TEG, checked by Lean

The premises and theorems of the three papers, as Lean 4 states them. Each line is copied from the file. The proofs are left out; the files have them. `lean <file>` prints nothing when every proof is accepted. A check shows the premises agree, or that a premise is needed.

Paper 1: Why Existence Cannot Ground Itself

`Grounding.lean`

PREMISES

premise

```
indiscernible : ∀ x y : Thing, (∀ p, hasProp x p ↔ hasProp y p) → x = y
```

Step 4: whatever tells two things apart is a property. So two things with all properties in common, none included, are one.

premise

```
something : ∃ x p, hasProp x p
```

Something exists. The paper takes this for granted.

premise

```
premise : ∀ S : Thing → Prop, (∃ x, S x) → (∀ x, S x → ∃ p, hasProp x p) →
  ∃ c, ¬ S c ∧ ∀ x, S x → causes c x
```

Step 1, the premise, as step 3 uses it: whatever has properties, taken alone or as a whole with others, has an account, and the account lies outside that whole. A loop or an endless chain is a whole whose parts account only for each other.

premise

```
time_is_property : ∀ x t, atTime x t → ∃ p, hasProp x p
```

Step 5: a place in time is a property.

STEPS 2 TO 6

definition

```
abbrev InExistence (x : W.Thing) : Prop
```

Step 2: existence is everything with properties.

theorem

```
theorem step3 : ∃ c, ¬ W.InExistence c ∧ ∀ x, W.InExistence x → W.causes c x
```

Step 3: there is a cause of everything in existence, and it has no properties.

theorem

```
theorem step4 : ∃ pp, ¬ W.InExistence pp ∧ (∀ x, W.InExistence x → W.causes pp x) ∧
    ∀ c, ¬ W.InExistence c → c = pp
```

Step 4: that cause is one. Call it Pure Potential.

theorem

```
theorem step5 (pp : W.Thing) (h : ¬ W.InExistence pp) (t : W.Time) : ¬ W.atTime pp t
```

Step 5: what has no properties has no place in time.

theorem

```
theorem step6 : ∃ pp, ¬ W.InExistence pp ∧ (∀ t, ¬ W.atTime pp t) ∧
    ∀ t x, W.atTime x t → W.causes pp x
```

Step 6: Pure Potential, with no place in time, causes everything that is at any instant.

CHECKS

check

```
def tiny : World
```

The premises do not contradict each other. In this world `true` is a rock, with one property, and `false` has none and causes the rock.

check

```
theorem links_are_not_enough : …
```

The premise read link by link does not reach step 3. Here the things are the numbers 0, 1, 2 and so on, each with its own property, and n + 1 causes n. Every thing has properties and a cause other than itself, and nothing is free of properties.

Paper 2: The Ground Must Choose

`FreeWillGround.lean`

PREMISES

from paper 1

```
pp_free  : ¬ ∃ p, hasProp pp p
```

Paper 1, steps 3 to 6: Pure Potential, which has no properties.

premise

```
binder_is_property : ∀ x, bindsWithin x → ∃ p, hasProp x p
```

Step 7: a binder inside a thing is a property of that thing.

premise

```
order_needs_binding : orderLasts → bindsWithin pp ∨ keeps pp
```

Step 8: for lasting order to have an account, something must bind the ground's causing: a binder inside the ground, or what the causing already made. Causing that is unbound and keeps nothing could give order only by chance.

observation

```
order_lasts : orderLasts
```

The observation.

STEPS 7 TO 9

definition

```
def Chooses (x : G.Thing) : Prop
```

Step 9's word: causing that nothing inside binds, and that keeps what it made.

theorem

```
theorem step7 : ¬ G.bindsWithin G.pp
```

Step 7: nothing inside Pure Potential binds its causing.

theorem

```
theorem step8 : G.keeps G.pp
```

Step 8: Pure Potential keeps what it made.

theorem

```
theorem step9 : G.Chooses G.pp
```

Step 9: what is left is choice.

CHECKS

check

```
theorem keeping_is_no_feature (h : ∀ x, G.keeps x → ∃ p, G.hasProp x p) : False
```

Keeping must not count as a feature of the ground: if every thing that keeps had a property for it, the premises would contradict each other.

check

```
def tiny : Ground
```

The premises do not contradict each other. `true` is a rock with one property; `false` is the ground, with none, and it keeps what it made.

Paper 3: What Is Aware Is the Ground

`Aware.lean`

PREMISES

from paper 1

```
indiscernible : ∀ x y : Thing, (∀ p, hasProp x p ↔ hasProp y p) → x = y
```

Paper 1, step 4: whatever tells two things apart is a property.

proved below

```
observation_premise : ∀ x, (∃ p, hasProp x p) → couldShowUp x
```

The premise of observation: whatever has properties could show up for someone.

observation

```
observer_unseen : ∀ a L, awareOf a L → ¬ couldShowUp a
```

The observer is never observed: what is aware enters no view.

observation

```
there_is_awareness : ∃ a L, awareOf a L
```

You know that there is awareness, by being it.

STEPS

theorem

```
theorem no_properties (a : A.Thing) (L : A.Line) (h : A.awareOf a L) :
    ¬ ∃ p, A.hasProp a p
```

What is aware has no properties.

theorem

```
theorem not_two (a b : A.Thing) (L M : A.Line) (ha : A.awareOf a L)
    (hb : A.awareOf b M) : a = b
```

Not two: what is aware of my line and what is aware of yours are the same.

theorem

```
theorem is_the_ground (a : A.Thing) (L : A.Line) (h : A.awareOf a L) : a = A.pp
```

What is aware in anyone is the ground.

THE PREMISE OF OBSERVATION, FROM KEEPING

from paper 1

```
made : ∀ x, (∃ p, hasProp x p) → ∃ t, madeAt x t
```

Paper 1, step 6: everything with properties is made at some instant.

premise

```
no_last : ∀ t, ∃ t', before t t'
```

Every instant has a later one. Without it, what is made at a last instant would never be fitted, and would need not show up for anyone.

from paper 2

```
keeping : ∀ x t t', madeAt x t → before t t' → fitsAt pp x t'
```

Paper 2, step 9: at each instant the causing fits what was made before.

premise

```
link : ∀ g x t, fitsAt g x t → showsUpFor x g
```

The link: fitting what was made needs access to it. What was made shows up for the one fitting.

theorem

```
theorem observation_premise : ∀ x, (∃ p, K.hasProp x p) → ∃ s, K.showsUpFor x s
```

The premise of observation follows: whatever has properties shows up for the ground.

theorem

```
theorem keeping_chain : …
```

Paper 3's chain still goes through: what is aware is the ground.

CHECKS

check

```
theorem aware_of_is_no_feature (h : ∀ a L, A.awareOf a L → ∃ p, A.hasProp a p) :
    False
```

Being aware of a line must not count as a feature: if everything aware of a line had a property for it, the premises would contradict each other.

check

```
def tinyAware : Aware
```

With paper 3's premises added, the tiny world is still a world: one awareness, two lines, and it is the ground.

check

```
theorem observation_is_its_own_premise : …
```

The premise of observation does not follow from paper 1's premise. In this world every thing with properties has its account outside itself, as paper 1 asks, yet the rock could never show up for anyone.

check

```
theorem link_not_implied : …
```

The premise of observation does not give the link. Here everything with properties shows up for a creature, and the ground fits the rock, yet the rock does not show up for the ground. (`none` is the ground, `some true` the rock, `some false` the creature.)

check

```
theorem link_alone_is_not_enough : …
```

The link alone does not give the premise of observation: where nothing is fitted, the link is true and a rock can show up for no one. Keeping does the work.