

МАШИННО-ПРОВЕРЕННЫЙ ОТЧЁТ · LEAN 4

Аксиоматика «Обелиска»: аудит, ремонт и реализация ТВІ-пунктов

Разбор демонстрации пруф-ассистента «Обелиск» (вариант HOL, 2026-05-05). Каждое утверждение этого отчёта о выводимости или противоречивости проверено проверщиком Lean и снабжено точным списком использованных аксиом.

объект: `obeliskproofassistant.github.io`автор системы: **Георгий Дунаев**проверка: **Lean 4.32.0**, без **MathLib**дата: **2026-07-22****2**

независимых вывода
противоречия в
напечатанной системе

9

пунктов демо доказано
из напечатанного ядра

3

ТВІ-пункта (3.45–3.47)
реализовано и
верифицировано

0

ошибок · 0 sorry ·
полный аудит `#print
axioms`

Резюме. «Обелиск» — содержательная система: теория классов уровня Морса–Келли с аксиомой Тарского, ϵ -индукцией в изящной форме степени (3.14) и оригинальными парами для собственных классов. Но напечатанный список аксиом **противоречив**: в определении 3.34 ($\leq$) пропущена инъективность, и пустая функция делает $\leq$ тотальным. Отсюда двумя независимыми путями — через аксиому ограничения размера 3.37 и через аксиому Тарского 3.66 — выводится $\perp$. Ремонт минимален (см. §4). Попутно: пары Куратовского несовместимы с определением 3.29, в списке не хватает аксиом пары и объединения, а помеченные «ТВІ» пары Дунаева здесь реализованы и доказаны — они действительно работают для собственных классов, ради чего и задуманы.

1 · Что такое «Обелиск» математически

Ядро — теория *классов*, где «быть множеством» — определяемый предикат

(3.02: $X \text{ set} \leftrightarrow \exists Y. X \in Y$), а выделение классов импредикативно (3.01: формула φ может квантифицировать по классам — это уровень Морса–Келли, сильнее NBG). Поверх ядра: Heredity (3.12), Exponent (3.13), «Regularity» (3.14) — на самом деле $\in$ -индукция в классовой форме, Infinity (3.53), фон-неймановское ограничение размера (3.37) и аксиома Тарского (3.66) — родня Tarski–Grothendieck, на которой стоит Mizar. Метачасть (разделы 4 и 6 демо) добавляет кавычки $\langle \rightarrow \rangle / \llbracket _ \rrbracket$ и внутренний предикат выводимости с целью доказать теорему Лёба и вторую теорему Гёделя внутри системы — программа в духе рефлексивных замыканий Фефермана.

Это серьёзная работа с настоящими оригинальными идеями. Именно поэтому она заслуживает того инструмента, для которого и создаются пруф-ассистенты: машинной проверки.

2 • Находки

НАХОДКА 1 $\not\leq$ без инъективности $\Rightarrow$ противоречие через аксиому 3.37

3.34) Def. “Is not strictly larger than” binary relation:

$$(\forall (A:\text{Class}) \forall (B:\text{Class})) ((A \leq B) \leftrightarrow (\exists (f:\text{Class})) ((f \text{ class-function}) \wedge (f[A] \subseteq B)))$$

▲ нет ни инъективности, ни усл

Пустой класс $\emptyset$ — class-function по определениям 3.29–3.33 (все условия вакуумны: $\text{Dom}(\emptyset) = \emptyset$). Его образ пуст: eval (3.25) выдаёт default-значение V , а V — собственный класс (Рассел + Heredity), так что ни одно множество b не равно V . Значит $\emptyset[A] = \emptyset \subseteq V$, то есть $A \leq V$ для всех A, V — и «строго меньше» (3.36) не выполняется никогда. Тогда аксиома 3.37, $X \in V \leftrightarrow |X| < |V|$, говорит, что V пусто — а по аксиоме 3.53 в нём лежит ω . Противоречие.

✓ Lean: PrintedWleq_total, printed_inconsistent_route1 : False — только из {CCR, ext, heredity, infinity} + гипотеза 3.37. Доказательство не зависит от кодировки пар: пустая функция не строит ни одной пары.

НАХОДКА 2 Тот же баг убивает универсумы: аксиома Тарского 3.66 противоречива, теорема 3.65 опровержима

Предикат `Tight` (3.41) использует $<$: раз $<$ ложно всегда, `Tight(U)` означает « $\forall V \subseteq U: V \not\subseteq U$ ». Но универсум (3.64) содержит ω и Hereditary-замкнут (3.38), откуда $\emptyset \in U$; при этом $\emptyset \subseteq U$, и `Tight` требует $\emptyset \notin U$. Значит **универсумов не существует вовсе**: аксиома Тарского 3.66 даёт $\perp$ сама по себе, а заявленная в демо *теорема* 3.65 « $\forall$ universe» — опровержима в напечатанной системе.

```
✓ Lean: no_universe , printed_inconsistent_route2 : False ,
printed_3_65_refutable .
```

НАХОДКА 3**Пары Куратовского + квантор $\exists!$ по классам = «фантомные партнёры» в 3.29**

Для собственного класса Y пара Куратовского вырождается: $\{x, Y\} = \{x\}$, откуда $(x, Y) = (x, x)$. Поэтому если граф F содержит диагональную пару (x, x) , единственность из 3.29 ($\exists! (y : \text{Class}) ((x, y) \in F)$) недостижима: $y := V$ — второй свидетель. Тожественная функция перестаёт быть class-function. Это *доказывает*, что машинерия функций «Обелиска» обязана опираться на собственные пары Дунаева (3.46), а не на 3.48 — конструкция 3.45–3.47 не украшение, а необходимость.

```
✓ Lean: kpair_phantom , no_unique_partner .
```

НАХОДКА 4**В списке аксиом нет Пары и Объединения — и их нельзя восстановить**

`Pairful` — часть определения универсума (3.64), и теорема 3.65 фактически анонсирует аксиому пары для V . Теги $\{\{\{a\}\}, n\}$ в парах Дунаева должны быть множествами — без аксиомы пары это недоказуемо. Восстановить пару из (исправленного) ограничения размера нельзя: $\leq$ определяется через class-functions, чьи графы уже состоят из упорядоченных пар — циркулярность. Аудит Lean подтверждает: `dpair_inj` требует ровно аксиому `pairing` (и, что приятно, не требует объединения).

```
✓ Lean: #print axioms dpair_inj → {CCR, ext, heredity, infinity, pairing}.
```

3 · Что выживает: девять пунктов демо,
доказанных из напечатанного ядра

Аудит честный в обе стороны: значительная часть заявленных теорем действительно выводима из напечатанных аксиом {CCR, ext, Heredity, Exponent, Regularity, Infinity} — и теперь у каждой есть машинное доказательство с точным списком использованных аксиом.

ДЕМО	УТВЕРЖДЕНИЕ	ТЕОРЕМА LEAN	АКСИОМЫ
3.07	$\emptyset \subseteq X$	thm_3_07	CCR
3.11	интенциональность	thm_3_11	CCR
3.18	Ord(On)	thm_3_18	только CCR
3.55	$\text{Ind}(\omega)$	thm_3_55	CCR·ext·her·inf
3.56	$\text{Ind}(\text{On})$	thm_3_56	CCR·her·exp·inf
3.57	$\text{Tr}(\omega)$	thm_3_57	CCR·ext·her·inf
3.58	$\text{Ord}(\omega)$	thm_3_58	CCR·ext·her·inf
3.59	$\omega \in \text{On}$	thm_3_59	CCR·ext·her·inf
3.67	индукция на транзитивном классе	thm_3_67	CCR + Regularity

Три наблюдения, которых нет в демо. **(а)** Пункт 3.67 выводится из одной лишь 3.14 — подтверждение, что «Regularity» Дунаева — это в точности $\in$ -индукция; формулировка через $\mathcal{A}(X) \subseteq X \rightarrow V \subseteq X$ корректна и красива. **(б)** $\text{Ind}(\omega)$ доказуема из ядра классическим трюком: либо индуктивный класс существует, либо $\omega = V$ — что противоречит 3.53 + Расселу (exists_inductive). **(в)** $\text{Ind}(\text{On})$ спасает транзитивность: для транзитивного а верно $a \cup \{a\} \subseteq \mathcal{A}(a)$, и множественность следует из Heredity+Exponent — без аксиом пары и объединения (isSet_succ_of_Tr). Единственный пункт, которому добавленные аксиомы правда нужны, — $\text{Ind}(V)$ (3.54), из-за нетранзитивных множеств.

4 · ТВІ-пункты 3.45–3.47 — реализованы и доказаны

Пары и списки Дунаева в демо помечены «TBI» (to be implemented). Здесь они реализованы, и их корректность доказана для произвольных классов, включая собственные — то, ради чего конструкция и задумана, и что пары Куратовского дать не могут (Находка 3):

```
dpr1_dpair : pr1(A,B) = A           -- для ЛЮБЫХ классов A, B
dpr2_dpair : pr2(A,B) = B
dpair_inj  : (A,B) = (C,D) → A = C ∧ B = D
dpair_proper_showcase : pr1(V,0n) = V ∧ pr2(V,0n) = 0n  -- собственные кла
dsingle_ne_dnil : [A] ≠ []           -- даже при A = ∅: маркер {0,n} p
```

Последняя строка — проверка главного дизайнерского решения конструкции: маркер $\{0,n\}$ делает длину списка восстановимой всегда, включая пустые компоненты. Это аккуратное усиление стандартного тегирования $A \times \{1\} \cup B \times \{2\}$, которое проблему $[\emptyset]$ vs $[]$ не решает.

5 • Минимальный ремонт

- 1 **3.34':** $A \leq B \iff \exists f (f \text{ class-function} \wedge A \subseteq \text{Dom } f \wedge f[A] \subseteq B \wedge f \text{ инъективна на } A)$. С этим исправлением 3.37 становится классическим фон-неймановским ограничением размера, а 3.41/3.66 — осмысленными.
- 2 **Добавить аксиомы Пары и Объединения** — стандартные для МК/NBG, верные во всех интендированных моделях; без них недоказуемы 3.54, 3.65 и собственные TBI-конструкции демо (Находка 4).
- 3 **3.29 читать через пары Дунаева** (либо ограничить $\exists!$ множествами) — иначе фантомные партнёры (Находка 3).
- 4 **Зафиксировать следствие:** 3.37' влечёт глобальный выбор (von Neumann, 1925) — это стоит сделать осознанным решением системы.
- 5 **Раздел 6:** внешняя теория в 6.02/6.03 должна быть сильнее Q — Q не доказывает условий выводимости Гильберта–Бернайса–Лёба; стандартный выбор — EA или IS_1 .

Согласованность отремонтированного ядра: модель — $V_{\kappa+1}$ для κ —

недостижимого предела недостижимых (множества = V_κ , классы = $V_{\kappa+1}$).

Система калибруется в окрестности МК + Limitation of Size + универсумы

Тарски–Гротендика — строго сильнее ZFC, как и задумано.

6 • Куда это развивать: программа «первоклассной математики»

Статья А (материал готов — этот отчёт + Lean-файл): «The Obelisk class theory: a machine-checked audit and repair» — находки, ремонт, верифицированная положительная часть, пары Дунаева как теоремы. Формат: ITP/CPP или arXiv с артефактом. **Статья В**: точная калибровка силы — взаимная интерпретируемость с МК+LoS+TG; кардинальная арифметика классов из 3.35 ($|On|=On$, $Card(On)$, $\neg Card(V)$ — пункты 3.60–3.63). **Статья С**: метатеория раздела 6 — типизированная кавычка/eval как модальность в духе GL, связь F с рефлексивным замыканием Фефермана и теориями истины; стратификация eval для класс-кванторов (обход теоремы Тарского); формализованные Лёб и Gödel II поверх существующих механизаций (Paulson в Isabelle; Lean-библиотека Foundation). **Инженерное**: экспорт доказательств «Обелиска» в проверяемый формат (Lean/Metamath) — тогда каждая строка демо получает независимую машинную проверку автоматически.

Что этот отчёт утверждает — и чего не утверждает

Проверен *напечатанный* список аксиом демо от 2026-05-05 — то, что опубликовано. Внутренняя реализация «Обелиска» может отличаться от печати (например, $\leq$ в коде может быть инъективным); тогда находки 1–2 — о выводе демо, а не о ядре системы. Формализация моделирует CCR лин-предикатами — это надмножество синтаксической схемы; существенно, что *каждый* использованный в доказательствах инстанс comprehension записывается формулой языка самого демо (Рассел, Dom, img, Ind и т.п. — собственные определения системы). Все прочие определения перенесены дословно, с сохранением нумерации.

7 • Воспроизведение

```
# установка Lean (однократно)
curl https://elan.lean-lang.org/elan-init.sh -sSf | sh -s -- -y --default-t

# проверка (из папки obelisk-analysis)
lean Obelisk.lean          # exit 0, без sorry; в конце – аудит #print axioms
```

▼ Полный исходник Obelisk.lean (самодостаточный, ~660 строк)

```
/-
```

```
=====
A machine-checked analysis of the axiom system of the "Obelisk"
proof assistant (higher-order-logic variant, demo dated 2026-05-05,
https://obeliskproofassistant.github.io/), by Georgy Dunaev.
```

```
Checked with Lean 4.32.0.  Self-contained: no Mathlib, only core Lean
```

```
Contents.
```

- Part 0. The base theory exactly as printed in the demo:
 - impredicative class comprehension (3.01), extensionality (3.02),
 hereditary membership (3.11), hereditary subset (3.12), exponent (3.13), regularity-as- ϵ -induction (3.53),
 infinity (3.53). These axioms are consistent (model: take an uncountable stage V_α of the cumulative hierarchy with α a
 cardinal $> \omega$, classes = subsets of V_α).
- Part 1. THE PRINTED SYSTEM IS INCONSISTENT, in two independent ways:
 - (a) definition 3.34 of $\leq$ has no injectivity requirement, so the empty class-function witnesses $A \leq B$ for ALL A, B ; hence the
 size-limitation axiom 3.37 forces V to have no members, contradicting axiom 3.53 (ω is a set);
 - (b) the same $\leq$ -bug propagates through "Tight" (3.41) into the universe predicate (3.64): NO class can be a universe,
 Tarski's axiom 3.66 is itself contradictory, and the announced theorem 3.65 ("V is a universe") is refutable as printed.
 Also: with Kuratowski pairs (3.48), "concentrated" (3.29) phantom partners – for a proper class Y , $(x, Y) = (x, x)$ – so
 uniqueness in 3.29 fails at every diagonal point. This shows that the function machinery must use the Dunaev pairs (3.46), which
 turn require Pairing and Union axioms absent from the printed list.
- Part 2. What survives: the announced theorems that ARE derivable from the
 printed base axioms alone, machine-checked:
 - 3.07, 3.11, 3.18 (Ord(0_n)), 3.55 (Ind(ω)), 3.56 (Ind(0_n)),
 3.57 (Tr(ω)), 3.58 (Ord(ω)), 3.59 ($\omega \in 0_n$), 3.67 (ϵ -induction on a transitive class), and the components of "V is a universe"
 that do not mention Tight.

```

Part 3. The repair: add Pairing and Union (they are NOT recoverable
limitation-of-size – the  $\leq$  machinery itself presupposes ordinals,
pairs, a circularity – and they are required by the demo's TBI items).
With them we implement and verify the demo's items 3.44–3.47: Dunaev's
tagged lists/pairs, which – unlike Kuratowski's – are faithful on
PROPER classes:
 $\text{pr1}(A,B) = A$ ,  $\text{pr2}(A,B) = B$ , and  $(A,B) = (C,D) \rightarrow A = C \wedge B = D$ 
for arbitrary classes, and  $[A] \neq []$  even for  $A = \emptyset$ 
(the  $\{0,n\}$  marker device works as designed).

=====
-/

set_option linter.unusedVariables false

namespace Obelisk
noncomputable section

/-! ### Part 0. Language and the printed base axioms -/

axiom Class : Type
axiom Mem : Class → Class → Prop
infix:50 " ∈ " => Mem

/-- 3.02: `X` is a set iff it is a member of some class. -/
def isSet (X : Class) : Prop := ∃ Y, X ∈ Y
/-- 3.03. -/
def proper (K : Class) : Prop := ¬ isSet K

/-- 3.01 (CCR): the class `{x | φ}`. The demo's `φ : Fm` ranges over formulas
with quantifiers over `Class` (Morse–Kelley-style impredicative
comprehension); we model it by a Lean-level predicate. -/
axiom cls : (Class → Prop) → Class
axiom mem_cls : ∀ (φ : Class → Prop) (x : Class), x ∈ cls φ ↔ (isSet x → φ x)

/-- 3.10 Extensionality. (3.08/3.09 are Lean's `Eq` congruence, for free)
axiom extAx : ∀ {X Y : Class}, (∀ z, z ∈ X ↔ z ∈ Y) → X = Y

/-- 3.06. -/
def Sub (X Y : Class) : Prop := ∀ a, a ∈ X → a ∈ Y
infix:50 " ⊆ " => Sub

/-- 3.04, 3.05. -/
def Null : Class := cls (fun _ => False)
def V : Class := cls (fun _ => True)
/-- Powerclass (used by 3.13, 3.14). -/
def power (X : Class) : Class := cls (fun y => y ∈ X)

```

```

def power (A : Class) : Class := cls (fun y => y ≤ A)

/-- 3.12 Heredity. -/
axiom heredity : ∀ {A B : Class}, A ≤ B → isSet B → isSet A
/-- 3.13 Exponent. -/
axiom exponent : ∀ {X : Class}, isSet X → isSet (power X)
/-- 3.14 "Regularity" – the powerclass form of  $\epsilon$ -induction. -/
axiom regularity : ∀ {X : Class}, power X ≤ X → V ≤ X

/-! Printed definitions 3.15–3.52, verbatim. -/

/-- 3.19, 3.20. -/
def sUnion (K : Class) : Class := cls (fun x => ∃ y, y ∈ K ∧ x ∈ y)
def sInter (K : Class) : Class := cls (fun x => ∀ y, y ∈ K → x ∈ y)
/-- 3.42, 3.43, 3.49, 3.50. -/
def sing (a : Class) : Class := cls (fun x => x = a)
def upair (a b : Class) : Class := cls (fun x => x = a ∨ x = b)
def bUnion (A B : Class) : Class := cls (fun x => x ∈ A ∨ x ∈ B)
def bInter (A B : Class) : Class := cls (fun x => x ∈ A ∧ x ∈ B)
/-- 3.48 Kuratowski pair (a,b) = {{a,a},{a,b}}. -/
def kpair (a b : Class) : Class := upair (upair a a) (upair a b)
/-- 3.21 Domain. -/
def Dom (F : Class) : Class := cls (fun x => ∃ b, kpair x b ∈ F)
/-- 3.22 IF-THEN-ELSE-FI (the demo's semantic definition). -/
def IfC (ψ : Prop) (A B : Class) : Class :=
  cls (fun v => (ψ ∧ v ∈ A) ∨ (¬ψ ∧ v ∈ B))
/-- 3.25 Evaluation, with the demo's `ELSE V` default. -/
def app (F a : Class) : Class :=
  IfC (∃ s, kpair a s ∈ F) (sUnion (cls (fun s => kpair a s ∈ F))) V
/-- 3.26 Image. -/
def img (f A : Class) : Class := cls (fun b => ∃ a, a ∈ A ∧ app f a = b)
/-- The demo's `∃!` quantifier (4.05): over ALL classes, proper inclusion. -/
def ExistsU (p : Class → Prop) : Prop := ∃ y, p y ∧ ∀ z, p z → z = y

/-- 3.27–3.33. -/
def isSingleton (s : Class) : Prop := ExistsU (fun x => x ∈ s)
def setValued (F : Class) : Prop := ∀ x, x ∈ Dom F → isSet (app F x)
def concentrated (F : Class) : Prop :=
  ∀ x, x ∈ Dom F → isSet (app F x) → ExistsU (fun y => kpair x y ∈ F)
def flatProper (F : Class) : Prop :=
  ∀ x, x ∈ Dom F → proper (app F x) → ∀ b, kpair x b ∈ F → isSingleton b
def isFamily (F : Class) : Prop := concentrated F ∧ flatProper F
def isClassFunction (F : Class) : Prop := isFamily F ∧ setValued F

/-- 3.34 AS PRINTED – note: no injectivity, no domain condition. -/
def PrintedWleq (A B : Class) : Prop :=
  -- ...

```

```

    ∃ t, isClassFunction t ∧ img t A ⊆ B
/-- 3.36 as printed:  $A < B \leftrightarrow \neg(B \leq A)$ . -/
def PrintedWlt (A B : Class) : Prop := ¬ PrintedWleq B A

/-- 3.15, 3.16, 3.17. -/
def Tr (A : Class) : Prop := ∀ B, B ∈' A → B ⊆' A
def OrdC (A : Class) : Prop := Tr A ∧ ∀ B, B ∈' A → Tr B
def On : Class := cls OrdC
/-- 3.35 Class cardinality (as printed; uses the printed ≤). -/
def card (K : Class) : Class :=
  bInter On (sInter (cls (fun a => a ∈' On ∧ PrintedWleq K a)))

/-- 3.51, 3.52. -/
def IndC (X : Class) : Prop :=
  Null ∈' X ∧ ∀ a, a ∈' X → bUnion a (sing a) ∈' X
def omega : Class := cls (fun x => ∀ Q, IndC Q → x ∈' Q)
def succ (a : Class) : Class := bUnion a (sing a)

/-- 3.53 Infinity. -/
axiom infinity : isSet omega

/-- 3.38–3.41 and 3.64, verbatim (Tight uses the printed <). -/
def Hereditary (A : Class) : Prop :=
  ∀ B, B ∈' A → ∀ X, X ⊆' B → X ∈' A
def Powerful (U : Class) : Prop := ∀ B, B ∈' U → power B ∈' U
def Pairful (U : Class) : Prop :=
  ∀ A B, A ∈' U → B ∈' U → upair A B ∈' U
def Tight (U : Class) : Prop :=
  ∀ B, B ⊆' U → (PrintedWlt B U ↔ B ∈' U)
def isUniverse (U : Class) : Prop :=
  ((omega ∈' U) ∧ ((Tr U ∧ Hereditary U) ∧ (Powerful U ∧ Tight U))) ∧ F

/#! ### Basic lemmas -/

theorem not_mem_Null (x : Class) : ¬ x ∈' Null :=
  fun h => ((mem_cls _ x).mp h).2

theorem Null_sub (X : Class) : Null ⊆' X :=
  fun a ha => absurd ha (not_mem_Null a)

theorem mem_V {x : Class} : x ∈' V ↔ isSet x :=
  (fun h => ((mem_cls _ x).mp h).1, fun h => (mem_cls _ x).mpr (h, trivial))

theorem isSet_of_mem {x Y : Class} (h : x ∈' Y) : isSet x := ⟨Y, h⟩

/-- ∅ is a set (from Heredity and Infinity). -/

```

```

theorem isSet_Null : isSet Null := heredity (Null_sub omega) infinity

/-- Russell: the universal class is proper (uses CCR and Heredity). -/
theorem V_proper : proper V := fun hV =>
  have hsub : cls (fun x => ¬ x ∈' x) ≤' V :=
    fun a ha => mem_V.mpr ((mem_cls _ a).mp ha).1
  have hset : isSet (cls (fun x => ¬ x ∈' x)) := heredity hsub hV
  have hiff := mem_cls (fun x => ¬ x ∈' x) (cls (fun x => ¬ x ∈' x))
  (Classical.em (cls (fun x => ¬ x ∈' x) ∈' cls (fun x => ¬ x ∈' x))).e
    (fun h => (hiff.mp h).2 h)
    (fun h => h (hiff.mpr {hset, h}))

theorem mem_upair {x a b : Class} :
  x ∈' upair a b ↔ isSet x ∧ (x = a ∨ x = b) := mem_cls _ _

theorem mem_sing {x a : Class} : x ∈' sing a ↔ isSet x ∧ x = a := mem_c

theorem self_mem_sing {a : Class} (h : isSet a) : a ∈' sing a :=
  mem_sing.mpr {h, rfl}

theorem upair_self (a : Class) : upair a a = sing a :=
  extAx (fun z =>
    {fun h => mem_sing.mpr ((mem_upair.mp h).1, (mem_upair.mp h).2.elim
      fun h => mem_upair.mpr ((mem_sing.mp h).1, Or.inl (mem_sing.mp h).

theorem sing_inj {a b : Class} (ha : isSet a) (h : sing a = sing b) : a
  (mem_sing.mp (h ▸ self_mem_sing ha)).2

/-! ### Part 1. Inconsistency of the printed system -/

/-- The empty class has empty domain, hence is (vacuously) a class-function
theorem Dom_Null_empty (x : Class) : ¬ x ∈' Dom Null := fun hx =>
  match ((mem_cls _ x).mp hx).2 with
  | {_, hb} => not_mem_Null _ hb

theorem Null_concentrated : concentrated Null :=
  fun x hx => absurd hx (Dom_Null_empty x)
theorem Null_flatProper : flatProper Null :=
  fun x hx => absurd hx (Dom_Null_empty x)
theorem Null_setValued : setValued Null :=
  fun x hx => absurd hx (Dom_Null_empty x)

theorem Null_isClassFunction : isClassFunction Null :=
  {Null_concentrated, Null_flatProper}, Null_setValued

/-- Evaluation of the empty class-function falls into the `ELSE V` branch

```

```

theorem app_Null (a : Class) : app Null a = V := by
  have hcond : ¬ ∃ s, kpair a s ∈' Null :=
    fun (_, hs) => not_mem_Null _ hs
  apply extAx
  intro v
  constructor
  · intro hv
    have h' := (mem_cls _ v).mp hv
    exact match h'.2 with
      | Or.inl (hψ, _) => absurd hψ hcond
      | Or.inr (_, hvV) => hvV
  · intro hv
    exact (mem_cls _ v).mpr (isSet_of_mem hv, Or.inr (hcond, hv))

/-- Its image is empty: no SET can equal `V`, the default value. -/
theorem img_Null_empty (A b : Class) : ¬ b ∈' img Null A := fun hb =>
  match (mem_cls _ b).mp hb with
  | (hbset, _, _, happ) =>
    V_proper ((happ.symm.trans (app_Null _)) ▶ hbset)

/-- **Finding 1.** With definition 3.34 as printed, `A ≤ B` holds for A
  `A B` – the empty class-function is a witness. -/
theorem PrintedWleq_total (A B : Class) : PrintedWleq A B :=
  (Null, Null_isClassFunction, fun b hb => absurd hb (img_Null_empty A))

/-- Hence `<` never holds... -/
theorem PrintedWlt_never (A B : Class) : ¬ PrintedWlt A B :=
  fun h => h (PrintedWleq_total B A)

/-- **Theorem (inconsistency, route 1).** The printed base axioms together
  with the printed size-limitation axiom 3.37 derive `False`:
  3.37 forces `V` to be empty, but  $\omega \in V$  by 3.53. -/
theorem printed_inconsistent_route1
  (sizeLimitation : ∀ X : Class, X ∈' V ↔ PrintedWlt (card X) (card V))
  False :=
  PrintedWlt_never (card omega) (card V)
  ((sizeLimitation omega).mp (mem_V.mpr infinity))

/-- **Theorem (inconsistency, route 2).** The ≤-bug propagates through
  `Tight` (3.41): no class is a universe –  $\emptyset$  must both belong (via
  Hereditary, since  $\omega \in U$ ) and not belong (via Tight, since < never
  holds) to it. -/
theorem no_universe (U : Class) (hU : isUniverse U) : False :=
  have hω : omega ∈' U := hU.1.1
  have hHer : Hereditary U := hU.1.2.1.2
  have hTight : Tight U := hU.1.2.2.2

```

```

have hNull : Null ∈' U := hHer omega hw Null (Null_sub omega)
PrintedWlt_never Null U ((hTight Null (Null_sub U)).mpr hNull)

/-- Hence Tarski's axiom 3.66 is itself contradictory over the base... -/
theorem printed_inconsistent_route2
  (tarski : ∀ x, isSet x → ∃ u, ((isUniverse u ∧ x ∈' u) ∧ isSet u))
  False :=
  match tarski Null isSet_Null with
  | ⟨u, ⟨hu, _⟩, _⟩ => no_universe u hu

/-- ...and the demo's announced THEOREM 3.65 ("V is a universe") is refuted
    as printed. -/
theorem printed_3_65_refutable : ¬ isUniverse V := no_universe V

/-- **Finding (phantom partners).** With Kuratowski pairs, a proper class
    collapses the pair: `{x, y} = {x}` and hence `(x, y) = (x, x)`. So
    uniqueness demanded by "concentrated" (3.29) fails at every diagonal
    point, and the function machinery cannot rest on 3.48 – it needs the
    Dunaev pairs 3.46. -/
theorem upair_proper_collapse {x y : Class} (hy : proper y) :
  upair x y = sing x := by
  apply extAx
  intro z
  constructor
  · intro hz
    have h' := (mem_cls _ z).mp hz
    exact match h'.2 with
    | 0r.inl h => mem_sing.mpr ⟨h'.1, h⟩
    | 0r.inr h => absurd (h ▶ h'.1) hy
  · intro hz
    have h' := (mem_cls _ z).mp hz
    exact mem_upair.mpr ⟨h'.1, 0r.inl h'.2⟩

theorem kpair_phantom {x y : Class} (hy : proper y) :
  kpair x y = kpair x x := by
  show upair (upair x x) (upair x y) = upair (upair x x) (upair x x)
  rw [upair_proper_collapse hy, upair_self]

/-- No `F` containing a diagonal pair `(x,x)` has a UNIQUE partner at
    `y := V` is a phantom second witness. -/
theorem no_unique_partner {F x : Class} (hx : isSet x)
  (hxx : kpair x x ∈' F) : ¬ ExistsU (fun y => kpair x y ∈' F) :=
  fun ⟨y₀, _, huniq⟩ =>
  have h1 : x = y₀ := huniq x hxx
  have h2 : V = y₀ := huniq V (by
    show knair x V ∈' F

```

```

      rw [kpair_phantom V_proper]; exact hxx)
V_proper ((h1.trans h2.symm) ▸ hx)

/-! ### Part 2.  What survives: theorems derivable from the printed base

/-- 3.07:  $\emptyset \subseteq X$ . -/
theorem thm_3_07 (X : Class) : Null  $\subseteq$  X := Null_sub X

/-- 3.11 "Intensionality": sets with the same containers are equal. -/
theorem thm_3_11 {X Y : Class} (hX : isSet X) (_hY : isSet Y)
  (h :  $\forall W, X \in' W \leftrightarrow Y \in' W$ ) : X = Y :=
  have hXX : X  $\in'$  sing X := self_mem_sing hX
  (((mem_cls _ Y).mp ((h (sing X)).mp hXX)).2).symm

/-- Members of `On` are exactly the ordinal sets. -/
theorem mem_On {B : Class} : B  $\in'$  On  $\leftrightarrow$  isSet B  $\wedge$  OrdC B := mem_cls _ _

/-- 3.18: `On` is transitive... -/
theorem Tr_On : Tr On := fun B hB b hb =>
  have h := mem_On.mp hB
  mem_On.mpr (isSet_of_mem hb,
    h.2.2 b hb,
    fun c hc => h.2.2 c (h.2.1 b hb c hc))

/-- ...and 3.18 in full: `Ord(On)` – the class of all ordinals is an ordinal
    class (no Burali-Forti paradox: `On` is proper, which is fine here)
theorem thm_3_18 : OrdC On :=
  (Tr_On, fun B hB => (mem_On.mp hB).2.1)

theorem mem_omega {x : Class} :
  x  $\in'$  omega  $\leftrightarrow$  isSet x  $\wedge$   $\forall Q$ , IndC Q  $\rightarrow$  x  $\in'$  Q := mem_cls _ _

theorem Null_mem_omega : Null  $\in'$  omega :=
  mem_omega.mpr (isSet_Null, fun _ hQ => hQ.1)

/-- A classical detour available in the printed system: SOME inductive
    exists – otherwise  $\omega = V$ , contradicting 3.53 + Russell. -/
theorem exists_inductive :  $\exists Q$ , IndC Q :=
  (Classical.em ( $\exists Q$ , IndC Q)).elim id (fun hno => by
    have hwV : omega = V := extAx (fun x =>
      (fun h => mem_V.mpr (isSet_of_mem h),
        fun h => mem_omega.mpr (mem_V.mp h,
          fun Q hQ => absurd (Q, hQ) hno)))
    exact absurd (hwV ▸ infinity) V_proper)

theorem isSet_succ_of_mem_omega {a : Class} (ha : a  $\in'$  omega) :

```

```

theorem isSet_succ_of_mem_omega {a : Class} (ha : a ∈ omega) :
  isSet (succ a) :=
  match exists_inductive with
  | ⟨Q, hQ⟩ => ⟨Q, hQ.2 a ((mem_omega.mp ha).2 Q hQ)⟩

/-- 3.55: ω is inductive – provable from the printed base alone. -/
theorem thm_3_55 : IndC omega :=
  ⟨Null_mem_omega,
    fun a ha => mem_omega.mpr
      (isSet_succ_of_mem_omega ha,
        fun Q hQ => hQ.2 a ((mem_omega.mp ha).2 Q hQ))⟩

/-- The induction principle of ω: ω is contained in every inductive class -/
theorem omega_sub {Q : Class} (hQ : IndC Q) : omega ≤' Q :=
  fun x hx => (mem_omega.mp hx).2 Q hQ

theorem mem_succ {x a : Class} :
  x ∈' succ a ↔ isSet x ∧ (x ∈' a ∨ x = a) := by
  constructor
  · intro h
    have h' := (mem_cls _ x).mp h
    exact ⟨h'.1, h'.2.elim Or.inl (fun hs => Or.inr ((mem_cls _ x).mpr h))
  · intro ⟨hset, h⟩
    exact (mem_cls _ x).mpr ⟨hset,
      h.elim Or.inl (fun he => Or.inr ((mem_cls _ x).mpr ⟨hset, he⟩))⟩

theorem sub_succ {a : Class} : a ≤' succ a :=
  fun x hx => mem_succ.mpr ⟨isSet_of_mem hx, Or.inl hx⟩

theorem mem_bInter {x A B : Class} :
  x ∈' bInter A B ↔ isSet x ∧ x ∈' A ∧ x ∈' B := mem_cls _ _

/-- 3.57: ω is transitive (by ω-induction on {x ∈ ω | x ≤ ω}). -/
theorem thm_3_57 : Tr omega := by
  have hInd : IndC (bInter omega (cls (fun x => x ≤' omega))) := by
    constructor
    · exact mem_bInter.mpr ⟨isSet_Null, Null_mem_omega,
      (mem_cls _ _).mpr ⟨isSet_Null, Null_sub omega⟩⟩
    · intro a ha
      have h := mem_bInter.mp ha
      have haw : a ∈' omega := h.2.1
      have hasub : a ≤' omega := ((mem_cls _ a).mp h.2.2).2
      have hsw : succ a ∈' omega := thm_3_55.2 a haw
      exact mem_bInter.mpr ⟨isSet_of_mem hsw, hsw,
        (mem_cls _ _).mpr ⟨isSet_of_mem hsw,
          fun x hx => (mem_succ.mp hx).2.elim
            (fun h1 => h1.2.1 haw)
            (fun h2 => h2.2.2 hasub)
            (fun h3 => h3.2.1 haw)
            (fun h4 => h4.2.2 hasub)
            (fun h5 => h5.2.1 haw)
            (fun h6 => h6.2.2 hasub)
            (fun h7 => h7.2.1 haw)
            (fun h8 => h8.2.2 hasub)
            (fun h9 => h9.2.1 haw)
            (fun h10 => h10.2.2 hasub)
            (fun h11 => h11.2.1 haw)
            (fun h12 => h12.2.2 hasub)
            (fun h13 => h13.2.1 haw)
            (fun h14 => h14.2.2 hasub)
            (fun h15 => h15.2.1 haw)
            (fun h16 => h16.2.2 hasub)
            (fun h17 => h17.2.1 haw)
            (fun h18 => h18.2.2 hasub)
            (fun h19 => h19.2.1 haw)
            (fun h20 => h20.2.2 hasub)
            (fun h21 => h21.2.1 haw)
            (fun h22 => h22.2.2 hasub)
            (fun h23 => h23.2.1 haw)
            (fun h24 => h24.2.2 hasub)
            (fun h25 => h25.2.1 haw)
            (fun h26 => h26.2.2 hasub)
            (fun h27 => h27.2.1 haw)
            (fun h28 => h28.2.2 hasub)
            (fun h29 => h29.2.1 haw)
            (fun h30 => h29.2.2 hasub)
            (fun h31 => h31.2.1 haw)
            (fun h32 => h32.2.2 hasub)
            (fun h33 => h33.2.1 haw)
            (fun h34 => h34.2.2 hasub)
            (fun h35 => h35.2.1 haw)
            (fun h36 => h36.2.2 hasub)
            (fun h37 => h37.2.1 haw)
            (fun h38 => h38.2.2 hasub)
            (fun h39 => h39.2.1 haw)
            (fun h40 => h40.2.2 hasub)
            (fun h41 => h41.2.1 haw)
            (fun h42 => h42.2.2 hasub)
            (fun h43 => h43.2.1 haw)
            (fun h44 => h44.2.2 hasub)
            (fun h45 => h45.2.1 haw)
            (fun h46 => h46.2.2 hasub)
            (fun h47 => h47.2.1 haw)
            (fun h48 => h48.2.2 hasub)
            (fun h49 => h49.2.1 haw)
            (fun h50 => h50.2.2 hasub)
            (fun h51 => h51.2.1 haw)
            (fun h52 => h52.2.2 hasub)
            (fun h53 => h53.2.1 haw)
            (fun h54 => h54.2.2 hasub)
            (fun h55 => h55.2.1 haw)
            (fun h56 => h56.2.2 hasub)
            (fun h57 => h57.2.1 haw)
            (fun h58 => h58.2.2 hasub)
            (fun h59 => h59.2.1 haw)
            (fun h60 => h60.2.2 hasub)
            (fun h61 => h61.2.1 haw)
            (fun h62 => h62.2.2 hasub)
            (fun h63 => h63.2.1 haw)
            (fun h64 => h64.2.2 hasub)
            (fun h65 => h65.2.1 haw)
            (fun h66 => h66.2.2 hasub)
            (fun h67 => h67.2.1 haw)
            (fun h68 => h68.2.2 hasub)
            (fun h69 => h69.2.1 haw)
            (fun h70 => h70.2.2 hasub)
            (fun h71 => h71.2.1 haw)
            (fun h72 => h72.2.2 hasub)
            (fun h73 => h73.2.1 haw)
            (fun h74 => h74.2.2 hasub)
            (fun h75 => h75.2.1 haw)
            (fun h76 => h76.2.2 hasub)
            (fun h77 => h77.2.1 haw)
            (fun h78 => h78.2.2 hasub)
            (fun h79 => h79.2.1 haw)
            (fun h80 => h80.2.2 hasub)
            (fun h81 => h81.2.1 haw)
            (fun h82 => h82.2.2 hasub)
            (fun h83 => h83.2.1 haw)
            (fun h84 => h84.2.2 hasub)
            (fun h85 => h85.2.1 haw)
            (fun h86 => h86.2.2 hasub)
            (fun h87 => h87.2.1 haw)
            (fun h88 => h88.2.2 hasub)
            (fun h89 => h89.2.1 haw)
            (fun h90 => h90.2.2 hasub)
            (fun h91 => h91.2.1 haw)
            (fun h92 => h92.2.2 hasub)
            (fun h93 => h93.2.1 haw)
            (fun h94 => h94.2.2 hasub)
            (fun h95 => h95.2.1 haw)
            (fun h96 => h96.2.2 hasub)
            (fun h97 => h97.2.1 haw)
            (fun h98 => h98.2.2 hasub)
            (fun h99 => h99.2.1 haw)
            (fun h100 => h100.2.2 hasub)
            (fun h101 => h101.2.1 haw)
            (fun h102 => h102.2.2 hasub)
            (fun h103 => h103.2.1 haw)
            (fun h104 => h104.2.2 hasub)
            (fun h105 => h105.2.1 haw)
            (fun h106 => h106.2.2 hasub)
            (fun h107 => h107.2.1 haw)
            (fun h108 => h108.2.2 hasub)
            (fun h109 => h109.2.1 haw)
            (fun h110 => h110.2.2 hasub)
            (fun h111 => h111.2.1 haw)
            (fun h112 => h112.2.2 hasub)
            (fun h113 => h113.2.1 haw)
            (fun h114 => h114.2.2 hasub)
            (fun h115 => h115.2.1 haw)
            (fun h116 => h116.2.2 hasub)
            (fun h117 => h117.2.1 haw)
            (fun h118 => h118.2.2 hasub)
            (fun h119 => h119.2.1 haw)
            (fun h120 => h120.2.2 hasub)
            (fun h121 => h121.2.1 haw)
            (fun h122 => h122.2.2 hasub)
            (fun h123 => h123.2.1 haw)
            (fun h124 => h124.2.2 hasub)
            (fun h125 => h125.2.1 haw)
            (fun h126 => h126.2.2 hasub)
            (fun h127 => h127.2.1 haw)
            (fun h128 => h128.2.2 hasub)
            (fun h129 => h129.2.1 haw)
            (fun h130 => h130.2.2 hasub)
            (fun h131 => h131.2.1 haw)
            (fun h132 => h132.2.2 hasub)
            (fun h133 => h133.2.1 haw)
            (fun h134 => h134.2.2 hasub)
            (fun h135 => h135.2.1 haw)
            (fun h136 => h136.2.2 hasub)
            (fun h137 => h137.2.1 haw)
            (fun h138 => h138.2.2 hasub)
            (fun h139 => h139.2.1 haw)
            (fun h140 => h140.2.2 hasub)
            (fun h141 => h141.2.1 haw)
            (fun h142 => h142.2.2 hasub)
            (fun h143 => h143.2.1 haw)
            (fun h144 => h144.2.2 hasub)
            (fun h145 => h145.2.1 haw)
            (fun h146 => h146.2.2 hasub)
            (fun h147 => h147.2.1 haw)
            (fun h148 => h148.2.2 hasub)
            (fun h149 => h149.2.1 haw)
            (fun h150 => h150.2.2 hasub)
            (fun h151 => h151.2.1 haw)
            (fun h152 => h152.2.2 hasub)
            (fun h153 => h153.2.1 haw)
            (fun h154 => h154.2.2 hasub)
            (fun h155 => h155.2.1 haw)
            (fun h156 => h156.2.2 hasub)
            (fun h157 => h157.2.1 haw)
            (fun h158 => h158.2.2 hasub)
            (fun h159 => h159.2.1 haw)
            (fun h160 => h160.2.2 hasub)
            (fun h161 => h161.2.1 haw)
            (fun h162 => h162.2.2 hasub)
            (fun h163 => h163.2.1 haw)
            (fun h164 => h164.2.2 hasub)
            (fun h165 => h165.2.1 haw)
            (fun h166 => h166.2.2 hasub)
            (fun h167 => h167.2.1 haw)
            (fun h168 => h168.2.2 hasub)
            (fun h169 => h169.2.1 haw)
            (fun h170 => h170.2.2 hasub)
            (fun h171 => h171.2.1 haw)
            (fun h172 => h172.2.2 hasub)
            (fun h173 => h173.2.1 haw)
            (fun h174 => h174.2.2 hasub)
            (fun h175 => h175.2.1 haw)
            (fun h176 => h176.2.2 hasub)
            (fun h177 => h177.2.1 haw)
            (fun h178 => h178.2.2 hasub)
            (fun h179 => h179.2.1 haw)
            (fun h180 => h180.2.2 hasub)
            (fun h181 => h181.2.1 haw)
            (fun h182 => h182.2.2 hasub)
            (fun h183 => h183.2.1 haw)
            (fun h184 => h184.2.2 hasub)
            (fun h185 => h185.2.1 haw)
            (fun h186 => h186.2.2 hasub)
            (fun h187 => h187.2.1 haw)
            (fun h188 => h188.2.2 hasub)
            (fun h189 => h189.2.1 haw)
            (fun h190 => h190.2.2 hasub)
            (fun h191 => h191.2.1 haw)
            (fun h192 => h192.2.2 hasub)
            (fun h193 => h193.2.1 haw)
            (fun h194 => h194.2.2 hasub)
            (fun h195 => h195.2.1 haw)
            (fun h196 => h196.2.2 hasub)
            (fun h197 => h197.2.1 haw)
            (fun h198 => h198.2.2 hasub)
            (fun h199 => h199.2.1 haw)
            (fun h200 => h200.2.2 hasub)
            (fun h201 => h201.2.1 haw)
            (fun h202 => h202.2.2 hasub)
            (fun h203 => h203.2.1 haw)
            (fun h204 => h204.2.2 hasub)
            (fun h205 => h205.2.1 haw)
            (fun h206 => h206.2.2 hasub)
            (fun h207 => h207.2.1 haw)
            (fun h208 => h208.2.2 hasub)
            (fun h209 => h209.2.1 haw)
            (fun h210 => h210.2.2 hasub)
            (fun h211 => h211.2.1 haw)
            (fun h212 => h212.2.2 hasub)
            (fun h213 => h213.2.1 haw)
            (fun h214 => h214.2.2 hasub)
            (fun h215 => h215.2.1 haw)
            (fun h216 => h216.2.2 hasub)
            (fun h217 => h217.2.1 haw)
            (fun h218 => h218.2.2 hasub)
            (fun h219 => h219.2.1 haw)
            (fun h220 => h220.2.2 hasub)
            (fun h221 => h221.2.1 haw)
            (fun h222 => h222.2.2 hasub)
            (fun h223 => h223.2.1 haw)
            (fun h224 => h224.2.2 hasub)
            (fun h225 => h225.2.1 haw)
            (fun h226 => h226.2.2 hasub)
            (fun h227 => h227.2.1 haw)
            (fun h228 => h228.2.2 hasub)
            (fun h229 => h229.2.1 haw)
            (fun h230 => h230.2.2 hasub)
            (fun h231 => h231.2.1 haw)
            (fun h232 => h232.2.2 hasub)
            (fun h233 => h233.2.1 haw)
            (fun h234 => h234.2.2 hasub)
            (fun h235 => h235.2.1 haw)
            (fun h236 => h236.2.2 hasub)
            (fun h237 => h237.2.1 haw)
            (fun h238 => h238.2.2 hasub)
            (fun h239 => h239.2.1 haw)
            (fun h240 => h240.2.2 hasub)
            (fun h241 => h241.2.1 haw)
            (fun h242 => h242.2.2 hasub)
            (fun h243 => h243.2.1 haw)
            (fun h244 => h244.2.2 hasub)
            (fun h245 => h245.2.1 haw)
            (fun h246 => h246.2.2 hasub)
            (fun h247 => h247.2.1 haw)
            (fun h248 => h248.2.2 hasub)
            (fun h249 => h249.2.1 haw)
            (fun h250 => h250.2.2 hasub)
            (fun h251 => h251.2.1 haw)
            (fun h252 => h252.2.2 hasub)
            (fun h253 => h253.2.1 haw)
            (fun h254 => h254.2.2 hasub)
            (fun h255 => h255.2.1 haw)
            (fun h256 => h256.2.2 hasub)
            (fun h257 => h257.2.1 haw)
            (fun h258 => h258.2.2 hasub)
            (fun h259 => h259.2.1 haw)
            (fun h260 => h260.2.2 hasub)
            (fun h261 => h261.2.1 haw)
            (fun h262 => h262.2.2 hasub)
            (fun h263 => h263.2.1 haw)
            (fun h264 => h264.2.2 hasub)
            (fun h265 => h265.2.1 haw)
            (fun h266 => h266.2.2 hasub)
            (fun h267 => h267.2.1 haw)
            (fun h268 => h268.2.2 hasub)
            (fun h269 => h269.2.1 haw)
            (fun h270 => h270.2.2 hasub)
            (fun h271 => h271.2.1 haw)
            (fun h272 => h272.2.2 hasub)
            (fun h273 => h273.2.1 haw)
            (fun h274 => h274.2.2 hasub)
            (fun h275 => h275.2.1 haw)
            (fun h276 => h276.2.2 hasub)
            (fun h277 => h277.2.1 haw)
            (fun h278 => h278.2.2 hasub)
            (fun h279 => h279.2.1 haw)
            (fun h280 => h280.2.2 hasub)
            (fun h281 => h281.2.1 haw)
            (fun h282 => h282.2.2 hasub)
            (fun h283 => h283.2.1 haw)
            (fun h284 => h284.2.2 hasub)
            (fun h285 => h285.2.1 haw)
            (fun h286 => h286.2.2 hasub)
            (fun h287 => h287.2.1 haw)
            (fun h288 => h288.2.2 hasub)
            (fun h289 => h289.2.1 haw)
            (fun h290 => h290.2.2 hasub)
            (fun h291 => h291.2.1 haw)
            (fun h292 => h292.2.2 hasub)
            (fun h293 => h293.2.1 haw)
            (fun h294 => h294.2.2 hasub)
            (fun h295 => h295.2.1 haw)
            (fun h296 => h296.2.2 hasub)
            (fun h297 => h297.2.1 haw)
            (fun h298 => h298.2.2 hasub)
            (fun h299 => h299.2.1 haw)
            (fun h300 => h300.2.2 hasub)
            (fun h301 => h301.2.1 haw)
            (fun h302 => h302.2.2 hasub)
            (fun h303 => h303.2.1 haw)
            (fun h304 => h304.2.2 hasub)
            (fun h305 => h305.2.1 haw)
            (fun h306 => h306.2.2 hasub)
            (fun h307 => h307.2.1 haw)
            (fun h308 => h308.2.2 hasub)
            (fun h309 => h309.2.1 haw)
            (fun h310 => h310.2.2 hasub)
            (fun h311 => h311.2.1 haw)
            (fun h312 => h312.2.2 hasub)
            (fun h313 => h313.2.1 haw)
            (fun h314 => h314.2.2 hasub)
            (fun h315 => h315.2.1 haw)
            (fun h316 => h316.2.2 hasub)
            (fun h317 => h317.2.1 haw)
            (fun h318 => h318.2.2 hasub)
            (fun h319 => h319.2.1 haw)
            (fun h320 => h320.2.2 hasub)
            (fun h321 => h321.2.1 haw)
            (fun h322 => h322.2.2 hasub)
            (fun h323 => h323.2.1 haw)
            (fun h324 => h324.2.2 hasub)
            (fun h325 => h325.2.1 haw)
            (fun h326 => h326.2.2 hasub)
            (fun h327 => h327.2.1 haw)
            (fun h328 => h328.2.2 hasub)
            (fun h329 => h329.2.1 haw)
            (fun h330 => h330.2.2 hasub)
            (fun h331 => h331.2.1 haw)
            (fun h332 => h332.2.2 hasub)
            (fun h333 => h333.2.1 haw)
            (fun h334 => h334.2.2 hasub)
            (fun h335 => h335.2.1 haw)
            (fun h336 => h336.2.2 hasub)
            (fun h337 => h337.2.1 haw)
            (fun h338 => h338.2.2 hasub)
            (fun h339 => h339.2.1 haw)
            (fun h340 => h340.2.2 hasub)
            (fun h341 => h341.2.1 haw)
            (fun h342 => h342.2.2 hasub)
            (fun h343 => h343.2.1 haw)
            (fun h344 => h344.2.2 hasub)
            (fun h345 => h345.2.1 haw)
            (fun h346 => h346.2.2 hasub)
            (fun h347 => h347.2.1 haw)
            (fun h348 => h348.2.2 hasub)
            (fun h349 => h349.2.1 haw)
            (fun h350 => h350.2.2 hasub)
            (fun h351 => h351.2.1 haw)
            (fun h352 => h352.2.2 hasub)
            (fun h353 => h353.2.1 haw)
            (fun h354 => h354.2.2 hasub)
            (fun h355 => h355.2.1 haw)
            (fun h356 => h356.2.2 hasub)
            (fun h357 => h357.2.1 haw)
            (fun h358 => h358.2.2 hasub)
            (fun h359 => h359.2.1 haw)
            (fun h360 => h360.2.2 hasub)
            (fun h361 => h361.2.1 haw)
            (fun h362 => h362.2.2 hasub)
            (fun h363 => h363.2.1 haw)
            (fun h364 => h364.2.2 hasub)
            (fun h365 => h365.2.1 haw)
            (fun h366 => h366.2.2 hasub)
            (fun h367 => h367.2.1 haw)
            (fun h368 => h368.2.2 hasub)
            (fun h369 => h369.2.1 haw)
            (fun h370 => h370.2.2 hasub)
            (fun h371 => h371.2.1 haw)
            (fun h372 => h372.2.2 hasub)
            (fun h373 => h373.2.1 haw)
            (fun h374 => h374.2.2 hasub)
            (fun h375 => h375.2.1 haw)
            (fun h376 => h376.2.2 hasub)
            (fun h377 => h377.2.1 haw)
            (fun h378 => h378.2.2 hasub)
            (fun h379 => h379.2.1 haw)
            (fun h380 => h380.2.2 hasub)
            (fun h381 => h381.2.1 haw)
            (fun h382 => h382.2.2 hasub)
            (fun h383 => h383.2.1 haw)
            (fun h384 => h384.2.2 hasub)
            (fun h385 => h385.2.1 haw)
            (fun h386 => h386.2.2 hasub)
            (fun h387 => h387.2.1 haw)
            (fun h388 => h388.2.2 hasub)
            (fun h389 => h389.2.1 haw)
            (fun h390 => h390.2.2 hasub)
            (fun h391 => h391.2.1 haw)
            (fun h392 => h392.2.2 hasub)
            (fun h393 => h393.2.1 haw)
            (fun h394 => h394.2.2 hasub)
            (fun h395 => h395.2.1 haw)
            (fun h396 => h396.2.2 hasub)
            (fun h397 => h397.2.1 haw)
            (fun h398 => h398.2.2 hasub)
            (fun h399 => h399.2.1 haw)
            (fun h400 => h400.2.2 hasub)
            (fun h401 => h401.2.1 haw)
            (fun h402 => h402.2.2 hasub)
            (fun h403 => h403.2.1 haw)
            (fun h404 => h404.2.2 hasub)
            (fun h405 => h405.2.1 haw)
            (fun h406 => h406.2.2 hasub)
            (fun h407 => h407.2.1 haw)
            (fun h408 => h408.2.2 hasub)
            (fun h409 => h409.2.1 haw)
            (fun h410 => h410.2.2 hasub)
            (fun h411 => h411.2.1 haw)
            (fun h412 => h412.2.2 hasub)
            (fun h413 => h413.2.1 haw)
            (fun h414 => h414.2.2 hasub)
            (fun h415 => h415.2.1 haw)
            (fun h416 => h416.2.2 hasub)
            (fun h417 => h417.2.1 haw)
            (fun h418 => h418.2.2 hasub)
            (fun h419 => h419.2.1 haw)
            (fun h420 => h420.2.2 hasub)
            (fun h421 => h421.2.1 haw)
            (fun h422 => h422.2.2 hasub)
            (fun h423 => h423.2.1 haw)
            (fun h424 => h424.2.2 hasub)
            (fun h425 => h425.2.1 haw)
            (fun h426 => h426.2.2 hasub)
            (fun h427 => h427.2.1 haw)
            (fun h428 => h428.2.2 hasub)
            (fun h429 => h429.2.1 haw)
            (fun h430 => h430.2.2 hasub)
            (fun h431 => h431.2.1 haw)
            (fun h432 => h432.2.2 hasub)
            (fun h433 => h433.2.1 haw)
            (fun h434 => h434.2.2 hasub)
            (fun h435 => h435.2.1 haw)
            (fun h436 => h436.2.2 hasub)
            (fun h437 => h437.2.1 haw)
            (fun h438 => h438.2.2 hasub)
            (fun h439 => h439.2.1 haw)
            (fun h440 => h440.2.2 hasub)
            (fun h441 => h441.2.1 haw)
            (fun h442 => h442.2.2 hasub)
            (fun h443 => h443.2.1 haw)
            (fun h444 => h444.2.2 hasub)
            (fun h445 => h445.2.1 haw)
            (fun h446 => h446.2.2 hasub)
            (fun h447 => h447.2.1 haw)
            (fun h448 => h448.2.2 hasub)
            (fun h449 => h449.2.1 haw)
            (fun h450 => h450.2.2 hasub)
            (fun h451 => h451.2.1 haw)
            (fun h452 => h452.2.2 hasub)
            (fun h453 => h453.2.1 haw)
            (fun h454 => h454.2.2 hasub)
            (fun h455 => h455.2.1 haw)
            (fun h456 => h456.2.2 hasub)
            (fun h457 => h457.2.1 haw)
            (fun h458 => h458.2.2 hasub)
            (fun h459 => h459.2.1 haw)
            (fun h460 => h460.2.2 hasub)
            (fun h461 => h461.2.1 haw)
            (fun h462 => h462.2.2 hasub)
            (fun h463 => h463.2.1 haw)
            (fun h464 => h464.2.2 hasub)
            (fun h465 => h465.2.1 haw)
            (fun h466 => h466.2.2 hasub)
            (fun h467 => h467.2.1 haw)
            (fun h468 => h468.2.2 hasub)
            (fun h469 => h469.2.1 haw)
            (fun h470 => h470.2.2 hasub)
            (fun h471 => h471.2.1 haw)
            (fun h472 => h472.2.2 hasub)
            (fun h473 => h473.2.1 haw)
            (fun h474 => h474.2.2 hasub)
            (fun h475 => h475.2.1 haw)
            (fun h476 => h476.2.2 hasub)
            (fun h477 => h477.2.1 haw)
            (fun h478 => h478.2.2 hasub)
            (fun h479 => h479.2.1 haw)
            (fun h480 => h480.2.2 hasub)
            (fun h481 => h481.2.1 haw)
            (fun h482 => h482.2.2 hasub)
            (fun h483 => h483.2.1 haw)
            (fun h484 => h484.2.2 hasub)
            (fun h485 => h485.2.1 haw)
            (fun h486 => h486.2.2 hasub)
            (fun h487 => h487.2.1 haw)
            (fun h488 => h488.2.2 hasub)
            (fun h489 => h489.2.1 haw)
            (fun h490 => h490.2.2 hasub)
            (fun h491 => h491.2.1 haw)
            (fun h492 => h492.2.2 hasub)
            (fun h493 => h493.2.1 haw)
            (fun h494 => h494.2.2 hasub)
            (fun h495 => h495.2.1 haw)
            (fun h496 => h496.2.2 hasub)
            (fun h497 => h497.2.1 haw)
            (fun h498 => h498.2.2 hasub)
            (fun h499 => h499.2.1 haw)
            (fun h500 => h500.2.2 hasub)
            (fun h501 => h501.2.1 haw)
            (fun h502 => h502.2.2 hasub)
            (fun h503 => h503.2.1 haw)
            (fun h504 => h504.2.2 hasub)
            (fun h505 => h505.2.1 haw)
            (fun h506 => h506.2.2 hasub)
            (fun h507 => h507.2.1 haw)
            (fun h508 => h508.2.2 hasub)
            (fun h509 => h509.2.1 haw)
            (fun h510 => h510.2.2 hasub)
            (fun h511 => h511.2.1 haw)
            (fun h512 => h512.2.2 hasub)
            (fun h513 => h513.2.1 haw)
            (fun h514 => h514.2.2 hasub)
            (fun h515 => h515.2.1 haw)
            (fun h516 => h516.2.2 hasub)
            (fun h517 => h517.2.1 haw)
            (fun h518 => h518.2.2 hasub)
            (fun h519 => h519.2.1 haw)
            (fun h520 => h520.2.2 hasub)
            (fun h521 => h521.2.1 haw)
            (fun h522 => h522.2.2 hasub)
            (fun h523 => h523.2.1 haw)
            (fun h524 => h524.2.2 hasub)
            (fun h525 => h525.2.1 haw)
            (fun h526 => h526.2.2 hasub)
            (fun h527 => h527.2.1 haw)
            (fun h528 => h528.2.2 hasub)
            (fun h529 => h529.2.1 haw)
            (fun h530 => h530.2.2 hasub)
            (fun h531 => h531.2.1 haw)
            (fun h532 => h532.2.2 hasub)
            (fun h533 => h533.2.1 haw)
            (fun h534 => h
```

```

      (fun h' => hasub x h') (fun h' => h'.symm ▶ haw)))
intro B hB
exact ((mem_cls _ B).mp (mem_bInter.mp (omega_sub hInd B hB)).2.2).2

/-- Every member of  $\omega$  is transitive (again by  $\omega$ -induction). -/
theorem Tr_of_mem_omega {B : Class} (hB : B ∈' omega) : Tr B := by
  have hInd : IndC (bInter omega (cls Tr)) := by
    constructor
    · exact mem_bInter.mpr {isSet_Null, Null_mem_omega,
      (mem_cls _ _).mpr {isSet_Null,
        fun C hC => absurd hC (not_mem_Null C)}}
    · intro a ha
      have h := mem_bInter.mp ha
      have haw : a ∈' omega := h.2.1
      have haTr : Tr a := ((mem_cls _ a).mp h.2.2).2
      have hsw : succ a ∈' omega := thm_3_55.2 a haw
      refine mem_bInter.mpr {isSet_of_mem hsw, hsw,
        (mem_cls _ _).mpr {isSet_of_mem hsw, ?_}}
      intro C hC
      cases (mem_succ.mp hC).2 with
      | inl h' => exact fun x hx => sub_succ x (haTr C h' x hx)
      | inr h' => exact fun x hx => sub_succ x (h' ▶ hx)
      exact ((mem_cls _ B).mp (mem_bInter.mp (omega_sub hInd B hB)).2.2).2

/-- 3.58:  $\omega$  is an ordinal class. -/
theorem thm_3_58 : OrdC omega :=
  {thm_3_57, fun B hB => Tr_of_mem_omega hB}

/-- 3.59:  $\omega \in 0_n$ . -/
theorem thm_3_59 : omega ∈' 0_n := mem_0n.mpr {infinity, thm_3_58}

/-- For a TRANSITIVE set,  $\text{succ } a \subseteq \mathcal{P}(a)$ , so  $\text{succ } a$  is a set without
pairing/union axiom – this rescues Ind( $0_n$ ) in the printed base. -/
theorem isSet_succ_of_Tr {a : Class} (haTr : Tr a) (ha : isSet a) :
  isSet (succ a) :=
  heredity
  (fun x hx => (mem_cls _ x).mpr {isSet_of_mem hx,
    (mem_succ.mp hx).2.elim (fun h => haTr x h)
    (fun h => fun z hz => h ▶ hz)}})
  (exponent ha)

/-- 3.56:  $0_n$  is inductive – provable from the printed base alone. -/
theorem thm_3_56 : IndC 0_n := by
  constructor
  · exact mem_0n.mpr {isSet_Null,
    fun B hB => absurd hB (not_mem_Null B),

```

```

    fun B hB => absurd hB (not_mem_Null B))
  · intro a ha
    have h := mem_On.mp ha
    have haTr : Tr a := h.2.1
    refine mem_On.mpr (isSet_succ_of_Tr haTr h.1, ?_, ?_)
  · intro B hB
    cases (mem_succ.mp hB).2 with
    | inl h' => exact fun x hx => sub_succ x (haTr B h' x hx)
    | inr h' => exact fun x hx => sub_succ x (h' ▶ hx)
  · intro B hB
    cases (mem_succ.mp hB).2 with
    | inl h' => exact h.2.2 B h'
    | inr h' => exact h'.symm ▶ haTr

/-- 3.67: the general induction principle on a transitive class, derived
from the printed "Regularity" 3.14 – which is exactly  $\epsilon$ -induction in
powerclass form. -/
theorem thm_3_67 {K A : Class} (hK : Tr K) (h : bInter K (power A)  $\subseteq$ ' A)
  K  $\subseteq$ ' A := by
  have step : power (cls (fun x => x  $\in$ ' K  $\rightarrow$  x  $\in$ ' A))  $\subseteq$ '
    cls (fun x => x  $\in$ ' K  $\rightarrow$  x  $\in$ ' A) := by
    intro y hy
    have h' := (mem_cls _ y).mp hy
    refine (mem_cls _ y).mpr (h'.1, fun hyK => ?_)
    have hyA : y  $\subseteq$ ' A := fun x hx =>
      ((mem_cls _ x).mp (h'.2 x hx)).2 (hK y hyK x hx)
    exact h y (mem_bInter.mpr (h'.1, hyK,
      (mem_cls _ y).mpr (h'.1, hyA)))
  intro k hk
  exact ((mem_cls _ k).mp
    (regularity step k (mem_V.mpr (isSet_of_mem hk)))).2 hk

/-- The Tight-free components of "V is a universe" (towards 3.65), proved
from the printed base. -/
theorem V_Tr : Tr V := fun _B _hB b hb => mem_V.mpr (isSet_of_mem hb)
theorem V_Hereditary : Hereditary V :=
  fun _B hB X hX => mem_V.mpr (heredity hX (mem_V.mp hB))
theorem V_Powerful : Powerful V :=
  fun _B hB => mem_V.mpr (exponent (mem_V.mp hB))
theorem omega_mem_V : omega  $\in$ ' V := mem_V.mpr infinity

/-! ### Part 3. The repair: Pairing and Union, and the TBI items 3.44-

Pairing and Union are absent from the printed axiom list, yet:
(i) `Pairful` is part of the universe definition 3.64, and 3.65
announces `Pairful V` – i.e. the pairing axiom itself;

```

(ii) the Dunaev lists 3.45 (marked TBI in the demo) need the tags
``{{{a}},n}`` to be sets;
 (iii) pairing cannot be recovered from a repaired limitation-of-size
 axiom, because $\leq$ is defined through class functions whose graphs
 already consist of ordered pairs – a circularity.
 So we add them. (Both hold in every intended model.) -/

```
axiom pairing : ∀ {a b : Class}, isSet a → isSet b → isSet (upair a b)
axiom unionAx : ∀ {x : Class}, isSet x → isSet (sUnion x)
```

```
theorem isSet_sing {a : Class} (h : isSet a) : isSet (sing a) :=
  upair_self a ▸ pairing h h
```

```
theorem bUnion_eq_sUnion_upair {A B : Class} (hA : isSet A) (hB : isSet B) :
  bUnion A B = sUnion (upair A B) :=
  extAx (fun x =>
    (fun h =>
      have h' := (mem_cls _ x).mp h
      (mem_cls _ x).mpr ⟨h'.1, h'.2.elim
        (fun hx => ⟨A, mem_upair.mpr ⟨hA, Or.inl rfl⟩, hx⟩)
        (fun hx => ⟨B, mem_upair.mpr ⟨hB, Or.inr rfl⟩, hx⟩)⟩,
      fun h =>
        have h' := (mem_cls _ x).mp h
        (mem_cls _ x).mpr ⟨h'.1,
          match h'.2 with
          | ⟨_, hy, hxy⟩ => (mem_upair.mp hy).2.elim
            (fun e => Or.inl (e ▸ hxy)) (fun e => Or.inr (e ▸ hxy))⟩)))
```

```
theorem isSet_succ {a : Class} (ha : isSet a) : isSet (succ a) := by
  show isSet (bUnion a (sing a))
  rw [bUnion_eq_sUnion_upair ha (isSet_sing ha)]
  exact unionAx (pairing ha (isSet_sing ha))
```

```
/-- 3.54: V is inductive – NOW provable. (This one genuinely needs the
  added axioms: for a non-transitive set `a`, the  $\mathcal{P}$ -trick of
  `isSet_succ_of_Tr` does not apply.) -/
```

```
theorem thm_3_54 : IndC V :=
  (mem_V.mpr isSet_Null, fun a ha => mem_V.mpr (isSet_succ (mem_V.mp ha)))
```

```
/-- The `Pairful` component of 3.65 – literally the pairing axiom. -/
```

```
theorem V_Pairful : Pairful V :=
  fun _A _B hA hB => mem_V.mpr (pairing (mem_V.mp hA) (mem_V.mp hB))
```

```
/-! ##### The Dunaev tagged lists (3.44–3.47) – the demo's TBI, implemented
```

```
3.44 numerals; 3.45 mark(A,n) = {{0,n}} ∪ { {{{a}},n} | a ∈ A };
```

3.46 $(A,B) := [A,B] = \text{mark}(A,1) \cup \text{mark}(B,2);$

3.47 $\text{pr1}(L) = \{ a \mid \{\{\{a\}\}, 1\} \in L \}, \quad \text{pr2}(L) = \{ b \mid \{\{\{b\}\}, 2\} \in L \}$

The point of the construction: it is faithful on PROPER classes, w
Kuratowski pairs collapse (see `kpair\_phantom`). We verify this.

```
def zero : Class := Null
def one  : Class := sing Null
def two  : Class := upair Null one

theorem isSet_one : isSet one := isSet_sing isSet_Null
theorem isSet_two : isSet two := pairing isSet_Null isSet_one

theorem Null_ne_one : Null ≠ one := fun h =>
  have h1 : Null ∈' one := self_mem_sing isSet_Null
  not_mem_Null Null (h.symm ▸ h1)

theorem Null_ne_two : Null ≠ two := fun h =>
  have h1 : one ∈' two := mem_upair.mpr {isSet_one, Or.inr rfl}
  not_mem_Null one (h.symm ▸ h1)

theorem one_ne_two : one ≠ two := fun h =>
  have h2 : one ∈' two := mem_upair.mpr {isSet_one, Or.inr rfl}
  have h3 : one ∈' one := h.symm ▸ h2
  Null_ne_one ((mem_sing.mp h3).2.symm)

/-- The tag `{{{a}}}, n`. -/
def tag (a n : Class) : Class := upair (sing (sing a)) n
/-- 3.45. -/
def markC (A n : Class) : Class :=
  cls (fun x => x = upair Null n ∨ ∃ a, a ∈' A ∧ x = tag a n)
/-- 3.46: the Dunaev ordered pair. -/
def dpair (A B : Class) : Class := bUnion (markC A one) (markC B two)
/-- 3.47. -/
def dpr1 (L : Class) : Class := cls (fun a => tag a one ∈' L)
def dpr2 (L : Class) : Class := cls (fun a => tag a two ∈' L)

theorem isSet_tag {a n : Class} (ha : isSet a) (hn : isSet n) :
  isSet (tag a n) := pairing (isSet_sing (isSet_sing ha)) hn

theorem mem_markC {x A n : Class} :
  x ∈' markC A n ↔ isSet x ∧
    (x = upair Null n ∨ ∃ a, a ∈' A ∧ x = tag a n) := mem_cls _ _

/-! Disambiguation kit: tags, markers and numerals never collide. -/
```

```

theorem sing_sing_ne_one {a : Class} (ha : isSet a) : sing (sing a) ≠ 1
  fun h =>
    have h1 : sing a ∈' sing (sing a) := self_mem_sing (isSet_sing ha)
    have h2 : sing a ∈' one := h ▸ h1
    have h3 : sing a = Null := (mem_sing.mp h2).2
    not_mem_Null a (h3 ▸ self_mem_sing ha)

theorem sing_sing_ne_two {a : Class} (ha : isSet a) : sing (sing a) ≠ 2
  fun h =>
    have h1 : Null ∈' two := mem_upair.mpr (isSet_Null, Or.inl rfl)
    have h2 : Null = sing a := (mem_sing.mp (h.symm ▸ h1)).2
    not_mem_Null a (h2.symm ▸ self_mem_sing ha)

/-- A tag `{{a}},n)` with `n ≠ 0` is never a marker `{0,m}`. -/
theorem tag_ne_marker {a n m : Class} (ha : isSet a) (hn : n ≠ Null) :
  tag a n ≠ upair Null m :=
  fun h =>
    have h1 : Null ∈' upair Null m := mem_upair.mpr (isSet_Null, Or.inl rfl)
    have h2 : Null ∈' tag a n := h.symm ▸ h1
    (mem_upair.mp h2).2.elim
      (fun e =>
        have h3 : sing a ∈' Null := e.symm ▸ self_mem_sing (isSet_sing
          not_mem_Null _ h3)
        (fun e => hn e.symm))

theorem tag_one_ne_tag_two {a c : Class} (ha : isSet a) (hc : isSet c)
  tag a one ≠ tag c two :=
  fun h =>
    have h1 : one ∈' tag a one := mem_upair.mpr (isSet_one, Or.inr rfl)
    have h2 : one ∈' tag c two := h ▸ h1
    (mem_upair.mp h2).2.elim
      (fun e =>
        have h0 : Null ∈' one := self_mem_sing isSet_Null
        have h3 : Null ∈' sing (sing c) := e ▸ h0
        have h4 : Null = sing c := (mem_sing.mp h3).2
        not_mem_Null c (h4.symm ▸ self_mem_sing hc))
    one_ne_two

theorem tag_inj {a c n : Class} (ha : isSet a) (hc : isSet c)
  (hn : sing (sing a) ≠ n) (h : tag a n = tag c n) : a = c :=
  have h1 : sing (sing a) ∈' tag a n :=
    mem_upair.mpr (isSet_sing (isSet_sing ha), Or.inl rfl)
  have h2 : sing (sing a) ∈' tag c n := h ▸ h1
  (mem_upair.mp h2).2.elim
    (fun e => sing_inj ha (sing_inj (isSet_sing ha) e))
    (fun e => absurd e hn)

```

```

      (fun e => absurd e m1))

theorem mem_dpair {x A B : Class} :
  x ∈' dpair A B ↔ isSet x ∧ (x ∈' markC A one ∨ x ∈' markC B two) :=
  mem_cls _ _

theorem tag1_mem_dpair {a A B : Class} (ha : a ∈' A) :
  tag a one ∈' dpair A B :=
  have ht : isSet (tag a one) := isSet_tag (isSet_of_mem ha) isSet_one
  mem_dpair.mpr ⟨ht, Or.inl (mem_markC.mpr ⟨ht, Or.inr ⟨a, ha, rfl⟩)⟩⟩

theorem tag2_mem_dpair {b A B : Class} (hb : b ∈' B) :
  tag b two ∈' dpair A B :=
  have ht : isSet (tag b two) := isSet_tag (isSet_of_mem hb) isSet_two
  mem_dpair.mpr ⟨ht, Or.inr (mem_markC.mpr ⟨ht, Or.inr ⟨b, hb, rfl⟩)⟩⟩

theorem mem_A_of_tag1 {a A B : Class} (ha : isSet a)
  (h : tag a one ∈' dpair A B) : a ∈' A :=
  (mem_dpair.mp h).2.elim
  (fun h1 => (mem_markC.mp h1).2.elim
    (fun e => absurd e (tag_ne_marker ha (Ne.symm Null_ne_one)))
    (fun ⟨a', ha', he⟩ =>
      (tag_inj ha (isSet_of_mem ha') (sing_sing_ne_one ha) he).symm
    (fun h2 => (mem_markC.mp h2).2.elim
      (fun e => absurd e (tag_ne_marker ha (Ne.symm Null_ne_one)))
      (fun ⟨b, hb, he⟩ =>
        absurd he (tag_one_ne_tag_two ha (isSet_of_mem hb))))))

theorem mem_B_of_tag2 {b A B : Class} (hb : isSet b)
  (h : tag b two ∈' dpair A B) : b ∈' B :=
  (mem_dpair.mp h).2.elim
  (fun h1 => (mem_markC.mp h1).2.elim
    (fun e => absurd e (tag_ne_marker hb (Ne.symm Null_ne_two)))
    (fun ⟨a, ha, he⟩ =>
      absurd he.symm (tag_one_ne_tag_two (isSet_of_mem ha) hb)))
  (fun h2 => (mem_markC.mp h2).2.elim
    (fun e => absurd e (tag_ne_marker hb (Ne.symm Null_ne_two)))
    (fun ⟨b', hb', he⟩ =>
      (tag_inj hb (isSet_of_mem hb') (sing_sing_ne_two hb) he).symm

/-- **The TBI items, verified.** First projection recovers the first
  component of a Dunaev pair – for ARBITRARY classes... -/
theorem dpr1_dpair (A B : Class) : dpr1 (dpair A B) = A :=
  extAx (fun x =>
    (fun h =>
      have h' := (mem_cls _ x).mp h
      . . . . .

```

```

      mem_A_of_tag1 h'.1 h'.2,
      fun h => (mem_cls _ x).mpr (isSet_of_mem h, tag1_mem_dpair h)))

/-- ...and the second likewise. -/
theorem dpr2_dpair (A B : Class) : dpr2 (dpair A B) = B :=
  extAx (fun x =>
    {fun h =>
      have h' := (mem_cls _ x).mp h
      mem_B_of_tag2 h'.1 h'.2,
      fun h => (mem_cls _ x).mpr (isSet_of_mem h, tag2_mem_dpair h)})

/-- Hence the Dunaev pair is injective on ARBITRARY classes (proper inc
    – exactly what Kuratowski pairs cannot deliver (`kpair_phantom`). -
theorem dpair_inj {A B C D : Class} (h : dpair A B = dpair C D) :
  A = C ∧ B = D :=
  {by rw [← dpr1_dpair A B, h, dpr1_dpair],
   by rw [← dpr2_dpair A B, h, dpr2_dpair]}

/-- Showcase: it works where Kuratowski provably fails – on proper clas
theorem dpair_proper_showcase :
  dpr1 (dpair V 0n) = V ∧ dpr2 (dpair V 0n) = 0n :=
  {dpr1_dpair V 0n, dpr2_dpair V 0n}

/-- The `{0,n}` marker device works: a mark is never empty, so list len
    is always recoverable – in particular `[A] ≠ []` even for `A = ∅`.
theorem mark_ne_Null (A n : Class) (hn : isSet n) : markC A n ≠ Null :=
  fun h =>
    have hm : upair Null n ∈' markC A n :=
      mem_markC.mpr (pairing isSet_Null hn, Or.inl rfl)
    not_mem_Null _ (h ▸ hm)

theorem dsingle_ne_dnil (A : Class) : markC A one ≠ Null :=
  mark_ne_Null A one isSet_one

end
end Obelisk

/-! ### Audit: exact axiom dependencies of the main results -/
#print axioms Obelisk.printed_inconsistent_route1
#print axioms Obelisk.printed_inconsistent_route2
#print axioms Obelisk.printed_3_65_refutable
#print axioms Obelisk.no_unique_partner
#print axioms Obelisk.thm_3_18
#print axioms Obelisk.thm_3_55
#print axioms Obelisk.thm_3_56
#print axioms Obelisk.thm_3_58

```

```
#print axioms Obelisk.thm_3_67
#print axioms Obelisk.dpair_inj
#print axioms Obelisk.dsingle_ne_dnil
```

Отчёт подготовлен Claude (Anthropic) 22 июля 2026 по просьбе Алексея Комиссарова. Все формальные утверждения проверены Lean 4.32.0; файл самодостаточен (без Mathlib) и компилируется без ошибок и без sorry.

Источники: [obeliskproofassistant.github.io](https://github.com/obeliskproofassistant) → [georgydunaev.github.io](https://github.com/georgydunaev),
репозиторий [georgydunaev/variousreleases](https://github.com/georgydunaev/variousreleases) (output.html/output.pdf, 2026-05-05).