Computational Back-and-Forth Arguments in Constructive Type Theory

A Proof Pearl

Dominik Kirst
Saarland University, Saarland Informatics Campus

ITP'22, Haifa, Israel, August 9th

What is the Back-and-Forth Method?[1]

[1] See Silver's "Who invented Cantor's back-and-forth argument?" (1994) for a historic overview.

What is the Back-and-Forth Method?[1]

Back-and-forth method

From Wikipedia, the free encyclopedia

In mathematical logic, especially set theory and model theory, the **back-and-forth method** is a method for showing isomorphism between countably infinite structures satisfying specified conditions. In particular it can be used to prove that

- any two countably infinite densely ordered sets (i.e., linearly ordered in such a way that between any two members there is another) without endpoints are isomorphic. An isomorphism between linear orders is simply a strictly increasing bijection. This result implies, for example, that there exists a strictly increasing bijection between the set of all rational numbers and the set of all real algebraic numbers.
- any two countably infinite atomless Boolean algebras are isomorphic to each other.
- any two equivalent countable atomic models of a theory are isomorphic.
- the Erdős–Rényi model of random graphs, when applied to countably infinite graphs, always produces a unique graph, the Rado graph.
- any two many-complete recursively enumerable sets are recursively isomorphic.

[1] See Silver's "Who invented Cantor's back-and-forth argument?" (1994) for a historic overview.

What is the Back-and-Forth Method?[1]

Back-and-forth method

From Wikipedia, the free encyclopedia

In mathematical logic, especially set theory and model theory, the **back-and-forth method** is a method for showing isomorphism between countably infinite structures satisfying specified conditions. In particular it can be used to prove that

- any two countably infinite densely ordered sets (i.e., linearly ordered in such a way that between any two members there is another) without endpoints are isomorphic. An isomorphism between linear orders is simply a strictly increasing bijection. This result implies, for example, that there exists a strictly increasing bijection between the set of all rational numbers and the set of all real algebraic numbers.
- any two countably infinite atomless Boolean algebras are isomorphic to each other.
- any two equivalent countable atomic models of a theory are isomorphic.
- the Erdős–Rényi model of random graphs, when applied to countably infinite graphs, always produces a unique graph, the Rado graph.
- any two many-complete recursively enumerable sets are recursively isomorphic.

General technique to establish isomorphisms for countable structures:

- Any two countable unbounded dense linear orders are isomorphic (Cantor (1895))
- Any two one-one interreducible sets are recursively isomorphic (Myhill (1957))
- Many more examples in model theory, Boolean algebras, and graph theory

[1] See Silver's "Who invented Cantor's back-and-forth argument?" (1994) for a historic overview.

What will this talk be about?

What will this talk be about?

- Formalisation of the abstract method in a constructive foundation

What will this talk be about?

- Formalisation of the abstract method in a constructive foundation
- Instantiation to Cantor's and Myhill's isomorphism theorems

What will this talk be about?

- Formalisation of the abstract method in a constructive foundation
- Instantiation to Cantor's and Myhill's isomorphism theorems
- Observations about the computational interpretation of the results

What will this talk be about?

- Formalisation of the abstract method in a constructive foundation
- Instantiation to Cantor's and Myhill's isomorphism theorems
- Observations about the computational interpretation of the results
- Mechanisation of all results in Coq:

`www.ps.uni-saarland.de/extras/back-and-forth`

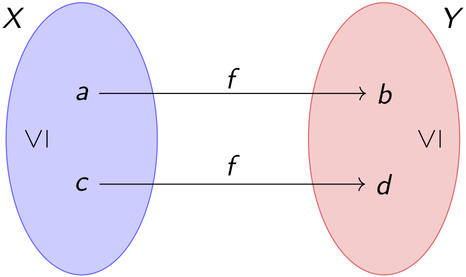
The Informal Argument: Cantor's Isomorphism Theorem

Assume two countable unbounded dense linear orders $(X, <)$ and $(Y, <)$, think $(\mathbb{Q}, <)$.

The Informal Argument: Cantor's Isomorphism Theorem

Assume two countable unbounded dense linear orders $(X, <)$ and $(Y, <)$, think $(\mathbb{Q}, <)$.

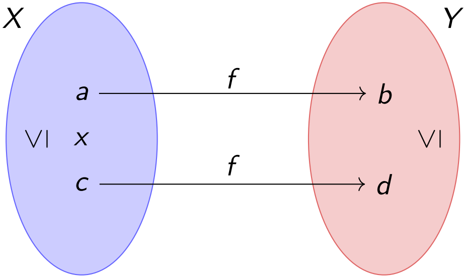
Given a partial order-isomorphism $f : X \to Y$, how do we extend it to a new element x?



The Informal Argument: Cantor's Isomorphism Theorem

Assume two countable unbounded dense linear orders $(X, <)$ and $(Y, <)$, think $(\mathbb{Q}, <)$.

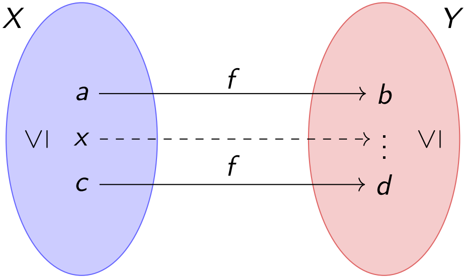
Given a partial order-isomorphism $f : X \to Y$, how do we extend it to a new element x?



The Informal Argument: Cantor's Isomorphism Theorem

Assume two countable unbounded dense linear orders $(X, <)$ and $(Y, <)$, think $(\mathbb{Q}, <)$.

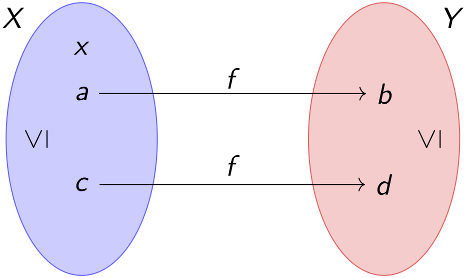
Given a partial order-isomorphism $f : X \to Y$, how do we extend it to a new element x?



The Informal Argument: Cantor's Isomorphism Theorem

Assume two countable unbounded dense linear orders $(X, <)$ and $(Y, <)$, think $(\mathbb{Q}, <)$.

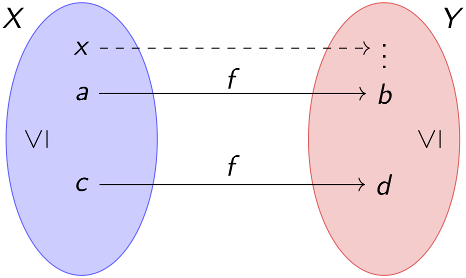
Given a partial order-isomorphism $f : X \to Y$, how do we extend it to a new element x?



The Informal Argument: Cantor's Isomorphism Theorem

Assume two countable unbounded dense linear orders $(X, <)$ and $(Y, <)$, think $(\mathbb{Q}, <)$.

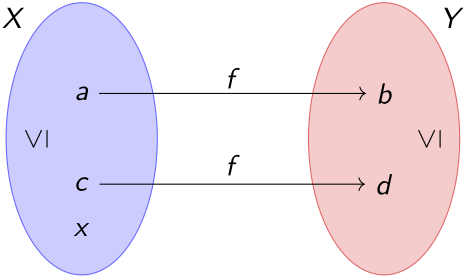
Given a partial order-isomorphism $f : X \to Y$, how do we extend it to a new element x?



The Informal Argument: Cantor's Isomorphism Theorem

Assume two countable unbounded dense linear orders $(X, <)$ and $(Y, <)$, think $(\mathbb{Q}, <)$.

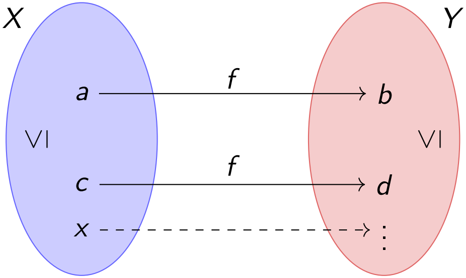
Given a partial order-isomorphism $f : X \to Y$, how do we extend it to a new element x?



The Informal Argument: Cantor's Isomorphism Theorem

Assume two countable unbounded dense linear orders $(X, <)$ and $(Y, <)$, think $(\mathbb{Q}, <)$.

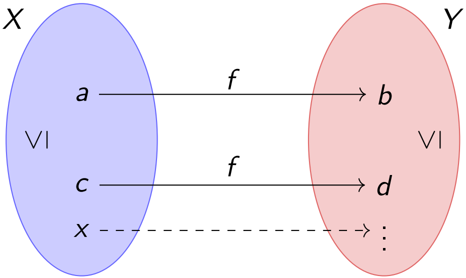
Given a partial order-isomorphism $f : X \to Y$, how do we extend it to a new element x?



The Informal Argument: Cantor's Isomorphism Theorem

Assume two countable unbounded dense linear orders $(X, <)$ and $(Y, <)$, think $(\mathbb{Q}, <)$.

Given a partial order-isomorphism $f : X \to Y$, how do we extend it to a new element x?



Obtain a total order-isomorphism F:

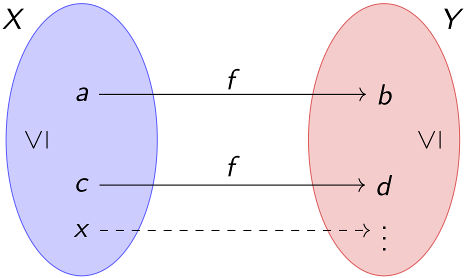
$$f_0 := \emptyset$$
$$f_{n+1} := \{(x_n, \text{matching partner for } x_n)\} \cup f_n$$

$$F := \bigcup_{n \in \mathbb{N}} f_n$$

The Informal Argument: Cantor's Isomorphism Theorem

Assume two countable unbounded dense linear orders $(X, <)$ and $(Y, <)$, think $(\mathbb{Q}, <)$.

Given a partial order-isomorphism $f : X \to Y$, how do we extend it to a new element x?



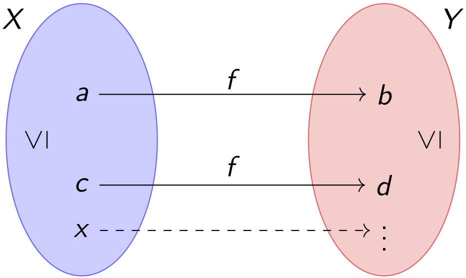
Obtain a total order-isomorphism F:

$$
\begin{aligned}
f_0 &:= \emptyset \\
f_{n+1} &:= \{(x_n, \text{matching partner for } x_n)\} \cup \\
&\quad \{(\text{matching partner for } y_n, y_n)\} \cup f_n \\
F &:= \bigcup_{n \in \mathbb{N}} f_n
\end{aligned}
$$

The Informal Argument: Cantor's Isomorphism Theorem

Assume two countable unbounded dense linear orders $(X, <)$ and $(Y, <)$, think $(\mathbb{Q}, <)$.

Given a partial order-isomorphism $f : X \to Y$, how do we extend it to a new element x?



Obtain a total order-isomorphism F:

$$
\begin{aligned}
f_0 &:= \emptyset \\
f_{n+1} &:= \{(x_n, \text{matching partner for } x_n)\} \cup \\
&\quad\ \{(\text{matching partner for } y_n, y_n)\} \cup f_n \\
F &:= \bigcup_{n \in \mathbb{N}} f_n
\end{aligned}
$$

F is correct: $\forall x x'.\ x < x' \leftrightarrow F\,x < F\,x'$

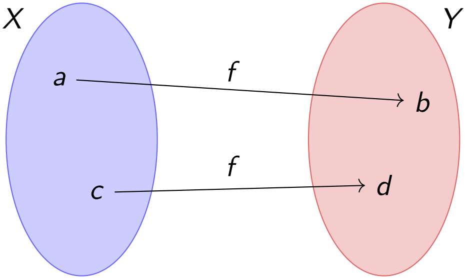
The Informal Argument: Countable Cantor-Bernstein Theorem

Assume two countable sets X and Y with injections $i_1 : X \to Y$ and $i_2 : Y \to X$.

The Informal Argument: Countable Cantor-Bernstein Theorem

Assume two countable sets X and Y with injections $i_1 : X \to Y$ and $i_2 : Y \to X$.

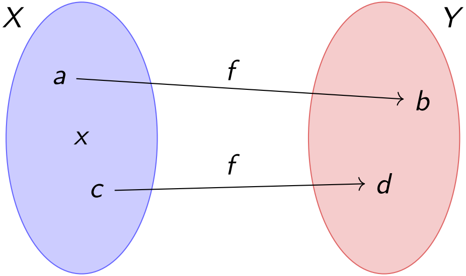
Given a partial one-to-one mapping $f : X \to Y$, how do we extend it to a new element x?



The Informal Argument: Countable Cantor-Bernstein Theorem

Assume two countable sets X and Y with injections $i_1 : X \to Y$ and $i_2 : Y \to X$.

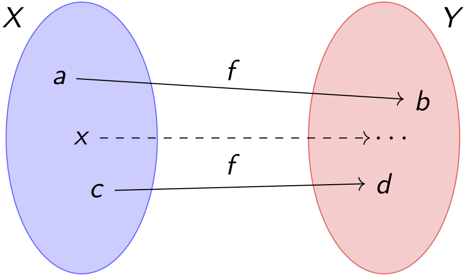
Given a partial one-to-one mapping $f : X \to Y$, how do we extend it to a new element x?



The Informal Argument: Countable Cantor-Bernstein Theorem

Assume two countable sets X and Y with injections $i_1 : X \to Y$ and $i_2 : Y \to X$.

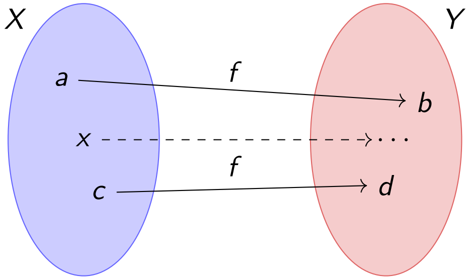
Given a partial one-to-one mapping $f : X \to Y$, how do we extend it to a new element x?



The Informal Argument: Countable Cantor-Bernstein Theorem

Assume two countable sets X and Y with injections $i_1 : X \to Y$ and $i_2 : Y \to X$.

Given a partial one-to-one mapping $f : X \to Y$, how do we extend it to a new element x?



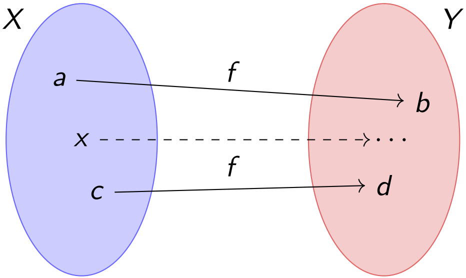
Obtain a total one-to-one mapping F:

$$
\begin{aligned}
f_0 &:= \emptyset \\
f_{n+1} &:= \{(x_n, \text{matching partner for } x_n)\} \cup \\
&\quad\;\{(\text{matching partner for } y_n, y_n)\} \cup f_n \\
F &:= \bigcup_{n \in \mathbb{N}} f_n
\end{aligned}
$$

The Informal Argument: Countable Cantor-Bernstein Theorem

Assume two countable sets X and Y with injections $i_1 : X \to Y$ and $i_2 : Y \to X$.

Given a partial one-to-one mapping $f : X \to Y$, how do we extend it to a new element x?

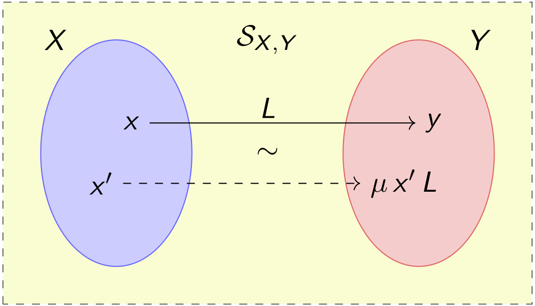


Obtain a total one-to-one mapping F:

$$\begin{aligned} f_0 &:= \emptyset \\ f_{n+1} &:= \{(x_n, \text{matching partner for } x_n)\} \cup \\ &\quad \{(\text{matching partner for } y_n, y_n)\} \cup f_n \\ F &:= \bigcup_{n \in \mathbb{N}} f_n \end{aligned}$$

F is correct: $\forall x x'.\ x = x' \leftrightarrow F\,x = F\,x'$

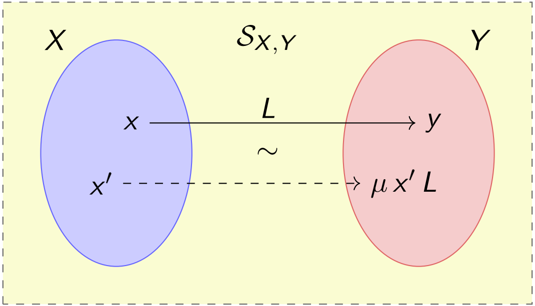
The Abstract Argument: Assumptions



Assumed data:

- Countable types $X, Y : \mathbb{T}$
- Abstract structure $\mathcal{A} : \mathbb{T} \times \mathbb{T} \to \mathbb{T}$
- Structure $\mathcal{S}_{X,Y} : \mathcal{A}(X, Y)$ on X and Y
- $\sim : (X \times X) \to (Y \times Y) \to \mathbb{P}$
- $\mu : X \to \mathcal{L}(X \times Y) \to Y$

The Abstract Argument: Assumptions



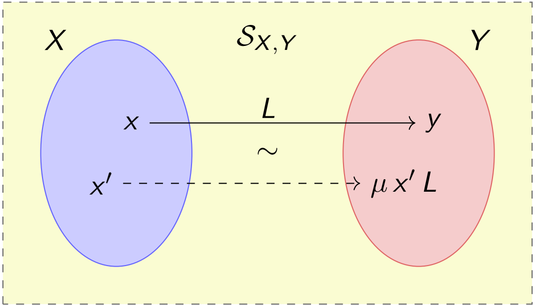
Assumed data:

- Countable types $X, Y : \mathbb{T}$
- Abstract structure $\mathcal{A} : \mathbb{T} \times \mathbb{T} \to \mathbb{T}$
- Structure $\mathcal{S}_{X,Y} : \mathcal{A}(X, Y)$ on X and Y
- $\sim : (X \times X) \to (Y \times Y) \to \mathbb{P}$
- $\mu : X \to \mathcal{L}(X \times Y) \to Y$

Assumed properties:

- If $(x, x') \sim (y, y')$, then $x = x'$ iff $y = y'$.
- If L respects $\sim$, then so does $\nu\, x\, L := (x, \mu\, x\, L) :: L$.

The Abstract Argument: Assumptions



Assumed data:

- Countable types $X, Y : \mathbb{T}$
- Abstract structure $\mathcal{A} : \mathbb{T} \times \mathbb{T} \to \mathbb{T}$
- Structure $\mathcal{S}_{X,Y} : \mathcal{A}(X, Y)$ on X and Y
- $\sim : \forall XY.\, \mathcal{A}(X, Y) \to (X \times X) \to (Y \times Y) \to \mathbb{P}$
- $\mu : \forall XY.\, \mathcal{A}(X, Y) \to X \to \mathcal{L}(X \times Y) \to Y$
- Structure reversal $\mathcal{S}_{X,Y}^{-1} : \mathcal{A}(Y, X)$

Assumed properties:

- If $(x, x') \sim (y, y')$, then $x = x'$ iff $y = y'$.
- If L respects $\sim$, then so does $\nu\, x\, L := (x, \mu\, x\, L) :: L$.
- $(\mathcal{S}_{X,Y}^{-1})^{-1} = \mathcal{S}_{X,Y}$ and if $(x, x') \sim (y, y')$, then $(y, y') \sim (x, x')$

The Abstract Argument: Construction

The Abstract Argument: Construction

Define finite approximations of the isomorphism, following the enumeration of X and Y:

$$
\begin{aligned}
L_{\_} &: \forall XY.\, \mathcal{A}(X, Y) \to \mathbb{N} \to \mathcal{L}(X \times Y) \\
L_0 &:= [] \\
L_{n+1} &:= (\nu\, y_n\, (\nu\, x_n\, L_n)^{-1})^{-1}
\end{aligned}
$$

The Abstract Argument: Construction

Define finite approximations of the isomorphism, following the enumeration of X and Y:

$$L_{\_} \;:\; \forall XY.\, \mathcal{A}(X,Y) \to \mathbb{N} \to \mathcal{L}(X \times Y)$$

$$L_0 \;:=\; []$$

$$L_{n+1} \;:=\; (\nu\, y_n\, (\nu\, x_n\, L_n)^{-1})^{-1}$$

Obtain the isomorphism by lookup in an approximation after enough iterations:

$$F\, x_n \;:=\; L_{n+1}[x_n] \qquad\qquad F^{-1}\, x_n \;:=\; L^{-1}_{n+1}[y_n]$$

The Abstract Argument: Construction

Define finite approximations of the isomorphism, following the enumeration of X and Y:

$$L_{\_} \; : \; \forall XY.\, \mathcal{A}(X,Y) \to \mathbb{N} \to \mathcal{L}(X \times Y)$$

$$L_0 \; := \; []$$

$$L_{n+1} \; := \; (\nu\, y_n\, (\nu\, x_n\, L_n)^{-1})^{-1}$$

Obtain the isomorphism by lookup in an approximation after enough iterations:

$$F\, x_n \; := \; L_{n+1}[x_n] \qquad\qquad F^{-1}\, x_n \; := \; L_{n+1}^{-1}[y_n]$$

Goal: F preserves the abstract structure, i.e. $\forall xx'.\, (x,x') \sim (F\,x, F\,x')$.

The Abstract Argument: Verification

Lemma

L_n respects $\sim$ for every $n : \mathbb{N}$.

The Abstract Argument: Verification

Lemma

L_n respects $\sim$ for every $n : \mathbb{N}$.

Proof.

By induction on n with $L_0 = []$ trivial.

The Abstract Argument: Verification

Lemma

L_n respects $\sim$ for every $n : \mathbb{N}$.

Proof.

By induction on n with $L_0 = []$ trivial. Now $L_{n+1} = (\nu\, y_n\, (\nu\, x_n\, L_n)^{-1})^{-1}$ respects $\sim$ since so do:

The Abstract Argument: Verification

Lemma

L_n respects $\sim$ for every $n : \mathbb{N}$.

Proof.

By induction on n with $L_0 = []$ trivial. Now $L_{n+1} = (\nu\, y_n\, (\nu\, x_n\, L_n)^{-1})^{-1}$ respects $\sim$ since so do:

L_n (by inductive hypothesis)

□

The Abstract Argument: Verification

Lemma

L_n respects $\sim$ for every $n : \mathbb{N}$.

Proof.

By induction on n with $L_0 = []$ trivial. Now $L_{n+1} = (\nu\, y_n\, (\nu\, x_n\, L_n)^{-1})^{-1}$ respects $\sim$ since so do:

$$\begin{aligned} L_n &\quad \text{(by inductive hypothesis)} \\ \nu\, x_n\, L_n &\quad \text{(by the assumption on } \mu\text{)} \end{aligned}$$

□

The Abstract Argument: Verification

Lemma

L_n respects $\sim$ for every $n : \mathbb{N}$.

Proof.

By induction on n with $L_0 = []$ trivial. Now $L_{n+1} = (\nu\, y_n\, (\nu\, x_n\, L_n)^{-1})^{-1}$ respects $\sim$ since so do:

$$
\begin{aligned}
L_n \quad & \text{(by inductive hypothesis)} \\
\nu\, x_n\, L_n \quad & \text{(by the assumption on } \mu\text{)} \\
(\nu\, x_n\, L_n)^{-1} \quad & \text{(by the symmetry assumption)}
\end{aligned}
$$

□

The Abstract Argument: Verification

Lemma

L_n respects $\sim$ for every $n : \mathbb{N}$.

Proof.

By induction on n with $L_0 = []$ trivial. Now $L_{n+1} = (\nu\, y_n\, (\nu\, x_n\, L_n)^{-1})^{-1}$ respects $\sim$ since so do:

$$
\begin{aligned}
L_n &\quad \text{(by inductive hypothesis)}\\
\nu\, x_n\, L_n &\quad \text{(by the assumption on } \mu\text{)}\\
(\nu\, x_n\, L_n)^{-1} &\quad \text{(by the symmetry assumption)}\\
\nu\, y_n\, (\nu\, x_n\, L_n)^{-1} &\quad \text{(by the assumption on } \mu\text{)}
\end{aligned}
$$

□

The Abstract Argument: Verification

Lemma

L_n respects $\sim$ for every $n : \mathbb{N}$.

Proof.

By induction on n with $L_0 = []$ trivial. Now $L_{n+1} = (\nu\, y_n\, (\nu\, x_n\, L_n)^{-1})^{-1}$ respects $\sim$ since so do:

$$
\begin{aligned}
L_n &\quad \text{(by inductive hypothesis)}\\
\nu\, x_n\, L_n &\quad \text{(by the assumption on } \mu\text{)}\\
(\nu\, x_n\, L_n)^{-1} &\quad \text{(by the symmetry assumption)}\\
\nu\, y_n\, (\nu\, x_n\, L_n)^{-1} &\quad \text{(by the assumption on } \mu\text{)}\\
(\nu\, y_n\, (\nu\, x_n\, L_n)^{-1})^{-1} &\quad \text{(by the symmetry assumption)}
\end{aligned}
$$

□

The Abstract Argument: Verification

Lemma

L_n respects $\sim$ for every $n : \mathbb{N}$.

Proof.

By induction on n with $L_0 = []$ trivial. Now $L_{n+1} = (\nu\, y_n\, (\nu\, x_n\, L_n)^{-1})^{-1}$ respects $\sim$ since so do:

$$
\begin{aligned}
L_n & \quad \text{(by inductive hypothesis)} \\
\nu\, x_n\, L_n & \quad \text{(by the assumption on } \mu\text{)} \\
(\nu\, x_n\, L_n)^{-1} & \quad \text{(by the symmetry assumption)} \\
\nu\, y_n\, (\nu\, x_n\, L_n)^{-1} & \quad \text{(by the assumption on } \mu\text{)} \\
(\nu\, y_n\, (\nu\, x_n\, L_n)^{-1})^{-1} & \quad \text{(by the symmetry assumption)}
\end{aligned}
$$

□

Theorem (Isomorphism)

F satisfies $(x, x') \sim (F\,x, F\,x')$ for all $x, x' : X$ and is inverted by F^{-1}.

The Abstract Argument in Coq

```
(* Assumptions regarding the invertable abstract structure 𝒜 *)
struc : Type → Type → Type
srev : ∀ X Y, struc X Y → struc Y X
srev_invol : ∀ X Y (S : struc X Y), srev (srev S) = S

(* Assumptions regarding the abstract isomorphism property ∼ *)
iso : ∀ X Y, struc X Y → X → X → Y → Y → Prop
iso_eq : ∀ X Y (S : struc X Y) x x' y y', iso S x x' y y' → x = x' ↔ y = y'
iso_rev : ∀ X Y (S : struc X Y) x x' y y', iso S x x' y y' → iso (srev S) y y' x x'

(* Assumptions regarding the one-step extension function μ *)
find : ∀ X Y, struc X Y → X → list (X * Y) → Y
find_iso : ∀ X Y (S : struc X Y) L x, x ∉ dom L → tiso S L → tiso S ((x, find S x L) :: L)

(* After 100 lines of abstract proofs: *)
Theorem back_and_forth :
  { F & { G | inverse F G ∧ ∀ x x', iso SXY x x' (F x) (F x') } }.
```

The Computational Argument[2]

Informally, the abstract theorem we have proven states:

Countable structures with a structure-preserving extension function are isomorphic.

[2] Employing synthetic computability theory as pioneered by Richman (1983) and Bauer (2006).

The Computational Argument[2]

Informally, the abstract theorem we have proven states:

Countable structures with a structure-preserving extension function are isomorphic.

By working in a constructive meta-theory, we can also interpret it effectively:

Computational interpretation: enumerable and discrete structures with a structure-preserving extension algorithm are computably isomorphic.

[2] Employing synthetic computability theory as pioneered by Richman (1983) and Bauer (2006).

The Computational Argument[2]

Informally, the abstract theorem we have proven states:

Countable structures with a structure-preserving extension function are isomorphic.

By working in a constructive meta-theory, we can also interpret it effectively:

Computational interpretation: enumerable and discrete structures with a structure-preserving extension algorithm are computably isomorphic.

So in the upcoming instantiations we always prove two theorems simultaneously,

[2] Employing synthetic computability theory as pioneered by Richman (1983) and Bauer (2006).

The Computational Argument[2]

Informally, the abstract theorem we have proven states:

Countable structures with a structure-preserving extension function are isomorphic.

By working in a constructive meta-theory, we can also interpret it effectively:

Computational interpretation: enumerable and discrete structures with a structure-preserving extension algorithm are computably isomorphic.

So in the upcoming instantiations we always prove two theorems simultaneously, and by implementation in Coq we even have extractable algorithms at hand!

[2] Employing synthetic computability theory as pioneered by Richman (1983) and Bauer (2006).

Cantor's Isomorphism Theorem[3]

[3] Discovered by Cantor (1895), previously mechanised by Giese and Schönegge (1995) and Marzion (2020).

Cantor's Isomorphism Theorem[3]

$$\text{Structure } \mathcal{A}(X, Y) \;:=\; X \text{ and } Y \text{ are unboundend dense linear orders}$$

$$(x, x') \sim (y, y') \;:=\; (x = x' \leftrightarrow y = y') \land (x < x' \leftrightarrow y < y')$$

[3] Discovered by Cantor (1895), previously mechanised by Giese and Schönegge (1995) and Marzion (2020).

Cantor's Isomorphism Theorem[3]

$$\text{Structure } \mathcal{A}(X, Y) \;:=\; X \text{ and } Y \text{ are unboundend dense linear orders}$$
$$(x, x') \sim (y, y') \;:=\; (x = x' \leftrightarrow y = y') \land (x < x' \leftrightarrow y < y')$$

Lemma

There is a function μ such that if L is a partial order-isomorphism, so is $(x, \mu\, x\, L) :: L$.

[3] Discovered by Cantor (1895), previously mechanised by Giese and Schönegge (1995) and Marzion (2020).

Cantor's Isomorphism Theorem[3]

$$\text{Structure } \mathcal{A}(X, Y) \;:=\; X \text{ and } Y \text{ are unboundend dense linear orders}$$
$$(x, x') \sim (y, y') \;:=\; (x = x' \leftrightarrow y = y') \land (x < x' \leftrightarrow y < y')$$

Lemma

There is a function μ such that if L is a partial order-isomorphism, so is $(x, \mu\, x\, L) :: L$.

Theorem (Cantor)

All countable unbounded dense linear orders are isomorphic.

[3] Discovered by Cantor (1895), previously mechanised by Giese and Schönegge (1995) and Marzion (2020).

Cantor's Isomorphism Theorem[3]

$$\text{Structure } \mathcal{A}(X,Y) \;:=\; X \text{ and } Y \text{ are unboundend dense linear orders}$$
$$(x,x') \sim (y,y') \;:=\; (x = x' \leftrightarrow y = y') \land (x < x' \leftrightarrow y < y')$$

Lemma

There is a function μ such that if L is a partial order-isomorphism, so is $(x, \mu\, x\, L) :: L$.

Theorem (Cantor)

All countable unbounded dense linear orders are isomorphic.

Theorem (Computational Cantor)

All decidable linear orders over enumerable domain with computable witnesses for density and unboundedness are computably isomorphic.

[3] Discovered by Cantor (1895), previously mechanised by Giese and Schönegge (1995) and Marzion (2020).

Cantor's Isomorphism Theorem in Coq

```
(* After 150 lines to define partner and prove step_morph: *)
Theorem Cantor X Y (OX : dulo X) (OY : dulo Y) (RX : retract X nat) (RY : retract Y nat) :
  { F : X → Y & { G | inverse F G ∧ ∀ x x', x < x' ↔ (F x) < (F x') } }.
Proof.
  unshelve edestruct back_and_forth as [F[G[H1 H2]]].
  - intros A B. exact (dulo A * dulo B).                                (* structure *)
  - intros A B [OA OB]. exact (OB, OA).                                 (* srev *)
  - cbn. intros A B [OA OB]. reflexivity.                                (* srev_invol *)
  - intros A B [OA OB] a a' b b'. exact ((a = a' ↔ b = b') ∧ (a < a' ↔ b < b')).   (* iso *)
  - cbn. tauto.                                                          (* iso_eq *)
  - cbn. tauto.                                                          (* iso_rev *)
  - cbn. intros A B [OA OB] f a. exact (partner OA OB f a).              (* find *)
  - cbn. intros A B [OA OB] f x. unfold step, tiso. now apply step_morph.   (* find_iso *)

  - exact X.
  - exact Y.
  - exact (OX, OY).
  - exact RX.
  - exact RY.

  - cbn in *. exists F, G. split; try apply H1. apply H2.
Qed.
```

Myhill's Isomorphism Theorem[4]

"One-one interreducible sets of numbers are recursively isomorphic."

[4] Discovered by Myhill (1957), previously mechanised by Forster, Jahn, and Smolka (2022).

Myhill's Isomorphism Theorem[4]

"One-one interreducible sets of numbers are recursively isomorphic."

A function $f : X \to Y$ is a many-one reduction from $p : X \to \mathbb{P}$ to $q : Y \to \mathbb{P}$ if

$$\forall x : X.\, p\, x \leftrightarrow q\,(f\, x).$$

If f is injective (bijective), it is a one-one reduction (recursive isomorphism).

[4] Discovered by Myhill (1957), previously mechanised by Forster, Jahn, and Smolka (2022).

Myhill's Isomorphism Theorem[4]

"One-one interreducible sets of numbers are recursively isomorphic."

A function $f : X \to Y$ is a many-one reduction from $p : X \to \mathbb{P}$ to $q : Y \to \mathbb{P}$ if

$$\forall x : X.\, p\,x \leftrightarrow q\,(f\,x).$$

If f is injective (bijective), it is a one-one reduction (recursive isomorphism).

$$\begin{aligned} \text{Structure } \mathcal{A}(X, Y) &:= \text{ a pair } (p, q) \text{ of one-one interreducible predicates} \\ (x, x') \sim (y, y') &:= (x = x' \leftrightarrow y = y') \wedge (p\,x \leftrightarrow q\,y) \end{aligned}$$

[4]Discovered by Myhill (1957), previously mechanised by Forster, Jahn, and Smolka (2022).

Myhill's Isomorphism Theorem[4]

"One-one interreducible sets of numbers are recursively isomorphic."

A function $f : X \to Y$ is a many-one reduction from $p : X \to \mathbb{P}$ to $q : Y \to \mathbb{P}$ if

$$\forall x : X.\, p\, x \leftrightarrow q\,(f\, x).$$

If f is injective (bijective), it is a one-one reduction (recursive isomorphism).

$$\begin{aligned} \text{Structure } \mathcal{A}(X, Y) \; &:= \; \text{a pair } (p, q) \text{ of one-one interreducible predicates} \\ (x, x') \sim (y, y') \; &:= \; (x = x' \leftrightarrow y = y') \land (p\, x \leftrightarrow q\, y) \end{aligned}$$

Theorem (Myhill)

One-one interreducible unary predicates on (retracts of) $\mathbb{N}$ are recursively isomorphic.

[4] Discovered by Myhill (1957), previously mechanised by Forster, Jahn, and Smolka (2022).

Myhill's Isomorphism Theorem in Coq

```
(* After 100 lines to define mstep and prove step_corr: *)
Theorem Myhill X Y (SXY : bireduction X Y) (RX : retract X nat) (RY : retract Y nat) :
  { F : X → Y & { G | inverse F G ∧ reduction p q F } }.
Proof.
  unshelve edestruct back_and_forth as [F[G[H1 H2]]].
  - intros A B. exact (bireduction A B).                                       (* structure *)
  - intros A B S. cbn in *. apply (@Build_bireduction B A eY eX q p g f); apply S.   (* srev *)
  - cbn. intros A B []. reflexivity.                                           (* srev_invol *)
  - intros A B S a a' b b'. cbn in S. exact ((a = a' ↔ b = b') ∧ (p a ↔ q b)).  (* iso *)
  - cbn. tauto.                                                                (* iso_eq *)
  - cbn. tauto.                                                                (* iso_rev *)
  - cbn. intros A B S C a. exact (mstep f eX eY C a).                          (* find *)
  - cbn. intros A B S C a. unfold step, tiso. apply step_corr; apply S.        (* find_iso *)

  - exact X.
  - exact Y.
  - exact SXY.
  - exact RX.
  - exact RY.

  - cbn in *. exists F, G. split; try apply H1. intros x. now apply H2.
Qed.
```

Bonus: Computational Cantor-Bernstein Theorem[5]

[5] Discovered and previously mechanised by Forster, Jahn, and Smolka (2022).

Bonus: Computational Cantor-Bernstein Theorem[5]

If we forget about the computational interpretation of the previous theorem, we can observe:

[5]Discovered and previously mechanised by Forster, Jahn, and Smolka (2022).

Bonus: Computational Cantor-Bernstein Theorem[5]

If we forget about the computational interpretation of the previous theorem, we can observe:

Corollary (Countable Cantor-Bernstein)

Countable sets X and Y with injections $X \to Y$ and $Y \to X$ admit a bijection $X \to Y$.

[5] Discovered and previously mechanised by Forster, Jahn, and Smolka (2022).

Bonus: Computational Cantor-Bernstein Theorem[5]

If we forget about the computational interpretation of the previous theorem, we can observe:

Corollary (Countable Cantor-Bernstein)

Countable sets X and Y with injections $X \to Y$ and $Y \to X$ admit a bijection $X \to Y$.

Proof.

Pick as p and q the full predicates on X and Y, then use the previous theorem. □

[5] Discovered and previously mechanised by Forster, Jahn, and Smolka (2022).

Bonus: Computational Cantor-Bernstein Theorem[5]

If we forget about the computational interpretation of the previous theorem, we can observe:

Corollary (Countable Cantor-Bernstein)

Countable sets X and Y with injections $X \to Y$ and $Y \to X$ admit a bijection $X \to Y$.

Proof.

Pick as p and q the full predicates on X and Y, then use the previous theorem. □

Though Pradic and Brown (2019) show that the general Cantor-Bernstein theorem is equivalent to excluded middle, the restriction to countable sets holds constructively!

[5] Discovered and previously mechanised by Forster, Jahn, and Smolka (2022).

What are the Take-Home Messages?

What are the Take-Home Messages?

- The back-and-forth method can be described abstractly via a very general interface:
  - Natural formulation in constructive type theory
  - Neat construction and verification fully exploiting symmetry

What are the Take-Home Messages?

- The back-and-forth method can be described abstractly via a very general interface:
  - Natural formulation in constructive type theory
  - Neat construction and verification fully exploiting symmetry
- Instantiation just requires a notion of structure and a one-step extension function:
  - Examples given for Cantor's and Myhill's isomorphism theorems
  - Should apply to all other examples and might save your time

What are the Take-Home Messages?

- The back-and-forth method can be described abstractly via a very general interface:
  - Natural formulation in constructive type theory
  - Neat construction and verification fully exploiting symmetry
- Instantiation just requires a notion of structure and a one-step extension function:
  - Examples given for Cantor's and Myhill's isomorphism theorems
  - Should apply to all other examples and might save your time
- By doing all proofs constructively, you get computational results for free:
  - No need to work with an explicit model of computation
  - Method is pretty constructive by default, only little care needs to be taken

What are the Take-Home Messages?

- The back-and-forth method can be described abstractly via a very general interface:
  - Natural formulation in constructive type theory
  - Neat construction and verification fully exploiting symmetry
- Instantiation just requires a notion of structure and a one-step extension function:
  - Examples given for Cantor's and Myhill's isomorphism theorems
  - Should apply to all other examples and might save your time
- By doing all proofs constructively, you get computational results for free:
  - No need to work with an explicit model of computation
  - Method is pretty constructive by default, only little care needs to be taken

Thanks for listening!

Bibliography

Bauer, A. (2006). First steps in synthetic computability theory. *Electronic Notes in Theoretical Computer Science*, 155:5–31.

Cantor, G. (1895). Beiträge zur Begründung der transfiniten Mengenlehre. *Mathematische Annalen*, 46(4):481–512.

Forster, Y., Jahn, F., and Smolka, G. (2022). A Constructive and Synthetic Theory of Reducibility: Myhill's Isomorphism Theorem and Post's Problem for Many-one and Truth-table Reducibility in Coq (Full Version). working paper or preprint.

Forster, Y., Kirst, D., and Smolka, G. (2019). On synthetic undecidability in Coq, with an application to the entscheidungsproblem. In *Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs*, pages 38–51.

Giese, M. and Schönegge, A. (1995). *Any Two Countable, Densely Ordered Sets Without Endpoints are Isomorphic: A Formal Proof with KIV*. Univ., Fak. für Informatik.

Marzion, E. (2020). Visualizing Cantor's theorem on dense linear orders using Coq. Blog post.

Myhill, J. (1957). Creative sets. *Journal of Symbolic Logic*, 22(1).

Pradic, P. and Brown, C. E. (2019). Cantor-Bernstein implies excluded middle. *arXiv preprint arXiv:1904.09193*.

Richman, F. (1983). Church's thesis without tears. *The Journal of symbolic logic*, 48(3):797–803.

Silver, C. L. (1994). Who invented Cantor's back-and-forth argument? *Modern Logic*, 4(1):74–78.

Synthetic Computability Theory

content

Myhill Construction

content