Formalizing Minimal Systems and Recurrence in Topological Dynamics in Lean 4

Eric Ceglie
ETH Zürich, Rämistrasse 101, 8092 Zürich
eceglie@ethz.ch

Leandro Zehnder
ETH Zürich, HG J12, Rämistrasse 101, 8092 Zürich
lzehnde@ethz.ch

May 8, 2025

Abstract

We give a fully machine-checked formalization in Lean 4 using `mathlib4` of two existence results in compact topological dynamics: the existence of minimal invariant subsystems and the existence of recurrent points under continuous monoid actions. The full code can be found on github.com/ericceg/Topological-Dynamics-in-lean4.

Contents

Preamble

Throughout, we refer to [BS02, Section 2] and [EW10, Exercises 4.2.1 and 4.2.2]. We would also like to thank Prof. Manfred Einsiedler for introducing the authors to the topic of dynamical systems with his lecture "Dynamical Systems and Ergodic Theory", held in spring 2024 at ETH Zurich. Moreover, the authors would like to thank Jaume De Dios Pont for his help and guidance with the Lean implementation and for introducing the authors to Lean in a seminar held by him at ETH Zurich in the spring of 2025.

In all that follows $\mathbb{N}$ shall refer to the non-negative integers and $\mathbb{N}^+ := \mathbb{N} \setminus \{0\}$.

1 Introduction

We begin with a short introduction to topological dynamics where we are interested in studying so-called *dynamical systems*.

Definition 1.1 (Dynamical system)**.** Let X be a set and $T : X \to X$ be some map. Then we call this a *dynamical system*.

Often we will want to look at a less general setting. One interesting case is the case where X is a topological space (possibly with some additional properties such as compactness) and $T : X \to X$ is continuous (or even a homeomorphism). This is the setting of *topological dynamics*.

An important concept is that of an *orbit*. Orbits are usually defined via actions and, in fact, a map T induces an action from $\mathbb{N}$ on X as follows:

$$\forall n \in \mathbb{N} : \forall x \in X : \quad n + x := T^{[n]}(x), \tag{1}$$

where we iteratively define $T^{[n]}(x) = T\big(T^{[n-1]}(x)\big)$ for all $n \in \mathbb{N}^+$ and $x \in X$ and $T^{[0]}(x) = x$. This now allows us to consider orbits.

Definition 1.2 (Orbit)**.** We call the set:

$$\mathrm{Orb}(x) := \{y \in X \mid \exists n \in \mathbb{N} : T^{[n]}(x) = y\} \tag{2}$$

the **orbit** of x.

We now want to consider so-called *minimal actions*.

Definition 1.3. (Minimal action) $T : X \to X$ is called *minimal* if every orbit is dense (in X).

We consider a classical example.

Example 1.4. Let $X = \mathbb{R}/\mathbb{Z} \cong \mathbb{S}^1$. Pick $\alpha \in \mathbb{R} \setminus \mathbb{Q}$ some irrational number and consider the corresponding homeomorphism:

$$T : X \to X, [x] \mapsto [x + \alpha]. \tag{3}$$

This is indeed a homeomorphism, and it is a standard fact from topology that for every $n \in \mathbb{N}$ the set of the form

$$\{x + n\alpha \mod 1 | n \in \mathbb{N}\} \tag{4}$$

is dense in $\mathbb{R}/\mathbb{Z}$. Hence the action induced by this T is minimal on the circle.

1.1 A prerequisite result

It can be shown (and it is already shown in `mathlib4`) that:

Theorem 1.5. *Let X be a compact topological space and let $T : X \to X$ be a homeomorphism. Then the following are equivalent:*

(a) T is minimal.

(b) For any $E \subseteq X$ which is closed and invariant under T (i.e. $T(E) \subseteq E$) we have $E = \varnothing$ or $E = X$.

This result will come in handy in formulating our new results.

2 Reformulation using Actions

As `mathlib4` tries to be as general as possible, we will reformulate the previous definitions and theorems using actions.

Definition 2.1. A set M equipped with a binary operation $+\colon M \times M \to M$ is called an *additive monoid* if the following hold:

1. For all $a, b, c \in M$ we have $(a + b) + c = a + (b + c)$.
2. There exists an element $0 \in M$ such that $a + 0 = 0 + a = a$ for all $a \in M$.

Definition 2.2. An *additive action* of an additive monoid M on a set X is a map

$$M \times X \to X,\ (m, x) \mapsto mx$$

such that:

1. For all $x \in X$ we have $0x = x$.
2. For all $m, n \in M$ and $x \in X$ we have $(m + n)x = m(nx)$.

Let M be an additive monoid acting on a compact topological space X.

Definition 2.3. For any $x \in X$ define the *orbit* of x by

$$\mathrm{Orb}(x) := \{mx \mid m \in M\}.$$

Definition 2.4. (X, M) is called *minimal* if every orbit is dense.

This definition is already implemented in `mathlib4`, see [Kud21].

```
class AddAction.IsMinimal (M α : Type*)
    [AddMonoid M] [TopologicalSpace α] [AddAction M α] : Prop where
  dense_orbit : ∀ x : α, Dense (AddAction.orbit M x)
```

Definition 2.5. We say that the action of M on X is *continuous* if for every $m \in M$ the map $X \to X$, $x \mapsto mx$ is continuous.

3 Existence of Minimal Subsystems

The goal of this section is to prove the following theorem.

Theorem 3.1. *Let M be an additive monoid acting continuously on a non-empty compact topological space X. Then there exists a closed non-empty $Y \subseteq X$ such that $MY \subseteq Y$ and the restricted action*

$$M \times Y \to Y,\ (m, y) \mapsto my$$

is minimal.

```
theorem exists_minimal_invariant_subset {M X : Type*}
    [h_X_top : TopologicalSpace X] [h_X_compact : CompactSpace X]
    [h_X_nonempty : Nonempty X] [h_M_monoid : AddMonoid M]
    [h_M_X_action : AddAction M X]
    [h_action_continuous : ContinuousConstVAdd M X] :
    ∃ (Y : Set X) (h_SubAction : AddActionRestriction M X Y),
    have : AddAction M Y := h_SubAction.SubAction
    Y.Nonempty ∧
    IsClosed Y ∧
    AddAction.IsMinimal M Y
```

We begin by reformulating Theorem 1.5 in our more general setting.

Theorem 3.2. *Let M be an additive monoid acting continuously on a topological space X. The following are equivalent:*

- *(X, M) is minimal.*
- *For every closed M-invariant subset $E \subseteq X$ we have $E = \emptyset$ or $E = X$.*

Conveniently, this reformulation is already implemented in `mathlib4`, see [Kud21].

```
theorem isMinimal_iff_isClosed_vadd_invariant (M : Type u_1)
    {α : Type u_3} [AddMonoid M] [TopologicalSpace α]
    [AddAction M α] [ContinuousConstVAdd M α] :
    AddAction.IsMinimal M α
    ↔ ∀ (s : Set α), IsClosed s → (∀ (c : M), c +ᵥ s ⊆ s) → s = ∅ ∨ s =
     Set.univ
```

For the proof of Theorem 3.1 we need two more lemmata that are not yet implemented in `mathlib4`.

Lemma 3.3. *Let M be an additive monoid acting on a set X and let $Y \subseteq X$ be an M-invariant subset, i.e. assume that $MY \subseteq Y$. Then the restricted action*

$$M \times Y \to Y,\ (m, y) \mapsto my$$

is a well-defined additive action of M on Y.

```
class AddActionRestriction (M : Type*) (X : Type*) (Y : Set X)
    [AddMonoid M] [add_action_orig : AddAction M X] where
    SubAction : AddAction M Y
    SubAction_eq_Action : ∀ (c : M) (x : Y), ↑(SubAction.vadd c x) =
    add_action_orig.vadd c ↑x -- we need to use this instead of ↑(c +ᵥ x) =
    c +ᵥ ↑x to make it clear which action to use

def invariant_subset_restricted_action {M X : Type*} {Y : Set X}
    [h_M_monoid : AddMonoid M] [h_M_X_action : AddAction M X]
    (h_Y_invariant : ∀ c : M, ∀ y ∈ Y, c +ᵥ y ∈ Y) :
    AddActionRestriction M X Y
```

Lemma 3.4. *Let M be an additive monoid acting continuously on a compact topological space X, and let $Y \subseteq X$ be an M-invariant subset. Then the restricted action of M on Y is continuous.*

```
class AddActionRestrictionContinuous (M X : Type*) (Y : Set X)
    [h_X_top : TopologicalSpace X]  [h_M_monoid : AddMonoid M]
    [h_M_X_action : AddAction M X]
    [h_action_continuous : ContinuousConstVAdd M X] where
    (RestrictedAction : AddActionRestriction M X Y)
    (SubAction := RestrictedAction.SubAction)
    (SubActionContinuous : ContinuousConstVAdd M Y)

def restriction_of_continuous_action_is_continuous {M X : Type*}
    [h_X_top : TopologicalSpace X]  [h_M_monoid : AddMonoid M]
    [h_M_X_action : AddAction M X]
    [h_action_continuous : ContinuousConstVAdd M X] (Y : Set X)
    (h_Y_invariant : ∀ c : M, ∀ y ∈ Y, c +ᵥ y ∈ Y) :
    AddActionRestrictionContinuous M X Y
```

Besides Lemma 3.3 and Lemma 3.4, we state some more general results that will be needed and that are already implemented in `mathlib4`.

Theorem 3.5 (Zorn's Lemma)**.** *Let S be a set of subsets of a set α. Assume that for every chain $C \subseteq S$ there exists an element $l \in S$ such that*

$$\forall s \in C : \ l \subseteq s.$$

Then there exists an element $m \in S$ with

$$\forall a \in S : \ a \subseteq m \implies a = m.$$

```
theorem zorn_superset
    {α : Type u_1} (S : Set (Set α))
    (h : ∀ c ⊆ S, IsChain (fun (x1 x2 : Set α) => x1 ⊆ x2) c
        → ∃ lb ∈ S, ∀ s ∈ c, lb ⊆ s) :
            ∃ (m : Set α), Minimal (fun (x : Set α) => x ∈ S) m
```

Theorem 3.6 (Cantor's Intersection Theorem)**.** *The intersection of a non-empty directed family of non-empty compact closed sets is non-empty.*

```
theorem IsCompact.nonempty_sInter_of_directed_nonempty_isCompact_isClosed
    {S : Set (Set X)} [hS : Nonempty S] (hSd : DirectedOn ( · ⊇ · ) S)
    (hSn : ∀ U ∈ S, U.Nonempty) (hSc : ∀ U ∈ S, IsCompact U)
    (hScl : ∀ U ∈ S, IsClosed U) :
    (⋂₀ S).Nonempty
```

We now turn to the proof of Theorem 3.1. In what follows, we aim to be as detailed as possible to highlight the parallel with the implementation of the proof in Lean.

Proof of Theorem 3.1. Let M be an additive monoid acting on a non-empty compact topological space X. Define the family

$$S := \{Y \subseteq X \mid Y \neq \emptyset,\ Y \text{ closed},\ MY \subseteq Y\}.$$

We want to apply Zorn's lemma to find a minimal element in S. Let $C \subseteq S$ be any chain. Observe that if $C = \emptyset$ then we can take $l := X$ and we are done. Hence we may assume that $C \neq \emptyset$, which is important to be able to apply Theorem 3.6. Now define

$$l := \bigcap_{Y \in C} Y \subseteq X.$$

We verify that $l \in S$.

- Since for all $Y \in C$ we have $Y \subseteq X$ we obtain $l \subseteq X$.
- Since l is an intersection of closed sets it is closed.
- First observe that for every $Y \in C$ we have $Y \in S$ and thus $Y \subseteq X$ and Y is closed by definition of S. Since X is compact this implies that Y is compact. Moreover, every $Y \in C$ is non-empty by definition of S. Hence by Theorem 3.6 we obtain $l \neq \emptyset$.
- Let $y \in l$ and $m \in M$ be arbitrary. Then we have

$$\forall Y \in C :\ my \in Y$$

since $C \subseteq S$. This implies

$$my \in \bigcap_{Y \in C} Y = l.$$

Since $y \in l$ and $m \in M$ were arbitrary we obtain $Ml \subseteq l$.

This proves $l \in S$. Moreover, by definition of l we have

$$\forall Y \in C : \ l \subseteq Y.$$

Hence by Theorem 3.5 there exists an element $Y \in S$ such that

$$\forall Z \in S : \ Z \subseteq Y \implies Z = Y. \tag{1}$$

Observe that by definition of S we have $MY \subseteq Y$ and thus by Lemma 3.3 the restricted action

$$M \times Y \to Y, \ (m, y) \mapsto my$$

is well-defined. Moreover, since M acts continuously on X, by Lemma 3.4 we obtain that the restricted action of M on Y is again continuous.

We now show that (Y, M) is minimal using Proposition 3.2. Let $E \subseteq Y$ be any closed subset with $ME \subseteq E$ and assume that $E \neq \emptyset$. Then we have $E \in S$ by definition of S. Hence using (1) we obtain $E = Y$. This proves that (Y, M) is minimal by Proposition 3.2. □

The full implementation into Lean 4 can be found on github.com/ericceg/Topological-Dynamics-in-lean4.

4 Recurrent points

We now want to continue our discussion to recurrent points. Naively we are interested in points x where the orbit of T at that point comes back to it infinitely often. This means that there exists some strictly increasing $(n_k)_{k\in\mathbb{N}} \in \mathbb{N}^{\mathbb{N}}$ such that:

$$x = \lim_{k\to+\infty} T^{[n_k]}(x), \tag{5}$$

or more formally:

Definition 4.1. (Recurrent point) Let X be a topological space, $T : X \to X$ a homeomorphism. Then $x \in X$ is called a *recurrent point* or *recurrent* if:

$$x \in \bigcap_{n\in\mathbb{N}} \overline{\mathrm{Orb}(T^{[n]}(x))}, \tag{6}$$

where we consider the orbit under the action induced by T. The set $\bigcap_{n\in\mathbb{N}} \overline{\mathrm{Orb}(x)}$, written as $\omega^+(x)$ is called the omega-limit of x.

This definition can also be formulated in a weaker form using an additive monoid M as opposed to $\mathbb{N}$. We will implement this definition as follows in Lean 4:

```
def StronglyRecurrent {X : Type*} [TopologicalSpace X] {M : Type*}
    [AddMonoid M] [AddAction M X] (x: X): Prop:=
    x ∈ ⋂ (m : M), closure (AddAction.orbit M (m + x))
```

We will now be interested in when these recurrent points exist. It is in fact a corollary from theorem 3.1 that the following is true:

Theorem 4.2. *Let X be a compact topological space, $T : X \to X$ be a homeomorphism. Then there exists a recurrent point of X.*

We present a proof from Einsiedler (see [EW10]).

Proof. There exists a closed and non-empty $T-$invariant subspace $Y \subseteq X$ such that $T|_Y : Y \to Y$ is minimal by theorem 3.1. As Y is non-empty, we may pick $x \in Y$. We claim that x is recurrent. It is clear that $\omega^+(x)$ is $T|_Y-$invariant and closed (the intersection of closed subsets is closed). Thus we also have that $\omega^+(x) \subseteq Y$ and hence by theorem 1.5 we have that:

$$\omega^+(x) \in \{Y, \varnothing\}. \tag{7}$$

But $\omega^+(x)$ is non-empty! indeed:

$$\forall n \in \mathbb{N} : \mathrm{Orb}(T^{[n]}(x)) \supseteq \mathrm{Orb}(T^{[n+1]}(x)). \tag{8}$$

Hence this also holds for their closures and thus:

$$\omega^+(x) = \bigcap_{n\in\mathbb{N}} \mathrm{Orb}(T^{[n]}(x)) \neq \varnothing. \tag{9}$$

Thus we conclude that $\omega^+(x) = Y$ and as $x \in Y$ then x is a recurrent point. □

Indeed, this proof shows that an action is minimal if and only if every point is recurrent.

4.1 Implementation in Lean 4

The author first tried to implement the proof of theorem 4.2 faithfully and in the strong setting we presented above. This led to a number of problems which we will discuss further on. We remark that a draft for the proof of the theorem in the setting above and following the proof above consists of roughly 300 lines of code. The revised and completed version of the proof that works in a more general setting only consists of roughly 150 lines of code. The full code can be found on github.com/ericceg/Topological-Dynamics-in-lean4.

We prove the following revised version of theorem 4.2:

Theorem 4.3. *Let X be a compact, non-empty topological space and let M be an additive monoid acting continuously on X from the left. Then there exists a recurrent point $x \in X$ of that action.*

This theorem is implemented in Lean 4 as follows:

```
theorem exists_strongly_recurrent
  {X : Type*} [TopologicalSpace X] [CompactSpace X] [Nonempty X]
  {M : Type*} [AddMonoid M]
  [ContinuousAddAction M X] :
  ∃ x, StronglyRecurrent (M := M) (X := X) x
```

We use the definition of recurrent as we gave it in section 4 above. Henceforth in this chapter X shall always denote an arbitrary but non-empty and compact topological space and M shall be a monoid acting additively and continuously from the left on X.

The proof now works differently to how it worked in the naive case. We have split it up into multiple lemmata. We first show that if an action is minimal, all points are recurrent.

Lemma 4.4. *If the action of M on X is minimal, then every point of X is recurrent.*

Notice that we used this in the naive proof above but for the action induced by $T|_Y$ in the setting above. The proof of this theorem relies on theorem 1.5. Indeed it suffices to show that for all $x \in X$:

$$\forall m \in M : x \in \overline{\mathrm{Orb}(m + x)}. \tag{10}$$

But by minimality of the action we immediately know:

$$\forall m \in M : \overline{\mathrm{Orb}(m + x)} = X, \tag{11}$$

using theorem 1.5 or in the case of the Lean implementation (see [Kud21]):

```
theorem MulAction.dense_orbit [IsMinimal M α] (x : α) : Dense (orbit M x)
```

The Lean 4 implementation of lemma 4.4 is:

```
lemma Minimal_strongly_recurrent (x : X)
    [AddAction.IsMinimal M X] :
    StronglyRecurrent (M := M) x
```

A tedious issue when we tried to implement the naive code was dealing with conversions from points in a subspace to the upstairs points or with subactions of actions on subspaces. This partially explains the length of the naive proof as we have to tell Lean at every usage of one of these translations (which are obvious to the mathematician) that they are indeed true and provide it with a proof.

We provide some examples. The following code snippets serve to illustrate the inconvenience of dealing with subspaces and subactions in a strict setting as the one we tried to implement here.

In the following snippet we obtain our subspace Y as we did in the proof above using theorem 3.1. We then need to tell Lean that there is a subaction on that space and what it is. Then we get to pick our point in Y using that it is non-empty. But this we do by picking a point (x in the code below) in the set X with the hypothesis that $x \in Y$. We then need to define a new point x' as an element of Y which is equal to x upstairs.

```
...
obtain ⟨Y, hSubAction, hY_nonempty, hY_closed, hY_minimal⟩ :=
  exists_minimal_invariant_subset (M := ℕ) (X := X)

let _ : AddAction ℕ Y := hSubAction.SubAction

-- Y is not empty, so there exists some element, call it x
  obtain ⟨x, hx⟩ := hY_nonempty
...
let x' : Y := ⟨x, hx⟩
...
```

This serves to illustrate some of the problems arising from trying to deal with subspaces as one would in usual mathematics. Another example is:

```
    change Subtype.val (c + z) ∈ Z
    rw [hSubAction.SubAction_eq_Action c z]
```

where we use the lemma:

```
hSubAction.SubAction_eq_Action : ∀ c (y : Y), ↑(c + y) = c + (y : X)
```

This effectively means rewriting `↑ (c + z)` to `c + (z : X)` or as one would write naively:

$$\forall m \in M \forall z \in Y : m + z = m + z, \tag{12}$$

but where the left $+$ is the additive action of M on Y and the $+$ on the right is the additive action of M on X and on the left we consider z as an element of $Y \subseteq X$ and on the right we consider z as an element of X.

Remark 4.5 (The coercion operator $\uparrow$)**.** A set in Lean does not consist of elements as it does in naive set theory or even ZFC[1]. A set is actually a collection of things with proofs that they lie in some set. For example, the information that $x \in Y$ (where $Y \subseteq X$) would be displayed by $\langle$`x, hxY`$\rangle$, where `x` would be some point $x \in X$ (or in Lean notation: `x : X`) and `hxY` is a proof that $x \in Y$. The *coercion operator* now allows us to move between the two. Now if $y \in Y$ is defined as an element of Y and we know that $Y \subseteq X$ then $\uparrow$ `y` makes `y` into an element of X.

Now in our new and improved proof we want to minimize issues and difficulty arising from this. We therefore use two lemmata. A first one says that if X and M are again as usual, any point of Y is recurrent under a minimal action on Y. Its Lean 4 implementation is:

```
lemma StronglyRecurrent.of_minimal_subaction
    {X : Type*} [TopologicalSpace X]
    {M : Type*} [AddMonoid M] [AddAction M X]
    {Y : Set X} [MySubAddAction M X Y]
    (hmin: AddAction.IsMinimal M Y)
    {y : Y} :
  StronglyRecurrent (M := M) (X := X) (y : X)
```

We then go on to show a second lemma, which says that if a point $y \in Y$ is recurrent (notice that this means that y is recurrent with respect to the downstairs action $M \times Y \to Y$) then y (but now as an element of X) is also recurrent with respect to the upstairs action $M \times X \to X$. The Lean 4 implementation looks as follows:

[1] Even in ZFC a set is still defined by its element. This is the axiom of existentionality: $\forall x \forall y (\forall z : z \in x \iff z \in y) \implies x = y$.

```
lemma StronglyRecurrent.of_subaction
    {X : Type*} [TopologicalSpace X]
    {M : Type*} [AddMonoid M] [AddAction M X]
    {Y : Set X} [MySubAddAction M X Y]
    {y : Y}
    (hy : StronglyRecurrent (M := M) (X := Y) y) :
  StronglyRecurrent (M := M) (X := X) (y : X)
```

This is again "trivial" to a human mathematician as the orbit of y under the downstairs action $M \times Y \to Y$ is the "same" as the orbit under the upstairs action $M \times X \to X$, so it makes sense that the recurrent property is unchanged. We remark that this lemma still has to deal with the difficulties we discussed. In fact the exact same argument is found in its proof, namely:

```
    have h1 : ↑(m +ᵥ y : Y) = m +ᵥ (y : X) :=
            MySubAddAction.SubAction_eq_Action m y
```

We summarize the two lemmata in regular mathematical terms.

Lemma 4.6 (StronglyRecurrent.of minimal subaction)**.** *Let $Y \subseteq X$ and assume that the subaction $M \times Y \to Y$ is minimal, then every point of Y is recurrent.*

Lemma 4.7 (StronglyRecurrent.of subaction)**.** *Let $Y \subseteq X$ be non-empty, let $y \in Y$ and assume that y is a recurrent point of the subaction $M \times Y \to Y$. Then y is also a recurrent point of the action $M \times X \to X$.*

Notice that if we combine the two lemmata, we get immediately that:

Corollary 4.8. *Let $Y \subseteq X$ and assume that the subaction $M \times Y \to Y$ is minimal, then every point of Y is a recurrent point of the action $M \times X \to X$.*

This consequence is immediate and did not require an explicit formulation as a separate theorem. We are not able to understand the full proof of our theorem.

```
theorem exists_strongly_recurrent
  {X : Type*} [TopologicalSpace X] [CompactSpace X] [Nonempty X]
  {M : Type*} [AddMonoid M]
  [ContinuousAddAction M X] :
  ∃ x, StronglyRecurrent (M := M) (X := X) x := by {
```

The proof begins by applying theorem 1.5 to obtain a non-empty, closed subset Y of X such that the subaction $M \times Y \to Y$ on Y is minimal.

```
  obtain ⟨Y, hSubAction, hY_nonempty, hY_closed, hY_minimal⟩ :=
  exists_minimal_invariant_subset (M := M) (X := X)
```

We then invoke the hypothesis that $Y \neq \varnothing$ to obtain a point $x_0 \in X$ together with a proof of the proposition $x_0 \in Y$.

```
  obtain ⟨x0, hx0Y⟩ := hY_nonempty
```

Now, x_0 is formally not a point of Y. So we call $y \in Y$ the element of Y which lifts to x_0, i.e. `↑y = x0`. A human mathematician would omit this step as it boils down to just saying $y := x_0$.

```
  let y : Y := ⟨x0, hx0Y⟩
```

Now, we establish that the action $M \times Y \to Y$ is minimal. This is just using one of the properties of Y we got from applying theorem 1.5.

```
  haveI : AddAction.IsMinimal M Y := hY_minimal
```

We can now apply lemma 4.6 (which in Lean we called `Minimal_strongly_recurrent`) to show that the point y is a recurrent point of the subaction.

```
  have hy_recurrent_in_Y :
    StronglyRecurrent (M := M) (X := Y) y :=
    Minimal_strongly_recurrent y
```

We can then apply lemma 4.7 (which in Lean we called `StronglyRecurrent.of_subaction`) to show that the point x_0 is a recurrent point of the upstairs action.

```
  have hx_recurrent_in_X :
    StronglyRecurrent (M := M) (X := X) (y : X) :=
      StronglyRecurrent.of_subaction hy_recurrent_in_Y
```

We can then conclude as x_0 is a recurrent point of the upstairs action, which is whose existence we wanted to show.

```
  use x0
}
```

4.2 A recurrent point of the subaction is recurrent

The core of the proof is lemma 4.7 which says that if $y \in Y$ is a recurrent point of the action $M \times Y \to Y$, then y (but viewed as an element of X or even more specifically, the element of X to which y is lifted by the coercion operator) is also a recurrent point of the upstairs action $M \times X \to X$.

This is the core of the proof because it is the most opaque in the sense that it seems really trivial, which makes it the hardest part to implement in Lean.

The author would like to thank Jaume De Dios Pont for his implementation of this lemma.

We again walk through this proof line by line.

```
lemma StronglyRecurrent.of_subaction
    {X : Type*} [TopologicalSpace X]
    {M : Type*} [AddMonoid M] [AddAction M X]
    {Y : Set X} [MySubAddAction M X Y]
    {y : Y}
    (hy : StronglyRecurrent (M := M) (X := Y) y) :
  StronglyRecurrent (M := M) (X := X) (y : X) := by {
```

We begin by writing out the definition of strong recurrence. This turns the goal from ⊢`StronglyRecurrent↑ y` into ⊢ `∀(i : M),↑y∈closure (AddAction.orbit M (i+↑y))`.

```
    apply Set.mem_iInter.2
```

We want to show the new goal. For this we pick an arbitrary $m \in M$.

```
    intro m
```

We have as part of our assumption that $y \in Y$ is a recurrent point. Hence we know that $\forall m \in M : y \in \overline{\mathrm{Orb}(m+y)}$, where the action is $M \times Y \to Y, (n, z) \mapsto n + z$ but for $z = m + y$. Notice that the orbit (and its closure) are then subsets of Y. Here we now show that $y \in \overline{\mathrm{Orb}(m+y)}$ but where y is viewed as an element of X and the orbit is viewed as the orbit of the action $M \times X \to X$, so is a subset of X. We use:

```
/--closure_subtype :
  ∀ {α : Type*} [TopologicalSpace α] {s : Set S} (t : Set S),
  closure (t : Set α) = closure ((Subtype.val : S → α) '' t)-/
```

which is now applied as follows:

```
    have hX' : (y : X) ∈
      closure ((Subtype.val) '' (AddAction.orbit M (m + y) : Set Y)) :=
        (closure_subtype).1 ((Set.mem_iInter.1 hy) m)
```

We now show that the lift of the orbit of $m + y$ (where y is viewed as an element of Y) under the action $M \times Y \to Y$ (which is a subset of X) is actually a subset of the orbit of $m+ \uparrow y$ (so where $\uparrow y$ is the lift of y, i.e. now an element of X) under the action $M \times X \to X$. It is not hard to see, that actually there is equality here. But we won't need that.

```
    have hsubset :
        (Subtype.val) '' (AddAction.orbit M (m + y) : Set Y) ⊆
        (AddAction.orbit M (m + (y : X)) : Set X) := by
      rintro z ⟨z', ⟨c, rfl⟩, rfl⟩
```

We first bring up the action. We do this in two steps. First we show that if $y \in Y$ is an element of Y, then the lift of $m + y$ is the same as $m+$ the lift of y.

```
have h1 : ↑(m + y : Y) = m + (y : X) :=
      MySubAddAction.SubAction_eq_Action m y
```

Next, if y is viewed as an element in Y then $m + y$ is also viewed as an element of Y because Lean uses the action $M \times Y \to Y$. So we want to show that the lift of $c + (m + y)$ is the same as $c+$ the lift of $m + y$.

```
have h2 : ↑(c + (m + y) : Y) = c + ((m + y):X) :=
  MySubAddAction.SubAction_eq_Action c (m + y)
```

We apply hypothesis `h1` to change the hypothesis `h2`. Right now the hypothesis looks like: `have h2 : ↑(c + (m + y) : Y) = c + ((m + y):X)`. We use a rewrite to change it into: `↑(c + m + y) = c + m + ↑y`.

```
rw [h1] at h2
```

We use `h2` to obtain the inequality in the hypothesis `hrewrite`:

```
have hrewrite : ↑(c + (m + y) : Y) = c + m + (y : X) := h2
```

Now, we see that the right-hand side of `hrewrite` is just the ambient orbit upstairs in X.

```
have : (c + (m + (y : X)))
    ∈ (AddAction.orbit M (m + (y : X)) : Set X) := ⟨c, rfl⟩
```

This `simp` completes the proof of hypothesis `hsubset`.

```
simp [hrewrite]
```

We have now nearly shown what we claimed. We have shown that all the downstairs orbits are contained in the upstairs orbit. But we need to show the analogous thing for the respective closures of the orbits. Here, we use the theorem:

```
/--theorem closure_mono (h : s ⊆ t) : closure s ⊆ closure t-/
```

from the `mathlib4` library to conclude.

```
    exact (closure_mono hsubset) hX'
}
```

The full implementation into Lean 4 can be found on github.com/ericceg/Topological-Dynamics-in-lean4.

5 References

[BS02] Michael Brin and Garrett Stuck. *Introduction to dynamical systems*. Cambridge university press, 2002.

[EW10] Manfred Einsiedler and Thomas Ward. *Ergodic Theory: With a View Towards Number Theory*, volume 259 of *Graduate Texts in Mathematics*. Springer London, 2010.

[Kud21] Yury Kudryashov. Minimal.lean: Mathlib 4, dynamics, topological entropy. `https://github.com/leanprover-community/mathlib4/blob/master/Mathlib/Dynamics/Minimal.lean`, 2021. Accessed 2025-05-06.