

A zoo of partial continuity properties

Jean CASPAR

Internship with Yannick FORSTER

Cambium, INRIA Paris

Continuity principle

Consider a function $F : (Q \rightarrow A) \rightarrow R$ in Rocq, for arbitrary datatypes Q , A and R :

- Given an input function $f : Q \rightarrow A$, it may call f only a finite number of time, because it is computable and it terminates.
- This property can be seen as a continuity property.
- There are several way to formalise this property in Rocq.

Moduli

Consider these functions:

Definition F1 ($f : \text{nat} \rightarrow \text{nat}$) : $\text{nat} := f\ 0$.

Definition F2 ($f : \text{nat} \rightarrow \text{bool}$) : $\text{bool} := \text{if } f\ 0 \text{ then } f\ 1 \text{ else } f\ 2$.

Definition F3 ($f : \text{nat} \rightarrow \text{nat}$) : $\text{nat} := f\ (f\ 0)$.

The values passed to f are:

- 1 [0]
- 2 [0, 1] if $f(0) = \text{true}$ and [0, 2] if $f(0) = \text{false}$.
- 3 [0, $f(0)$]

If g is another function which coincide with f on these lists, $F(f) = F(g)$.

Definition (Modulus)

$I : \mathbb{L} Q$ is a modulus¹ of $F : (Q \rightarrow A) \rightarrow R$ at $f : Q \rightarrow A$ if

$$\forall g : Q \rightarrow A, (\forall q \in I, f(q) = g(q)) \rightarrow F(f) = F(g)$$

Definition (Modulus continuous)

$F : (Q \rightarrow A) \rightarrow R$ is modulus continuous if there is a function $M : (Q \rightarrow A) \rightarrow \mathbb{L} Q$ such that $M(f)$ is a modulus for F at f for each f .

¹Troelstra and Dalen [1988]

Moduli

Definition F1 (f : nat -> nat) : nat := f 0.

Definition F2 (f : nat -> bool) : bool := if f 0 then f 1 else f 2.

Definition F3 (f : nat -> nat) : nat := f (f 0).

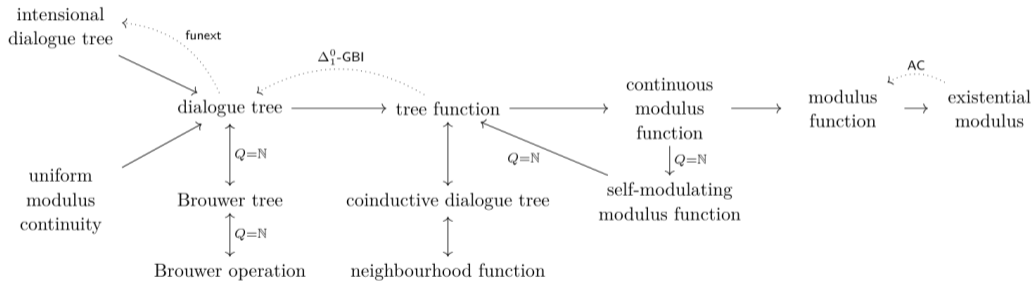
Definition M1 (f : nat -> nat) : list nat := [0].

Definition M2 (f : nat -> bool) : list nat :=
if f 0 then [0; 1] else [0; 2].

Definition M3 (f : nat -> nat) : list nat := [0; f 0].

Program

- Baillon, Forster, Mahboubi, Pédrot and Piquerez [2025] classified several definitions of continuity.



■ **Figure 1** Overview of implications. Dotted arrows indicate that axioms are used.

- I did the same, but for partial computable functions $(Q \multimap A) \multimap R$.
- We work in the type theory of Rocq, therefore with constructive logic.

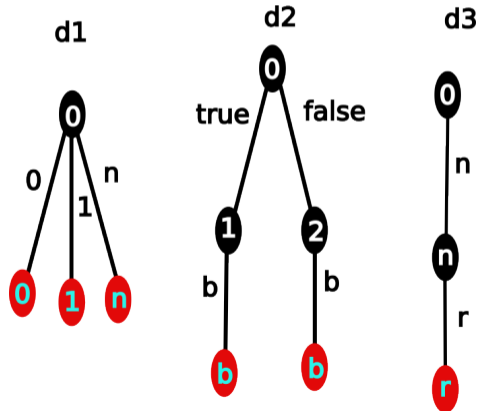
Dialogue tree

The programs we defined are “sequential”: the calls to f are made one after the other. This allows to see $F(f)$ as a program with access to an oracle for f . A further calls may depend on a precedent one, so 2nd order programs have a tree structure:

Definition F1 ($f : \text{nat} \rightarrow \text{nat}$) : $\text{nat} :=$
 f 0.

Definition F2 ($f : \text{nat} \rightarrow \text{bool}$) : $\text{bool} :=$
 if f 0 then f 1 else f 2.

Definition F3 ($f : \text{nat} \rightarrow \text{nat}$) : $\text{nat} :=$
 f (f 0).



Dialogue tree

Formally, we define a type of tree, dialogue trees², with leaves labelled by R and nodes labelled with Q that have A children:

```
Inductive dialogue_tree (Q A R : Type) :=  
  | Output : R -> dialogue_tree Q A R  
  | Ask : Q -> (A -> dialogue_tree Q A R).
```

We can evaluate a dialogue tree d : $\partial_d : (Q \rightarrow A) \rightarrow R$.

Definition (Dialogue tree-continuous)

F is dialogue tree-continuous if there exists a dialogue tree d such that for all f , $\partial_d(f) = F(f)$.

²Escardó [2013]

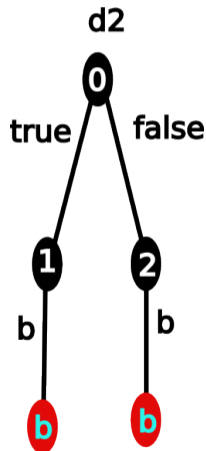
Dialogue tree

Suppose that that we want to evaluate $f : \mathbb{N} \rightarrow \mathbb{B}$ defined by $f(n) := n \bmod 2 \stackrel{?}{=} 0$.

```
Fixpoint eval (d : dialogue_tree) (f : Q -> A) : R :=  
  match d with  
  | Output r => r  
  | Ask q k => eval k (f q)  
  end.
```

```
Definition F2 (f : nat -> bool) : bool :=  
  if f 0 then f 1 else f 2.
```

- 1 We have $f(0) = \text{true}$, so we look at the true branch of d_2 .
- 2 We have $f(1) = \text{false}$. We go to the false branch of the node "1".
- 3 We arrive at an Output false node: we return false.

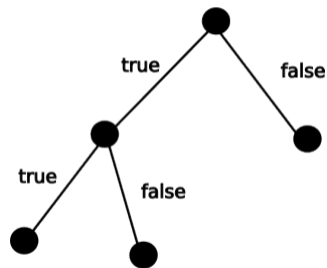


Extensional trees

Trees can also be presented differently that as an inductive data-structure: as a set of paths.

A binary tree T is represented as the set of paths along it. It is a function $\mathbb{L}\mathbb{B} \rightarrow \mathbb{B}$ that is:

- Prefix-closed: $T(l' \# l) = \text{true} \rightarrow T(l) = \text{true}$;
- Non-empty: $T[] = \text{true}$;
- Well-founded:
 $\forall (f : \mathbb{N} \rightarrow A), \exists (n : \mathbb{N}), T[f(n-1), \dots, f(0)] = \text{false}$.



$$T(l) := l \in \{[], [\text{true}], [\text{true}; \text{true}], [\text{false}; \text{true}], [\text{false}]\}.$$

Function tree

Dialogue trees are more complex: nodes and leaves holds data, so we will use a tree function³ $\tau : \mathbb{L} A \rightarrow Q + R$.

- $\tau(l) = \text{inl } q$ means that there the path l ends in a node labelled with q .
- $\tau(l) = \text{inr } r$ means that the path l goes through a leaf labelled with r .

Using the fact that τ is well-founded, we can evaluate a well-founded tree function τ :
 $\partial_\tau : (Q \rightarrow A) \rightarrow R$.

Definition (Tree function-continuity)

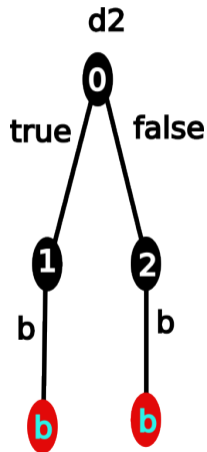
F is tree function-continuous if there exists a tree function τ such that for all f ,
 $\partial_\tau(f) = F(f)$.

³Baillon et al. [2025]

Function tree

```
Definition F2 (f : nat -> bool) : bool :=  
  if f 0 then f 1 else f 2.
```

```
Definition tau : list bool -> nat + bool :=  
  fun l => match l with  
  | [] => inl 0  
  | [b] => if b then inl 1 else inl 2  
  | l => l[1]  
  end.
```

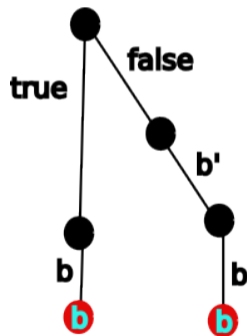


Neighborhood functions

If $Q = \mathbb{N}$, we can ask τ to ask questions in order: first $f(0)$, then $f(1)$, then $f(2)$, etc. This gives the notion of neighborhood function⁴ $\gamma : \mathbb{L}A \rightarrow \mathbb{O}R$.

```
Definition F2 (f : nat -> bool) : bool :=  
  if f 0 then f 1 else f 2.
```

```
Definition gamma : list bool -> option nat :=  
  fun l => match l with  
  | [] => None  
  | [_] => None  
  | [_; false] => None  
  | l => if l[0] then l[1] else l[2]  
  end.
```



⁴Kleene [1962]

Classification

Consider two different continuity definitions $C_1(F)$ and $C_2(F)$. We want to classify them.

- Sometimes, $\forall F : (Q \rightarrow A) \rightarrow R, C_1(F) \rightarrow C_2(F)$ is true.
- Sometimes, $\neg \forall F : (Q \rightarrow A) \rightarrow R, C_1(F) \rightarrow C_2(F)$ is true.
- Sometimes, neither of them are provable.

Classification

How can we prove that a formula is unprovable?

- With proof theory.
- By using models.
- By showing they imply a known unprovable formula.

We want to work inside Rocq to prove results about Rocq functions.

We will use the latter method.

Constructive reverse mathematics

We look for formulas P and Q such that:

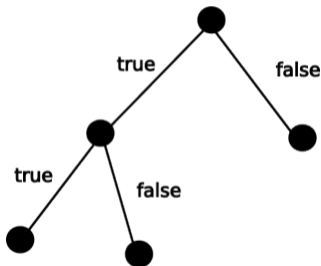
- P and Q are known to be unprovable;
- It is consistent to suppose P and Q ;
- $P \rightarrow (\forall F, C_1(F) \rightarrow C_2(F))$;
- $(\forall F, C_1(F) \rightarrow C_2(F)) \rightarrow Q$.

We will try to find P such that $P \iff (\forall F, C_1(F) \rightarrow C_2(F))$.

For example, ZF, “every vector space has a basis” is equivalent to AC, so it is unprovable but consistent.

Bar induction

We can turn every intensional tree into an extensional tree.



$$T(I) := I \in \{[], [\text{true}], [\text{true}; \text{true}], [\text{false}; \text{true}], [\text{false}]\}.$$

However, the converse is unprovable. It is equivalent to a statement called “bar induction”⁵. The statement for binary trees is called “fan theorem”.

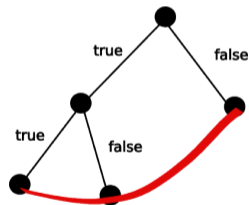
⁵Troelstra and Dalen [1988]

Bar induction

Formally, it states that every “bar” is “inductive”. It is often restricted to decidable or monotonic bars.

A bar is a predicate $P : \mathbb{L} A \rightarrow \mathbb{P}$ such that for every infinite path $f : \mathbb{N} \rightarrow A$,

$$\exists (n : \mathbb{N}), P[f(n-1), \dots, f(0)].$$



A bar is the border of a well-founded tree. Every path that is long enough must leave the tree, and meet the bar.

An inductive bar is the border of an inductive tree.

```
Inductive inductive_bar (P : list A -> Prop) : list A -> Prop :=
| Base (l : list A) : P l -> inductive_bar P l
| Hered (l : list A) : (forall (a : A), inductive_bar P (a :: l))
  -> inductive_bar P l.
```

Brouwer operation

- A Brouwer operation⁶ is an “inductively defined” neighborhood function.
- A neighborhood $\gamma : \mathbb{L} A \rightarrow \mathbb{O} R$ is a Brouwer operation if $\exists r, \gamma([]) = \text{Some } r$ or if $\gamma([]) = \text{None}$ and for all $a : A, \lambda l. \gamma(a :: l)$ is a Brouwer operation.
- $P(l) := \exists r, \gamma(l) = \text{Some } r$ is a decidable bar.
- The following are equivalent:
 - Every neighborhood function is a Brouwer operation.
 - Every neighborhood-continuous function is Brouwer operation-continuous.
 - Decidable bar induction holds.

⁶Troelstra and Dalen [1988]

Internship

- We will now look at my internship.
- I adapted the continuity definitions to partial functions $F : (Q \multimap A) \multimap R$, and classified them.

Partiality

We are interested in computable partial functions, as in pure OCaml functions. If $F : (Q \rightarrow A) \rightarrow R$ is partial and computable, and $F(f)$ terminates, there must also have been a finite number of calls to f .

This work is parameterized by a partiality monad \mathcal{P} , equipped with:

- A deterministic relation $\downarrow : \mathcal{P} A \rightarrow A \rightarrow \mathbb{P}$;
- A step evaluation function $\mathcal{P} A \rightarrow \mathbb{N} \rightarrow \mathbb{O}A$;
- A constant `undef` : $\mathcal{P} A$;
- The minimizer $\mu : (\mathbb{N} \rightarrow \mathbb{B}) \rightarrow \mathbb{N}$;
- Various axioms specifying them.

A monad implementing this specification is the monad mapping a type A to the type of monotonic functions $\mathbb{N} \rightarrow \mathbb{O}A$.

Partial dialogue tree

The partial version of dialogue trees should be straightforward:

```
Inductive dialogue_tree (Q A R : Type) :=  
  | Output : R -> dialogue_tree Q A R  
  | Ask : Q -> (A -> partial (dialogue_tree Q A R))  
    -> dialogue_tree Q A R.
```

- The children of the trees may be undefined.
- But this is rejected by Rocq! The partiality monad is abstract, so Rocq cannot look through it to see that it is strictly positive.
- The existing implementation of the partiality monad also doesn't work. I had to implement the delay monad⁷, defined through a coinductive type.

⁷Capretta [2005]

Delay monad

```
Inductive delayF (A T : Type) : Type :=  
| DelayF : T -> delayF A T  
| NowF : A -> delayF A T.
```

```
CoInductive delay (A : Type) := {  
  delay_go : delayF A (delay A);  
}.
```

- Values of type $\text{delay } A$ are of the form $\text{Delay}(\text{Delay}(\dots))$ with potentially an infinite number of Delay , or a $\text{Now } a$ at the end.
- It is isomorphic to monotonic functions $\mathbb{N} \rightarrow \mathbb{O} A$, but it is strictly positive.
- I did not have the time to look further at partial dialogue trees.

Partial moduli

Given $F : (Q \multimap A) \multimap R$, $f : Q \multimap A$ and $r : R$ with $F(f) \Downarrow r$, there are several definitions of moduli in the partial case. I is a modulus of $F(f)$ if:

Weak $\forall g, (\forall q \in I, \exists a, f(q) \Downarrow a \wedge g(q) \Downarrow a) \rightarrow F(g) \Downarrow r$

Extensional $\forall g, (\forall q \in I, f(q) \approx g(q)) \rightarrow F(g) \Downarrow r$

Monotonic $\forall g, (\forall q \in I, \forall a, f(q) \Downarrow a \rightarrow g(q) \Downarrow a) \rightarrow F(g) \Downarrow r$

Strong f terminates on every q in I + any of the above (all equivalent)

Partial modulus continuity

There are also various way for a function to have a modulus at each input on which it terminates:

Existential modulus F has a modulus l at each f such that $F f \Downarrow$;

Modulus There exists a function $M : (Q \multimap A) \multimap \mathbb{L} Q$ computing F 's modulus out of f when $F f \Downarrow$;

Continuous modulus The function M itself has a modulus where it is defined;

Self-modulating modulus The function M computes its own modulus, ie. if $F f \Downarrow$ and $M f \Downarrow l$ then l is modulus for both F and M at f .

Each “modulus” in these definitions can be taken as any of the moduli definitions:

$4 + 4 + (4 \times 4) + (4 \times 4) = 40$ definitions!

Some modular analysis

If we leave the world of Rocq function and assumes additional principles, like computational LPO— $\forall p : \mathcal{P} A, \{ p \downarrow \} + \{ \neg p \downarrow \}$ —then we can conflate or separate some principles:

- We can transform a monotonic modulus into a strong one. However, this process is not continuous.
- $F(f) \downarrow \text{true} \iff (f(0) \text{ does not terminate})$ has an extensional modulus computed by a strongly self-modulating function, but is non-monotonic.

Some modular analysis

The function $F : (\mathbb{B} \rightarrow \mathbb{1}) \rightarrow \mathbb{1}$, which terminates if $f(\text{true})$ or $f(\text{false})$ terminates, has:

- A monotonic self-modulating monotonic modulus;
- A monotonic modulus that has a strong modulus;
- A strong modulus;
- But no strong modulus that has a monotonic modulus.

F is defined through the “parallel or”.

$\mathcal{P}A$ has a the structure of ω -cpo: $p \preceq q \iff \forall a, p \not\downarrow a \rightarrow q \not\downarrow a$. Monotonic sequences have a supremum, and ω -continuous functions preserves them.

ω -continuity lies in the hierarchy of modulus definitions for $Q = \mathbb{N}$:
Strong continuity $\rightarrow \omega$ -continuity \rightarrow monotonic modulus.

Partial tree functions

Tree function $\tau : \mathbb{L} A \rightarrow Q + R$ and neighborhood function $\gamma : \mathbb{L} A \rightarrow \mathbb{O} R$ can be made partial easily.

In the total case, they are equivalent for $Q = \mathbb{N}$. In the partial case, neighborhood-continuity implies strictly tree function-continuity:

Definition $F (f : \text{nat} \rightarrow \text{nat}) : \text{partial nat} := f 1$.

This function is tree function-continuous but not neighborhood-continuous:

- A tree function may ask $f(1)$ directly.
- A neighborhood function must ask questions in order, so it should ask $f(0)$, which could be undefined, before $f(1)$.

Tree function-continuity implies strictly strong self-modulating strong modulus-continuity.

Partial bar induction

- $\exists r, \gamma(I) \not\downarrow$ Some r is not decidable. It is decidable, however, if $\gamma(I) \not\downarrow$.
- In the partial case, we only have:

$$\forall (f : \mathbb{N} \rightarrow A), F(f) \not\downarrow \rightarrow \exists (n : \mathbb{N})(r : R), \\ (\forall i < n, f(i) \not\downarrow) \wedge \gamma([f(n-1), \dots, f(0)])$$

- We consider the notion of T -bar, for T a “continuous semi-computable tree”: a function F and its neighborhood γ . $T(f)$ means $F(f) \not\downarrow$, $T(I)$ means $\gamma(I) \not\downarrow$.
- I adapted the definition of inductive bars to T -bars: every lists that appears is “guarded” by a T .

Definition (Partial bar induction)

Every T -decidable monotonic T -bar is T -inductive.

Strictness

- We must suppose that γ is strict: if $\gamma(I)$ terminates, $\exists I' \exists r$ such that $\gamma(I' \# I) \not\downarrow$ Some r .
- This is a kind of well-foundedness: to check that $T(f)$, we only need to check a finite amount of data.
- We suppose that $\mathbb{N} \rightarrow \mathbb{L} A$, which is true for $A = \mathbb{N}$.
- This allows to turn every neighborhood function into a strict one.

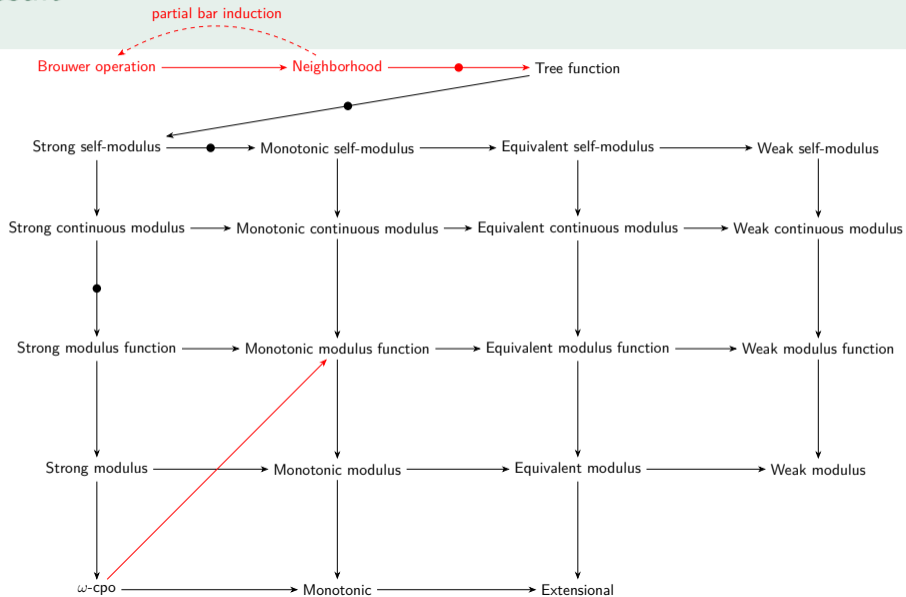
Partial Brouwer operation

Theorem

For all $F : (\mathbb{N} \rightarrow A) \rightarrow R$, F being neighborhood-continuous implying F being Brouwer operation continuous is equivalent to partial bar induction.

- Brouwer operations are relativised to a tree.
- F is Brouwer operation-continuous if there exists a neighborhood function γ which is a Brouwer operation for the tree $T := (F, \gamma)$.
- Partial bar induction should be implied by AC and implies usual bar induction.

Final result



Conclusion

- Continuity definitions:
 - Idea: a function $F(f)$ may call its input only finitely many times.
 - 3 classes of formalisations: moduli, extensional and intensional trees.
- Classification:
 - “Sandwiching” propositions between known consistent and unprovable propositions.
 - Finding equivalences: constructive reverse mathematics.
- Bar induction: Principle that allows to turn extensional trees into intensional trees.
- I used these to study the notions of continuity for partial computable functions.
- Future work: replace partiality by arbitrary effects.

Baire space

Definition (Continuity)

A function $f : X \rightarrow Y$ is continuous if $f^{-1}(U)$ is open if U is open.

Definition (Baire space)

The Baire space is the set $\mathbb{N}^{\mathbb{N}}$ with the product topology.

Theorem

2nd order functions definable in Heyting arithmetic can be interpreted as continuous functions from the Baire space to \mathbb{N} . These are exactly the modulus-continuous functions.