Rational Codata as Syntax-with-Binding

Correct-by-Construction Foundations of the Modal μ-Calculus

Sean Watters

University of Strathclyde

9th June 2025

Plan

We will:

1. Introduce the μ-calculus, and its Fischer-Ladner closure.
2. Sketch our (now complete!) formalised proof of the closure's finiteness.
3. Discuss the presentation of rational cotrees as syntax with binding, its role in the proof, and future plans in this direction.
4. Aim to keep it high-level and skip the gory details!

The Modal μ-Calculus

Syntax: For all propositional atoms $a \in \mathsf{At}$ and variable names $x \in \mathsf{Var}$:

$$\varphi := a \mid \neg a \mid x \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Box\varphi \mid \Diamond\varphi \mid \mu x.\varphi \mid \nu x.\varphi$$

Notes:

- The fixpoint operators μ and ν are variable binders.
- The syntax is strictly positive — this matters when giving semantics to fixpoint formulas.

The Modal μ-Calculus

Syntax: For all propositional atoms $a \in \mathsf{At}$ and variable names $x \in \mathsf{Var}$:

$$\varphi := a \mid \neg a \mid x \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Box\varphi \mid \Diamond\varphi \mid \mu x.\varphi \mid \nu x.\varphi$$

Notes:

- The fixpoint operators μ and ν are variable binders.
- The syntax is strictly positive — this matters when giving semantics to fixpoint formulas.

Semantics:

- Kripke semantics, where μ and ν let us reason about unbounded/infinite behaviour.
- Satisfiability and model checking are decidable.
- The μ-calculus subsumes temporal and dynamic logics such as LTL, CTL*, and PDL.

Fixpoint Unfolding

At the heart of the μ-calculus is the semantic equivalence:

$$\eta x.\ \varphi \ \equiv \ \varphi[x := \eta x.\varphi]$$

We call $\varphi[x := \eta x.\varphi]$ the **unfolding** of $\eta x.\varphi$.

For example: let $E(p) := \mu x.\ p \vee \Diamond x$. Then:

Fixpoint Unfolding

At the heart of the μ-calculus is the semantic equivalence:

$$\eta x.\ \varphi \ \equiv\ \varphi[x := \eta x.\varphi]$$

We call $\varphi[x := \eta x.\varphi]$ the **unfolding** of $\eta x.\varphi$.

For example: let $E(p) := \mu x.\ p \vee \Diamond x$. Then:

$$\begin{aligned} & E(p) \\ \equiv\ & p \vee \Diamond(E(p)) \\ \equiv\ & p \vee \Diamond(p \vee \Diamond(E(p))) \\ \equiv\ & \dots \end{aligned}$$

The Closure

Definition

The **closure** of a formula φ is the minimal set which contains φ, and is closed under taking unfoldings of fixpoint formulas, and direct subformulas of non-fixpoint formulas.

In other words, it is the minimal set C satisfying:

$$\begin{gathered}\varphi \in C \\ \bigcirc\varphi \in C \Rightarrow \varphi \in C, \text{ where } \bigcirc \in \{\Box, \Diamond\} \\ \varphi \star \psi \in C \Rightarrow \varphi \in C \text{ and } \psi \in C, \text{ where } \star \in \{\wedge, \vee\} \\ \eta x.\varphi \in C \Rightarrow \varphi[x := \mu x.\varphi] \in C, \text{ where } \eta \in \{\mu, \nu\}\end{gathered}$$

Finiteness of the Closure (1)

Theorem

For all φ, the closure of φ is finite. (Kozen, 1983)

Proof Sketch (Kozen):

1. Define the *expansion* of a formula as the sequential instantiation of all its free variables.
2. Define an alternative, structurally inductive procedure for computing the closure via the expansion map.
3. Prove the alternative definition correct by induction.

Not so simple in a formal setting!

Finiteness of the Closure (2)

Proof Sketch (Our Approach in Agda):

1. Implement the closure coinductively as a non-wellfounded cotree, via the standard definition. (Trivially correct).

Finiteness of the Closure (2)

Proof Sketch (Our Approach in Agda):

1. Implement the closure coinductively as a non-wellfounded cotree, via the standard definition. (Trivially correct).
2. Implement the alternative definition of the closure as an inductive syntax-with-binding representation of a rational cotree (a “tree with back-edges”). (Finite by construction).

Finiteness of the Closure (2)

Proof Sketch (Our Approach in Agda):

1. Implement the closure coinductively as a non-wellfounded cotree, via the standard definition. (Trivially correct).
2. Implement the alternative definition of the closure as an inductive syntax-with-binding representation of a rational cotree (a “tree with back-edges”). (Finite by construction).
3. Define the “unfolding” of such a tree-with-back-edges to a cotree.

Finiteness of the Closure (2)

Proof Sketch (Our Approach in Agda):

1. Implement the closure coinductively as a non-wellfounded cotree, via the standard definition. (Trivially correct).
2. Implement the alternative definition of the closure as an inductive syntax-with-binding representation of a rational cotree (a "tree with back-edges"). (Finite by construction).
3. Define the "unfolding" of such a tree-with-back-edges to a cotree.
4. Prove the coinductive definition closure bisimilar to the unfolding of the inductive approximation.

Finiteness of the Closure (2)

Proof Sketch (Our Approach in Agda):

1. Implement the closure coinductively as a non-wellfounded cotree, via the standard definition. (Trivially correct).
2. Implement the alternative definition of the closure as an inductive syntax-with-binding representation of a rational cotree (a "tree with back-edges"). (Finite by construction).
3. Define the "unfolding" of such a tree-with-back-edges to a cotree.
4. Prove the coinductive definition closure bisimilar to the unfolding of the inductive approximation.
5. Prove that we can transport the correctness proofs across the bisimulation, to show that the finite-by-construction algorithm does in fact compute the closure.

Non-Wellfounded Cotrees

```
mutual
  record ∞NWFTree (X : Set) : Set where
    coinductive
    field
      head : X
      subtree : NWFTree X

  data NWFTree (X : Set) : Set where
    leaf : NWFTree X
    node1 : ∞NWFTree X → NWFTree X
    node2 : ∞NWFTree X → ∞NWFTree X → NWFTree X
    nodeη : ∞NWFTree X → NWFTree X
```

Rational Trees

Key Insight: Variables are pointers to their binders! (Ghani, Hamana, Uustalu & Vene, 2006).

```
mutual
  data RTree (X : Set) (n : ℕ) : Set where
    step : (x : X) → (t : RTree-step X n) → RTree X n
    var : (x : Fin n) → RTree X n

  data RTree-step (X : Set) (n : ℕ) : Set where
    leaf : RTree-step X n
    node1 : RTree X n → RTree-step X n
    node2 : RTree X n → RTree X n → RTree-step X n
    nodeη : RTree X (suc n) → RTree-step X n

data Scope (X : Set) : ℕ → Set where
  []   : Scope X zero
  _::_ : ∀ {n} → (t : RTree X n) → {{_ : NonVar t}}
       → (Γ₀ : Scope X n) → Scope X (suc n)
```

Unfolding Trees

```
head : ∀ {X n} → (Γ : Scope X n) → RTree X n → X
head Γ (step x t)         = x
head (t :: Γ) (var zero) = head Γ t
head (t :: Γ) (var (suc x)) = head Γ (var x)

mutual
  unfold : ∀ {X n} → (Γ : Scope X n) → RTree X n → ∞NWFTree X
  unfold Γ t .∞NWFTree.head = head Γ t
  unfold Γ t .∞NWFTree.subtree = unfold-subtree Γ t

  unfold-subtree : ∀ {X n} → (Γ : Scope X n) → RTree X n → NWFTree X
  unfold-subtree Γ (step x leaf)          = leaf
  unfold-subtree Γ (step x (node1 t)) = node1 (unfold Γ t)
  unfold-subtree Γ (step x (node2 tl tr)) = node2 (unfold Γ tl) (unfold Γ tr)
  unfold-subtree Γ (step x (nodeη t)) = nodeη (unfold ((step x (nodeη t)) :: Γ) t)
  unfold-subtree (t :: Γ) (var zero)      = unfold-subtree Γ t
  unfold-subtree (t :: Γ) (var (suc x)) = unfold-subtree Γ (var x)
```

Progress

Theorem

The direct coinductive definition of the closure is bisimilar to the unfolding of the inductive syntax-with-binding definition.

Theorem

Let T be the tree with back-edges produced by the inductive closure algorithm applied to φ. Then for all formulas ψ, there is a path to ψ in T iff there is a path to ψ in the closure of ϕ. That is, T really is the closure of φ.

To-do/future work:

- Tighten the size bounds.
- What's the general, abstract story about this presentation of rational codata? (Rational fixpoint of containers??)

Thanks!

References:

- *Results on the Propositional μ-Calculus.* Kozen, 1983.
- *Representing Cyclic Structures as Nested Datatypes.* Ghani, Hamana, Uustalu & Vene, 2006.