Tournaments, $\bar{\omega}$-criticality, and directed paths: the formal statements

A machine-checked companion to the `digraph-theory` library
for mathematicians with no Rocq background

The `LLM4Rocq/digraph-theory` project

June 12, 2026

https://github.com/LLM4Rocq/digraph-theory

Abstract

This document lets a mathematician audit, without reading any proof or learning any proof assistant, what exactly has been machine-checked in the `digraph-theory` library: **for every** $k \in \{3, 4, 5\}$ **there are infinitely many** k-$\bar{\omega}$**-critical tournaments** (Conjecture 5.10 of Aboulker–Aubian–Charbit–Lopes, arXiv:2310.04265, at every value of k for which it is humanly known), that **Question 5.9 of the same paper fails at each such** k, and **the** $\delta = 3$ **case of the Cheng–Keevash path conjecture** (every oriented digraph with minimum out-degree ≥ 3 contains a directed path of length 6), among other results.

Each result page shows the *verbatim* formal statement, a plain mathematical decoding, and — the audit contract — the complete list of definitions the statement depends on, each explained in the Dictionary (Chapter 2). The lists are *generated from the compiled library* and checked in continuous integration: no definition can be silently missing. Proofs never appear; they are the machine's job. Every result here is axiom-free (`Print Assumptions`: closed under the global context).

Contents

Chapter 1

How to read a formal statement

This chapter is the only Rocq you need. Statements in this document are shown *verbatim* from the library source (grey boxes, with a clickable `[file:line]` link to GitHub); this page explains the ambient conventions once and for all.

1.1 What the machine guarantees, and what you must check

A proof assistant divides the work cleanly:

- **The machine checks the proofs.** Every theorem in this document has been verified by the Rocq kernel down to the axioms of its type theory, and every one reports `Print Assumptions: Closed under the global context` — no axioms, no admitted steps, nothing assumed. Proofs are therefore *not shown*: reading them adds nothing to your confidence.
- **You check the statements.** The only way a formalization can be wrong for a mathematician is that the formal statement does not say what the informal one means. That is exactly what this document is for: each result page lists *every* definition its statement depends on (the blue "Everything this statement needs" panel), and each definition has a Dictionary entry with the verbatim formal text and a note on why it captures the intended notion.
- **The lists themselves are machine-generated.** A script computes the statement's dependency closure from the compiled library and CI fails if a needed definition has no Dictionary entry, or if a quoted text drifts from the source. What you read is what the kernel saw.

1.2 Reading the notation

Finite types. All graphs here are finite. A vertex set is a `finType` (a type with a finite enumeration); `#|T|` is its cardinality, `{set T}` the type of its subsets, `{perm T}` the type of its permutations. Quantification `[forall v : T, P v]` over a finite type is an (effectively computable) Boolean.

Booleans as statements. MathComp states decidable properties as Boolean computations: `==` is decidable equality, `&&`, `||`, `~~` are and/or/not, and a Boolean `b` used as a statement abbreviates `b = true`. For finite objects this is a presentation choice, not a logical restriction: every such Boolean is provably equivalent to the corresponding ordinary proposition.

Numbers. `nat` is $\{0, 1, 2, \dots\}$; `n.+1`, `n.-1`, `n.*2` are $n{+}1$, $n{-}1$ (truncated), $2n$; `<=` and `<` on `nat` are the usual order (the `%N` marker just selects this arithmetic when several are in scope). `'Z_n` is the ring $\mathbb{Z}/n\mathbb{Z}$; for `z : 'Z_n`, `val z` is its representative in $\{0, \dots, n-1\}$.

The recurring index convention. The circulant families need "every odd $n = 2m+1$ with $m \geq 3$". In the formal statements this appears as a parameter `m' : nat` with the abbreviations

$m := \texttt{m'.+1}$ and $n := \texttt{m.*2.+1}$, plus a hypothesis `(3 <= m'.+1)%N`. So `m'` ranges over $\{2, 3, 4, \dots\}$, m over $\{3, 4, 5, \dots\}$ and n over $\{7, 9, 11, \dots\}$ — exactly the intended family, with the non-degeneracy $n \geq 3$ built into the shape of n.

Sections. Library files fix common parameters with `Variable`/`Hypothesis` inside a `Section`. A statement quoted from inside a section is implicitly universally quantified over these parameters; where it matters, the result page says so in the Decoded box. (After the section ends, Rocq performs this quantification mechanically; the quoted in-file form and the exported form denote the same theorem.)

Structures. "`T : tournament`" reads "T is a tournament": a type bundled with an arc relation satisfying the tournament axioms (Section 2.5). Writing `v : T` for a vertex of the bundled T is the formal counterpart of "$v \in V(T)$".

1.3 The two headline objects

Everything in this document concerns two invariants, both defined in the Dictionary with full pedigree:

- $\bar{\omega}(T)$, the *tournament clique number* (Section 2.10): the minimum over all orderings of $V(T)$ of the clique number of the *backedge graph* (Section 2.9). Written with the omega-bar notation in the source.
- $\ell(D)$, the length of a longest directed simple path (Section 2.22).

Chapter 2

The Dictionary: every definition you need

Each entry: the mathematical notion, the verbatim formal text, and — where there is anything to argue — why the formal object is the intended one. Result pages link here; the blue panels list exactly which entries each result needs.

2.1 Digraphs and arcs ($u \longrightarrow v$)

A digraph is a (finite, in this document) vertex type equipped with an arc relation. The library roots its own hierarchy on a bare relation:

```
HB.mixin Record HasArc V := { arc : rel V }.
Notation "u --> v" := (arc u v) (at level 30) : digraph_scope.
```

[theories/core/digraph.v:35]

`rel V` is the type of Boolean binary relations on `V`; `u --> v` is notation for `arc u v`, "there is an arc from u to v". A `diGraphType` is a finite type bundled with such a relation, with no further axioms.

Why this is the right definition. No axioms at this level means nothing is smuggled in: loops and 2-cycles are not yet excluded. They are excluded exactly where the mathematics requires it — by the oriented/tournament axioms below. Isomorphism of digraphs is the evident notion (Section 2.2).

2.2 Digraph isomorphism

```
Definition dgiso (D1 D2 : diGraphType) : Prop :=
  exists f : D1 -> D2, bijective f /\ forall u v, (f u --> f v) =
     (u --> v).
```

[theories/core/digraph.v:97]

A bijection between the vertex types that preserves and reflects arcs ($fu \to fv$ iff $u \to v$, as Booleans).

2.3 Oriented digraphs

An *oriented digraph* (an "oriented graph" in the combinatorics literature) has no loops and no digons:

```
HB.mixin Record DiGraph_IsOriented V of DiGraph V := {
  arc_irrefl : irreflexive (arc : rel V);
  arc_asymm  : forall u v : V, arc u v -> arc v u = false
}.
```

[theories/core/oriented.v:33]

`arc_irrefl` forbids loops; `arc_asymm` says $u \to v$ and $v \to u$ cannot both hold. The bundled structure is `orientedDigraph`.

2.4 Out-degree

```
Definition outdeg v := #|[set w | v --> w]|.
```

[theories/core/oriented.v:47]

The number of out-neighbours of v: $d^+(v) = |\{w : v \to w\}|$. The path results state their hypothesis as "`forall v, k <= outdeg v`", i.e. minimum out-degree $\delta^+ \geq k$.

2.5 Tournaments

A tournament orients every pair: for distinct u, v exactly one of $u \to v$, $v \to u$ holds.

```
HB.mixin Record Oriented_IsTournament V of Oriented V := {
  arc_total : forall u v : V, (u != v) = (arc u v) (+) (arc v u)
}.
```

[theories/core/tournament.v:32]

`(+)` is exclusive or, so the axiom reads: $u \neq v$ iff exactly one of the two arcs is present. Together with irreflexivity (inherited from `Oriented`) this is precisely the textbook definition. The bundled structure is `tournament`.

Why this is the right definition. On the diagonal the axiom specialises to false $=$ $(u \to u) \oplus (u \to u)$, which holds trivially — irreflexivity is genuinely carried by the `Oriented` layer, and totality by this one; no case is lost.

2.6 Transitive tournaments

```
Definition transb := [forall u : T, [forall v : T, [forall w : T,
  (u --> v) ==> (v --> w) ==> (u --> w)]]].
```

[theories/core/tournament.v:112]

The Boolean "the arc relation is transitive". Transitive tournaments are exactly the linearly ordered ones; they are the tournaments with $\bar{\omega} = 1$ (`omegabar_transb`, Section 6.1).

2.7 The directed triangle C_3

```
Definition C3 : Type := 'Z_3.

Lemma arcC3E (u v : C3) : (u --> v) = (v == u + 1 :> 'Z_3).
```

[theories/core/tournament.v:183]

Vertices $\mathbb{Z}/3$, arcs $u \to u+1$: the 3-cycle $0 \to 1 \to 2 \to 0$, the smallest non-transitive tournament.

2.8 Vertex orders as permutations

$\bar{\omega}$ quantifies over *orderings of the vertex set*. The library represents an ordering by a permutation p, compared through the position of the enumeration:

```
Definition ltp p u v := enum_rank (p u) < enum_rank (p v).

Definition realize : {perm V} := perm rfun_inj.

Lemma ltp_realizeE u v : ltp realize u v = r u v.
```

[theories/core/order.v:39]

`ltp p u v` reads "u precedes v in the order named by p".

Why this is the right definition. Two directions need checking. (i) Every `ltp p` is an irreflexive, transitive, total comparison — a strict linear order on the vertices. (ii) Conversely, *every* strict linear order arises this way: `realize` turns any irreflexive transitive total relation `r` into a permutation, and `ltp_realizeE` states `ltp realize u v = r u v`. So quantifying over `{perm T}` is quantifying over all $|T|!$ vertex orders — no order is missed, which is what makes the *minimum* in $\bar{\omega}$ honest.

2.9 The backedge graph

Fix a tournament T and an order p. The backedge graph has an (undirected) edge between u and v when the arc between them points *backwards* with respect to p:

```
Definition backedge_rel : rel T :=
  [rel u v | ltp p u v && (v --> u) || ltp p v u && (u --> v)].

Definition backedge : sgraph := SGraph backedge_sym backedge_irrefl.

Lemma backedgeE (u v : backedge) :
  (u -- v) = ltp p u v && (v --> u) || ltp p v u && (u --> v).
```

[theories/core/order.v:136]

`sgraph` is the simple-graph type of the `coq-graph-theory` library, whose clique machinery (`cliques`, `omega`) is reused unchanged; `u -- v` is its edge relation.

Why this is the right definition. A set K is a clique of the backedge graph iff, listing K in p-increasing order, every later vertex beats every earlier one — i.e. K is a "reverse-ordered" subtournament under p. This is the standard combinatorial reading of backedge cliques.

2.10 The tournament clique number $\bar{\omega}$

```
Definition omegab_at (T : tournament) (p : {perm T}) : nat :=
  ω([set: backedge p]).

Definition omegabar (T : tournament) : nat :=
  omegab_at [arg min_(p < (1%g : {perm T})) omegab_at p].
```

[theories/invariants/omegabar.v:29]

`omegab_at p` is the clique number of the backedge graph of the order p (graph-theory's `omega` of the full vertex set); `omegabar` takes the *minimum* over all p (`[arg min_(...)]` is MathComp's minimiser over the finite type `{perm T}`). The display notation in the source is the omega-bar of the quoted statements.

Why this is the right definition. $\bar{\omega}(T) = \min_{\prec} \omega(\mathrm{BE}(T, \prec))$ over all $|T|!$ linear orders $\prec$, by the realization argument of Section 2.8. This matches Aboulker–Aubian–Charbit–Lopes (arXiv:2310.04265, where the invariant measures the distance of T from transitivity); $\bar{\omega}(T) = 1$ iff T is transitive (Section 6.1).

2.11 k-$\bar{\omega}$-critical tournaments

```
Definition kcritical (k : nat) (T : tournament) : bool :=
  (ω̄(T) == k) && [forall v : T, ω̄(del_tournament v) == k.-1].
```

[theories/invariants/critical.v:21]

T is k-critical when $\bar{\omega}(T) = k$ and deleting *any single* vertex drops $\bar{\omega}$ to $k-1$. The `[forall v : T, ...]` ranges over all vertices; `k.-1` is $k-1$.

2.12 Subtournaments and vertex deletion

```
Definition sub_tournament (T : tournament) (S : {set T}) :
   tournament :=
  {x : T | x \in S} : tournament.

Definition del_tournament (T : tournament) (v : T) : tournament :=
  sub_tournament [set~ v].

Lemma sub_arcE (u v : {x | P x}) : (u --> v) = (val u --> val v).
```

[theories/core/tournament.v:171]

For $S \subseteq V(T)$, `sub_tournament S` is the induced subtournament: its vertices are the elements of S (a dependent-pair type, whose first projection `val` is the underlying vertex) and its arcs are inherited verbatim (`sub_arcE`). Deleting v is the special case $S = V \setminus \{v\}$ (`[set~ v]`).

2.13 Directed domination

```
Definition dominatesb (X : {set T}) : bool :=
  [forall v : T, (v \in X) || [exists x in X, x --> v]].

Definition domnum : nat := #|[arg min_(X < [set: T] | dominatesb X)
   #|X|]|.
```

[theories/invariants/domination.v:28]

u dominates v if $u = v$ or $u \to v$; $\mathrm{dom}(T)$ is the least size of a set dominating every vertex. Used in the general bound $\mathrm{dom}(T) \le \bar{\omega}(T)$ (Section 6.4).

2.14 Automorphisms

```
Definition autb p : bool :=
  [forall u : D, [forall v : D, arc (p u) (p v) == arc u v]].

Definition dgaut : {set {perm D}} := [set p | autb p].
```

[theories/constructions/automorphism.v:25]

A permutation of the vertices preserving arcs in both directions; `dgaut` is the set of all of them (a group).

2.15 Vertex-transitivity

```
Definition vertex_transitiveb : bool :=
  [forall u : D, [forall v : D,
    [exists p : {perm D}, (p \in dgaut) && (p u == v)]]].
```

[theories/constructions/automorphism.v:55]

For every pair of vertices some automorphism carries one to the other. All three critical families here are vertex-transitive — which is why checking one deletion suffices (Section 6.3).

2.16 Lexicographic substitution $S[H]$

```
Definition lexprod : Type := (D1 * D2)%type.

Lemma lexprod_arcE (u v : lexprod) :
  (u --> v) = (u.1 --> v.1) || (u.1 == v.1) && (u.2 --> v.2).

Definition lexprod_tournament (T1 T2 : tournament) : tournament :=
  lexprod T1 T2 : tournament.
```

[theories/constructions/product.v:25]

Vertices are pairs (s, h); between different S-blocks the arc follows the S-arc of the first coordinates, inside a block it follows H (`lexprod_arcE` is the characterising equation: the arc is `(u.1 --> v.1)|| (u.1 == v.1)&& (u.2 --> v.2)`, and for tournaments the first disjunct can only fire when $u.1 \neq v.1$). This is the standard substitution/composition $S[H]$: every block is a copy of H, blocks are arranged as S.

2.17 Cayley digraphs

```
Definition cayley (gT : finGroupType) (A : {set gT}) : Type := gT.

Lemma cayley_arcE (x y : cayley A) : (x --> y) = ((x^-1 * y)%g \in
    A).
```

[theories/constructions/cayley.v:30]

For a finite group G and connection set $A \subseteq G$: vertices G, arc $x \to y$ iff $x^{-1}y \in A$. Over $\mathbb{Z}/n$ (written additively) this is the circulant: $x \to y$ iff $y - x \in A$.

2.18 The circulant tournament AC_n

The paper's witness platform: for odd $n = 2m + 1$, the circulant with connection set $g = \{1, \ldots, m-1\} \cup \{m+1\}$:

```
Definition ACset : {set 'Z_n} :=
  [set z : 'Z_n | ((0 < val z < m) || (val z == m.+1))%N].

Definition AC : tournament := cayley ACset : tournament.

Lemma AC_arcE (x y : AC) : (x --> y) = (y - x \in ACset).
```

[theories/constructions/circulant.v:63]

(`val z` is the representative of $z \in \mathbb{Z}/n$ in $\{0, \ldots, n-1\}$; the set comprehension says $0 < \mathrm{val}\, z < m$ or $\mathrm{val}\, z = m + 1$.) Since g contains exactly one of $\{z, -z\}$ for each $z \neq 0$, AC_n is a tournament; as a Cayley digraph it is vertex-transitive. Throughout, m is `m'.+1` and n is `m.*2.+1` (Chapter 1).

In statements quoted from section contexts, the local abbreviations `ACm`, `m`, `n` denote `AC m'`, `m'.+1`, `m.*2.+1`.

2.19 The $k = 4$ family $\mathrm{AC}_n[C_3]$

`T4` abbreviates `lexprod_tournament ACm (C3 : tournament)`: the substitution of the directed triangle into AC_n (Sections 2.7, 2.16 and 2.18), a vertex-transitive tournament on $3n$ vertices.

2.20 The $k = 5$ family $\mathrm{AC}_n[\mathrm{AC}_n]$

`T5` abbreviates `lexprod_tournament ACm ACm`: AC_n substituted into itself, a vertex-transitive tournament on n^2 vertices.

2.21 Directed simple paths

```
Definition dipath x s := path arc x s && uniq (x :: s).
```

[theories/core/dipath.v:36]

`path arc x s` (MathComp) says consecutive elements of `x :: s` are joined by arcs; `uniq` makes the path simple. Its *length* (number of arcs) is `size s`.

2.22 The longest-path length $\ell(D)$

```
Definition has_dipath k := [exists x : D, exists t : k.-tuple D,
   dipath x t].

Definition ell := \max_(k < #|D| | has_dipath k) (k : nat).
```

[theories/core/dipath.v:144]

$\ell(D)$ is the largest k for which a directed simple path with k arcs exists (`\max_(...)` over the finite range; a simple path has at most $|D| - 1 < |D|$ arcs, so the bounded maximum is the true maximum). The display notation in the source is the script ell of the quoted statements.

2.23 Strong connectivity

```
Definition strongb := [forall x : D, forall y : D, connect arc x y].
```

[theories/invariants/strong.v:37]

Every vertex reaches every other by a directed walk (`connect` is MathComp's reflexive-transitive closure).

Chapter 3

The unified theorem: Conjecture 5.10 at $k = 3, 4, 5$

Aboulker, Aubian, Charbit and Lopes (arXiv:2310.04265) conjecture (their Conjecture 5.10) that for every $k \geq 3$ there are infinitely many k-$\bar{\omega}$-critical tournaments, and ask (their Question 5.9) whether a tournament with $\bar{\omega}(T) \geq k$ must contain a *bounded-size* subtournament with $\bar{\omega} \geq k$. The library settles every humanly known case of the conjecture and refutes the question at each such k.

Trust. Every theorem in this chapter is verified by the Rocq kernel and reports `Closed under the global context` (`Print Assumptions`) — no axioms, no admitted steps. The witnesses are fully explicit: AC_n, $\mathrm{AC}_n[C_3]$, $\mathrm{AC}_n[\mathrm{AC}_n]$ (Sections 2.18 to 2.20). The small instances were independently cross-checked by exact and SAT-based computation (`scripts/`).

3.1 The headline: every $k \in \{3, 4, 5\}$

```
Theorem conjecture_5_10_at_345 (k : nat) : (3 <= k <= 5)%N ->
  forall N : nat, exists T : tournament, kcritical k T /\ (N <
     #|T|)%N.
```

[theories/applications/unified.v:66]

Decoded. For every k with $3 \leq k \leq 5$ and every bound N there exists a k-$\bar{\omega}$-critical tournament on more than N vertices: the k-critical tournaments form an infinite family. ("T is k-critical" unfolds per Section 2.11: $\bar{\omega}(T) = k$ and $\bar{\omega}(T - v) = k - 1$ for every vertex v.)

Everything this statement needs[a]

- Digraphs and arcs (u –> v) (§2.1)
- Tournaments (irreflexive + total) (§2.5)
- Subtournaments and vertex deletion (§2.12)
- Vertex orders as permutations (ltp, realize) (§2.8)
- The backedge graph of an ordered tournament (§2.9)
- The tournament clique number omegabar (§2.10)
- k-omegabar-critical tournaments (§2.11)
- plus the standard finite-mathematics vocabulary of the Glossary (Appendix A)

[a]This list is *generated* from the build: a script computes every constant the formal statement references, transitively through library definitions, and CI fails if any of them lacks an entry in the Dictionary (`scripts/statement_closure.py`).

3.2 $k = 3$: infinitely many, and Question 5.9 fails

```
Theorem conjecture_5_10_at_k3 (N : nat) :
  exists T : tournament, kcritical 3 T /\ (N < #|T|)%N.
```

[theories/applications/unified.v:33]

```
Theorem question_5_9_fails_at_k3 (L : nat) :
  exists T : tournament,
    [/\ kcritical 3 T, (L < #|T|)%N
      & forall S : {set T}, (3 <= ω̄(sub_tournament S))%N -> S =
          [set: T]].
```

[theories/applications/unified.v:47]

Decoded. First statement: for every N there is a 3-critical tournament on more than N vertices (the witness is AC_n itself, $n = 2\max(2, N)+3$). Second: moreover the witness can be chosen so that *its only subtournament with $\bar{\omega} \geq 3$ is the whole tournament* — formally, any vertex subset S whose induced subtournament has $\bar{\omega} \geq 3$ must be all of V (`[set: T]` is the full set). Hence no function $\ell(3)$ as in Question 5.9 can exist: witnesses have unbounded size but no proper witnessing subtournament at all.

Everything this statement needs[a]

- Digraphs and arcs (u –> v) (§2.1)
- Tournaments (irreflexive + total) (§2.5)
- Subtournaments and vertex deletion (§2.12)
- Vertex orders as permutations (ltp, realize) (§2.8)
- The backedge graph of an ordered tournament (§2.9)
- The tournament clique number omegabar (§2.10)
- k-omegabar-critical tournaments (§2.11)
- plus the standard finite-mathematics vocabulary of the Glossary (Appendix A)

[a]This list is *generated* from the build: a script computes every constant the formal statement references, transitively through library definitions, and CI fails if any of them lacks an entry in the Dictionary (`scripts/statement_closure.py`).

3.3 $k = 4$

```
Theorem conjecture_5_10_at_k4 (N : nat) :
  exists T : tournament, kcritical 4 T /\ (N < #|T|)%N.
```

[theories/applications/k4/k4_main.v:74]

```
Theorem question_5_9_fails_at_k4 (L : nat) :
  exists T : tournament,
    [/\ kcritical 4 T, (L < #|T|)%N
```

```
      & forall S : {set T}, (4 <= ω̄(sub_tournament S))%N -> S =
          [set: T]].
```

[theories/applications/k4/k4_main.v:89]

Decoded. As for $k = 3$, with witness $\mathrm{AC}_n[C_3]$ on $3n$ vertices (Section 2.19; the underlying values $\bar{\omega} = 4$, $\bar{\omega}(T - v) = 3$ are Sections 4.4 and 4.5).

Everything this statement needs[a]

- Digraphs and arcs (u –> v) (§2.1)
- Tournaments (irreflexive + total) (§2.5)
- Subtournaments and vertex deletion (§2.12)
- Vertex orders as permutations (ltp, realize) (§2.8)
- The backedge graph of an ordered tournament (§2.9)
- The tournament clique number omegabar (§2.10)
- k-omegabar-critical tournaments (§2.11)
- plus the standard finite-mathematics vocabulary of the Glossary (Appendix A)

[a] This list is *generated* from the build: a script computes every constant the formal statement references, transitively through library definitions, and CI fails if any of them lacks an entry in the Dictionary (`scripts/statement_closure.py`).

3.4 $k = 5$

```
Theorem conjecture_5_10_at_k5 (N : nat) :
  exists T : tournament, kcritical 5 T /\ (N < #|T|)%N.
```

[theories/applications/k5/main.v:108]

```
Theorem question_5_9_fails_at_k5 (L : nat) :
  exists T : tournament,
    [/\ kcritical 5 T, (L < #|T|)%N
      & forall S : {set T}, (5 <= ω̄(sub_tournament S))%N -> S =
          [set: T]].
```

[theories/applications/k5/main.v:127]

Decoded. As above, with witness $\mathrm{AC}_n[\mathrm{AC}_n]$ on n^2 vertices (Section 2.20).

Everything this statement needs[a]

- Digraphs and arcs (u –> v) (§2.1)
- Tournaments (irreflexive + total) (§2.5)
- Subtournaments and vertex deletion (§2.12)
- Vertex orders as permutations (ltp, realize) (§2.8)
- The backedge graph of an ordered tournament (§2.9)
- The tournament clique number omegabar (§2.10)
- k-omegabar-critical tournaments (§2.11)
- plus the standard finite-mathematics vocabulary of the Glossary (Appendix A)

[a] This list is *generated* from the build: a script computes every constant the formal statement references, transitively through library definitions, and CI fails if any of them lacks an entry in the Dictionary (`scripts/statement_closure.py`).

Chapter 4

The three critical families

The three witnesses share one platform: the circulant tournament AC_n on $n = 2m+1 \geq 7$ vertices (Section 2.18), used directly for $k = 3$ and composed with itself or with C_3 by lexicographic substitution (Section 2.16) for $k = 5, 4$. All statements below are quoted from inside sections fixing `m' : nat` with `(3 <= m'.+1)%N` (see Chapter 1 for the m', m, n convention): each is universally quantified over all odd $n \geq 7$.

$k = 3$: the tournament AC_n

4.1 $\bar{\omega}(\mathrm{AC}_n) = 3$

```
Theorem omegabar_AC : (3 <= m)%N -> ω̄((AC m' : tournament)) = 3.
```

[theories/applications/k5/acn_base.v:374]

Decoded. For every odd $n = 2m + 1$ with $m \geq 3$, the tournament clique number of AC_n is exactly 3. `((AC m' : tournament)` is AC_n viewed as a tournament; the hypothesis `3 <= m` is $m \geq 3$.)

Everything this statement needs[a]

- Digraphs and arcs (u –> v) (§2.1)
- Tournaments (irreflexive + total) (§2.5)
- Vertex orders as permutations (ltp, realize) (§2.8)
- The backedge graph of an ordered tournament (§2.9)
- The tournament clique number omegabar (§2.10)
- Cayley digraphs (§2.17)
- The circulant tournament ACn (n = 2m+1) (§2.18)
- plus the standard finite-mathematics vocabulary of the Glossary (Appendix A)

[a] This list is *generated* from the build: a script computes every constant the formal statement references, transitively through library definitions, and CI fails if any of them lacks an entry in the Dictionary (`scripts/statement_closure.py`).

4.2 $\bar{\omega}(\mathrm{AC}_n - v) = 2$ for every v

```
Theorem omegabar_AC_del (v : ACm) :
  (3 <= m)%N -> ω̄(del_tournament v) = 2.
```

[theories/applications/k5/acn_base.v:423]

Decoded. Deleting any single vertex v from AC_n drops the invariant to 2.

Everything this statement needs[a]

- Digraphs and arcs (u –> v) (§2.1)
- Tournaments (irreflexive + total) (§2.5)
- Subtournaments and vertex deletion (§2.12)
- Vertex orders as permutations (ltp, realize) (§2.8)
- The backedge graph of an ordered tournament (§2.9)
- The tournament clique number omegabar (§2.10)
- Cayley digraphs (§2.17)
- The circulant tournament ACn (n = 2m+1) (§2.18)
- plus the standard finite-mathematics vocabulary of the Glossary (Appendix A)

[a] This list is *generated* from the build: a script computes every constant the formal statement references, transitively through library definitions, and CI fails if any of them lacks an entry in the Dictionary (`scripts/statement_closure.py`).

4.3 AC_n is 3-critical; its basic stats

```
Theorem AC_kcritical3 : (3 <= m)%N -> kcritical 3 (AC m' :
    tournament).
```

[theories/applications/k5/acn_base.v:431]

```
Lemma card_AC : #|AC| = n.
```

[theories/constructions/circulant.v:129]

```
Lemma AC_vertex_transitive : vertex_transitiveb AC.
```

[theories/constructions/circulant.v:134]

Decoded. AC_n is 3-$\bar{\omega}$-critical (Section 2.11), has exactly n vertices, and is vertex-transitive (Section 2.15).

Everything this statement needs[a]

- Digraphs and arcs (u –> v) (§2.1)
- Tournaments (irreflexive + total) (§2.5)
- Subtournaments and vertex deletion (§2.12)
- Vertex orders as permutations (ltp, realize) (§2.8)
- The backedge graph of an ordered tournament (§2.9)
- The tournament clique number omegabar (§2.10)
- k-omegabar-critical tournaments (§2.11)

- Cayley digraphs (§2.17)
- The circulant tournament ACn (n = 2m+1) (§2.18)
- plus the standard finite-mathematics vocabulary of the Glossary (Appendix A)

[a]This list is *generated* from the build: a script computes every constant the formal statement references, transitively through library definitions, and CI fails if any of them lacks an entry in the Dictionary (`scripts/statement_closure.py`).

$k = 4$: the tournament $\mathrm{AC}_n[C_3]$

4.4 $\bar{\omega}(\mathrm{AC}_n[C_3]) = 4$

```
Theorem omegabar_T4 : ω̄(T4) = 4.
```

[theories/applications/k4/k4_value.v:511]

Decoded. Here `T4` abbreviates the substitution $\mathrm{AC}_n[C_3]$ (Section 2.19). For every odd $n = 2m+1 \geq 7$, its tournament clique number is exactly 4.

Everything this statement needs[a]

- Digraphs and arcs (u –> v) (§2.1)
- Tournaments (irreflexive + total) (§2.5)
- The directed triangle C3 (§2.7)
- Vertex orders as permutations (ltp, realize) (§2.8)
- The backedge graph of an ordered tournament (§2.9)
- The tournament clique number omegabar (§2.10)
- Lexicographic substitution S[H] (§2.16)
- Cayley digraphs (§2.17)
- The circulant tournament ACn (n = 2m+1) (§2.18)
- The k = 4 family ACn[C3] (§2.19)
- plus the standard finite-mathematics vocabulary of the Glossary (Appendix A)

[a]This list is *generated* from the build: a script computes every constant the formal statement references, transitively through library definitions, and CI fails if any of them lacks an entry in the Dictionary (`scripts/statement_closure.py`).

4.5 $\bar{\omega}(\mathrm{AC}_n[C_3] - v) = 3$ for every v

```
Theorem omegabar_T4_del (v : T4) : ω̄(del_tournament v) = 3.
```

[theories/applications/k4/k4_main.v:51]

Decoded. Deleting any vertex of $\mathrm{AC}_n[C_3]$ drops the invariant to 3 (by vertex-transitivity all deletions are isomorphic, so this is one computation plus symmetry — see Section 6.3).

Everything this statement needs[a]

- Digraphs and arcs (u –> v) (§2.1)

- Tournaments (irreflexive + total) (§2.5)
- The directed triangle C3 (§2.7)
- Subtournaments and vertex deletion (§2.12)
- Vertex orders as permutations (ltp, realize) (§2.8)
- The backedge graph of an ordered tournament (§2.9)
- The tournament clique number omegabar (§2.10)
- Lexicographic substitution S[H] (§2.16)
- Cayley digraphs (§2.17)
- The circulant tournament ACn (n = 2m+1) (§2.18)
- The k = 4 family ACn[C3] (§2.19)
- plus the standard finite-mathematics vocabulary of the Glossary (Appendix A)

[a] This list is *generated* from the build: a script computes every constant the formal statement references, transitively through library definitions, and CI fails if any of them lacks an entry in the Dictionary (`scripts/statement_closure.py`).

4.6 $\mathrm{AC}_n[C_3]$ is 4-critical, on $3n$ vertices

```
Theorem T4_kcritical4 : kcritical 4 T4.
```

[theories/applications/k4/k4_main.v:59]

```
Lemma card_T4 : #|T4| = (n * 3)%N.
```

[theories/applications/k4/k4_main.v:67]

Decoded. The two previous values say precisely that $\mathrm{AC}_n[C_3]$ is 4-$\bar{\omega}$-critical; its order is $n \cdot 3$.

Everything this statement needs[a]

- Digraphs and arcs (u –> v) (§2.1)
- Tournaments (irreflexive + total) (§2.5)
- The directed triangle C3 (§2.7)
- Subtournaments and vertex deletion (§2.12)
- Vertex orders as permutations (ltp, realize) (§2.8)
- The backedge graph of an ordered tournament (§2.9)
- The tournament clique number omegabar (§2.10)
- k-omegabar-critical tournaments (§2.11)
- Lexicographic substitution S[H] (§2.16)
- Cayley digraphs (§2.17)
- The circulant tournament ACn (n = 2m+1) (§2.18)
- The k = 4 family ACn[C3] (§2.19)
- plus the standard finite-mathematics vocabulary of the Glossary (Appendix A)

[a] This list is *generated* from the build: a script computes every constant the formal statement references, transitively through library definitions, and CI fails if any of them lacks an entry in the Dictionary (`scripts/statement_closure.py`).

$k = 5$: the tournament $\mathrm{AC}_n[\mathrm{AC}_n]$

4.7 $\bar{\omega}(\mathrm{AC}_n[\mathrm{AC}_n]) = 5$

```
Theorem omegabar_T5 : ω̄(T5) = 5.
```

[theories/applications/k5/main.v:54]

Decoded. `T5` abbreviates $\mathrm{AC}_n[\mathrm{AC}_n]$ (Section 2.20); its tournament clique number is exactly 5, for every odd $n = 2m{+}1 \geq 7$.

Everything this statement needs[a]

- Digraphs and arcs (u –> v) (§2.1)
- Tournaments (irreflexive + total) (§2.5)
- Vertex orders as permutations (ltp, realize) (§2.8)
- The backedge graph of an ordered tournament (§2.9)
- The tournament clique number omegabar (§2.10)
- Lexicographic substitution S[H] (§2.16)
- Cayley digraphs (§2.17)
- The circulant tournament ACn (n = 2m+1) (§2.18)
- The k = 5 family ACn[ACn] (§2.20)
- plus the standard finite-mathematics vocabulary of the Glossary (Appendix A)

[a] This list is *generated* from the build: a script computes every constant the formal statement references, transitively through library definitions, and CI fails if any of them lacks an entry in the Dictionary (`scripts/statement_closure.py`).

4.8 $\bar{\omega}(\mathrm{AC}_n[\mathrm{AC}_n] - v) = 4$; 5-criticality; order n^2

```
Theorem omegabar_T5_del (v : T5) : ω̄(del_tournament v) = 4.
```

[theories/applications/k5/main.v:62]

```
Theorem T5_kcritical5 : kcritical 5 T5.
```

[theories/applications/k5/main.v:71]

```
Lemma card_T5 : #|T5| = (n * n)%N.
```

[theories/applications/k5/main.v:79]

Decoded. Every single-vertex deletion has $\bar{\omega} = 4$; hence $\mathrm{AC}_n[\mathrm{AC}_n]$ is 5-$\bar{\omega}$-critical, on $n \cdot n$ vertices.

Everything this statement needs[a]

- Digraphs and arcs (u –> v) (§2.1)
- Tournaments (irreflexive + total) (§2.5)
- Subtournaments and vertex deletion (§2.12)
- Vertex orders as permutations (ltp, realize) (§2.8)
- The backedge graph of an ordered tournament (§2.9)
- The tournament clique number omegabar (§2.10)
- k-omegabar-critical tournaments (§2.11)
- Lexicographic substitution S[H] (§2.16)
- Cayley digraphs (§2.17)
- The circulant tournament ACn (n = 2m+1) (§2.18)
- The k = 5 family ACn[ACn] (§2.20)
- plus the standard finite-mathematics vocabulary of the Glossary (Appendix A)

[a] This list is *generated* from the build: a script computes every constant the formal statement references, transitively through library definitions, and CI fails if any of them lacks an entry in the Dictionary (`scripts/statement_closure.py`).

Chapter 5

Directed paths: the Cheng–Keevash results

Thomassé's path conjecture, in the oriented form studied by Cheng and Keevash ("Conjecture 1"), asserts that every oriented digraph with minimum out-degree $\delta^+ \geq k$ contains a directed simple path of length $2k$. The library proves the cases $k = 2$ and $k = 3$ and the general Lemma 7 of Cheng–Keevash, uniform in δ.

Trust. All results axiom-free (`Print Assumptions` closed). Hypotheses are stated on arbitrary finite oriented digraphs (Section 2.3) — no hidden strong-connectivity or regularity assumptions; where the *proof* reduces to a strong subdigraph, that reduction is part of the formalized proof, not of the statement.

5.1 The $\delta = 3$ path theorem

```
Theorem ck_conj1_delta3 (D : orientedDigraph) :
  0 < #|D| -> (forall v : D, 3 <= outdeg v) -> 6 <= ell D.
```

[theories/applications/ck3/ck3_main.v:147]

```
Corollary ck_conj1_delta3_path (D : orientedDigraph) :
  0 < #|D| -> (forall v : D, 3 <= outdeg v) ->
  exists x : D, exists s : seq D, dipath x s /\ size s = 6.
```

[theories/applications/ck3/ck3_main.v:158]

Decoded. Let D be any nonempty finite oriented digraph in which every vertex has out-degree at least 3. Then $\ell(D) \geq 6$ (Section 2.22): D contains a directed simple path with 6 arcs (7 distinct vertices). The second form unfolds ℓ into the explicit existence of such a path (Section 2.21). This is the $k = 3$ case of Cheng–Keevash Conjecture 1.

Everything this statement needs[a]

- Digraphs and arcs (u –> v) (§2.1)
- Oriented digraphs (irreflexive + asymmetric) (§2.3)
- Out-degree (§2.4)
- Directed simple paths (§2.21)
- The longest-dipath length ell(D) (§2.22)
- plus the standard finite-mathematics vocabulary of the Glossary (Appendix A)

[a] This list is *generated* from the build: a script computes every constant the formal statement references, transitively through library definitions, and CI fails if any of them lacks an entry in the Dictionary (`scripts/statement_closure.py`).

5.2 The $\delta = 2$ case

```
Corollary ck_conj1_delta2 (D : orientedDigraph) :
  0 < #|D| -> (forall v : D, 2 <= outdeg v) -> 4 <= ell D.
```

[theories/applications/ck3/lemma7.v:698]

Decoded. Same statement one level down: minimum out-degree ≥ 2 forces a directed simple path of length 4.

Everything this statement needs[a]

- Digraphs and arcs (u –> v) (§2.1)
- Oriented digraphs (irreflexive + asymmetric) (§2.3)
- Out-degree (§2.4)
- Directed simple paths (§2.21)
- The longest-dipath length ell(D) (§2.22)
- plus the standard finite-mathematics vocabulary of the Glossary (Appendix A)

[a] This list is *generated* from the build: a script computes every constant the formal statement references, transitively through library definitions, and CI fails if any of them lacks an entry in the Dictionary (`scripts/statement_closure.py`).

5.3 Cheng–Keevash Lemma 7, uniform in δ

```
Theorem lemma7 (D : orientedDigraph) (delta : nat) :
  0 < #|D| -> (forall v : D, delta <= outdeg v) ->
  2 * delta <= ell D \/
  exists S : {set D},
    [/\ S != set0, #|S| <= delta
      & forall v, v \in S -> 2 * delta - ell D <= outdeg_in S v].
```

[theories/applications/ck3/lemma7.v:646]

Decoded. In a strong (Section 2.23) oriented digraph that is δ-out-regular-from-below ($d^+(v) \geq \delta$ everywhere, $\delta \geq 1$) and where the longest path is short ($\ell(D) \leq 2\delta$), the vertex set of any longest path supports the structural "kernel" conclusion quantified in the statement — the engine behind the path theorems. (This page quotes the statement in full; the surrounding section fixes the oriented digraph `D`, δ, and the strongness/degree hypotheses visible in the quoted text.)

Everything this statement needs[a]

- Digraphs and arcs (u –> v) (§2.1)
- Oriented digraphs (irreflexive + asymmetric) (§2.3)
- Out-degree (§2.4)

- Directed simple paths (§2.21)
- The longest-dipath length ell(D) (§2.22)
- plus the standard finite-mathematics vocabulary of the Glossary (Appendix A)

[a]This list is *generated* from the build: a script computes every constant the formal statement references, transitively through library definitions, and CI fails if any of them lacks an entry in the Dictionary (`scripts/statement_closure.py`).

5.4 Theorem 4, oriented version

```
Theorem ck_theorem4_oriented (D : orientedDigraph) (delta : nat) :
  0 < #|D| -> (forall v : D, delta <= outdeg v) ->
  2 * delta - (delta - 1)./2 <= ell D.
```

[theories/applications/ck3/lemma7.v:684]

Decoded. Every nonempty oriented digraph with minimum out-degree $\geq \delta$ satisfies $\ell(D) \geq 2\delta - \lfloor(\delta-1)/2\rfloor$ (`./2` is truncated halving). For even δ this is sharper than the $\lceil 3\delta/2\rceil$ stated by Cheng–Keevash.

Everything this statement needs[a]

- Digraphs and arcs (u –> v) (§2.1)
- Oriented digraphs (irreflexive + asymmetric) (§2.3)
- Out-degree (§2.4)
- Directed simple paths (§2.21)
- The longest-dipath length ell(D) (§2.22)
- plus the standard finite-mathematics vocabulary of the Glossary (Appendix A)

[a]This list is *generated* from the build: a script computes every constant the formal statement references, transitively through library definitions, and CI fails if any of them lacks an entry in the Dictionary (`scripts/statement_closure.py`).

5.5 No short-path strong counterexample at $\delta = 3$

```
Lemma no_short_strong3 (H : orientedDigraph) :
  (forall v : H, outdeg v = 3) -> 0 < #|H| -> strongb H ->
  ell H <= 5 -> False.
```

[theories/applications/ck3/ck3_main.v:28]

Decoded. There is no strong oriented digraph with $\delta^+ \geq 3$ and $\ell \leq 5$: the reduction target of the main theorem, stated separately.

Everything this statement needs[a]

- Digraphs and arcs (u –> v) (§2.1)
- Oriented digraphs (irreflexive + asymmetric) (§2.3)
- Out-degree (§2.4)
- Directed simple paths (§2.21)

- The longest-dipath length ell(D) (§2.22)
- Strong connectivity (§2.23)
- plus the standard finite-mathematics vocabulary of the Glossary (Appendix A)

[a] This list is *generated* from the build: a script computes every constant the formal statement references, transitively through library definitions, and CI fails if any of them lacks an entry in the Dictionary (`scripts/statement_closure.py`).

Chapter 6

General tournament theory

The reusable theorems behind the families — of independent interest for anyone working with $\bar{\omega}$.

6.1 $\bar{\omega} = 1$ exactly for transitive tournaments

```
Lemma omegabar_transb : 0 < #|T| -> (ω̄(T) == 1) = transb T.
```

[theories/invariants/omegabar.v:74]

Decoded. For a nonempty tournament, $\bar{\omega}(T) = 1$ iff T is transitive (Section 2.6) — the sanity anchor for reading $\bar{\omega}$ as a distance from transitivity. (The statement is a Boolean equation between $\bar{\omega}(T) = 1$ and `transb T`, both Booleans.)

Everything this statement needs[a]

- Digraphs and arcs (u –> v) (§2.1)
- Tournaments (irreflexive + total) (§2.5)
- Transitive tournaments (§2.6)
- Vertex orders as permutations (ltp, realize) (§2.8)
- The backedge graph of an ordered tournament (§2.9)
- The tournament clique number omegabar (§2.10)
- plus the standard finite-mathematics vocabulary of the Glossary (Appendix A)

[a] This list is *generated* from the build: a script computes every constant the formal statement references, transitively through library definitions, and CI fails if any of them lacks an entry in the Dictionary (`scripts/statement_closure.py`).

6.2 The substitution lower bound

```
Theorem omegabar_lexprod_ge :
  ω̄(S) + ω̄(H) <= ω̄(lexprod_tournament S H) + 1.
```

[theories/invariants_advanced/substitution.v:168]

Decoded. $\bar{\omega}(S) + \bar{\omega}(H) \leq \bar{\omega}(S[H]) + 1$ for any tournaments S, H (quoted inside a section fixing nonempty `S`, `H`): substitution composes criticality levels. This is what makes the tower $\mathrm{AC}_n \to \mathrm{AC}_n[C_3] \to \mathrm{AC}_n[\mathrm{AC}_n]$ work.

Everything this statement needs[a]

- Digraphs and arcs (u –> v) (§2.1)
- Tournaments (irreflexive + total) (§2.5)
- Vertex orders as permutations (ltp, realize) (§2.8)
- The backedge graph of an ordered tournament (§2.9)
- The tournament clique number omegabar (§2.10)
- Lexicographic substitution S[H] (§2.16)
- plus the standard finite-mathematics vocabulary of the Glossary (Appendix A)

[a]This list is *generated* from the build: a script computes every constant the formal statement references, transitively through library definitions, and CI fails if any of them lacks an entry in the Dictionary (`scripts/statement_closure.py`).

6.3 Vertex-transitive tournaments: one deletion decides criticality

```
Theorem vt_kcritical (k : nat) (v0 : T) :
  vertex_transitiveb (T : diGraphType) ->
  kcritical k T = (ω̄(T) == k) && (ω̄(del_tournament v0) == k.-1).
```

[theories/invariants_advanced/transitive.v:57]

```
Theorem omegabar_del_vt :
  vertex_transitiveb (T : diGraphType) ->
  forall u v : T, ω̄(del_tournament u) = ω̄(del_tournament v).
```

[theories/invariants_advanced/transitive.v:47]

Decoded. In a vertex-transitive tournament all single-vertex deletions have the same $\bar{\omega}$ (second statement), so k-criticality reduces to the value of $\bar{\omega}(T)$ and of one deletion $\bar{\omega}(T - v_0)$ (first statement, a Boolean equivalence).

Everything this statement needs[a]

- Digraphs and arcs (u –> v) (§2.1)
- Tournaments (irreflexive + total) (§2.5)
- Subtournaments and vertex deletion (§2.12)
- Vertex orders as permutations (ltp, realize) (§2.8)
- The backedge graph of an ordered tournament (§2.9)
- The tournament clique number omegabar (§2.10)
- k-omegabar-critical tournaments (§2.11)
- Automorphisms of a digraph (§2.14)
- Vertex-transitivity (§2.15)
- plus the standard finite-mathematics vocabulary of the Glossary (Appendix A)

[a]This list is *generated* from the build: a script computes every constant the formal statement references, transitively through library definitions, and CI fails if any of them lacks an entry in the Dictionary (`scripts/statement_closure.py`).

6.4 Domination bounds the clique number

```
Theorem domnum_le_omegabar : domnum <= ω̄(T).
```

[theories/invariants/domination.v:114]

Decoded. $\mathrm{dom}(T) \le \bar{\omega}(T)$ for every tournament: a minimum backedge-clique under an optimal order dominates (Section 2.13). This is the standard lower-bound tool for $\bar{\omega}$.

Everything this statement needs[a]

- Digraphs and arcs (u –> v) (§2.1)
- Tournaments (irreflexive + total) (§2.5)
- Vertex orders as permutations (ltp, realize) (§2.8)
- The backedge graph of an ordered tournament (§2.9)
- The tournament clique number omegabar (§2.10)
- Directed domination (§2.13)
- plus the standard finite-mathematics vocabulary of the Glossary (Appendix A)

[a] This list is *generated* from the build: a script computes every constant the formal statement references, transitively through library definitions, and CI fails if any of them lacks an entry in the Dictionary (`scripts/statement_closure.py`).

6.5 C_3 is the unique 2-critical tournament

```
Theorem kcritical2_uniq : dgiso (C3 : diGraphType) T.
```

[theories/invariants/critical.v:124]

Decoded. Every 2-$\bar{\omega}$-critical tournament is isomorphic (Section 2.2) to the directed triangle (Section 2.7): the base of the criticality hierarchy is unique.

Everything this statement needs[a]

- Digraphs and arcs (u –> v) (§2.1)
- Digraph isomorphism (§2.2)
- The directed triangle C3 (§2.7)
- plus the standard finite-mathematics vocabulary of the Glossary (Appendix A)

[a] This list is *generated* from the build: a script computes every constant the formal statement references, transitively through library definitions, and CI fails if any of them lacks an entry in the Dictionary (`scripts/statement_closure.py`).

6.6 Proper subtournaments of critical tournaments

```
Lemma kcritical_proper_sub (k : nat) (T : tournament) (S : {set T})
    :
  kcritical k T -> S != [set: T] -> (ω̄(sub_tournament S) <= k.-1)%N.
```

[theories/invariants/critical.v:46]

Decoded. If T is k-$\bar{\omega}$-critical and $S \subsetneq V(T)$, then the induced subtournament on S has $\bar{\omega} \leq k-1$: criticality leaves no proper witness inside. This single lemma is what refutes Question 5.9 at every k (Chapter 3).

Everything this statement needs[a]

- Digraphs and arcs (u –> v) (§2.1)
- Tournaments (irreflexive + total) (§2.5)
- Subtournaments and vertex deletion (§2.12)
- Vertex orders as permutations (ltp, realize) (§2.8)
- The backedge graph of an ordered tournament (§2.9)
- The tournament clique number omegabar (§2.10)
- k-omegabar-critical tournaments (§2.11)
- plus the standard finite-mathematics vocabulary of the Glossary (Appendix A)

[a] This list is *generated* from the build: a script computes every constant the formal statement references, transitively through library definitions, and CI fails if any of them lacks an entry in the Dictionary (`scripts/statement_closure.py`).

Appendix A

Glossary of MathComp / graph-theory primitives

The external vocabulary the quoted statements draw on. Each entry is standard library material (MathComp 2.5 / `coq-graph-theory` 0.9.7), battle-tested far beyond this project; the entries below give the reading, not a re-development.

`finType` A type with a (computable) finite enumeration. All vertex types in this document are finite.

`#|T|`, `#|A|` Cardinality of a finite type / of a set.

`{set T}` Subsets of the finite type `T`; `x \in A` membership; `[set: T]` the full set; `[set~ v]` the complement of $\{v\}$; `A != B` Boolean disequality of sets.

`{perm T}` Permutations of `T` (the symmetric group); `enum_rank x` the position of x in the canonical enumeration.

`'Z_n` The ring $\mathbb{Z}/n\mathbb{Z}$ (for the $n \geq 2$ in use here); `val z` its representative in $\{0, \ldots, n-1\}$; arithmetic `+`, `-` is modular.

`rel T` Boolean binary relations on `T`.

`b = true`, `is_true` A Boolean used as a statement means "it computes to `true`" (Chapter 1).

`==`, `!=` Decidable equality / disequality (provably equivalent to `=` on these types).

`(+)` Exclusive or of Booleans.

`[forall x : T, P]`, `[exists x, P]` Boolean quantifiers over finite types.

`reflect P b` "The Boolean `b` is equivalent to the proposition `P`".

`n.+1`, `n.-1`, `n.*2`, `n./2` Successor, truncated predecessor, doubling, truncated halving on `nat`; `maxn` maximum.

`path e x s`, `uniq s`, `size s` MathComp sequences: `path` chains the relation `e` along `x :: s`; `uniq` no repetition; `size` length.

`connect e x y` Reflexive–transitive closure: a directed `e`-walk from x to y exists.

`sgraph` `coq-graph-theory`'s finite simple graphs; `u -- v` adjacency (symmetric, irreflexive).

`cliques`, `omega` `coq-graph-theory`'s cliques of an `sgraph` and its clique number ω (the maximum size of a set of pairwise adjacent vertices). $\bar{\omega}$ (Section 2.10) is built on these — the one place this library leans on its undirected companion, by design.

`[arg min_(i < i0) F]` MathComp's minimiser of `F` over a finite type, with default `i0`.

`Sub`, `val`, `{x | P x}` Dependent pairs: an element together with a proof; `val` forgets the proof (the underlying element).

Appendix B

The full catalog (auto-generated)

Every named declaration in the library's build chain, with its source location. Entries shown in the narrative chapters are marked ★. Everything else is internal proof machinery: it affects *whether* the theorems hold (the kernel checked that), never *what* they say.

theories/foundations/prelude.v

prelude_classical (lemma) prelude.v:29 — Classical logic is in scope (derived from boolp's axioms).

prelude_set (lemma) prelude.v:33 — The classical set layer is in scope.

prelude_ssr (lemma) prelude.v:37 — MathComp's small-scale reflection is in scope.

card_classes (lemma) prelude.v:47 — ** General counting Partition counting by a bounded nat-valued class function: any finite set splits as the sum of its class sizes.

card_classes_inj (lemma) prelude.v:68 — When the class function is injective on K (one element per class at most — e.g.

theories/foundations/interop_graph_theory.v

clique_set1 (lemma) interop_graph_theory.v:30

set1_cliques (lemma) interop_graph_theory.v:33

omega_ge1 (lemma) interop_graph_theory.v:36

omega_le_card (lemma) interop_graph_theory.v:41

omega_le1 (lemma) interop_graph_theory.v:47 — An edgeless subset has clique number at most 1.

omega_ge2 (lemma) interop_graph_theory.v:56 — Conversely, a single edge pushes ω to at least 2.

omega_hom (lemma) interop_graph_theory.v:69 — ω is monotone along injective edge-preserving maps (used to compare backedge graphs of a sub-tournament and its host).

K2_rel (definition) interop_graph_theory.v:88 — ** Smoke check — ω of a tiny hand-built sgraph The original M0 exit criterion: graph-theory's clique number is usable on a concrete graph.

K2_sym (fact) interop_graph_theory.v:89

K2_irrefl (fact) interop_graph_theory.v:91

K2 (definition) interop_graph_theory.v:93

omega_K2 (lemma) interop_graph_theory.v:95

theories/core/digraph.v

to_GT (definition) digraph.v:56 — ** Projection into graph-theory's record world (D7 glue) diGraph is graph-theory's notation for its relType record (re-exported by the interop file); RelType is its constructor — its DiGraph alias is shadowed by our structure module above.

converse (definition) digraph.v:60 — ** Converse digraph (type alias, mathcomp T^d-style)

converse_arcE (lemma) digraph.v:66

sub_arcE (lemma) digraph.v:78

induced_digraph (definition) digraph.v:86 — Object-level versions, convenient in statements (the : diGraphType cast goes through Rocq's reverse-coercion via the canonical instances).

del_vertex (definition) digraph.v:89

dgiso (definition) digraph.v:97 — ** Digraph isomorphism (Named dgiso: graph-theory's diso — re-exported through the interop — is the *simple-graph* iso

dgiso_refl (lemma) digraph.v:100

dgiso_sym (lemma) digraph.v:103

dgiso_trans (lemma) digraph.v:109

dgiso_card (lemma) digraph.v:117

theories/core/oriented.v

outdeg (definition) oriented.v:47

outdeg_in (definition) oriented.v:51 — Out-degree into a fixed vertex subset — the induced out-degree when v \in A.

outdeg_inT (lemma) oriented.v:53

outdeg_in_le (lemma) oriented.v:56

outdeg_in_mono (lemma) oriented.v:61

outdeg_in_sumE (lemma) oriented.v:67

outdeg_induced (lemma) oriented.v:76 — The induced subgraph on A has the inner out-degrees.

arc_pair_bound (lemma) oriented.v:92

oriented_arcs_bound (lemma) oriented.v:100 — O1, counting form: twice the arc count of DA is $\leq$ |A|(|A|–1).

oriented_avg_bound (lemma) oriented.v:119 — O1(a): some vertex of a nonempty A has inner out-degree $\leq$ (|A|–1)./2.

oriented_mindeg_card (lemma) oriented.v:144 — O1(b): if all inner out-degrees are at least k, then |A| $\geq$ 2k+1.

oriented_card (lemma) oriented.v:159 — O1(c): a nonempty oriented digraph with min out-degree $\geq$ k has $\geq$ 2k+1 vertices.

sub_oarc_irrefl (fact) oriented.v:175

sub_oarc_asymm (fact) oriented.v:178

induced_oriented (definition) oriented.v:187 — Object-level version for statements.

outsel (definition) oriented.v:195 — The alias takes f as a REAL parameter (M2 lesson: a type alias must use its parameters, or section discharge drops them and all selections collapse onto one type).

outsel_arcE (lemma) oriented.v:204

outsel_arc_sub (lemma) oriented.v:208

outsel_irrefl (fact) oriented.v:219

outsel_asymm (fact) oriented.v:222

ksel (definition) oriented.v:237

ksel_sub (lemma) oriented.v:239

ksel_card (lemma) oriented.v:245

outsel_ksel_outdeg (lemma) oriented.v:254 — O2's exit: in outsel ksel, every out-degree is exactly k.

theories/core/dipath.v

dipath (definition) dipath.v:36 — ** Directed simple paths

dipath_path (lemma) dipath.v:38

dipath_uniq (lemma) dipath.v:41

dipath_nil (lemma) dipath.v:44

dipath_rcons (lemma) dipath.v:47

cat_dipath (lemma) dipath.v:56

cat_dipath_cont (lemma) dipath.v:67 — Continuation form: append a path that starts AT the endpoint.

dipath_cons (lemma) dipath.v:83

dipath_drop (lemma) dipath.v:89

last_take (lemma) dipath.v:105

last_drop (lemma) dipath.v:108

take_path_last (lemma) dipath.v:117 — Endpoint of a prefix, on the (x :: s) indexing.

dipath_take (lemma) dipath.v:124

dipath_arc_nth (lemma) dipath.v:132 — The i-th vertex of the path (x, s), i ranging over 0..size s.

dipath_size (lemma) dipath.v:136

has_dipath (definition) dipath.v:144 — ** The longest-path length ℓ(D)

has_dipathP (lemma) dipath.v:146

ell (definition) dipath.v:155

ell_max (lemma) dipath.v:157

ellP (lemma) dipath.v:166

maxpath_endclosure (lemma) dipath.v:178 — ** K-A: out-neighbours of the endpoint of a maximum path lie on it

endmax (definition) dipath.v:188 — ** End-maximal paths and K-Ext

endmax_closure (lemma) dipath.v:191

endmax_dipath (lemma) dipath.v:195

endmax_ex (lemma) dipath.v:199 — K-Ext: every vertex starts some end-maximal path.

dicycle (definition) dipath.v:223 — ** Directed cycles as seqs

dicycle_rot (lemma) dipath.v:225

dicycle_next (lemma) dipath.v:229 — The closing/cyclic arcs: every cycle vertex beats its **next**.

dicycle_prev (lemma) dipath.v:232

dicycle_unroll (lemma) dipath.v:237 — Unrolling: a cycle yields a path from any of its vertices through all of it, ending at the predecessor (with the closing arc back).

dicycle_suffix (lemma) dipath.v:262 — Cycle from a path suffix plus a back-arc from the endpoint.

prev_head (lemma) dipath.v:290 — **prev** of the head of a uniq seq is its last element (eqType-level helper for cycle predecessor reasoning).

dipath_map (lemma) dipath.v:304 — ** M-ℓ: ℓ is monotone under injective arc-preserving embeddings

ell_embed (lemma) dipath.v:314

ell_outsel (lemma) dipath.v:330 — The two instances used by the reduction R: arc-sub-relations and induced subgraphs.

ell_induced (lemma) dipath.v:337

theories/core/tournament.v

asymm (fact) tournament.v:47

arcxx (lemma) tournament.v:65

arc_neq (lemma) tournament.v:68

arc_asym (lemma) tournament.v:71

arcNarc (lemma) tournament.v:78 — On distinct vertices, the two arc directions are complementary.

arc_or (lemma) tournament.v:83

arc_xorE (lemma) tournament.v:87 — Since both directions never hold together, xor and or coincide.

N_out (definition) tournament.v:95 — ** Neighbourhoods

N_in (definition) tournament.v:96

N_outE (lemma) tournament.v:98

N_inE (lemma) tournament.v:99

N_out_in_partition (lemma) tournament.v:101

transb (definition) tournament.v:112 — ** Decidable transitivity and the 3-cycle dichotomy

transbP (lemma) tournament.v:115

ntransbP (lemma) tournament.v:124 — A tournament fails transitivity exactly when it has a directed triangle.

ntransb_card (lemma) tournament.v:139 — An intransitive tournament has at least 3 vertices.

card_le2_transb (lemma) tournament.v:148

sub_arc_total (fact) tournament.v:160 — The subtype already carries the `Oriented` structure (core/oriented.v); only totality is added here.

sub_tournament (definition) tournament.v:171 — Object-level versions for use in statements.

del_tournament (definition) tournament.v:174

C3 (definition) tournament.v:183

C3_irrefl (fact) tournament.v:188

C3_total (fact) tournament.v:191

arcC3E (lemma) tournament.v:196

card_C3 (lemma) tournament.v:199

C3_Ntransb (lemma) tournament.v:203 — C3 is the directed triangle: it is *not* transitive.

TT (definition) tournament.v:215

TT_irrefl (fact) tournament.v:220

TT_total (fact) tournament.v:223

arcTTE (lemma) tournament.v:232

card_TT (lemma) tournament.v:235

TT_transb (lemma) tournament.v:238

theories/core/order.v

ltp (definition) order.v:39

ltp_irrefl (lemma) order.v:41

ltp_trans (lemma) order.v:44

ltp_total (lemma) order.v:47

ltp_asym (lemma) order.v:53

le_r_total (fact) order.v:66

le_r_trans (fact) order.v:73

mem_s (fact) order.v:82

size_s (fact) order.v:83

sorted_s (fact) order.v:84

rank_lt (fact) order.v:85

rfun (definition) order.v:87

rfun_inj (fact) order.v:89

realize (definition) order.v:96

ltp_realizeE (lemma) order.v:98

ltp_pullback (lemma) order.v:119 — Transport an order along an injective map: the pulled-back order on `U` is again realized by a permutation.

backedge_rel (definition) order.v:136

backedge_sym (fact) order.v:139

backedge_irrefl (fact) order.v:142

backedge (definition) order.v:145

backedgeE (lemma) order.v:147

backedge_neq (lemma) order.v:153 — An edge of the backedge graph is in particular an arc-disagreement, so its endpoints are distinct.

backedge_arc_forward (lemma) order.v:163 — If the order linearizes the arcs (no backward arc at all), the backedge graph is edgeless.

theories/invariants/omegabar.v

omegab_at (definition) omegabar.v:29 — ** Definition

omegabar (definition) omegabar.v:32

omegabar_min (lemma) omegabar.v:42

omegabar_witness (lemma) omegabar.v:45

omegabar_gt0 (lemma) omegabar.v:48

omegab_at_le1 (lemma) omegabar.v:56 — ** $\bar{\omega} = 1$ characterizes transitivity

omegabar_le1 (lemma) omegabar.v:63

★ **omegabar_transb** (lemma) omegabar.v:74

omegabar_embed (lemma) omegabar.v:108 — ** Monotonicity under arc-preserving embeddings Any injective map that reflects arcs both ways pushes $\bar{\omega}$ up: the witness order on the big tournament pulls back along the embedding (`ltp_pullback`) and the backedge graphs correspond edge-for-edge.

omegabar_dgiso (lemma) omegabar.v:126 — $\bar{\omega}$ is invariant under digraph isomorphism.

omegabar_sub (lemma) omegabar.v:137 — Sub-tournaments and vertex deletion.

omegabar_del (lemma) omegabar.v:143

omegabar_nil (lemma) omegabar.v:148 — $\bar{\omega}$ on degenerate and tiny tournaments.

omegabar_card_le2 (lemma) omegabar.v:156

omegabar_TT (lemma) omegabar.v:164 — ** The two reference values

omegabar_C3 (lemma) omegabar.v:169

theories/invariants/strong.v

strongb (definition) strong.v:37

strongP (lemma) strong.v:39

arc_closed (definition) strong.v:44 — ** S1: forward-closed sets and the sink trick

closed_connect (lemma) strong.v:46

rset (definition) strong.v:54

rset_id (lemma) strong.v:56

rset_closed (lemma) strong.v:59

rset_trans (lemma) strong.v:65

sink_exists (lemma) strong.v:73 — The sink trick: a reachable set of minimum size is closed and strongly connected inside.

outdeg_closed (lemma) strong.v:91 — Closed sets keep out-degrees in the induced subgraph.

connect_induced_closed (lemma) strong.v:100 — Connectivity relativizes to closed sets.

connect_cross (lemma) strong.v:116 — ** S2: cut crossing

maxpath_no_loopback (lemma) strong.v:129 — ** K-a1: no loopback arc at a maximum path's endpoint

disjoint_cycles_path (lemma) strong.v:161 — ** K-10: disjoint cycles force long paths

strongb_induced_closed (lemma) strong.v:255 — Strongness of the induced subgraph on a closed, internally connected set.

reduction (theorem) strong.v:267 — ** R: the composed out-regular reduction (dossier item R)

theories/invariants/critical.v

kcritical (definition) critical.v:21

kcriticalP (lemma) critical.v:24

card_del (lemma) critical.v:34 — Deleting a vertex removes exactly one vertex.

★ **kcritical_proper_sub** (lemma) critical.v:46 — ** Proper subtournaments of critical tournaments In a k-$\bar{\omega}$-critical tournament every *proper* subtournament has $\bar{\omega} \leq k - 1$: it omits some vertex, hence embeds into that vertex's deletion (the Question-5.9-failure mechanism, stated once; G1 of docs/k34_dossier.md).

C3_kcritical2 (lemma) critical.v:64 — ** C3 is 2-$\bar{\omega}$-critical

kcritical2_card3 (lemma) critical.v:95 — A 2-critical tournament has no 4th vertex: deleting it would leave the triangle, contradicting $\bar{\omega}(T - x) = 1$.

★ **kcritical2_uniq** (theorem) critical.v:124 — The explicit isomorphism C3 ≅ T sending 0,1,2 to the triangle.

theories/invariants/domination.v

dominatesb (definition) domination.v:28

dominatesbP (lemma) domination.v:31

dominatesb_setT (lemma) domination.v:43

domnum (definition) domination.v:46

domnum_min (lemma) domination.v:48

domnum_witness (lemma) domination.v:54

greedy_clique_dom (lemma) domination.v:63 — The greedy chain: inside any A there is a subset that dominates A and is a clique of the backedge graph of p (the $\prec$-minimum first, every later pick beating — and sitting $\prec$-after — all earlier ones).

★ **domnum_le_omegabar** (theorem) domination.v:114 — Paper Property 3.2 — the lower-bound engine for $\bar{\omega}$.

theories/constructions/automorphism.v

autb (definition) automorphism.v:25 — p is an automorphism: it preserves arcs (hence also non-arcs).

autbP (lemma) automorphism.v:28

dgaut (definition) automorphism.v:35

dgautE (lemma) automorphism.v:37

dgaut1 (lemma) automorphism.v:40

dgautM (lemma) automorphism.v:43

dgaut_group_set (lemma) automorphism.v:49

vertex_transitiveb (definition) automorphism.v:55 — Vertex-transitivity.

vertex_transitivebP (lemma) automorphism.v:59

theories/constructions/product.v

lexprod (definition) product.v:25

lexprod_arcE (lemma) product.v:31

card_lexprod (lemma) product.v:35

lexprod_irrefl (fact) product.v:45

lexprod_total (fact) product.v:50

lexprod_tournament (definition) product.v:65 — Object-level version for statements.

lexprod_vertex_transitive (lemma) product.v:73

theories/constructions/cayley.v

cayley (definition) cayley.v:30 — The carrier is the group; the (otherwise unused) connection set A is a real parameter of the type alias so that distinct connection sets carry distinct canonical instances.

cayley_arcE (lemma) cayley.v:39

cayley_irreflP (lemma) cayley.v:44 — ** Tournament characterization

cayley_totalP (lemma) cayley.v:51

translation (definition) cayley.v:66 — ** Translations and vertex-transitivity

translationE (lemma) cayley.v:68

translation_aut (lemma) cayley.v:71

cayley_vertex_transitive (lemma) cayley.v:78

theories/constructions/circulant.v

Zp_mulgE (lemma) circulant.v:36

Zp_invgE (lemma) circulant.v:39

Zp_1gE (lemma) circulant.v:42

val_Zp_opp (lemma) circulant.v:45

circulant_arcE (lemma) circulant.v:50 — Circulant arc characterization: `x -> y` iff the difference is a connection element.

ACset (definition) circulant.v:63

ACset_cond (lemma) circulant.v:72 — The connection set hits each pair {z, –z} (z ≠ 0) exactly once — this is the residue-arithmetic heart of "AC is a tournament".

AC_irrefl (fact) circulant.v:109 — AC is a tournament.

AC_total (fact) circulant.v:115

AC (definition) circulant.v:124

AC_arcE (lemma) circulant.v:126

★ **card_AC** (lemma) circulant.v:129

★ **AC_vertex_transitive** (lemma) circulant.v:134 — AC is vertex-transitive (as any Cayley digraph: translations act transitively) — the M2 exit fact feeding the M3 criticality reduction.

C3_vertex_transitive (lemma) circulant.v:146 — ** C is vertex-transitive C3's arc relation `v == u + 1` over 'Z_3 is translation-invariant, so the translations x x + t are automorphisms and act transitively (the Cayley-translation argument in miniature; G3 of docs/k34_dossier.md — feeds the k = 4 criticality reduction).

theories/invariants_advanced/transitive.v

del_aut_iso (lemma) transitive.v:27 — An automorphism sending `u` to `g u` restricts to an isomorphism between the vertex-deleted sub-tournaments, so $\bar{\omega}$ agrees on them.

★ **omegabar_del_vt** (theorem) transitive.v:47 — The marquee theorem: on a vertex-transitive tournament, $\bar{\omega}$ after deletion is the same whichever vertex is deleted.

★ **vt_kcritical** (theorem) transitive.v:57 — Hence k-$\bar{\omega}$-criticality reduces to a single deletion.

AC_del_uniform (lemma) transitive.v:70 — Demo: in the circulant tournament AC all single-vertex deletions have the same $\bar{\omega}$ — its criticality will need only the deletion at 0 (M4).

theories/invariants_advanced/substitution.v

hmin (definition) substitution.v:46 — The $\prec$-minimum of a block, and its position.

posmin (definition) substitution.v:47

posmin_le (lemma) substitution.v:49

pos_inj (lemma) substitution.v:52

posmin_neq (lemma) substitution.v:55

rS_irr (fact) substitution.v:63

rS_trans (fact) substitution.v:64

rS_total (fact) substitution.v:65

rH_irr (fact) substitution.v:73

rH_trans (fact) substitution.v:75

rH_total (fact) substitution.v:77

cross_edgeE (lemma) substitution.v:88 — Edge correspondences between `backedge p` and the auxiliary graphs.

block_edgeE (lemma) substitution.v:97

omegab_at_lexprod_ge (lemma) substitution.v:103 — The bound, for this arbitrary order.

★ **omegabar_lexprod_ge** (theorem) substitution.v:168 — The substitution lower bound: $\bar{\omega}(\mathrm{SH}) \geq \bar{\omega}(\mathrm{S}) + \bar{\omega}(\mathrm{H}) - 1$.

theories/applications/k5/acn_arc_facts.v

val_addE (lemma) acn_arc_facts.v:40 — ** Values of sums, opposites, differences in 'Z_n

val_oppE (lemma) acn_arc_facts.v:43

val_sub_le (lemma) acn_arc_facts.v:46

val_sub_gt (lemma) acn_arc_facts.v:60

AC_mem_val (lemma) acn_arc_facts.v:71 — ** Connection-set membership by value

AC_arc_lt (lemma) acn_arc_facts.v:77 — ** Gap characterizations under the value order

AC_arc_gt (lemma) acn_arc_facts.v:84

AC_band_arc (lemma) acn_arc_facts.v:104 — Within a band (gap $<$ m), arcs follow the value order — the working form of paper (iii).

AC_mem_Hi (lemma) acn_arc_facts.v:110 — ** The band facts — paper (i) and (ii)

AC_mem_Hi_opp (lemma) acn_arc_facts.v:117

AC_mem_Lo (lemma) acn_arc_facts.v:138

AC_mem_Lo_opp (lemma) acn_arc_facts.v:147

theories/applications/k5/acn_base.v

bucket_bound (lemma) acn_base.v:36 — ** A pigeonhole principle: sparse values in a bounded range

rid_irr (fact) acn_base.v:70

rid_trans (fact) acn_base.v:71

rid_total (fact) acn_base.v:72

backedge_qid_dist (lemma) acn_base.v:79 — Every backedge of the value order spans a gap of at least m.

omegab_at_qid_le3 (lemma) acn_base.v:94 — ** Upper bound: $\bar{\omega}(\mathrm{AC}) \leq 3$

omegabar_AC_le3 (lemma) acn_base.v:106

r0_irr (fact) acn_base.v:115

r0_trans (fact) acn_base.v:116

r0_total (fact) acn_base.v:117

omegab_at_q0_le2 (lemma) acn_base.v:123

omegabar_ACdel_le2 (lemma) acn_base.v:147

mem0A (fact) acn_base.v:155

oppA_disjoint (fact) acn_base.v:158

cardA (fact) acn_base.v:166

cardN0 (fact) acn_base.v:192

mem_shift (fact) acn_base.v:198

shift_comp (fact) acn_base.v:204

card_shift (fact) acn_base.v:207

shiftI (fact) acn_base.v:210

shift0 (fact) acn_base.v:214

autocorr_sym (fact) acn_base.v:217

val1 (fact) acn_base.v:224

val_zeta (fact) acn_base.v:229

N0_val_in (fact) acn_base.v:235

`NO_0` (fact) acn_base.v:244

`autocorr_lo` (lemma) acn_base.v:249 — The autocorrelation lemma, low half: for $0 <$ val t $\leq$ m, the sets N and N + t share at least two elements (m $\geq$ 3).

`autocorr2` (lemma) acn_base.v:310 — The autocorrelation lemma: every nonzero shift overlaps N in $\geq$ 2 points.

`domnum_AC_ge3` (lemma) acn_base.v:326 — dom(AC) $\geq$ 3: one or two translates of N cannot cover 'Z_n.

★ `omegabar_AC` (theorem) acn_base.v:374 — ** $\bar{\omega}$(AC) = 3

`ACdel_Ntransb` (lemma) acn_base.v:386

`omegabar_ACdel0` (lemma) acn_base.v:412 — ** $\bar{\omega}$(AC – v) = 2 and 3-criticality

★ `omegabar_AC_del` (theorem) acn_base.v:423

★ `AC_kcritical3` (theorem) acn_base.v:431

theories/applications/acn_bands.v

`radix_ltA` (lemma) acn_bands.v:34 — ** Radix lemmas for lexicographic keys

`radix_lt_inv` (lemma) acn_bands.v:43

`radix_eq_inv` (lemma) acn_bands.v:53

`band` (definition) acn_bands.v:72 — ** Bands

`band1P` (lemma) acn_bands.v:75

`band2P` (lemma) acn_bands.v:81

`band3P` (lemma) acn_bands.v:87

`band_ge1` (lemma) acn_bands.v:92

`band_le3` (lemma) acn_bands.v:95

`band12P` (lemma) acn_bands.v:99 — Two band dichotomies.

`AC_wrapF` (lemma) acn_bands.v:114 — A backward residue over a gap below m is never a connection element.

`band_gap` (lemma) acn_bands.v:128 — Within a (nonzero) band, value gaps stay below m.

theories/applications/k5/k5_lower.v

`ACm_pos` (fact) k5_lower.v:40

`ACdel_pos` (fact) k5_lower.v:43

`omegabar_T_ge5` (theorem) k5_lower.v:48 — ** $\bar{\omega}$(T) $\geq$ 5

`emb_mem` (fact) k5_lower.v:62

`f_inj` (fact) k5_lower.v:71

`f_arc` (fact) k5_lower.v:77

`omegabar_Tdel_ge4` (theorem) k5_lower.v:83

theories/applications/k5/in_neighbourhood.v

`gval_bounds` (lemma) in_neighbourhood.v:32 — Connection-set elements have value in `1, m+1` \ {m}.

`H17_xrange` (lemma) in_neighbourhood.v:42 — An in-neighbour of an Hi vertex sits in the window `vx-m-1, vx-1`.

`H17_dichot` (lemma) in_neighbourhood.v:66 — An in-neighbour of a Lo vertex is either below it or $\geq$ m positions up.

H17_side (lemma) in_neighbourhood.v:83 — H17: the common in-neighbourhood lies in the band determined by x – y.

H17_no_mixed (lemma) in_neighbourhood.v:107 — The working corollary: an Hi vertex and a Lo vertex can never both beat both an Hi vertex x and a Lo vertex y.

theories/applications/k5/cells.v

cidx (definition) cells.v:45 — ** Cells and the key

cidx_decode (lemma) cells.v:47

cidx_le8 (lemma) cells.v:56

key (definition) cells.v:65

key_inj (lemma) cells.v:67

cidx_mono (lemma) cells.v:75

key_samecell (lemma) cells.v:81

rk (definition) cells.v:94 — ** The realized key order

rk_irr (fact) cells.v:95

rk_trans (fact) cells.v:96

rk_total (fact) cells.v:97

qk (definition) cells.v:103

qkE (lemma) cells.v:105

beat_key (lemma) cells.v:115

beat_blocks (lemma) cells.v:126 — Between different blocks, the beaten block-difference is a connection element; inside a block the inner difference is.

beat_inner (lemma) cells.v:136

cidx_inj_clique (lemma) cells.v:148 — ** The cell Lemma: at most one clique vertex per cell

occ (definition) cells.v:190 — ** Occupancy and the cardinality bridge

occP (lemma) cells.v:192

card_clique_cidx (lemma) cells.v:199

theories/applications/k5/obstructions.v

obstrb (definition) obstructions.v:30 — The 20-disjunct obstruction pattern over the 8 cell-occupancy bits.

nat_n_split (lemma) obstructions.v:46

mem_sub_Lo_Hi1 (lemma) obstructions.v:55 — ** Membership computations for cross-band differences

mem_sub_Hi1_Lo (lemma) obstructions.v:72

mem_sub_Hi_m (lemma) obstructions.v:90

mem_sub_m_Hi (lemma) obstructions.v:107

opp_arcs_contra (lemma) obstructions.v:133 — Opposite residues cannot both be connection elements.

repA_band (lemma) obstructions.v:149

rep_Hi (lemma) obstructions.v:155

rep_Lo (lemma) obstructions.v:162

rep_Z (lemma) obstructions.v:169

beatc (lemma) obstructions.v:177

bands_neq (lemma) obstructions.v:187

triple_A (lemma) obstructions.v:200 — ** The seven core obstruction shapes *) (* (X, zero, X) triples — paper sets 1–8

triple_B (lemma) obstructions.v:261 — ** The seven core obstruction shapes *) (* (X, zero, X) triples — paper sets 1–8 *) Lemma triple_A (u v w : backedge (qk m')) (i j k : nat) : u \in K -> v \in K -> w \in K -> cidx (u : T5) = i -> cidx (v : T5) = j -> cidx (w : T5) = k -> (i < j)%N -> (j < k)%N -> (j %% 3)%N = 2%N -> ((i %% 3 == 0) && (k %% 3 == 0))%N || ((i %% 3 == 1) && (k %% 3 == 1))%N -> False.

quad_A (lemma) obstructions.v:286 — ** The seven core obstruction shapes *) (* (X, zero, X) triples — paper sets 1–8 *) Lemma triple_A (u v w : backedge (qk m')) (i j k : nat) : u \in K -> v \in K -> w \in K -> cidx (u : T5) = i -> cidx (v : T5) = j -> cidx (w : T5) = k -> (i < j)%N -> (j < k)%N -> (j %% 3)%N = 2%N -> ((i %% 3 == 0) && (k %% 3 == 0))%N || ((i %% 3 == 1) && (k %% 3 == 1))%N -> False.

quad_B (lemma) obstructions.v:320 — ** The seven core obstruction shapes *) (* (X, zero, X) triples — paper sets 1–8 *) Lemma triple_A (u v w : backedge (qk m')) (i j k : nat) : u \in K -> v \in K -> w \in K -> cidx (u : T5) = i -> cidx (v : T5) = j -> cidx (w : T5) = k -> (i < j)%N -> (j < k)%N -> (j %% 3)%N = 2%N -> ((i %% 3 == 0) && (k %% 3 == 0))%N || ((i %% 3 == 1) && (k %% 3 == 1))%N -> False.

quad_C (lemma) obstructions.v:361 — ** The seven core obstruction shapes *) (* (X, zero, X) triples — paper sets 1–8 *) Lemma triple_A (u v w : backedge (qk m')) (i j k : nat) : u \in K -> v \in K -> w \in K -> cidx (u : T5) = i -> cidx (v : T5) = j -> cidx (w : T5) = k -> (i < j)%N -> (j < k)%N -> (j %% 3)%N = 2%N -> ((i %% 3 == 0) && (k %% 3 == 0))%N || ((i %% 3 == 1) && (k %% 3 == 1))%N -> False.

quad_D (lemma) obstructions.v:412 — ** The seven core obstruction shapes *) (* (X, zero, X) triples — paper sets 1–8 *) Lemma triple_A (u v w : backedge (qk m')) (i j k : nat) : u \in K -> v \in K -> w \in K -> cidx (u : T5) = i -> cidx (v : T5) = j -> cidx (w : T5) = k -> (i < j)%N -> (j < k)%N -> (j %% 3)%N = 2%N -> ((i %% 3 == 0) && (k %% 3 == 0))%N || ((i %% 3 == 1) && (k %% 3 == 1))%N -> False.

square (lemma) obstructions.v:450 — ** The seven core obstruction shapes *) (* (X, zero, X) triples — paper sets 1–8 *) Lemma triple_A (u v w : backedge (qk m')) (i j k : nat) : u \in K -> v \in K -> w \in K -> cidx (u : T5) = i -> cidx (v : T5) = j -> cidx (w : T5) = k -> (i < j)%N -> (j < k)%N -> (j %% 3)%N = 2%N -> ((i %% 3 == 0) && (k %% 3 == 0))%N || ((i %% 3 == 1) && (k %% 3 == 1))%N -> False.

occ_and3F (lemma) obstructions.v:495 — ** Packaging: the occupancy bits never match an obstruction

occ_and4F (lemma) obstructions.v:505

no_obstruction (lemma) obstructions.v:517

theories/applications/k5/coverage.v

coverage5 (lemma) coverage.v:19

theories/applications/k5/k5_upper.v

omegab_at_qk_le5 (lemma) k5_upper.v:45 — ** $\bar{\omega}(\mathrm{T}) \leq 5$

omegabar_T_le5 (theorem) k5_upper.v:56

rd_irr (fact) k5_upper.v:68

rd_trans (fact) k5_upper.v:71

rd_total (fact) k5_upper.v:74

omegab_at_qd_le4 (lemma) k5_upper.v:83

omegabar_Tdel_le4 (theorem) k5_upper.v:109

theories/applications/k5/main.v

T5_vertex_transitive (lemma) main.v:46 — The product of the vertex-transitive AC with itself is vertex-transitive.

★ **omegabar_T5** (theorem) main.v:54 — ** $\bar{\omega}(\mathrm{T}) = 5$

★ **omegabar_T5_del** (theorem) main.v:62 — ** $\bar{\omega}(\mathrm{T} - \mathrm{v}) = 4$ for every vertex v

★ **T5_kcritical5** (theorem) main.v:71 — ** T is 5-$\bar{\omega}$-critical

★ **card_T5** (lemma) main.v:79 — ** T has order n

proper_sub_omegabar_le4 (theorem) main.v:84 — ** Every proper subtournament has $\bar{\omega} \leq 4$

★ **conjecture_5_10_at_k5** (theorem) main.v:108 — ** Conjecture 5.10 at k = 5: an infinite family For every N there is a 5-$\bar{\omega}$-critical tournament on more than N vertices.

★ **question_5_9_fails_at_k5** (theorem) main.v:127 — ** Question 5.9 fails at k = 5 No bound $\ell(5)$ exists: for every L there is a 5-$\bar{\omega}$-critical tournament on more than L vertices whose ONLY subtournament with $\bar{\omega} \geq 5$ is the whole tournament.

theories/applications/ck3/lemma7.v

card_ord_ltn (lemma) lemma7.v:26 — ** Ordinal segment cardinality (counting helper)

card_ord_seg (lemma) lemma7.v:37

ckC (definition) lemma7.v:58 — The endpoint cycle as a seq: the suffix of the path from index a.

ckCset (definition) lemma7.v:61 — Its vertex set.

ckB (definition) lemma7.v:64 — B = the out-neighbours of v_{a-1} on the cycle.

ckS (definition) lemma7.v:68 — S = B, the C-predecessors of B.

kernel_full (theorem) lemma7.v:606 — *** The package

kernel_S (theorem) lemma7.v:631 — The bare Lemma 7 witness.

★ **lemma7** (theorem) lemma7.v:646 — ** Lemma 7 (Cheng–Keevash), in full generality

★ **ck_theorem4_oriented** (theorem) lemma7.v:684 — ** Theorem 4 (oriented case) and the $\delta = 2$ case of Conjecture 1

★ **ck_conj1_delta2** (corollary) lemma7.v:698

theories/applications/ck3/ck3_main.v

★ **no_short_strong3** (lemma) ck3_main.v:28 — ** The endgame: no strong 3-outregular oriented digraph has $\ell \leq 5$

★ **ck_conj1_delta3** (theorem) ck3_main.v:147 — ** The main theorem: Conjecture 1 at $\delta = 3$

★ **ck_conj1_delta3_path** (corollary) ck3_main.v:158 — Unfolded form: an explicit directed simple path with 6 arcs.

ck_conj1_at_3 (corollary) ck3_main.v:170 — Conjecture-1-shaped alias.

theories/applications/k4/k4_lower.v

ACm_pos (fact) k4_lower.v:44

C3_pos (fact) k4_lower.v:47

omegabar_T4_ge4 (theorem) k4_lower.v:52 — ** $\bar{\omega}(\mathrm{T4}) \geq 4$ (L1)

emb_mem (fact) k4_lower.v:64

f_inj (fact) k4_lower.v:69

f_arc (fact) k4_lower.v:72

omegabar_T4del_ge3 (theorem) k4_lower.v:77

theories/applications/k4/k4_value.v

dband (definition) k4_value.v:34 — ** The inner band on C (h = 0 heavy, h ≠ 0 light)

dband2P (lemma) k4_value.v:36

dband1P (lemma) k4_value.v:39

dband_ge1 (lemma) k4_value.v:42

dband_le2 (lemma) k4_value.v:45

sidx (definition) k4_value.v:60 — ** Sub-bands, classes, and the merged key

sidx_decode (lemma) k4_value.v:62

sidx_le5 (lemma) k4_value.v:71

kappa (definition) k4_value.v:80

kappa_sidx (lemma) k4_value.v:82

kv (definition) k4_value.v:88

kv_inj (lemma) k4_value.v:90

kv_kappa_mono (lemma) k4_value.v:98

kv_lt_t (lemma) k4_value.v:105

kv_samekappa (lemma) k4_value.v:111

rv (definition) k4_value.v:124 — ** The realized merged order

rv_irr (fact) k4_value.v:125

rv_trans (fact) k4_value.v:126

rv_total (fact) k4_value.v:127

qv (definition) k4_value.v:133

qvE (lemma) k4_value.v:135

n_sub_ge_Sm (lemma) k4_value.v:140 — ** Residue arithmetic helpers (pure nat, m-relative)

n_sub_gt_Sm (lemma) k4_value.v:143

n_sub_eq_Sm (lemma) k4_value.v:146

sub_eq_m (lemma) k4_value.v:152

mem_m_F (lemma) k4_value.v:159 — Connection-set refutations by value.

mem_mid_F (lemma) k4_value.v:164

vals12 (lemma) k4_value.v:169

beat_kv (lemma) k4_value.v:187

beat_blocksV (lemma) k4_value.v:196

beat_innerV (lemma) k4_value.v:206

sidx_inj_clique (lemma) k4_value.v:217 — ** The sub-band Lemma: at most one clique vertex per σ (V1–V4)

occv (definition) k4_value.v:263 — ** Occupancy

occvP (lemma) k4_value.v:265

card_clique_sidx (lemma) k4_value.v:272

occ24_block (lemma) k4_value.v:280 — ** V5: both $\kappa = 4$ sub-bands occupied forces {(m,0),(0,i)}

occ24_5F (lemma) k4_value.v:312

occ24_1F (lemma) k4_value.v:340

occ5310_F (lemma) k4_value.v:392 — ** V6: the four-class chain (0,0) / high / low / κ2 dies

clique_card_le4 (lemma) k4_value.v:485

omegab_at_qv_le4 (lemma) k4_value.v:500 — ** The value bound

omegabar_T4_le4 (theorem) k4_value.v:508

★ **omegabar_T4** (theorem) k4_value.v:511

theories/applications/k4/k4_del.v

kd (definition) k4_del.v:55 — ** The d_then_c key

kd_inj (lemma) k4_del.v:57

kd_sidx_mono (lemma) k4_del.v:65

kd_samesidx (lemma) k4_del.v:72

rd_irr (fact) k4_del.v:87

rd_trans (fact) k4_del.v:90

rd_total (fact) k4_del.v:93

qd (definition) k4_del.v:99

qdE (lemma) k4_del.v:101

sidx_del_le4 (lemma) k4_del.v:105 — Survivors never occupy the deleted band $\sigma = 5$ (D0).

beat_kd (lemma) k4_del.v:147

beat_blocksD (lemma) k4_del.v:157

beat_innerD (lemma) k4_del.v:167

didx_inj_clique (lemma) k4_del.v:178 — ** D1: at most one clique vertex per band

occd (definition) k4_del.v:218 — ** Occupancy

occdP (lemma) k4_del.v:220

card_clique_didx (lemma) k4_del.v:227

occ023_F (lemma) k4_del.v:237 — B1 + B3 + B4: B3 pins the B1 block at m+1 and the B4 block at s $\geq$ m+2; then (s,0) cannot beat (m+1,·).

occ124_F (lemma) k4_del.v:299 — B2 + B3 + B5: B3 forces s' = m (D3f), pinning the B2 block at m; the B3-beats-B2 backedge then needs m $\in$ g.

occ0134_F (lemma) k4_del.v:356 — B1 + B2 + B4 + B5: the D3e core.

clique_card_le3 (lemma) k4_del.v:484

omegab_at_qd_le3 (lemma) k4_del.v:500 — ** The deletion bound

omegabar_T4del_le3 (theorem) k4_del.v:508

omegabar_T4del0 (theorem) k4_del.v:511

theories/applications/k4/k4_main.v

T4_vertex_transitive (lemma) k4_main.v:43 — The product of the vertex-transitive AC with the vertex-transitive C is vertex-transitive.

★ **omegabar_T4_del** (theorem) k4_main.v:51 — ** $\bar{\omega}$(T4 – v) = 3 for every vertex v

★ **T4_kcritical4** (theorem) k4_main.v:59 — ** T4 is 4-$\bar{\omega}$-critical

★ **card_T4** (lemma) k4_main.v:67 — ** T4 has order 3n

★ **conjecture_5_10_at_k4** (theorem) k4_main.v:74 — ** Conjecture 5.10 at k = 4: an infinite family

★ **question_5_9_fails_at_k4** (theorem) k4_main.v:89 — ** Question 5.9 fails at k = 4

theories/applications/unified.v

★ **conjecture_5_10_at_k3** (theorem) unified.v:33 — ** Conjecture 5.10 at k = 3: an infinite family

★ **question_5_9_fails_at_k3** (theorem) unified.v:47 — ** Question 5.9 fails at k = 3

★ **conjecture_5_10_at_345** (theorem) unified.v:66 — ** The unified headline: every k $\in$ {3, 4, 5}