigma_S}$ is a set of *relational presheaves* on $\mathcal{W}$, assigning a relational presheaf to each sort in Σ_S;
- $\mathcal{F} = \{\mathcal{I}(f) : [\![\tau_1]\!] \times \cdots \times [\![\tau_n]\!] \to [\![\tau]\!]\}_{f \in \Sigma_F}$ is a set of *relational morphisms*, assigning a relational morphism to each function symbol $f : \tau_1 \times \cdots \times \tau_n \to \tau$ given in Σ_F by the signature Σ.

Algebraic QLTL: Example

- Take the signature of directed graphs Gr:

$$\mathsf{Gr}_S = \{\mathsf{node}, \mathsf{edge}\},$$

Algebraic QLTL: Example

- Take the signature of directed graphs Gr:

$$\mathsf{Gr}_S = \{\mathsf{node}, \mathsf{edge}\}, \quad \mathsf{Gr}_F = \{s, t\},$$

Algebraic QLTL: Example

- Take the signature of directed graphs Gr:

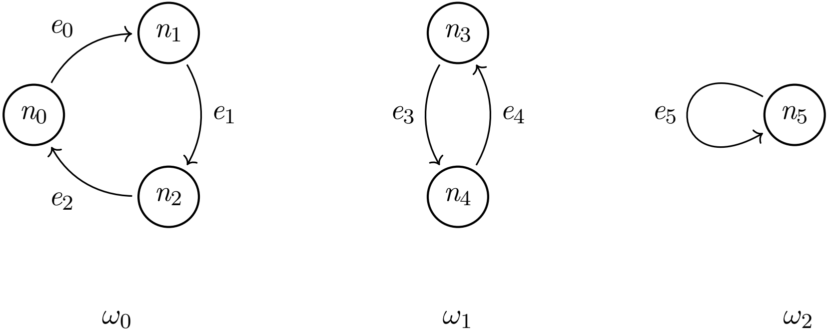
$$\mathsf{Gr}_S = \{\mathsf{node}, \mathsf{edge}\}, \quad \mathsf{Gr}_F = \{s, t\}, \text{ with } s, t : \mathsf{edge} \to \mathsf{node}$$

Algebraic QLTL: Example

- Take the signature of directed graphs Gr:

$$\mathsf{Gr}_S = \{\mathsf{node}, \mathsf{edge}\}, \quad \mathsf{Gr}_F = \{s, t\}, \text{ with } s, t : \mathsf{edge} \to \mathsf{node}$$

- A counterpart model on the signature Gr:

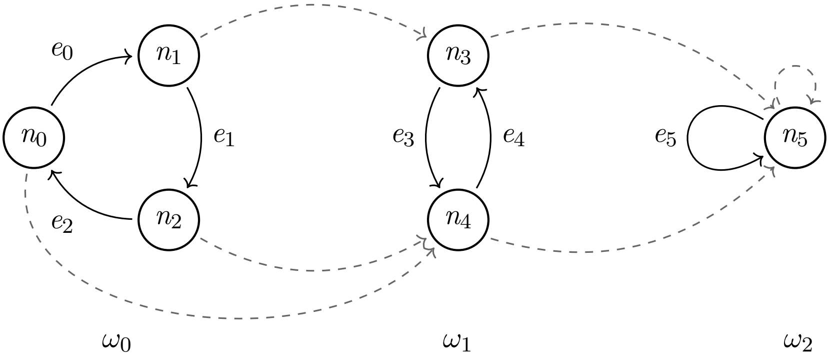


Algebraic QLTL: Example

- Take the signature of directed graphs Gr:

$$\mathsf{Gr}_S = \{\mathsf{node}, \mathsf{edge}\}, \quad \mathsf{Gr}_F = \{s, t\}, \text{ with } s, t : \mathsf{edge} \to \mathsf{node}$$

- A counterpart model on the signature Gr:

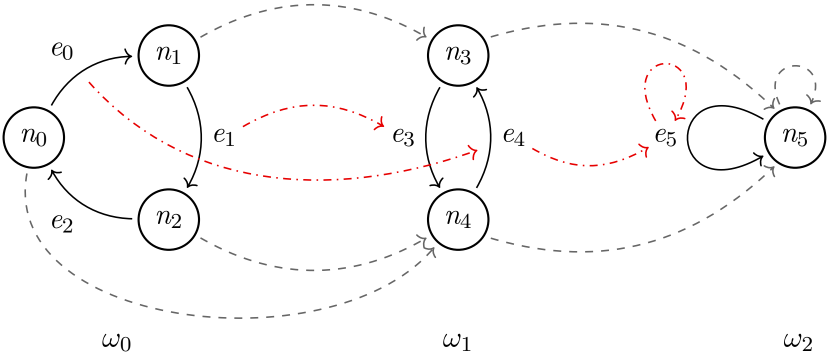


Algebraic QLTL: Example

- Take the signature of directed graphs Gr:

$$\mathrm{Gr}_S = \{\mathrm{node}, \mathrm{edge}\}, \quad \mathrm{Gr}_F = \{s, t\}, \text{ with } s, t : \mathrm{edge} \to \mathrm{node}$$

- A counterpart model on the signature Gr:

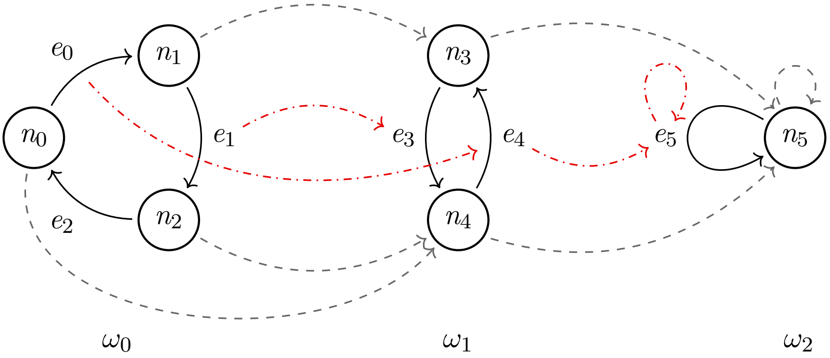


Algebraic QLTL: Example

- Take the signature of directed graphs Gr:

$$\mathsf{Gr}_S = \{\mathsf{node}, \mathsf{edge}\}, \quad \mathsf{Gr}_F = \{s, t\}, \text{ with } s, t : \mathsf{edge} \to \mathsf{node}$$

- A counterpart model on the signature Gr:



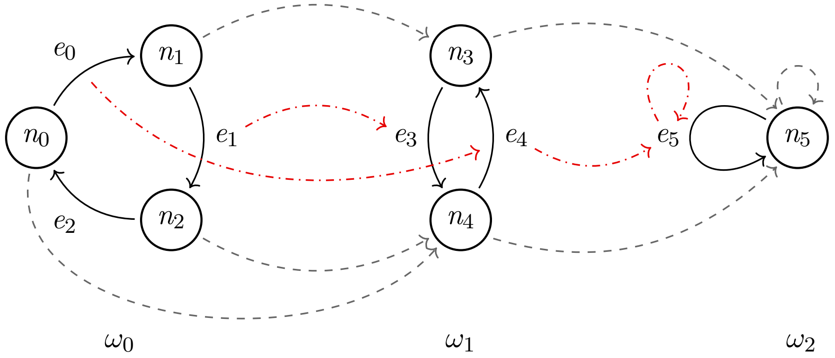
- *Algebraic* QLTL: equality between *terms*, instead of individuals:

Algebraic QLTL: Example

- Take the signature of directed graphs Gr:

$$\mathsf{Gr}_S = \{\mathsf{node}, \mathsf{edge}\}, \quad \mathsf{Gr}_F = \{s, t\}, \text{ with } s, t : \mathsf{edge} \to \mathsf{node}$$

- A counterpart model on the signature Gr:



- *Algebraic* QLTL: equality between *terms*, instead of individuals:

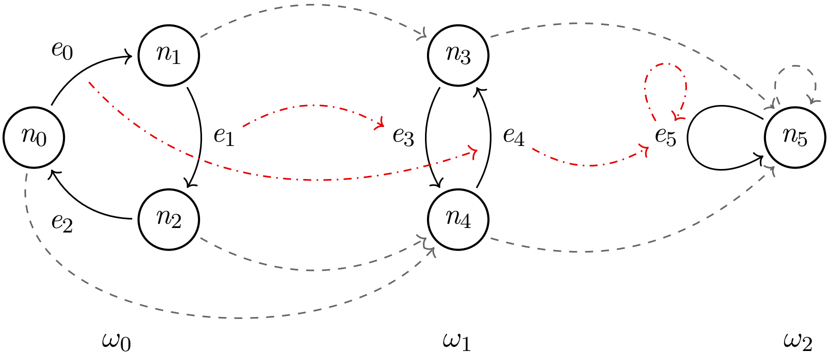
$$\textbf{loop}(e) := s(e) = t(e),$$

Algebraic QLTL: Example

- Take the signature of directed graphs Gr:

$$\mathsf{Gr}_S = \{\mathsf{node}, \mathsf{edge}\}, \quad \mathsf{Gr}_F = \{s, t\}, \text{ with } s, t : \mathsf{edge} \to \mathsf{node}$$

- A counterpart model on the signature Gr:



- *Algebraic* QLTL: equality between *terms*, instead of individuals:

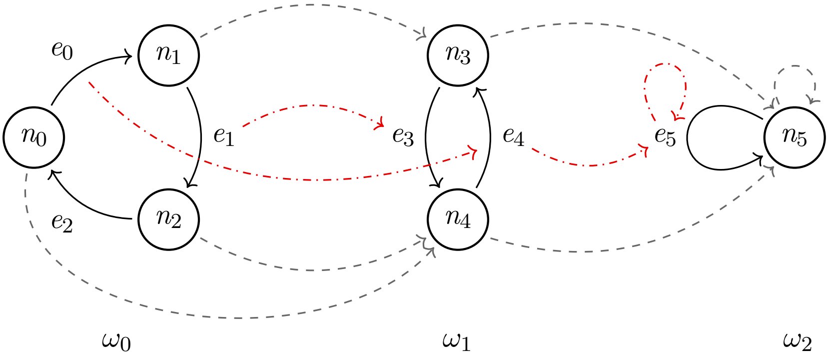
$$\mathbf{loop}(e) := s(e) = t(e), \quad e_5 \vDash_{\omega_2} \mathbf{loop}(x),$$

Algebraic QLTL: Example

- Take the signature of directed graphs Gr:

$$\mathsf{Gr}_S = \{\mathsf{node}, \mathsf{edge}\}, \quad \mathsf{Gr}_F = \{s, t\}, \text{ with } s, t : \mathsf{edge} \to \mathsf{node}$$

- A counterpart model on the signature Gr:



- *Algebraic* QLTL: equality between *terms*, instead of individuals:

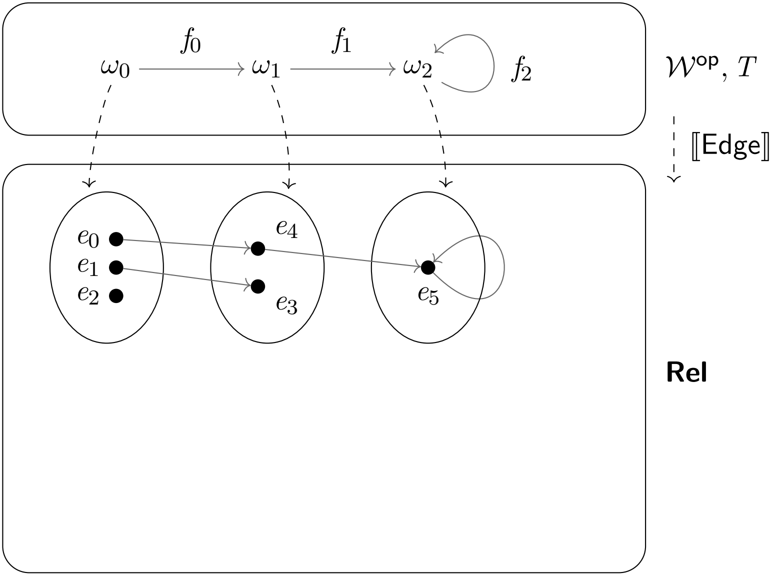
$$\mathbf{loop}(e) := s(e) = t(e), \quad e_5 \vDash_{\omega_2} \mathbf{loop}(x), \quad e_4 \vDash_{\omega_1} \mathsf{Next}(\mathbf{loop}(x)).$$

Algebraic QLTL with relational presheaves

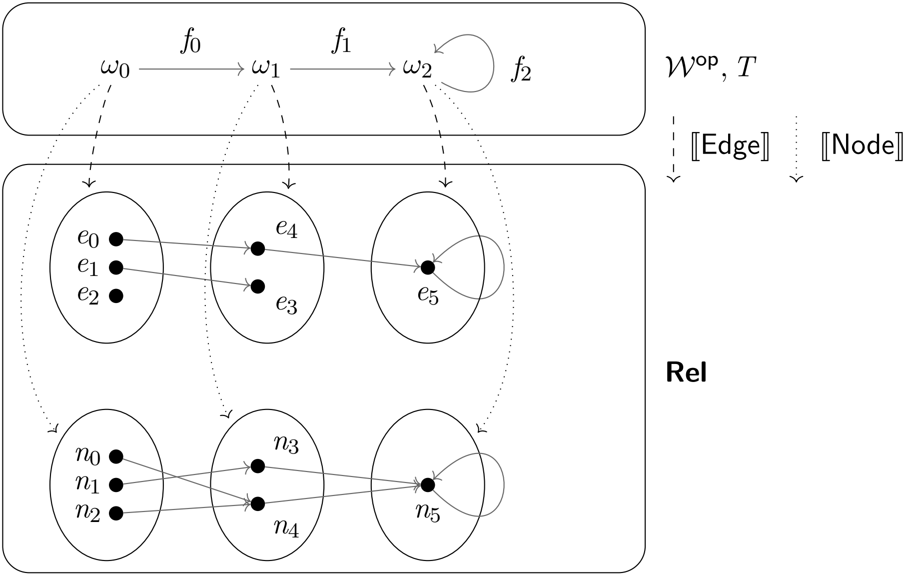
$$\begin{tikzcd} \omega_0 \arrow[r, "f_0"] & \omega_1 \arrow[r, "f_1"] & \omega_2 \arrow[loop right, "f_2"] \end{tikzcd}$$

$\mathcal{W}^{\mathsf{op}}$, T

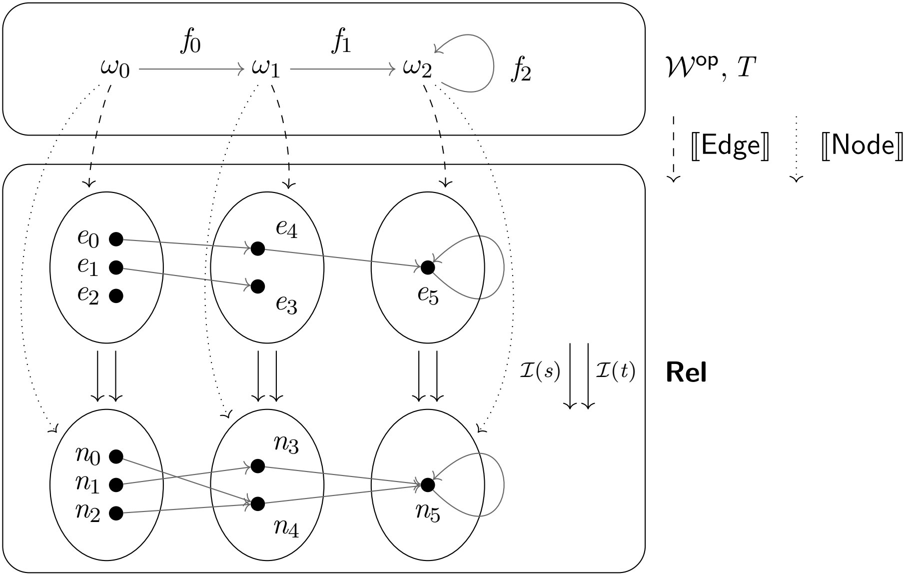
Algebraic QLTL with relational presheaves



Algebraic QLTL with relational presheaves



Algebraic QLTL with relational presheaves



Agda formalization

- Agda: *dependently typed programming language and proof assistant*

Agda formalization

- Agda: *dependently typed programming language and proof assistant*
- Can be used in practice to formalize mathematical constructions

Agda formalization

- Agda: *dependently typed programming language and proof assistant*
- Can be used in practice to formalize mathematical constructions
- Mechanization work:

Agda formalization

- Agda: *dependently typed programming language and proof assistant*
- Can be used in practice to formalize mathematical constructions
- Mechanization work:

1. A formalization of categorical QLTL and its models in Agda

Agda formalization

- Agda: *dependently typed programming language and proof assistant*
- Can be used in practice to formalize mathematical constructions
- Mechanization work:

1. A formalization of categorical QLTL and its models in Agda
2. Categorical semantics formalized using the agda-categories library

Agda formalization

- Agda: *dependently typed programming language and proof assistant*
- Can be used in practice to formalize mathematical constructions
- Mechanization work:

1. A formalization of categorical QLTL and its models in Agda
2. Categorical semantics formalized using the agda-categories library
3. Algebraic QLTL: worlds-as-algebras over any multi-sorted signature

- Agda: *dependently typed programming language and proof assistant*
- Can be used in practice to formalize mathematical constructions
- Mechanization work:

1. A formalization of categorical QLTL and its models in Agda
2. Categorical semantics formalized using the agda-categories library
3. Algebraic QLTL: worlds-as-algebras over any multi-sorted signature
4. A classical set-based semantics without the use of categorical logic

Agda formalization

- Agda: *dependently typed programming language and proof assistant*
- Can be used in practice to formalize mathematical constructions
- Mechanization work:
  1. A formalization of categorical QLTL and its models in Agda
  2. Categorical semantics formalized using the agda-categories library
  3. Algebraic QLTL: worlds-as-algebras over any multi-sorted signature
  4. A classical set-based semantics without the use of categorical logic
  5. Presentation of the *positive normal forms* of QLTL, also in Agda

Agda formalization

- Categorical semantics: **1388 lines** of Agda code
- Positive normal form: **1740 lines** of Agda code

Agda formalization

- Categorical semantics: **1388 lines** of Agda code
- Positive normal form: **1740 lines** of Agda code
- *Why is a formal presentation of our logic useful?*

Agda formalization

- Categorical semantics: **1388 lines** of Agda code
- Positive normal form: **1740 lines** of Agda code
- *Why is a formal presentation of our logic useful?*
- Formalizing constructions and semantics establishes their *correctness*

Agda formalization

- Categorical semantics: **1388 lines** of Agda code
- Positive normal form: **1740 lines** of Agda code
- *Why is a formal presentation of our logic useful?*
- Formalizing constructions and semantics establishes their *correctness*
- Provides a formal setting to *test and experiment with* temporal logics

Agda formalization

- Categorical semantics: **1388 lines** of Agda code
- Positive normal form: **1740 lines** of Agda code
- *Why is a formal presentation of our logic useful?*
- ☺ Formalizing constructions and semantics establishes their *correctness*
- ☺ Provides a formal setting to *test and experiment with* temporal logics
- ☺ Establishes a foundation to build *verified model checkers* for QTL

Agda formalization

- Categorical semantics: **1388 lines** of Agda code
- Positive normal form: **1740 lines** of Agda code
- *Why is a formal presentation of our logic useful?*
- Formalizing constructions and semantics establishes their *correctness*
- Provides a formal setting to *test and experiment with* temporal logics
- Establishes a foundation to build *verified model checkers* for QTL
- Gives a program to *convert* standard models into categorical ones:

Agda formalization

- Categorical semantics: **1388 lines** of Agda code
- Positive normal form: **1740 lines** of Agda code
- *Why is a formal presentation of our logic useful?*

☺ Formalizing constructions and semantics establishes their *correctness*

☺ Provides a formal setting to *test and experiment with* temporal logics

☺ Establishes a foundation to build *verified model checkers* for QTL

☺ Gives a program to *convert* standard models into categorical ones:

1. Define the semantics of the logic with *categorical* notions and models

Agda formalization

- Categorical semantics: **1388 lines** of Agda code
- Positive normal form: **1740 lines** of Agda code
- *Why is a formal presentation of our logic useful?*

☻ Formalizing constructions and semantics establishes their *correctness*

☻ Provides a formal setting to *test and experiment with* temporal logics

☻ Establishes a foundation to build *verified model checkers* for QTL

☻ Gives a program to *convert* standard models into categorical ones:

  1. Define the semantics of the logic with *categorical* notions and models
  2. Provide a standard *non-categorical* transition system as model

Agda formalization

- Categorical semantics: **1388 lines** of Agda code
- Positive normal form: **1740 lines** of Agda code
- *Why is a formal presentation of our logic useful?*

☺ Formalizing constructions and semantics establishes their *correctness*

☺ Provides a formal setting to *test and experiment with* temporal logics

☺ Establishes a foundation to build *verified model checkers* for QTL

☺ Gives a program to *convert* standard models into categorical ones:

1. Define the semantics of the logic with *categorical* notions and models
2. Provide a standard *non-categorical* transition system as model
3. Use the procedure `ClassicalToCategorical` to construct the categorical model so that the logic can be applied

Categorical semantics with agda-categories

- The *de-facto* (non-univalent) standard *category theory library* in Agda

Categorical semantics with agda-categories

- The *de-facto* (non-univalent) standard *category theory library* in Agda
- Some of the main design choices of the library:
    - Each category has an internalized notion of equality between *morphisms* (i.e. categories are enriched in setoids)

Categorical semantics with agda-categories

- The *de-facto* (non-univalent) standard *category theory library* in Agda
- Some of the main design choices of the library:
  - Each category has an internalized notion of equality between *morphisms* (i.e. categories are enriched in setoids)
  - $\rightarrow$ *avoids having to postulate extensionality principles*

Categorical semantics with agda-categories

- The *de-facto* (non-univalent) standard *category theory library* in Agda
- Some of the main design choices of the library:
  - Each category has an internalized notion of equality between *morphisms* (i.e. categories are enriched in setoids)
  - $\rightarrow$ *avoids having to postulate extensionality principles*
  - $\rightarrow$ *setoid-flavoured equational reasoning in proofs*

Categorical semantics with agda-categories

- The *de-facto* (non-univalent) standard *category theory library* in Agda
- Some of the main design choices of the library:
  - Each category has an internalized notion of equality between *morphisms* (i.e. categories are enriched in setoids)
  - $\rightarrow$ *avoids having to postulate extensionality principles*
  - $\rightarrow$ *setoid-flavoured equational reasoning in proofs*
  - $\rightarrow$ *additional coherences are required with working with setoids*:
    - In categories, $\circ\texttt{-resp-}\approx$: $f \approx h \rightarrow g \approx i \rightarrow f \circ g \approx h \circ i$

Categorical semantics with agda-categories

- The *de-facto* (non-univalent) standard *category theory library* in Agda
- Some of the main design choices of the library:
  - Each category has an internalized notion of equality between *morphisms* (i.e. categories are enriched in setoids)
  - $\rightarrow$ *avoids having to postulate extensionality principles*
  - $\rightarrow$ *setoid-flavoured equational reasoning in proofs*
  - $\rightarrow$ *additional coherences are required with working with setoids*:
    - In categories, `∘-resp-≈` : $f \approx h \rightarrow g \approx i \rightarrow f \circ g \approx h \circ i$
    - In functors, `F-resp-≈` : $C[f \approx g] \rightarrow D[F(f) \approx F(g)]$

Categorical semantics with agda-categories

- The *de-facto* (non-univalent) standard *category theory library* in Agda
- Some of the main design choices of the library:
  - Each category has an internalized notion of equality between *morphisms* (i.e. categories are enriched in setoids)
  - $\rightarrow$ *avoids having to postulate extensionality principles*
  - $\rightarrow$ *setoid-flavoured equational reasoning in proofs*
  - $\rightarrow$ *additional coherences are required with working with setoids*:
    - In categories, $\circ$`-resp-`$\approx$: $f \approx h \rightarrow g \approx i \rightarrow f \circ g \approx h \circ i$
    - In functors, `F-resp-`$\approx$: $C[f \approx g] \rightarrow D[F(f) \approx F(g)]$
  - Categories define two associativity fields, one of which is redundant:
    - `assoc` : $(h \circ g) \circ f \approx h \circ (g \circ f)$

Categorical semantics with agda-categories

- The *de-facto* (non-univalent) standard *category theory library* in Agda
- Some of the main design choices of the library:
  - Each category has an internalized notion of equality between *morphisms* (i.e. categories are enriched in setoids)
  - $\rightarrow$ *avoids having to postulate extensionality principles*
  - $\rightarrow$ *setoid-flavoured equational reasoning in proofs*
  - $\rightarrow$ *additional coherences are required with working with setoids*:
    - In categories, $\circ\texttt{-resp-}\approx\ :\ f \approx h \rightarrow g \approx i \rightarrow f \circ g \approx h \circ i$
    - In functors, $\texttt{F-resp-}\approx\ :\ C[f \approx g] \rightarrow D[F(f) \approx F(g)]$
  - Categories define two associativity fields, one of which is redundant:
    - $\texttt{assoc}\ :\quad (h \circ g) \circ f \approx h \circ (g \circ f)$
    - $\texttt{sym-assoc}\ :\ h \circ (g \circ f) \approx (h \circ g) \circ f$
  - $\rightarrow (C^{\mathsf{op}})^{\mathsf{op}}$ becomes definitionally equal to C, by swapping proofs twice.

Experience with agda-categories

- Extremely practical and flexible, no magic involved

Experience with agda-categories

- ☻ Extremely practical and flexible, no magic involved
- ☻ Design choices do not necessarily get in the way of practical applications

Experience with agda-categories

- ☺ Extremely practical and flexible, no magic involved
- ☺ Design choices do not necessarily get in the way of practical applications
- Main definitions used:

Experience with agda-categories

- ☻ Extremely practical and flexible, no magic involved
- ☻ Design choices do not necessarily get in the way of practical applications
- Main definitions used:
  - Categories, functors, natural transformations

Experience with agda-categories

- ☻ Extremely practical and flexible, no magic involved
- ☻ Design choices do not necessarily get in the way of practical applications
- Main definitions used:
  - Categories, functors, natural transformations
  - **Rel**: category of sets and relations

Experience with agda-categories

- ☺ Extremely practical and flexible, no magic involved
- ☺ Design choices do not necessarily get in the way of practical applications
- Main definitions used:
  - Categories, functors, natural transformations
  - **Rel**: category of sets and relations
  - Free categories generated from a quiver (`PathCategory`)

Experience with agda-categories

- ☺ Extremely practical and flexible, no magic involved
- ☺ Design choices do not necessarily get in the way of practical applications
- Main definitions used:
  - Categories, functors, natural transformations
  - **Rel**: category of sets and relations
  - Free categories generated from a quiver (`PathCategory`)
  - Presheaves, category of presheaves is complete

Experience with agda-categories

- ☺ Extremely practical and flexible, no magic involved
- ☺ Design choices do not necessarily get in the way of practical applications
- Main definitions used:
  - Categories, functors, natural transformations
  - **Rel**: category of sets and relations
  - Free categories generated from a quiver (`PathCategory`)
  - Presheaves, category of presheaves is complete
  - Relational presheaves and morphisms between them

Experience with agda-categories

- 😊 Extremely practical and flexible, no magic involved
- 😊 Design choices do not necessarily get in the way of practical applications
- Main definitions used:
  - Categories, functors, natural transformations
  - **Rel**: category of sets and relations
  - Free categories generated from a quiver (`PathCategory`)
  - Presheaves, category of presheaves is complete
  - Relational presheaves and morphisms between them
- 😐 Functoriality and setoid-equality preservation can be annoying to prove

Experience with agda-categories

- 🙂 Extremely practical and flexible, no magic involved
- 🙂 Design choices do not necessarily get in the way of practical applications
- Main definitions used:
  - Categories, functors, natural transformations
  - **Rel**: category of sets and relations
  - Free categories generated from a quiver (`PathCategory`)
  - Presheaves, category of presheaves is complete
  - Relational presheaves and morphisms between them
- 😐 Functoriality and setoid-equality preservation can be annoying to prove
- 😐 Relatively limited use of the constructions of the library in our setting

Embedding temporal logics in Agda

- The metalogic provided by Agda is *intuitionistic* in nature:

Embedding temporal logics in Agda

- The metalogic provided by Agda is *intuitionistic* in nature:

Negation is defined as $\neg A := A \rightarrow \bot$,
and the law of excluded middle $A \vee \neg A$ is *not* provable

Embedding temporal logics in Agda

- The metalogic provided by Agda is *intuitionistic* in nature:

  Negation is defined as $\neg A := A \to \bot$,
  and the law of excluded middle $A \vee \neg A$ is *not* provable

- *Consequence:* the logic we embed in Agda is also intuitionistic!

Embedding temporal logics in Agda

- The metalogic provided by Agda is *intuitionistic* in nature:

  Negation is defined as $\neg A := A \to \bot$,
  and the law of excluded middle $A \vee \neg A$ is *not* provable

- *Consequence:* the logic we embed in Agda is also intuitionistic!
- e.g., we also cannot prove in Agda that, for any choice of QLTL formula ϕ and counterpart model M,

$$M \models \neg\neg\phi \iff \phi.$$

Embedding temporal logics in Agda

- The metalogic provided by Agda is *intuitionistic* in nature:

  Negation is defined as $\neg A := A \to \bot$,
  and the law of excluded middle $A \vee \neg A$ is *not* provable

- *Consequence:* the logic we embed in Agda is also intuitionistic!
- e.g., we also cannot prove in Agda that, for any choice of QLTL formula ϕ and counterpart model M,

$$M \ \vDash \neg\neg\phi \Longleftrightarrow \phi.$$

- *Problem*: in temporal logic we usually define the essential operators of the logic, and then derive the other ones with negation:

Embedding temporal logics in Agda

- The metalogic provided by Agda is *intuitionistic* in nature:

  Negation is defined as $\neg A := A \to \bot$,
  and the law of excluded middle $A \vee \neg A$ is *not* provable

- *Consequence:* the logic we embed in Agda is also intuitionistic!
- e.g., we also cannot prove in Agda that, for any choice of QLTL formula ϕ and counterpart model M,

$$M \ \vDash \neg\neg\phi \Longleftrightarrow \phi.$$

- *Problem*: in temporal logic we usually define the essential operators of the logic, and then derive the other ones with negation:

$$\phi := \mathsf{true} \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \mathsf{Next}(\phi) \mid \phi_1 \,\mathsf{Until}\, \phi_2 \mid x = y \mid \exists x.\phi \mid P(x)$$

Embedding temporal logics in Agda

- The metalogic provided by Agda is *intuitionistic* in nature:

$$\text{Negation is defined as } \neg A := A \to \bot,$$
$$\text{and the law of excluded middle } A \vee \neg A \text{ is } \textit{not} \text{ provable}$$

- *Consequence:* the logic we embed in Agda is also intuitionistic!
- e.g., we also cannot prove in Agda that, for any choice of QLTL formula ϕ and counterpart model M,

$$M \ \vDash \neg\neg\phi \iff \phi.$$

- *Problem*: in temporal logic we usually define the essential operators of the logic, and then derive the other ones with negation:

$$\phi := \mathsf{true} \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \mathsf{Next}(\phi) \mid \phi_1 \,\mathsf{Until}\, \phi_2 \mid x = y \mid \exists x.\phi \mid P(x)$$

- In Agda, having $\wedge$ but not $\vee$ in QLTL is *not* equivalent to having both!

Embedding temporal logics in Agda

- The metalogic provided by Agda is *intuitionistic* in nature:

  Negation is defined as $\neg A := A \to \bot$,
  and the law of excluded middle $A \vee \neg A$ is *not* provable

- *Consequence:* the logic we embed in Agda is also intuitionistic!
- e.g., we also cannot prove in Agda that, for any choice of QLTL formula ϕ and counterpart model M,

$$M \ \vDash \neg\neg\phi \iff \phi.$$

- *Problem*: in temporal logic we usually define the essential operators of the logic, and then derive the other ones with negation:

$$\phi := \mathsf{true} \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \mathsf{Next}(\phi) \mid \phi_1 \, \mathsf{Until} \, \phi_2 \mid x = y \mid \exists x.\phi \mid P(x)$$

- In Agda, having $\wedge$ but not $\vee$ in QLTL is *not* equivalent to having both!
- *Problem:* it can be tricky to show contradictions with negated formulae

Positive normal forms

- *How can we tackle these issues?*

Positive normal forms

- *How can we tackle these issues?*
- Positive normal form (PNF):

*an extension of QLTL which can express **exactly** the same properties without using negation*

Positive normal forms

- *How can we tackle these issues?*
- Positive normal form (PNF):

*an extension of QLTL which can express **exactly** the same properties without using negation*

- Simplifies algorithms and model checking with temporal logics

Positive normal forms

- *How can we tackle these issues?*
- Positive normal form (PNF):

*an extension of QLTL which can express **exactly** the same properties without using negation*

- Simplifies algorithms and model checking with temporal logics
- Our approach:

1. Provide the logic in Agda with a *positive normal form*

Positive normal forms

- *How can we tackle these issues?*
- Positive normal form (PNF):

an extension of QLTL which can express ***exactly***
the same properties without using negation

- Simplifies algorithms and model checking with temporal logics
- Our approach:

1. Provide the logic in Agda with a *positive normal form*
2. *Separately* prove that the logic given is a PNF of the original logic

Positive normal forms

- *How can we tackle these issues?*
- Positive normal form (PNF):

an extension of QLTL which can express ***exactly***
the same properties without using negation

- Simplifies algorithms and model checking with temporal logics
- Our approach:

1. Provide the logic in Agda with a *positive normal form*
2. *Separately* prove that the logic given is a PNF of the original logic
3. Use LEM *only* in the PNF conversion proof, *not* when using the logic

Positive normal forms

- *How can we tackle these issues?*
- Positive normal form (PNF):

an extension of QLTL which can express ***exactly***
the same properties without using negation

- Simplifies algorithms and model checking with temporal logics
- Our approach:

1. Provide the logic in Agda with a *positive normal form*
2. *Separately* prove that the logic given is a PNF of the original logic
3. Use LEM *only* in the PNF conversion proof, *not* when using the logic

$\Rightarrow$ Guarantees that no extra expressivity is gained/lost

Positive normal forms

- *How can we tackle these issues?*
- Positive normal form (PNF):

*an extension of QLTL which can express **exactly** the same properties without using negation*

- Simplifies algorithms and model checking with temporal logics
- Our approach:

1. Provide the logic in Agda with a *positive normal form*
2. *Separately* prove that the logic given is a PNF of the original logic
3. Use LEM *only* in the PNF conversion proof, *not* when using the logic

$\Rightarrow$ Guarantees that no extra expressivity is gained/lost

$\Rightarrow$ Proving this theorem gives a program to *convert* formulae in PNF

Positive normal forms

- *How can we tackle these issues?*
- Positive normal form (PNF):

an extension of QLTL which can express ***exactly***
the same properties without using negation

- Simplifies algorithms and model checking with temporal logics
- Our approach:

1. Provide the logic in Agda with a *positive normal form*
2. *Separately* prove that the logic given is a PNF of the original logic
3. Use LEM *only* in the PNF conversion proof, *not* when using the logic

$\Rightarrow$ Guarantees that no extra expressivity is gained/lost

$\Rightarrow$ Proving this theorem gives a program to *convert* formulae in PNF

$\Rightarrow$ Directly prove another formula instead of working with contradiction

Conclusion and Future Work

In this work we present the categorical semantics of a counterpart-based temporal logic and formalize it in Agda using agda-categories.

- Many possible extensions of this work:

Conclusion and Future Work

In this work we present the categorical semantics of a counterpart-based temporal logic and formalize it in Agda using agda-categories.

- Many possible extensions of this work:
  - formalization of second-order QLTL to express set quantification

Conclusion and Future Work

In this work we present the categorical semantics of a counterpart-based temporal logic and formalize it in Agda using agda-categories.

- Many possible extensions of this work:
  - formalization of second-order QLTL to express set quantification
  - extending counterpart semantics to CTL, CTL* and their models

Conclusion and Future Work

In this work we present the categorical semantics of a counterpart-based temporal logic and formalize it in Agda using agda-categories.

- Many possible extensions of this work:
  - formalization of second-order QLTL to express set quantification
  - extending counterpart semantics to CTL, CTL* and their models
  - interfacing Agda with SMT solvers and model checkers for QLTL

Conclusion and Future Work

In this work we present the categorical semantics of a counterpart-based temporal logic and formalize it in Agda using agda-categories.

- Many possible extensions of this work:
  - formalization of second-order QLTL to express set quantification
  - extending counterpart semantics to CTL, CTL* and their models
  - interfacing Agda with SMT solvers and model checkers for QLTL
  - formalize syntax and models of the logic with indexed categories and morphisms between them, as in categorical logic [Jacobs, 2001]

Conclusion and Future Work

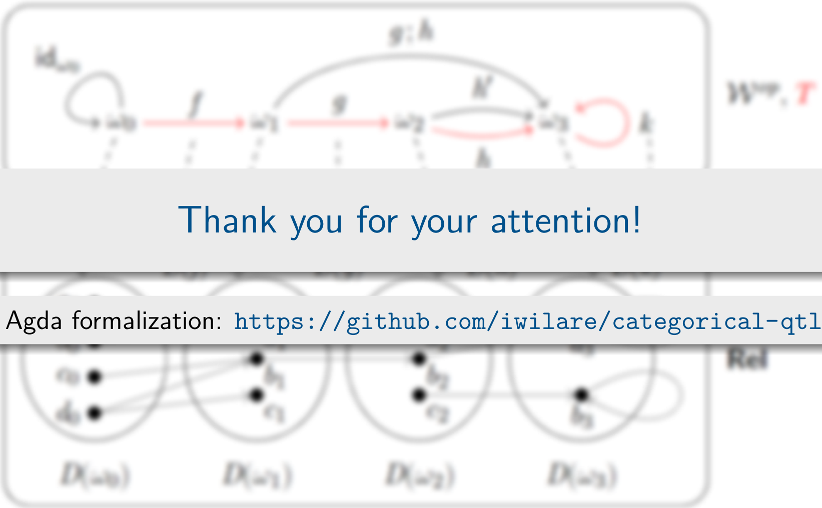
In this work we present the categorical semantics of a counterpart-based temporal logic and formalize it in Agda using agda-categories.

- Many possible extensions of this work:
  - formalization of second-order QLTL to express set quantification
  - extending counterpart semantics to CTL, CTL* and their models
  - interfacing Agda with SMT solvers and model checkers for QLTL
  - formalize syntax and models of the logic with indexed categories and morphisms between them, as in categorical logic [Jacobs, 2001]
- A study of formally-presented temporal logics is absent in the literature

Conclusion and Future Work

In this work we present the categorical semantics of a counterpart-based temporal logic and formalize it in Agda using agda-categories.

- Many possible extensions of this work:
  - formalization of second-order QLTL to express set quantification
  - extending counterpart semantics to CTL, CTL* and their models
  - interfacing Agda with SMT solvers and model checkers for QLTL
  - formalize syntax and models of the logic with indexed categories and morphisms between them, as in categorical logic [Jacobs, 2001]
- A study of formally-presented temporal logics is absent in the literature
- Formalizing and using category theory in proof assistants: [Carette, Hu, 2021], [Gross, Chlipala, Spivak, 2014]



Thank you for your attention!

Agda formalization: `https://github.com/iwilare/categorical-qtl`

PNF in Agda

- Formalized in Agda: PNF equivalence (using classical reasoning)

PNF in Agda

- Formalized in Agda: PNF equivalence (using classical reasoning)
- Two cases, using non-categorical semantics:

PNF in Agda

- Formalized in Agda: PNF equivalence (using classical reasoning)
- Two cases, using non-categorical semantics:
  - PNF with *partial functions* as counterpart relations

PNF in Agda

- Formalized in Agda: PNF equivalence (using classical reasoning)
- Two cases, using non-categorical semantics:
  - PNF with *partial functions* as counterpart relations
  - PNF with general relations (i.e. allow duplication of entities)

PNF in Agda

- Formalized in Agda: PNF equivalence (using classical reasoning)
- Two cases, using non-categorical semantics:
  - PNF with *partial functions* as counterpart relations
  - PNF with general relations (i.e. allow duplication of entities)
  - Expansion laws and equivalences in QLTL in both settings

Positive normal forms for QLTL

- PNF for QLTL:

$$\psi := \mathsf{true} \mid x = y \mid P(x), \quad \phi := \psi \mid \neg\psi \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid \exists x.\phi \mid \forall x.\phi$$

Positive normal forms for QLTL

- PNF for QLTL:

$$\psi := \mathsf{true} \mid x = y \mid P(x), \quad \phi := \psi \mid \neg\psi \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid \exists x.\phi \mid \forall x.\phi$$
$$\mid \mathsf{Next}(\phi) \mid \phi_1 \mathsf{Until} \phi_2 \mid \phi_1 \mathsf{WUntil} \phi_2$$

Positive normal forms for QLTL

- PNF for QLTL:

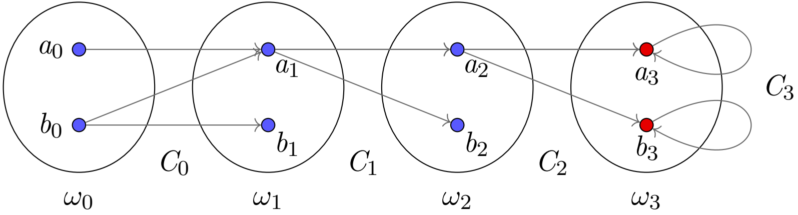
$$\psi := \mathsf{true} \mid x = y \mid P(x), \quad \phi := \psi \mid \neg\psi \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid \exists x.\phi \mid \forall x.\phi$$
$$\mid \mathsf{Next}(\phi) \mid \phi_1 \mathsf{Until} \phi_2 \mid \phi_1 \mathsf{WUntil} \phi_2 \mid \underline{\mathsf{NextF}}(\phi) \mid \phi_1 \underline{\mathsf{UntilF}} \phi_2 \mid \phi_1 \underline{\mathsf{Then}} \phi_2$$

Positive normal forms for QLTL

- PNF for QLTL:

$$\psi := \mathsf{true} \mid x = y \mid P(x), \quad \phi := \psi \mid \neg\psi \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid \exists x.\phi \mid \forall x.\phi$$

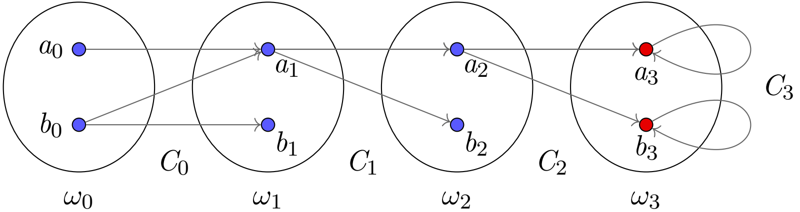
$$\mid \mathsf{Next}(\phi) \mid \phi_1 \mathsf{Until} \phi_2 \mid \phi_1 \mathsf{WUntil} \phi_2 \mid \underline{\mathsf{NextF}}(\phi) \mid \phi_1 \underline{\mathsf{UntilF}} \phi_2 \mid \phi_1 \underline{\mathsf{Then}} \phi_2$$



Positive normal forms for QLTL

- PNF for QLTL:

$$\psi := \text{true} \mid x = y \mid P(x), \quad \phi := \psi \mid \neg\psi \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid \exists x.\phi \mid \forall x.\phi$$
$$\mid \mathsf{Next}(\phi) \mid \phi_1 \mathsf{Until} \phi_2 \mid \phi_1 \mathsf{WUntil} \phi_2 \mid \underline{\mathsf{NextF}}(\phi) \mid \phi_1 \underline{\mathsf{UntilF}} \phi_2 \mid \phi_1 \underline{\mathsf{Then}} \phi_2$$

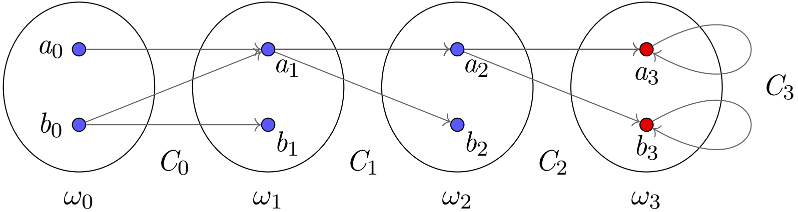


- $\neg\mathsf{Next}(\phi) \equiv \mathsf{NextF}(\neg\phi)$
- $\neg(\phi_1 \mathsf{Until} \phi_2) \equiv (\neg\phi_2)\mathsf{Then}(\neg\phi_1 \wedge \neg\phi_2)$
- $\neg(\phi_1 \mathsf{WUntil} \phi_2) \equiv (\neg\phi_2)\mathsf{UntilF}(\neg\phi_1 \wedge \neg\phi_2)$

Positive normal forms for QLTL

- PNF for QLTL:

$$\psi := \text{true} \mid x = y \mid P(x), \quad \phi := \psi \mid \neg\psi \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid \exists x.\phi \mid \forall x.\phi$$
$$\mid \mathsf{Next}(\phi) \mid \phi_1 \mathsf{Until} \phi_2 \mid \phi_1 \mathsf{WUntil} \phi_2 \mid \underline{\mathsf{NextF}}(\phi) \mid \phi_1 \underline{\mathsf{UntilF}} \phi_2 \mid \phi_1 \underline{\mathsf{Then}} \phi_2$$



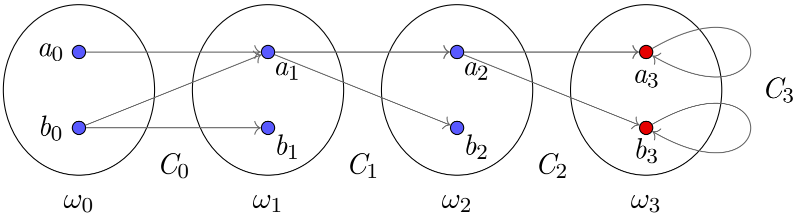
- $\neg\mathsf{Next}(\phi) \equiv \mathsf{NextF}(\neg\phi)$
- $\neg(\phi_1 \mathsf{Until} \phi_2) \equiv (\neg\phi_2)\mathsf{Then}(\neg\phi_1 \wedge \neg\phi_2)$
- $\neg(\phi_1 \mathsf{WUntil} \phi_2) \equiv (\neg\phi_2)\mathsf{UntilF}(\neg\phi_1 \wedge \neg\phi_2)$
- $b_0 \vDash_{\omega_0} \mathsf{NextF}(\mathsf{Blue}(x))$

Positive normal forms for QLTL

- PNF for QLTL:

$$\psi := \mathsf{true} \mid x = y \mid P(x), \quad \phi := \psi \mid \neg\psi \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid \exists x.\phi \mid \forall x.\phi$$

$$\mid \mathsf{Next}(\phi) \mid \phi_1 \mathsf{Until} \phi_2 \mid \phi_1 \mathsf{WUntil} \phi_2 \mid \underline{\mathsf{NextF}}(\phi) \mid \phi_1 \underline{\mathsf{UntilF}} \phi_2 \mid \phi_1 \underline{\mathsf{Then}} \phi_2$$



- $\neg\mathsf{Next}(\phi) \equiv \mathsf{NextF}(\neg\phi)$
- $\neg(\phi_1 \mathsf{Until} \phi_2) \equiv (\neg\phi_2)\mathsf{Then}(\neg\phi_1 \wedge \neg\phi_2)$
- $\neg(\phi_1 \mathsf{WUntil} \phi_2) \equiv (\neg\phi_2)\mathsf{UntilF}(\neg\phi_1 \wedge \neg\phi_2)$

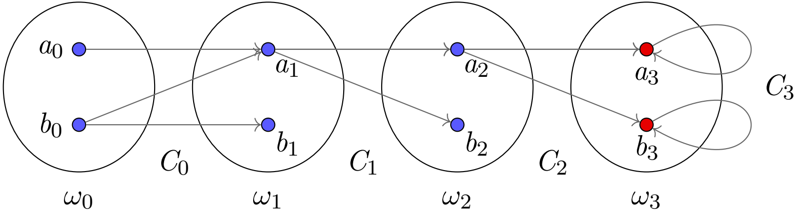
- $b_0 \vDash_{\omega_0} \mathsf{NextF}(\mathsf{Blue}(x))$
- $b_1 \vDash_{\omega_1} \mathsf{NextF}(\mathsf{Blue}(x))$

Positive normal forms for QLTL

- PNF for QLTL:

$$\psi := \mathsf{true} \mid x = y \mid P(x), \quad \phi := \psi \mid \neg\psi \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid \exists x.\phi \mid \forall x.\phi$$

$$\mid \mathsf{Next}(\phi) \mid \phi_1 \mathsf{Until} \phi_2 \mid \phi_1 \mathsf{WUntil} \phi_2 \mid \underline{\mathsf{NextF}}(\phi) \mid \phi_1 \underline{\mathsf{UntilF}} \phi_2 \mid \phi_1 \underline{\mathsf{Then}} \phi_2$$



- $\neg\mathsf{Next}(\phi) \equiv \mathsf{NextF}(\neg\phi)$
- $\neg(\phi_1 \mathsf{Until} \phi_2) \equiv (\neg\phi_2)\mathsf{Then}(\neg\phi_1 \wedge \neg\phi_2)$
- $\neg(\phi_1 \mathsf{WUntil} \phi_2) \equiv (\neg\phi_2)\mathsf{UntilF}(\neg\phi_1 \wedge \neg\phi_2)$

- $b_0 \vDash_{\omega_0} \mathsf{NextF}(\mathsf{Blue}(x))$
- $b_1 \vDash_{\omega_1} \mathsf{NextF}(\mathsf{Blue}(x))$
- $a_0 \vDash_{\omega_0} \mathsf{Blue}(x)\mathsf{UntilF}\,\mathsf{Red}(x)$