Chip-Firing Games on Graphs and Proof Formalization

Nathan Pflueger

Amherst College

12 March 2026

Budapest Semesters in Mathematics

Interactive demo: `https://npflueger.github.io/chipfiring/`
Lean4 Docs: `https://dhyeymavani.com/chip-firing-with-lean/docs/ChipFiringWithLean/RiemannRochForGraphs.html`

It is great to be back!

Me in Budapest, Fall 2007, excited to find courses with graphs in them!

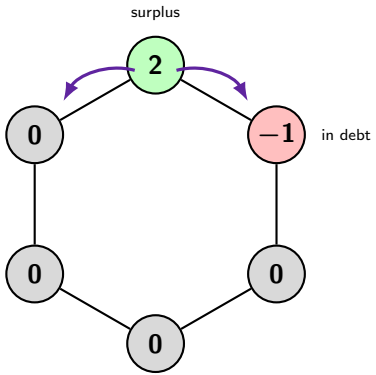
A Curious Economy

Imagine a network of people (a connected graph).

- Each person has an *integer* number of forints (chips).
- Negative numbers are debt.
- Money moves only by **chip-firing**: a person sends 1 forint to *each* neighbor simultaneously.

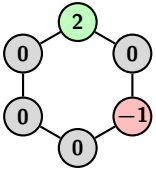
The Basic Game:
Can everyone be brought out of debt?
If so, the configuration is **winnable**.

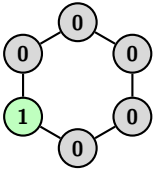
(Later, we'll discuss two more complex games: the *gonality game* and the *rank game*.)



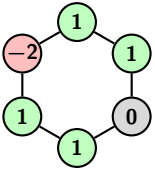
Examples on C_6

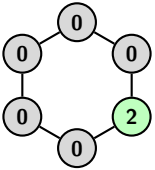
Winnable



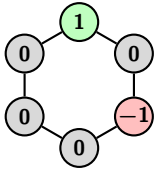


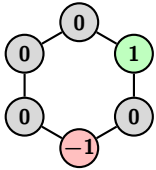
Winnable





Not winnable





Winnability on cycles

If you try examples like these, you might notice this pattern:

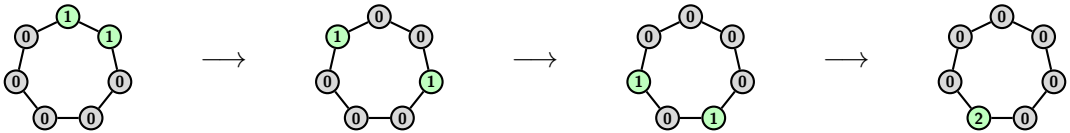
Theorem

On a cycle graph C_n, any configuration with ≥ 1 total forints is winnable.

Proof sketch.

Choose one forint as a "balancer." We claim that any other forint can be moved to any desired vertex if we also move the balancer in a mirrored way. In this way, use positive forints one by one to balance negative forints.

□



Winnability on cycles

Theorem

On a cycle graph C_n, any configuration with ≥ 1 total forints is winnable.

Definition

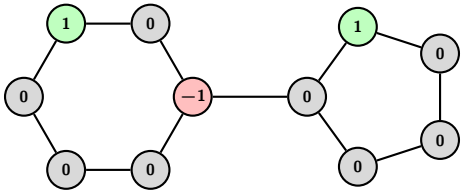
The *degree* of a chip configuration is the total number of forints (chips) it has.

Question

Given a graph G, what is the minimum degree d such that every configuration of degree $\geq d$ is winnable?

A barbell graph

Consider this *barbell graph*, joining C_6 to C_5 with a "bridge" edge.



This degree 1 configuration is not winnable. (Try it!) *The idea:* Winnability is *obstructed* by *two cycles.* But even so...

Theorem

Any configuration D with $\deg D \geq 2$ on a barbell graph is winnable.

Vague intuition: you need "as many chips as cycles" to guarantee winnability.

The genus

Classic graph theory fact: A connected graph G on V vertices has $E \geq V - 1$ edges. Equality $E = V - 1$ holds if and only if G has no cycles (i.e. is a tree).

Definition

The *genus* of a connected graph G is

$$\text{genus}(G) = E - V + 1,$$

where E is the number of edges and V is the number of vertices.

Informally: $\text{genus}(G)$ is the number of (independent) cycles in G.

Examples:

- If G is a tree, then $E = V - 1$, so $\text{genus}(G) = 0$.
- If $G = C_n$ is a cycle, then $E = V$, so $\text{genus}(G) = 1$.
- The barbell graph has two independent cycles, so $\text{genus}(G) = 2$.

A cheerful fact

Definition

The *genus* of a connected graph G is

$$\text{genus}(G) = E - V + 1,$$

where E is the number of edges and V is the number of vertices.

Theorem

If G has genus g, then

$$\deg D \geq g \quad \Longrightarrow \quad D \text{ is winnable.}$$

This is a consequence of the *Riemann–Roch theorem*, which I'll tell you about soon.

Quick terminology interlude

A **chip/forint configuration** on G is also called a **divisor** on G.

Why?? Historical reasons, rooted in abstract algebra...

But I'll start saying "divisor" now, because it's in our code. Just remember:

$$\text{Degee } d \text{ divisor} \quad \longleftrightarrow \quad \text{configuration of } d \text{ forints/chips}$$

Interlude: Lean 4

Lean 4 is a programming language and proof assistant: it lets us write definitions precisely, state theorems formally, and have the computer verify proofs line by line.

In collaboration with Dhyey Mavani (Amherst '25), we recently formalized core results on chip-firing games on finite graphs in Lean 4.

Dhyey Mavani

Interlude: Lean Definitions I

Graphs, divisors, and degree:

```
structure CFGraph (V : Type) [DecidableEq V] [Fintype V] [Nonempty V] where
  (edges : Multiset (V × V))
  (loopless : isLoopless edges)

def isLoopless (edges : Multiset (V × V)) : Prop :=
  ∀ v, (v, v) ∉ edges

def CFDiv (V : Type) := V → ℤ -- ''CFDiv'' means ''Chip Configuration''

def genus (G : CFGraph V) : ℤ :=
  Multiset.card G.edges - Fintype.card V + 1

def deg : CFDiv V → ℤ :=
  λ D => ∑ v, D v
```

Interlude: Lean Definitions II

Connectedness and winnability:

```
def graph_connected (G : CFGraph V) : Prop :=
  ∀ S : Finset V, (∃ (v w : V), v ∈ S ∧ w ∉ S) →
    (∃ v ∈ S, ∃ w ∉ S, num_edges G v w > 0)

def winnable (G : CFGraph V)
    (D : CFDiv V) : Prop :=
  ∃ D' : CFDiv V,
    (∀ v, D' v ≥ 0)
    ∧ linear_equiv G D D'
```

Note: "linearly equivalent" is the general term of "reachable by chip-firing moves."

The definitions match the mathematics closely, but every hidden convention has to be made explicit.

Interlude: A Lean Theorem

One key theorem, in Lean style:

```
lemma winnable_of_deg_ge_genus {G : CFGraph V} (h_conn : graph_connected G) (D : CFDiv V) :
     deg D ≥ genus G → winnable G D
```

This is the formalization of:

$$\deg(D) \geq g \quad \Longrightarrow \quad D \text{ is winnable.}$$

which simultaneously contains our observations about cycles and barbells.

A taste of Lean4 philosophy

This "theorem" is actually a *function*! Its input is a graph, a divisor, a *proof* that the graph is connected, and a *proof* that $d \geq g$, and its output is a proof that D is winnable.
In this case that function is constructive: there's an algorithm to find the chip-firing moves.

Behind the curtain: an embarrassing roadblock

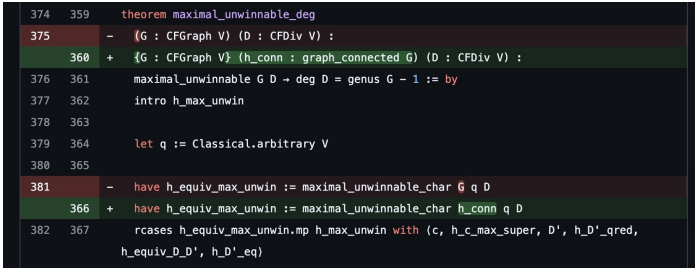
When we were writing our Lean4 code, we initially left some helper lemmas unproved, and proved the main theorems from them.

But when we tried to prove some of those helpers, we realized they weren't true as stated!

We'd forgotten to include a (formalized) *connectedness* hypothesis!

It's important to always be clear on what you need to assume.

Thanks to the magic of Github, this omission is recorded for posterity in `Commit 87d9193`, which remembers the 52 lines of code we needed to change. For example:



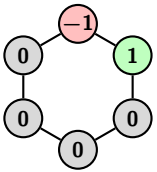
The gonality game

Back to math! Fix a graph G and an integer $k \geq 1$.

1. Alice chooses any divisor D of degree k on G.
2. Bob chooses a vertex and removes one chip there.
3. Alice tries to repair the resulting debt using chip-firing.

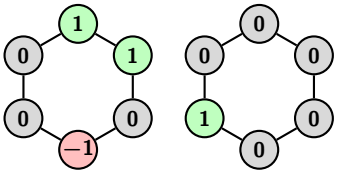
Question

For a given G and k, who has a winning strategy?



$k = 1$

Bob wins



$k = 2$

Alice wins

The gonality of a graph

Definition

The *gonality* of a graph G is the minimum integer k such that Alice has a winning strategy in the gonality game on G.
In other words: there exists D of degree k such that for every vertex v, $D - v$ is winnable.

The gonality is extremely mysterious! Much remains unknown. The most prominent question is to prove or disprove:

Gonality conjecture (Baker '07)

For every connected graph G of genus g, $\text{gonality}(G) \leq \lceil \frac{1}{2}g + 1 \rceil$.

What do we know about gonality?

- **Many specific families' gonalities have been computed or bounded:** For example, Ralph Morrison's **Williams/SMALL REU groups** have proved some *bounds* on gonality for glued grid graphs, queen's graphs, chess graphs, circulant graphs, and others, including some exact computations.
- A central question is about subdivision: how does gonality change as the edges are repeatedly subdivided? We now know that it *can go down*, falsifying an earlier conjecture of Baker.
- **Computation is NP-hard!** So we cannot hope for a fast general algorithm for *exact* answers.
- I highly recommend this paper as a friendly introduction to chip-firing in general, and gonality in particular:
  Chip-firing on the Platonic solids: a primer for studying graph gonality
  by Marchelle Beougher, Kexin Ding, Max Everett, Robin Huang, Chan Lee, Ralph Morrison, and Ben Weber.
  `https://arxiv.org/abs/2407.05158`

A gonality challenge for Lean4!

One of the few general facts about graph gonality is the following theorem of Baker.

Theorem (Baker '07)

For any connected graph G of genus g, there exists some $n \in \mathbb{N}$ such that, if we subdivide each edge of G into n edges, the resulting graph G_n has

$$\text{gonality}(G_n) \leq \left\lceil \frac{1}{2}g + 1 \right\rceil.$$

This *statement* is easy to formalize in Lean4. But Baker's proof is (for now, at least), *wildly beyond reach* of formalization. It relies on deep facts in **algebraic geometry**: deformation theory, arithmetic surfaces, degeneracy loci of vector bundles, intersection theory, and more.

Challenge

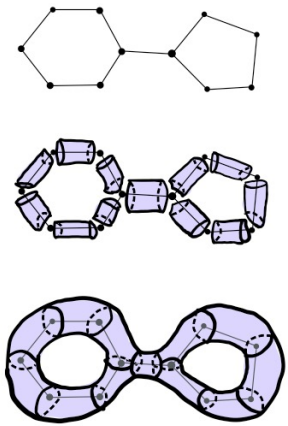
Give a purely combinatorial proof of this theorem, formalized in Lean4.

Algebraic geometry?!

I've been sweeping something important under the rug. The main reason people like me are interested in chip-firing is that it is a *combinatorial model* for algebro–geometric questions about *Riemann surfaces*.

A genus g graph is a model of a Riemann surface with g "handles." The edges represent cylinders.

By now we have a robust interface between algebraic geometry and chip firing, allowing each to prove theorems about the other. This is one of my favorite ways to prove theorems in algebraic geometry! It's also how Baker proved his subdivision gonality theorem.
A jewel in the crown of this analogy is the *Riemann–Roch theorem for graphs*, which I will tell you about now...



The rank game

Idea: we can *refine* the word "winnable" from a yes/no answer to something quantified. *How winnable* is a configuration? **The rank game:** Fix G, a configuration D, and an integer $r \geq 0$.

1. Bob removes a forint in r places.
2. Alice tries to clear all debt with chip-firing moves.

If Alice can win, we say "D has rank at least r."

Definition

The *rank* $r(D)$ of a divisor D on a graph G is the maximum integer r such that for every choice of r vertices, $D - v_1 - \cdots - v_r$ is winnable.

Think: $r(D) =$ "how winnable" D is.

Some facts about the rank

Definition

The *rank* $r(D)$ of a divisor D on a graph G is the maximum integer r such that for every choice of r vertices, $D - v_1 - \cdots - v_r$ is winnable.

1. The *definition of rank* was actually the last piece of the puzzle back in 2007!
2. gonality$(G) \leq k \Leftrightarrow$ there exists D with $\deg D = k$ and $r(D) \geq 1$.
3. D is winnable $\Leftrightarrow r(D) \geq 0$.

So all three of our games can be understood in terms of this one notion: the rank $r(D)$. Therefore this definition is a centerpiece of our Lean4 project.

The Riemann–Roch theorem for graphs

A natural question: Why should I believe that this "rank" has any nice structure? There exists a special "canonical configuration" K_G on G such that

Theorem (Riemann–Roch for graphs; Baker–Norine '07)

For all chip configurations D on G,

$$r(D) = \underbrace{\deg D - g}_{\text{main term}} + \underbrace{r(K_G - D) + 1}_{\text{correction term } (\geq 0)}$$

Bernhard Riemann
1826–1866

Gustav Roch
1839–1866

Matt Baker

Sergey Norine

How to think about Riemann–Roch

Theorem (Riemann–Roch for graphs; Baker–Norine '07)

For all chip configurations D on G,

$$r(D) = \underbrace{\deg D - g}_{\text{main term}} + \underbrace{r(K_G - D) + 1}_{\text{correction term } (\geq 0)}$$

1. Each of the g *cycles* is one *obstruction* to winning the rank game.
2. So the *expected* rank is $\deg D - g$.
3. *Special configurations* can have larger rank. Alice should hope for those in the rank game!

The canonical configuration

Theorem (Riemann–Roch for graphs; Baker–Norine '07)

For all chip configurations D on G,

$$r(D) = \underbrace{\deg D - g}_{\text{main term}} + \underbrace{r(K_G - D) + 1}_{\text{correction term } (\geq 0)}$$

What's this "canonical configuration?" K_G? It's a special chip configuration defined by:

$$K_G = \sum_{v \in V(G)} (\text{valence}(v) - 2) \cdot v.$$

This configuration is important because it is a "center of reflection" for a certain symmetry: the "maximal unwinnable configurations" are symmetric around it. But that's a story for another day (or you can read about it in our Lean4 docs!).

The formalization

Here's a peek at what this looks like in Lean4 syntax.

```
theorem riemann_roch_for_graphs {G : CFGraph V} (h_conn : graph_connected G) (D : CFDiv V)
    :
  rank G D - rank G (canonical_divisor G - D) = deg D - genus G + 1
  := by

  let K := canonical_divisor G

  -- Get key inequality
  have h_ineq := rank_degree_inequality h_conn D

  -- Get reverse inequality by applying to K-D
  have h_ineq_rev := rank_degree_inequality h_conn (K-D)

  -- Get degree of canonical divisor
  have h_deg_K : deg (canonical_divisor G) = 2 * genus G - 2 :=
    degree_of_canonical_divisor G
```

The formalization

Here's the end of the same proof.

```
-- Since rank is an integer and we have bounds, equality must hold
suffices rank G D - rank G (K-D) ≥ deg D - genus G + 1 ∧
         rank G D - rank G (K-D) ≤ deg D - genus G + 1 from
  le_antisymm (this.2) (this.1)

constructor
· -- Lower bound
  linarith
· -- Upper bound
  rw [deg.map_sub, h_deg_K] at h_ineq_rev
  have : canonical_divisor G - (K-D) = D := by
    rw [sub_sub_self]
  rw [this] at h_ineq_rev
  linarith
```

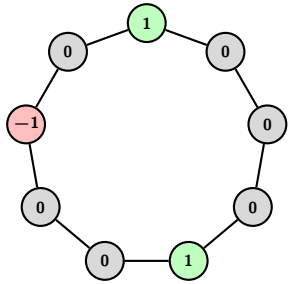
Cycles, revisited

Getting back to the graph we started with, our initial observations are much more quickly inferred using the Riemann–Roch theorem.

If D is a configuration of degree ≥ 1, then Riemann-Roch tells us that

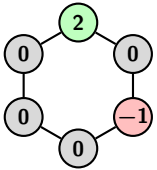
$$\begin{aligned} r(D) &= \deg D - g + (\text{correction term}) \\ &\geq \deg D - g \\ &= \deg D - 1 \\ &\geq 0. \end{aligned}$$

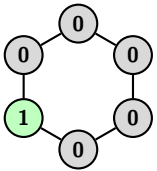
So D is winnable! No fussing with "balancers" needed; just One Big Theorem.



A degree-1 configuration on C_9

Happy chip-firing!






```
theorem riemann_roch_for_graphs
    {G : CFGraph V}
    (h_conn : graph_connected G)
    (D : CFDiv V) :
      rank G D
        - rank G (
    canonical_divisor G - D)
        = deg D - genus G + 1
    := by ...
```