On a Relative of the Ehrenfeucht-Fraïssé Game and Software for Helping with its Analysis

Marco Carmosino (IBM), Ron Fagin (IBM), Neil Immerman (UMass Amherst),
Phokion Kolaitis (IBM and UC Santa Cruz), **Jon Lenchner (IBM)**, and Rik Sengupta (IBM)*


Marco


Ron


Neil


Phokion


Rik

*With earlier contributions by Ken Regan, Nikhil Vyas, and Ryan Williams

Outline

- In the Beginning: A paper of Neil Immerman's from 1981 introducing a Game that captures Quantifier # and his Vision for the Game
- Rules of the Game & Fundamental Capture Theorem
- Why Quantifier # is a Potentially more Interesting Complexity Measure than the more usual Quantifier Rank

- **Quantifier Complexity of Boolean Functions**
- Example and First Results
- A Simplifying Concept in Game Analysis: The Notion of **Parallel Play**

Outline

- More Examples
- Culminating Result on the Quantifier Complexity of Boolean Functions
- The Road Ahead to Realize the Vision from Neil's 1981 Paper

- The **WHO-WINS** problem: how hard is it to decide who wins a generic **Ehrenfeucht-Fraïssé (EF) game**?
- How Hard is it to decide **WHO WINS** a **Multi-Structural (MS) game** (the game that captures Quantifier #)?

- Software to help with the Analysis of the Games

Back in 1981....

JOURNAL OF COMPUTER AND SYSTEM SCIENCES 22, 384–406 (1981)

Number of Quantifiers Is Better Than Number of Tape Cells*

NEIL IMMERMAN

Department of Mathematics,
Tufts University, Medford, Massachusetts 02155, and
Laboratory for Computer Science,
Massachusetts Institute of Technology,
Cambridge, Massachusetts 02139

Received January 5, 1981; revised January 15, 1981

Key Results from the Immerman Paper

1. Introduction of the **Quantifier Number, QN[] measure** – the number of quantifiers needed to express a property via a uniform sequence of FO sentences.

Key Results from the Immerman Paper

1. Introduction of the **Quantifier Number, QN[] measure** – the number of quantifiers needed to express a property via a uniform sequence of FO sentences.

2. Introduction of a combinatorial game that he called the **Separability Game**, that captures quantifier number.

Major focus of this talk!

Key Results from the Immerman Paper

1. Introduction of the **Quantifier Number, QN[] measure** – the number of quantifiers needed to express a property via a uniform sequence of FO sentences.

2. Introduction of a combinatorial game that he called the **Separability Game**, that captures quantifier number.

3. Proof that on *ordered structures*, and for $f(n) \geq \log n$,

Major focus of this talk!

$$\mathbf{NSPACE}[f(n)] \subseteq \mathbf{QN}[(f(n))^2/\log n] \subseteq \mathbf{DSPACE}[(f(n))^2].$$

Where the quantities inside $[\cdot]$ are meant in the $O()$ sense.

Key Results from the Immerman Paper

Plugging $f(n) = \log n$, we get:

$$\mathbf{NSPACE}[f(n)] \subseteq \mathbf{QN}[(f(n))^2/\log n] \subseteq \mathbf{DSPACE}[(f(n))^2]$$

$$\mathrm{NSPACE}[\log n] \subseteq \mathrm{QN}[\log n] \subseteq \mathrm{DSPACE}[\log^2 n]$$

Key Results from the Immerman Paper

Plugging $f(n) = \log n$, we get:

$$\mathbf{NSPACE}[f(n)] \subseteq \mathbf{QN}[(f(n))^2/\log n] \subseteq \mathbf{DSPACE}[(f(n))^2]$$

$$\mathrm{NSPACE}[\log n] \subseteq \mathrm{QN}[\log n] \subseteq \mathrm{DSPACE}[\log^2 n]$$

The left-hand inclusion implies that if we can find a single NP property of ordered structures that requires more than O(log n) quantifiers to express, we will have separated NL from NP!!

Key Results from the Immerman Paper

Plugging $f(n) = \log n$, we get:

$$\mathbf{NSPACE}[f(n)] \subseteq \mathbf{QN}[(f(n))^2/\log n] \subseteq \mathbf{DSPACE}[(f(n))^2]$$

$$\mathrm{NSPACE}[\log n] \subseteq \mathrm{QN}[\log n] \subseteq \mathrm{DSPACE}[\log^2 n]$$

The left-hand inclusion implies that if we can find a single NP property of ordered structures that requires more than O(log n) quantifiers to express, we will have separated NL from NP!!

How to prove such a thing?

That's what led Neil to his game!

However…

Number of Quantifiers Is Better Than Number of Tape Cells*

NEIL IMMERMAN

> Little is known about how to play the separability game. We leave it here as a jumping off point for further research. We urge others to study it, hoping that the separability game may become a viable tool for ascertaining some of the lower bounds which are "well believed" but have so far escaped proof.

Rediscovery of the Game

Back in 2020, a group of us at IBM rediscovered Immerman's Separability Game, but unaware of Neil's paper, we called it the "Multi-Structural Game" – a name we continue to use.

A few years later Neil joined our team.

To the Games….

Before getting to the Multi-Strunctural (MS) game, let’s review the more usual Ehrenfeucht-Fraïssé game.

Ehrenfeucht-Fraïssé (EF) Games

An **Ehrenfeucht-Fraïssé (EF) game** is played on two structures **A** and **B** for a fixed set of r rounds.

There are two players, a **Spoiler** and a **Duplicator**. It is useful to think of the players as each having an infinite set of uniquely colored pebbles. Each player has the same set of such pebbles.

Spoiler plays first. He chooses a structure and picks an element from the structure and places a pebble on it.

Duplicator plays next, picking an element from the other structure and placing a pebble of the same color on it.

They keep alternating turns, placing distinctly colored pebbles for the r rounds.

Duplicator wins if the two sets of pebbled elements, under the chosen element by element correspondence, gives rise to a partial isomorphism. **Spoiler** wins otherwise.

Multi-Structural (MS) Games

Multi-Structural games are played on two **sets, $\mathcal{A}$ and $\mathcal{B}$, of structures**.

Multi-Structural (MS) Games

Multi-Structural games are played on two **sets, $\mathcal{A}$ and $\mathcal{B}$, of structures**.

Spoiler and Duplicator take turns playing from the two **sets, $\mathcal{A}$ and $\mathcal{B}$** for a fixed number **r** of **rounds**. Spoiler plays first and picks a side to play on, $\mathcal{A}$ or $\mathcal{B}$, and places a pebble on an element from each structure in the set.

Multi-Structural (MS) Games

Multi-Structural games are played on two **sets, $\mathcal{A}$ and $\mathcal{B}$, of structures**.

Spoiler and Duplicator take turns playing from the two **sets, $\mathcal{A}$ and $\mathcal{B}$** for a fixed number **r** of **rounds**. Spoiler plays first and picks a side to play on, $\mathcal{A}$ or $\mathcal{B}$, and places a pebble on an element from each structure in the set.

Duplicator plays on the other side. On Duplicator's turn, she can make copies of any structures on her side, and if she likes, place pebbles on the elements of the structures in all possible ways.

Multi-Structural (MS) Games

Multi-Structural games are played on two **sets, $\mathcal{A}$ and $\mathcal{B}$, of structures**.

Spoiler and Duplicator take turns playing from the two **sets, $\mathcal{A}$ and $\mathcal{B}$** for a fixed number **r** of **rounds**. Spoiler plays first and picks a side to play on, $\mathcal{A}$ or $\mathcal{B}$, and places a pebble on an element from each structure in the set.

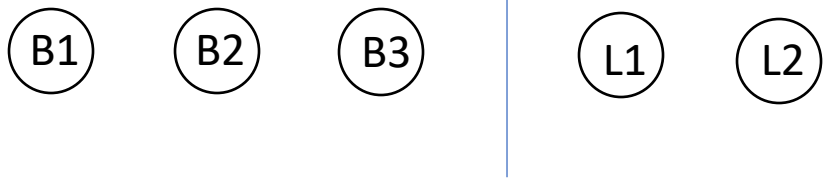
Duplicator plays on the other side. On Duplicator's turn, she can make copies of any structures on her side, and if she likes, place pebbles on the elements of the structures in all possible ways.

Play proceeds in this way for r rounds.

To win, Duplicator just needs to demonstrate a partial isomorphism between the pebbled elements picked from one pair of structures, $A \in \mathcal{A}$ and $B \in \mathcal{B}$. Spoiler wins otherwise.

Illustrative MS game on Linear Orders

- Pictorially we place smaller elements to the left and bigger elements to the right.

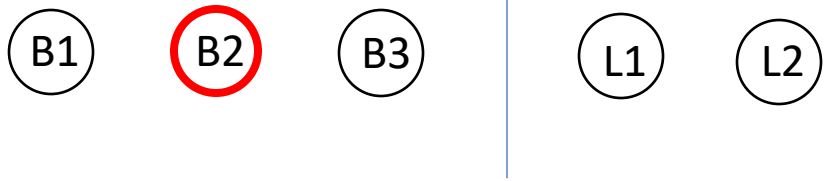


- Two sets of pebbled elements are partially isomorphic if they are in the same left-to-right order.

- Suppose we are playing a 2-round MS game.

Illustrative MS game on Linear Orders

- Pictorially we place smaller elements to the left and bigger elements to the right.

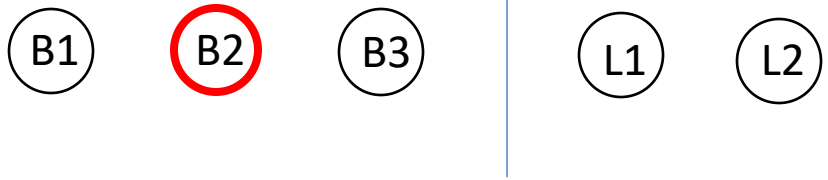


- Two sets of pebbled elements are partially isomorphic if they are in the same left-to-right order.

- Suppose we are playing a 2-round MS game.

Illustrative MS game on Linear Orders

- Pictorially we place smaller elements to the left and bigger elements to the right.



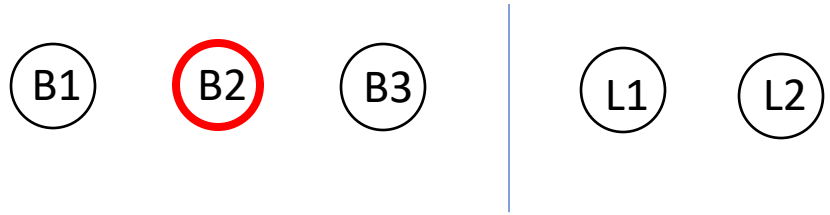
- Two sets of pebbled elements are partially isomorphic if they are in the same left-to-right order.

- Suppose we are playing a 2-round MS game

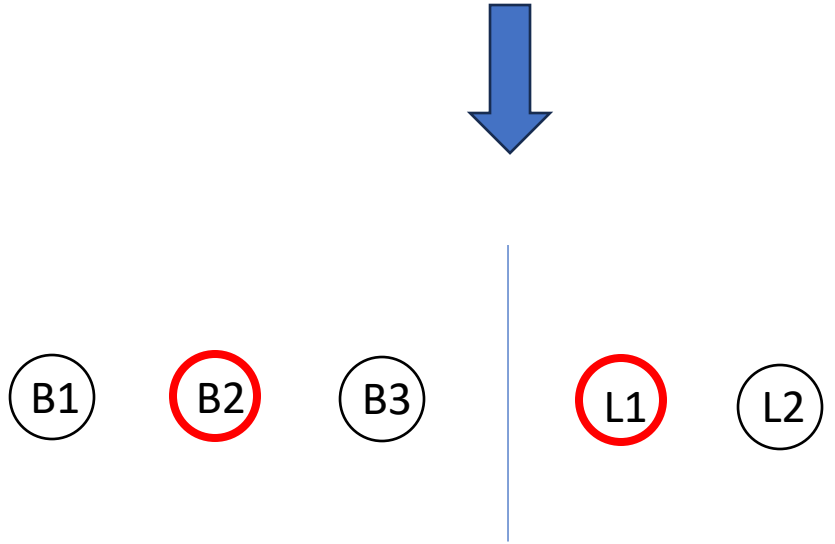
If Duplicator could not duplicate, she would lose!

Illustrative MS game on Linear Orders

- Pictorially we place smaller elements to the left and bigger elements to the right.



- Two sets of pebbled elements are partially isomorphic if they are in the same left-to-right order.

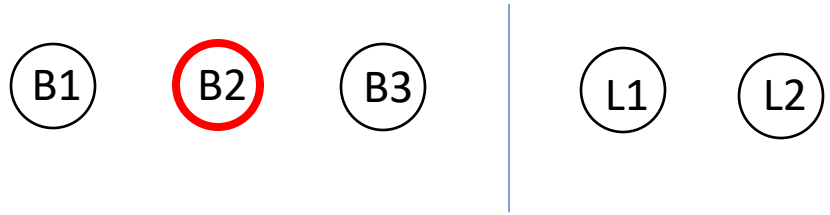


- Suppose we are playing a 2-round MS game

If Duplicator could not duplicate, she would lose!

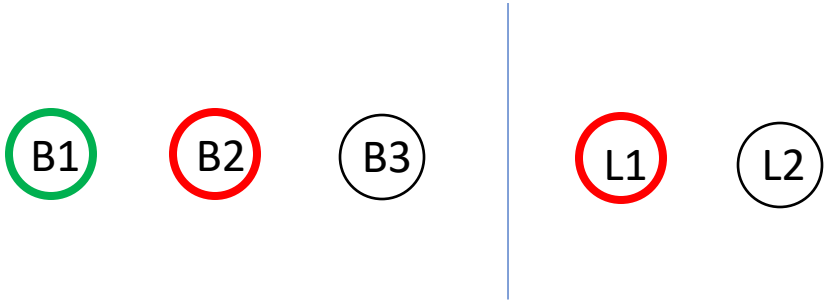
Illustrative MS game on Linear Orders

- Pictorially we place smaller elements to the left and bigger elements to the right.



- Two sets of pebbled elements are partially isomorphic if they are in the same left-to-right order.

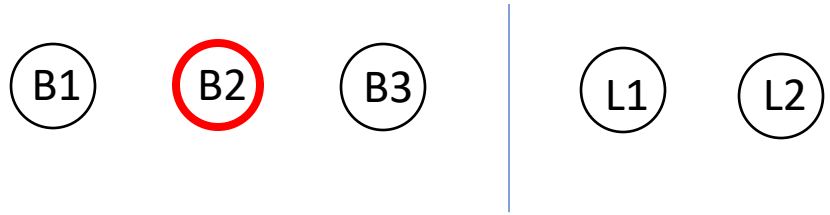
- Suppose we are playing a 2-round MS game



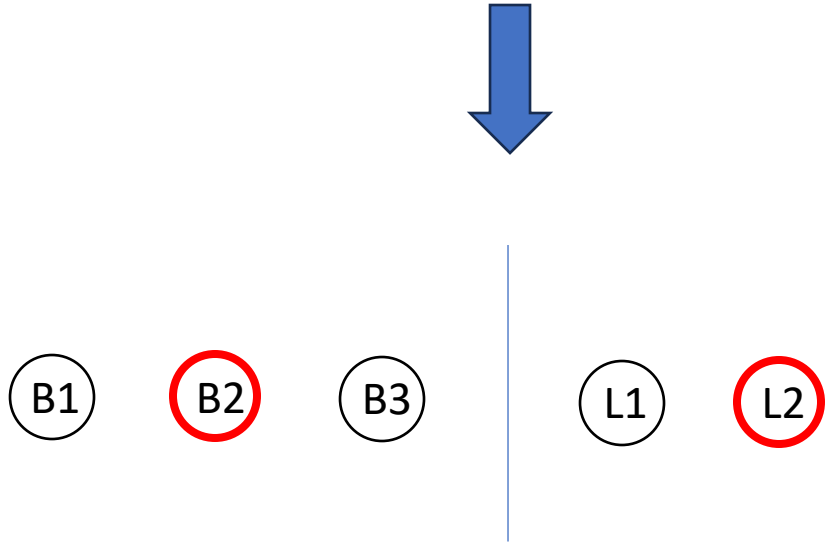
If Duplicator could not duplicate, she would lose!

Illustrative MS game on Linear Orders

- Pictorially we place smaller elements to the left and bigger elements to the right.



- Two sets of pebbled elements are partially isomorphic if they are in the same left-to-right order.

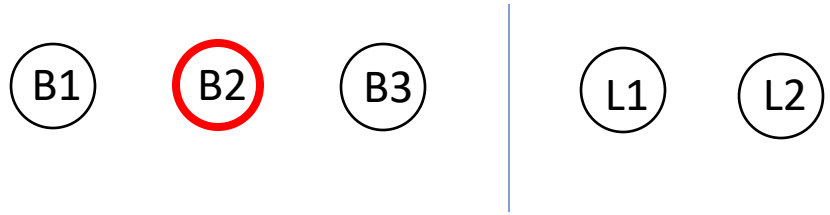


- Suppose we are playing a 2-round MS game

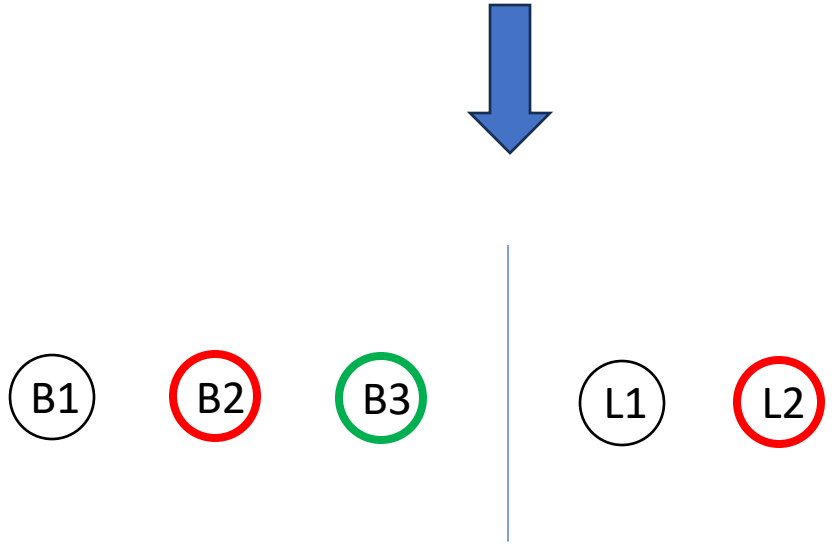
If Duplicator could not duplicate, she would lose!

Illustrative MS game on Linear Orders

- Pictorially we place smaller elements to the left and bigger elements to the right.



- Two sets of pebbled elements are partially isomorphic if they are in the same left-to-right order.

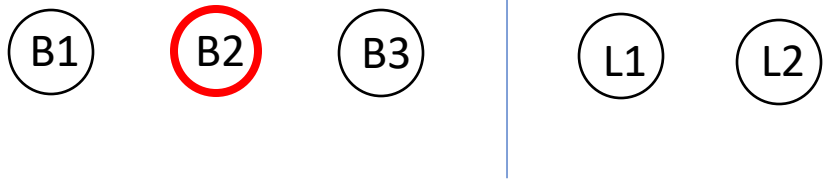


- Suppose we are playing a 2-round MS game

If Duplicator could not duplicate, she would lose!

Illustrative MS game on Linear Orders

- Pictorially we place smaller elements to the left and bigger elements to the right.

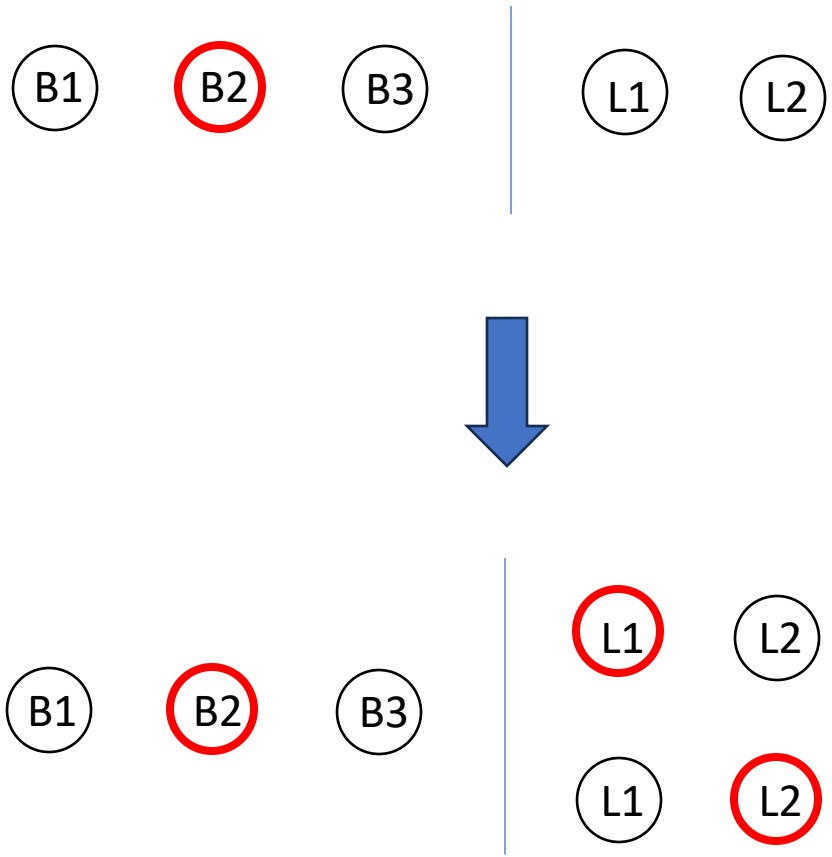


- Two sets of pebbled elements are partially isomorphic if they are in the same left-to-right order.

- Suppose we are playing a 2-round MS game.

Illustrative MS game on Linear Orders

- Pictorially we place smaller elements to the left and bigger elements to the right.
- Two sets of pebbled elements are partially isomorphic if they are in the same left-to-right order.
- Suppose we are playing a 2-round MS game



MS Games and the Oblivious Strategy

Note that in an MS game, it never hurts Duplicator to make as many copies of all structures as she needs and play all possible moves.

Such a strategy is always optimal, and we call it the “Oblivious Strategy”.

Equivalence Theorem for EF Games

Equivalence Theorem for Ehrenfeucht-Fraïssé Games: Spoiler wins the r-round EF game on structures A, B iff there is a FO sentence σ with quantifier rank r such that $A \vDash \sigma$ while $B \vDash \neg\sigma$.

Equivalence Theorem for Ehrenfeucht-Fraïssé Games (Alternative Formulation): A property P can be described by a FO sentence having quantifier rank r iff Spoiler can win an EF game of r rounds on every pair A, B of structures, with A satisfying P and B not satisfying P.

Equivalence Theorem for MS Games

Equivalence Theorem for Multi-Structural Games: Spoiler wins the r-round MS game on sets $\mathcal{A}$ and $\mathcal{B}$ of structures iff there is a FO sentence σ with r quantifiers such that $A \models \sigma$ for every $A \in \mathcal{A}$, while $B \models \neg\sigma$ for every $B \in \mathcal{B}$.

Equivalence Theorem for Multi-Structural Games (Alternative Formulation): A property P can be described by a FO sentence with r quantifiers iff Spoiler can win the multi-structural game of r rounds on sets $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$ of structures, where $\boldsymbol{\mathcal{A}}$ consists of all structures satisfying P and $\boldsymbol{\mathcal{B}}$ consists of all structures that don't satisfy P.

Important Nuance

The proof of the Equivalence Theorem shows that if Spoiler has a winning strategy, then Spoiler playing on the **left** corresponds to there being an **existential quantifier (∃)** in the separating sentence σ, while Spoiler playing on the **right** corresponds to there being a **universal quantifier (∀)** in σ.

Why Use Quantifier Number (QN) rather than Quantifier Rank (QR)?

In his 1981 paper, Neil provided the following construction showing that **every property of n-vertex ordered graphs can be expressed with a sentence having quantifier rank $\boldsymbol{\log n + 3}$**:

Let $\eta_i(x)$ be the formula that says x is the ith element in the n-vertex ordered graph G. $\eta_i(x)$ can be written as a formula, with one free variable, having at most quantifier rank $\log n + 1$. (That this can be done is an easy consequence of a result of Rosenstein's that we shall mention later.)

Let $E^{ij}(x, y)$ be a shorthand way of writing $E(x, y)$ if there is an edge between the ith and jth vertices in the ordered graph G, and $\neg E(x, y)$ if there is no such edge.

Why Use QN rather than QR?

The graph G can be completely described by the following sentence:

$$\sigma_G = \Lambda_{1\leq i,j\leq n} \exists x \exists y (\eta_i(x) \wedge \eta_j(y) \wedge E^{ij}(x,y)).$$

Since η_i, η_j have QR at most $\log n + 1$, σ_G has QR at most $\log n + 3$.

For a property P of n-vertex ordered graphs, let $\{G_i\}$ denote the family of all (n-vertex) graphs satisfying P with associated sentences σ_{G_i}. Then a sentence expressing P is just

$$\Sigma_P = \bigvee_i \sigma_{G_i}$$

which also has QR $\log n + 3$ [though QN may be huge! The above construction uses more than 2^{n^2} quantifiers!].

Why Use QN rather than QR?

In the case, of QN, it is easy to see that $\boldsymbol{n}$ **quantifiers** are sufficient to describe every property of n-vertex ordered graphs. Every n-vertex graph can be defined by the sentence

$$\Sigma_G = \exists x_1 \exists x_2 \cdots \exists x_n \underbrace{(x_1 < x_2 < \cdots < x_n \bigwedge_{1 \le i,j \le n} E^{ij}(x_i, x_j))}_{\sigma_G}.$$

So, for any property P of n-vertex ordered graphs, again with instances $\{G_i\}$, we can write:

$$\Sigma_P = \exists x_1 \exists x_2 \cdots \exists x_n \bigvee_i \sigma_{G_i}$$

which also has n quantifiers.

UPSHOT: There is a lot more leeway between $\log n + 1$ and n for QN, than between $\log n + 1$ and $\log n + 3$ for QR!!

Our Problems

In our first paper about the MS game, we showed how many quantifiers were necessary to distinguish **linear orders** of different sizes.

The next interesting ordered structures are the n-bit **binary strings**. We explored questions such as how many quantifiers are needed to distinguish one binary string from all others, or to distinguish one set of binary strings from its complement?

Setup

For the purpose of this talk, an **n-bit string** is a set of n elements equipped with

- a total ordering relation $<(,)$
- $\boldsymbol{min}$ and $\boldsymbol{max}$ constants that identify the 1st and last elements of the string
- a unary $1()$ relation that returns True iff a given element in the string is a 1.

Preliminary Observations

Observation 1: Since a Boolean function on n bits, $f: \{0,1\}^n \rightarrow \{0,1\}$, defines two complementary sets: $f^{-1}(0), f^{-1}(1)$, the number of quantifiers needed to express an arbitrary Boolean function on n bits is the same as the number of quantifiers needed to express an arbitrary set of n-bit strings.

By **expressing a set of n-bit strings** we mean finding a sentence that is true for the given set of n-bit strings and false for the complementary set of n-bit strings.

Preliminary Observations

Observation 2: Analogous to our observation about n-vertex graphs, every set S of n-bit strings can be described via a sentence having n existential quantifiers.

Preliminary Observations

Observation 2: Analogous to our observation about n-vertex graphs, every set S of n-bit strings can be described via a sentence having n existential quantifiers.

The sentence looks like this:

$$\exists x_1 \cdots \exists x_n ((x_1 < x_2 < \cdots < x_n) \wedge$$

"There exists n elements in increasing order"

Preliminary Observations

Observation 2: Analogous to our observation about n-vertex graphs, every set S of n-bit strings can be described via a sentence having n existential quantifiers.

The sentence looks like this:

$S = \{10 \cdots 0, \ldots, 01 \cdots 0\}$

$$\exists x_1 \cdots \exists x_n ((x_1 < x_2 < \cdots < x_n) \wedge$$

$$((1(x_1) \wedge \neg 1(x_2) \wedge \cdots \wedge \neg 1(x_n)) \vee$$

$$\cdots$$

$$(\neg 1(x_1) \wedge 1(x_2) \wedge \cdots \wedge \neg 1(x_n)))$$

"There exists n elements in increasing order"

One line for each of the strings in the set.

Quantifier Complexity of Boolean Functions: Main Results

Note. In everything that follows, $\log(n)$ denotes the base-2 logarithm of n.

► **Theorem A.** *Given an arbitrary $\varepsilon > 0$, every Boolean function on n-bit strings can be defined by a FO sentence having $(1+\varepsilon)\frac{n}{\log(n)} + O_\varepsilon(1)$ quantifiers, where the $O_\varepsilon(1)$ additive term depends only on ε and not n. Moreover, there are Boolean functions on n-bit strings that require $\frac{n}{\log(n)} + O(1)$ quantifiers to define.*

Quantifier Complexity of Boolean Functions: Main Results

Definition. Say that a family, $\{f_n\}_{n=1}^{\infty}$, of Boolean functions on n-bit strings is **sparse** if the cardinality of the set of strings mapping to 1 or to 0 under each f_n is polynomial in n.

Quantifier Complexity of Boolean Functions: Main Results

Definition. Say that a family, $\{f_n\}_{n=1}^{\infty}$, of Boolean functions on n-bit strings is **sparse** if the cardinality of the set of strings mapping to 1 or to 0 under each f_n is polynomial in n.

▶ **Theorem B.** *Given an arbitrary $\varepsilon > 0$, and a sparse family, $\{f_n\}_{n=1}^{\infty}$, of Boolean functions on n-bit strings, each function f_n can be defined by a FO sentence having $(1+\varepsilon)\log(n)+O_\varepsilon(1)$ quantifiers, where the $O_\varepsilon(1)$ additive term depends only on ε and not n or the choice of sparse family. Moreover, there are sparse families of Boolean functions on n-bit strings, the functions of which require $\log(n)$ quantifiers to define.*

First Results

- Since Rosenstein's work on linear orders in 1981, it has been known that quantifier rank (e.g., quantifier nesting depth) $\log n + 1$ is both necessary and sufficient to distinguish linear orders of size $\leq n$ from linear orders of size $> n$.

First Results

- Since Rosenstein's work on linear orders in 1981, it has been known that quantifier rank (e.g., quantifier nesting depth) $\log n + 1$ is both necessary and sufficient to distinguish linear orders of size $\leq n$ from linear orders of size $> n$.

- In a paper from LICS 2021 we were able to show that, similarly, for QN, $\log n + O(1)$ is both necessary and sufficient to distinguish linear orders of size $\leq n$ from all larger linear orders.

First Results

- Since Rosenstein's work on linear orders in 1981, it has been known that quantifier rank (e.g., quantifier nesting depth) $\log n + 1$ is both necessary and sufficient to distinguish linear orders of size $\leq n$ from linear orders of size $> n$.

- In a paper from LICS 2021 we were able to show that, similarly, for QN, $\log n + O(1)$ is both necessary and sufficient to distinguish linear orders of size $\leq n$ from all larger linear orders.

- A key fact that I shall use, but not prove, is that the sentences establishing the minimal QN for linear orders can be chosen to have **strictly alternating quantifier signatures**.

The Concept of Parallel Play

- Suppose that after k rounds of an r round MS game we can partition the pebbled structures as follows:

$$\mathcal{A}_1^k \cong \mathcal{B}_1^k$$

$$\mathcal{A}_2^k \cong \mathcal{B}_2^k$$

The Concept of Parallel Play

- Suppose that after k rounds of an r round MS game we can partition the pebbled structures as follows:

$$\mathcal{A}_1^k \cong \mathcal{B}_1^k$$

$$\mathcal{A}_2^k \cong \mathcal{B}_2^k$$

- Now imagine the remainder of the game being played as two separate **parallel games**.

- If Spoiler can win each of the parallel games by **playing on the same side of the board** in each of the remaining r-k rounds, then he can win the original game.

A Result about Linear Orders

In our distinguishing strings (a.k.a. characterizing Boolean functions) work we repeatedly leverage a variant of our LICS 2021 result about linear orders.

We need to be able to say that “between element x and element y in a string there is a linear order of precisely length n”.

A Result about Linear Orders

Remember, we know how to distinguish linear orders of sizes $\leq n$ from linear orders of sizes $> n$ with a logarithmic number of quantifiers.

σ_n

A Result about Linear Orders

Remember, we know how to distinguish linear orders of sizes $\leq n$ from linear orders of sizes $> n$ with a logarithmic number of quantifiers.

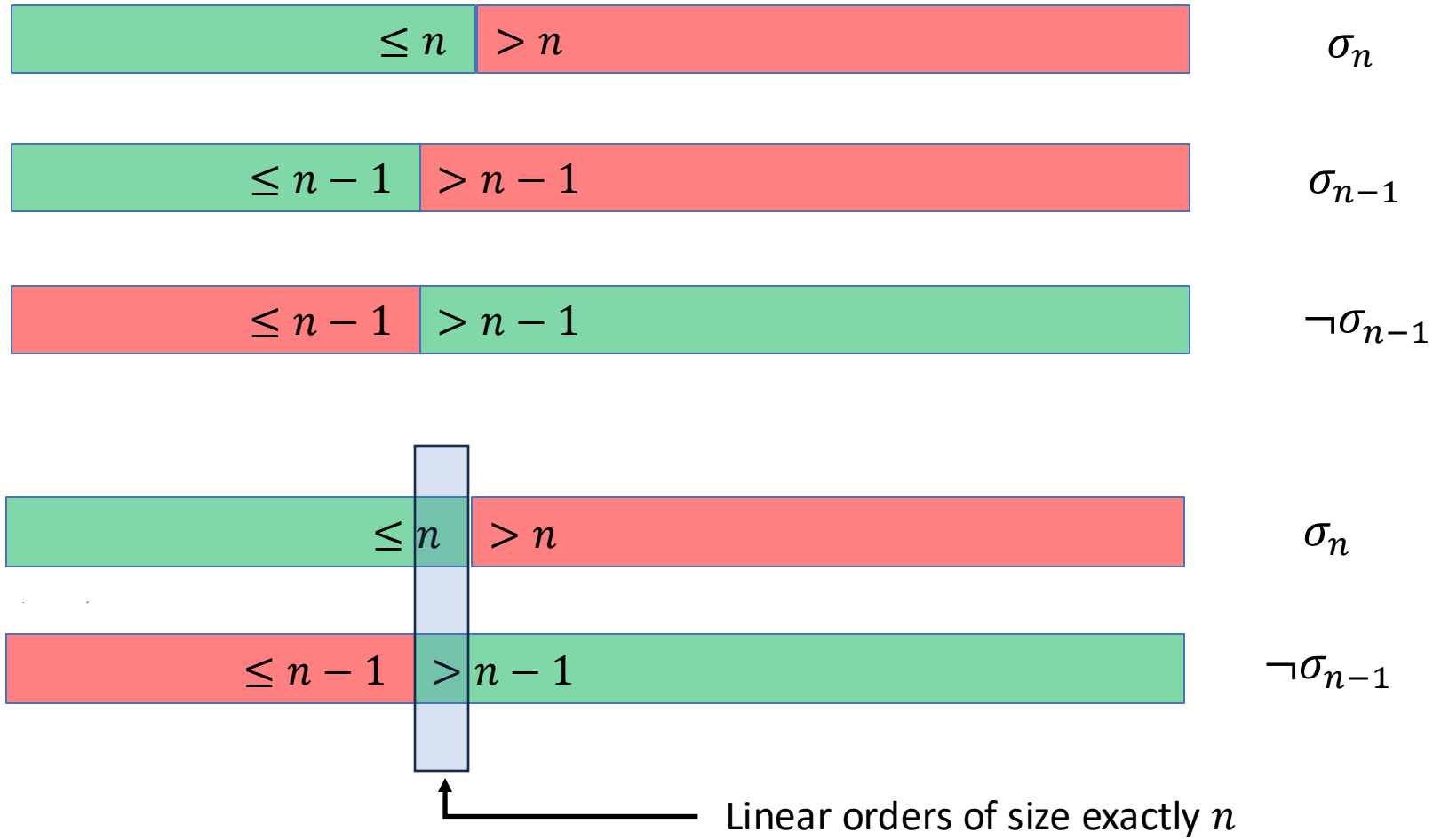
σ_n

σ_{n-1}

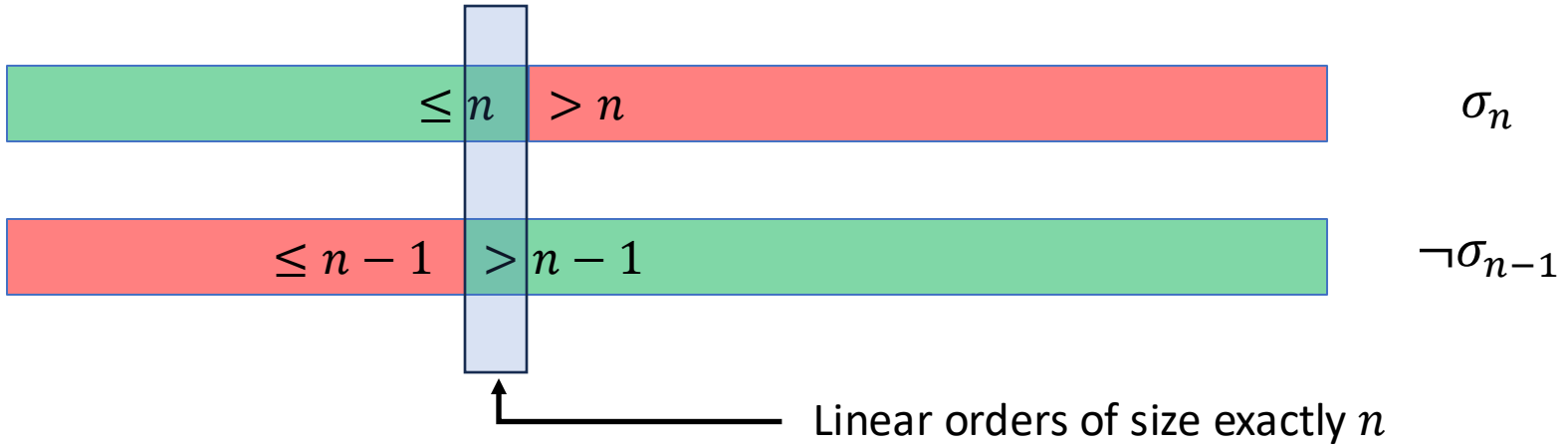
$\neg\sigma_{n-1}$

A Result about Linear Orders

Remember, we know how to distinguish linear orders of sizes $\leq n$ from linear orders of sizes $> n$ with a logarithmic number of quantifiers.



A Result about Linear Orders

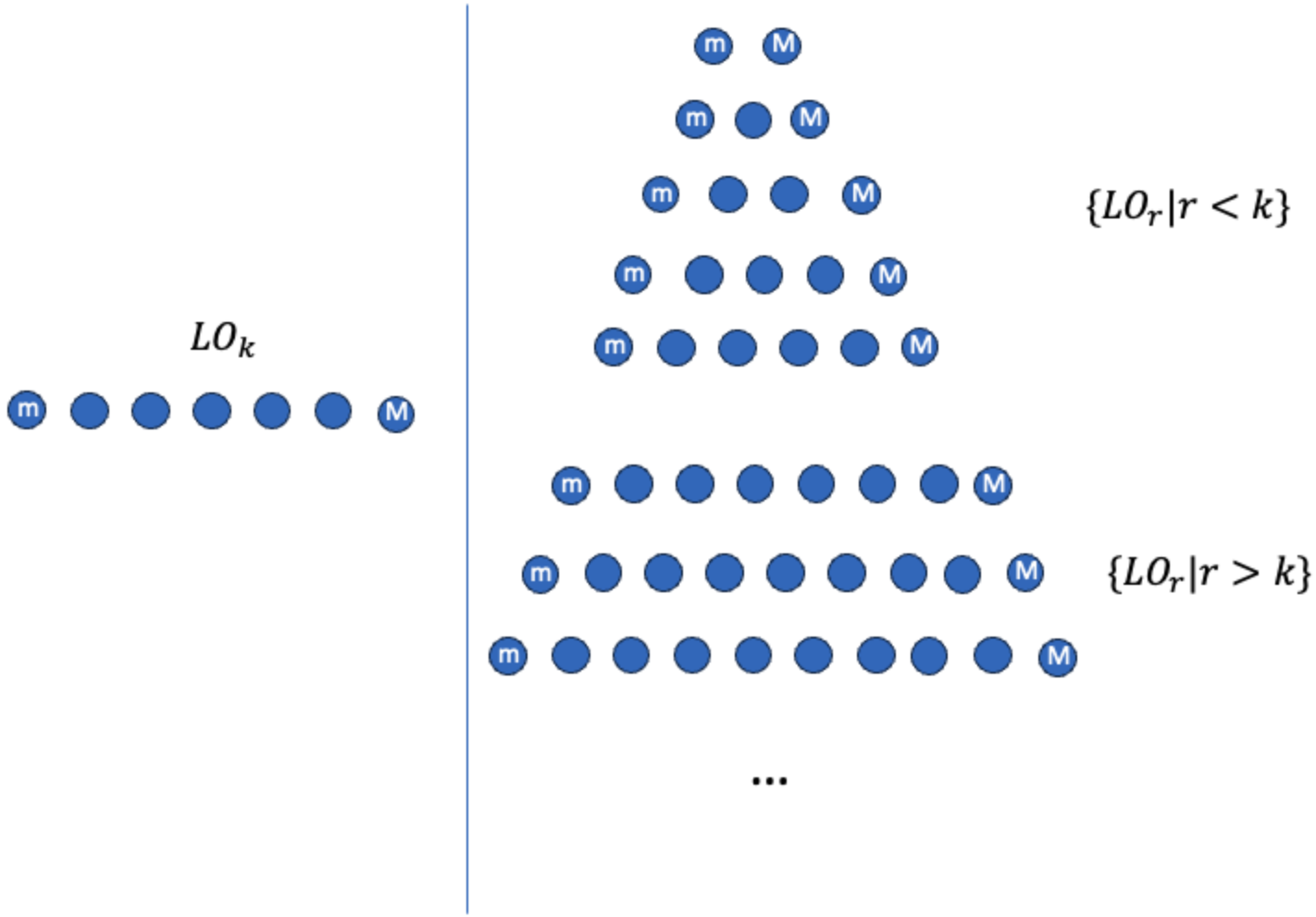


Taking the conjunction, $\sigma_n \wedge \neg\sigma_{n-1}$ we can distinguish the linear order of size n from all other linear orders with **twice this logarithmic number of quantifiers**.

But we can do better! In other words, we can say the same thing with fewer quantifiers.

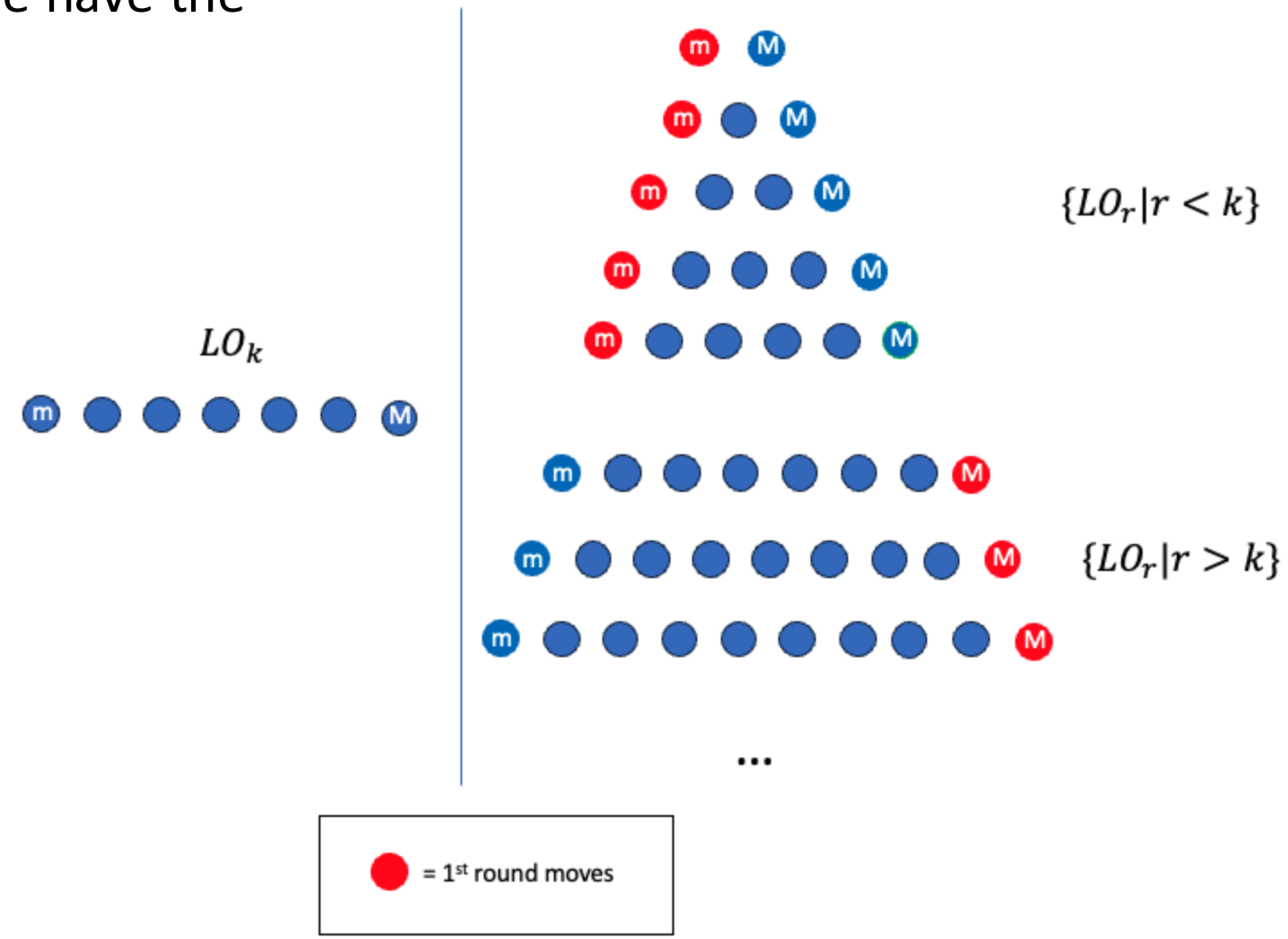
A Result about Linear Orders

From a game point of view, we have the following setup:



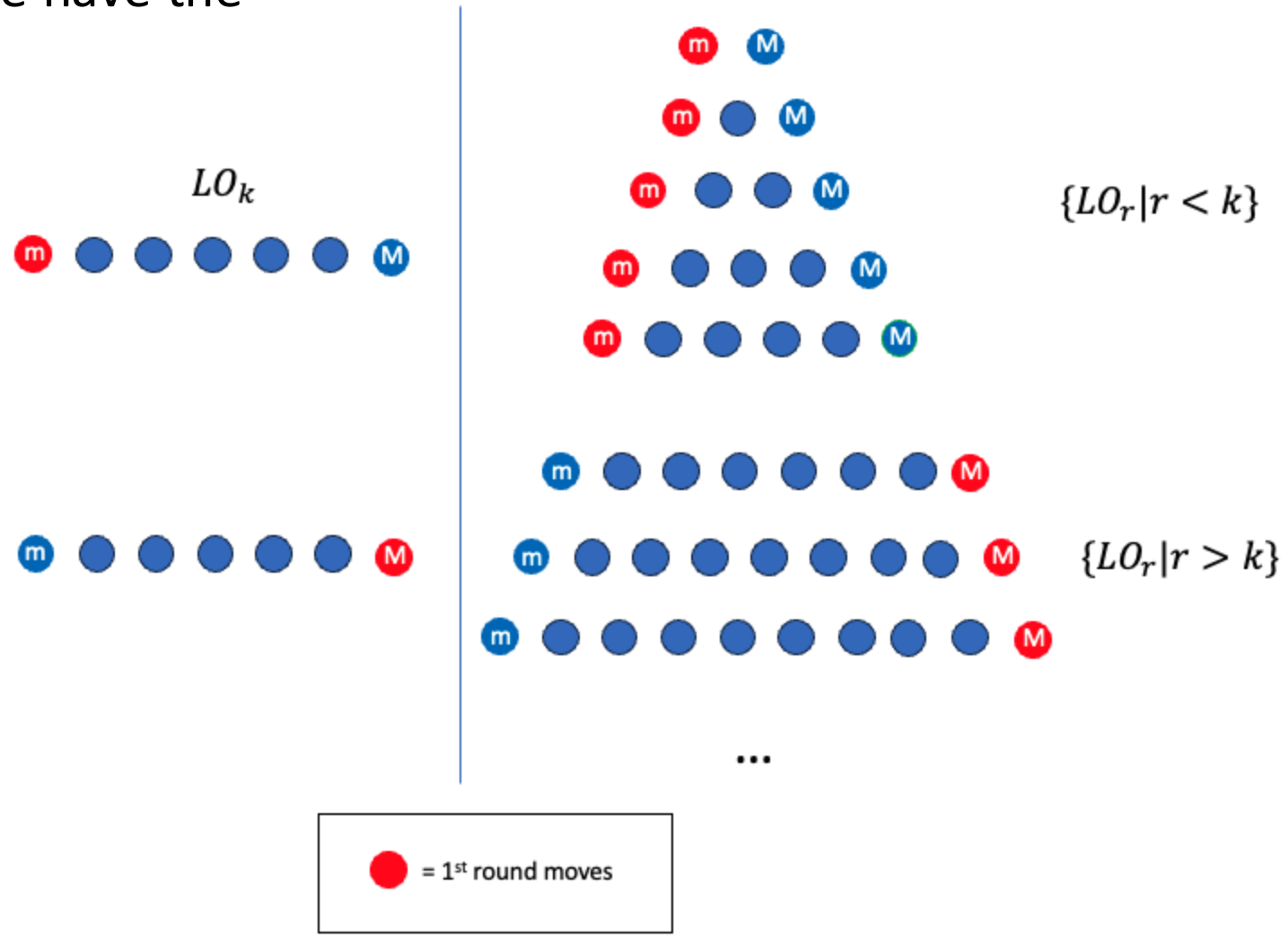
A Result about Linear Orders

From a game point of view, we have the following setup:



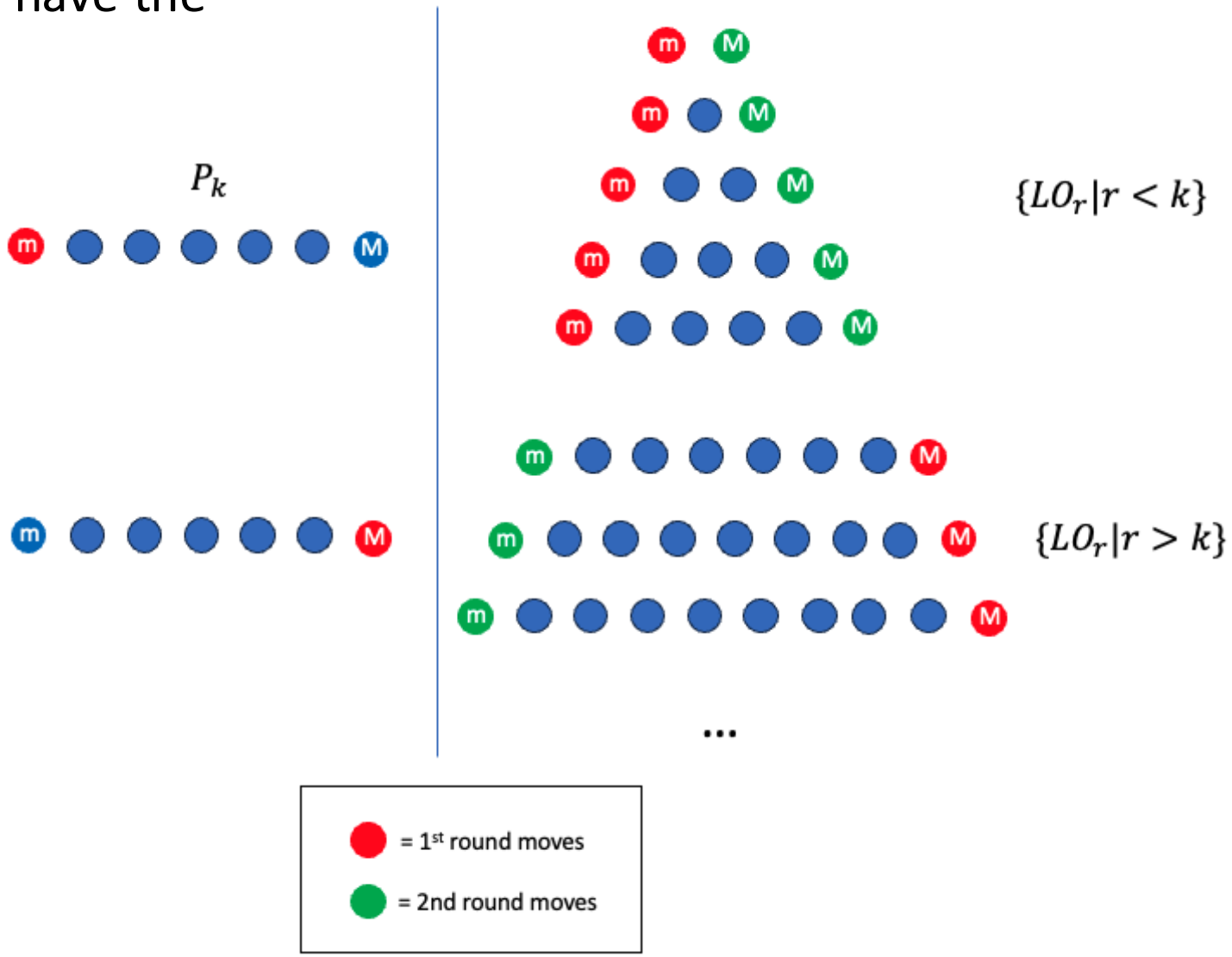
A Result about Linear Orders

From a game point of view, we have the following setup:



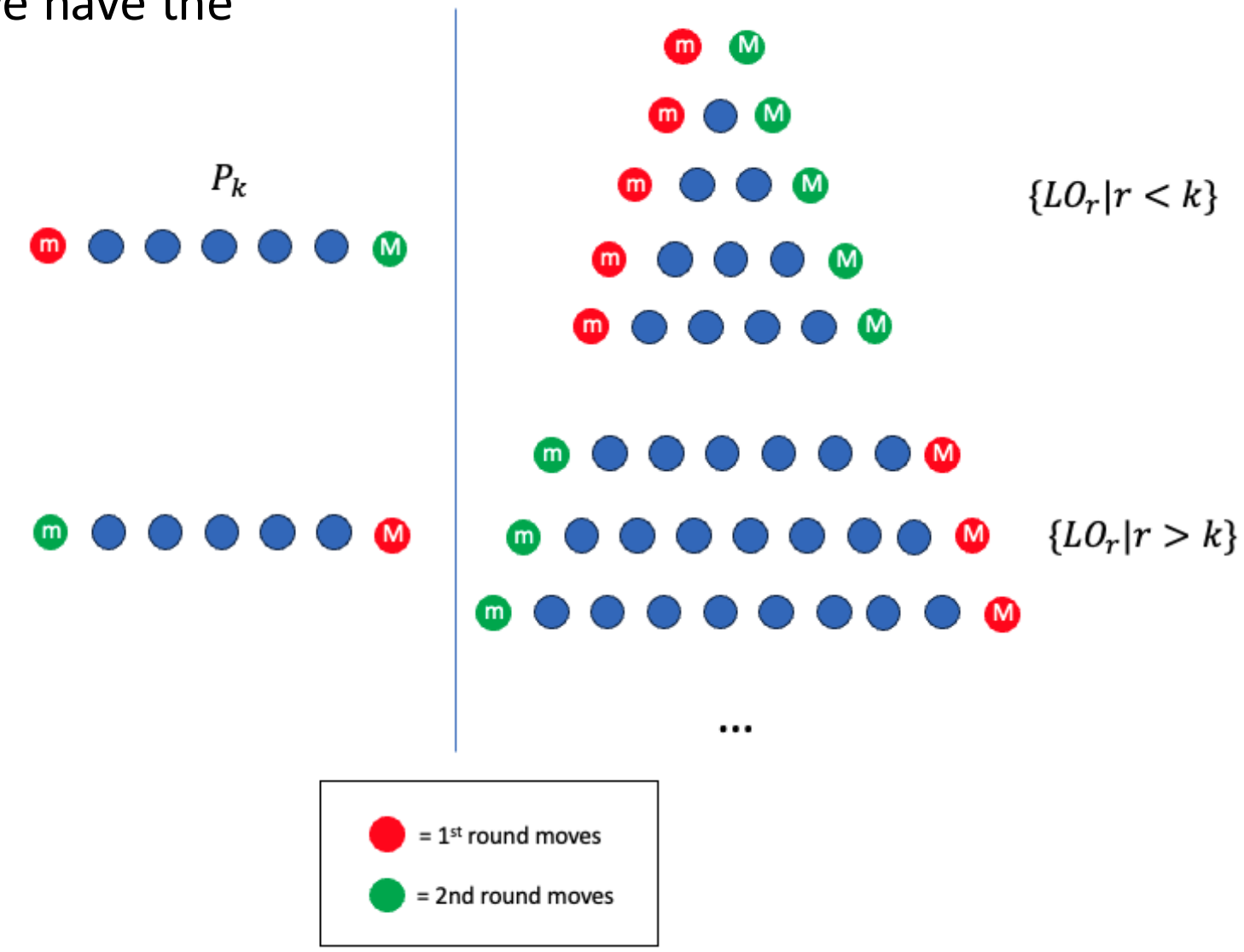
A Result about Linear Orders

From a game point of view, we have the following setup:



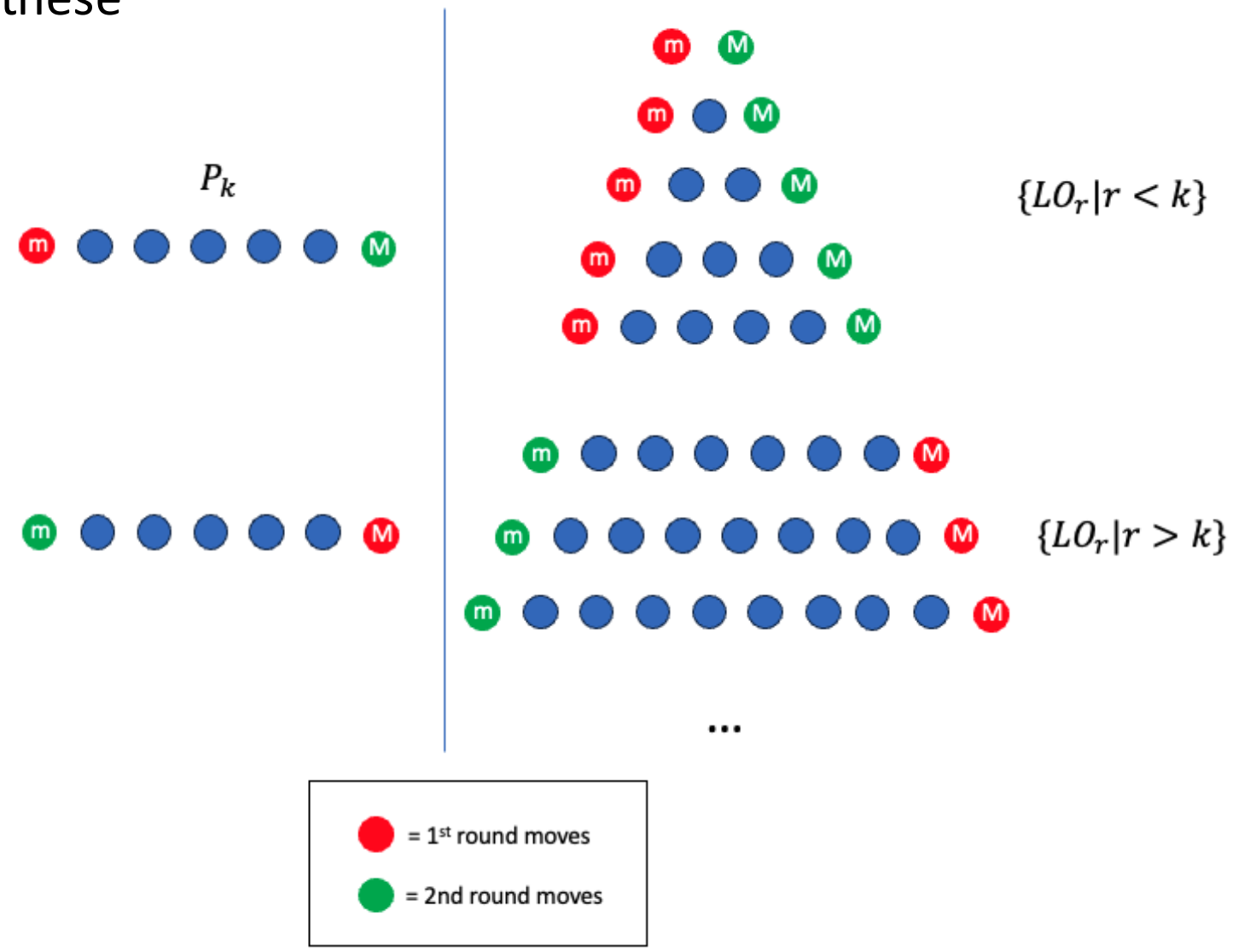
A Result about Linear Orders

From a game point of view, we have the following setup:



A Result about Linear Orders

- Spoiler is almost ready to play these two games in parallel…

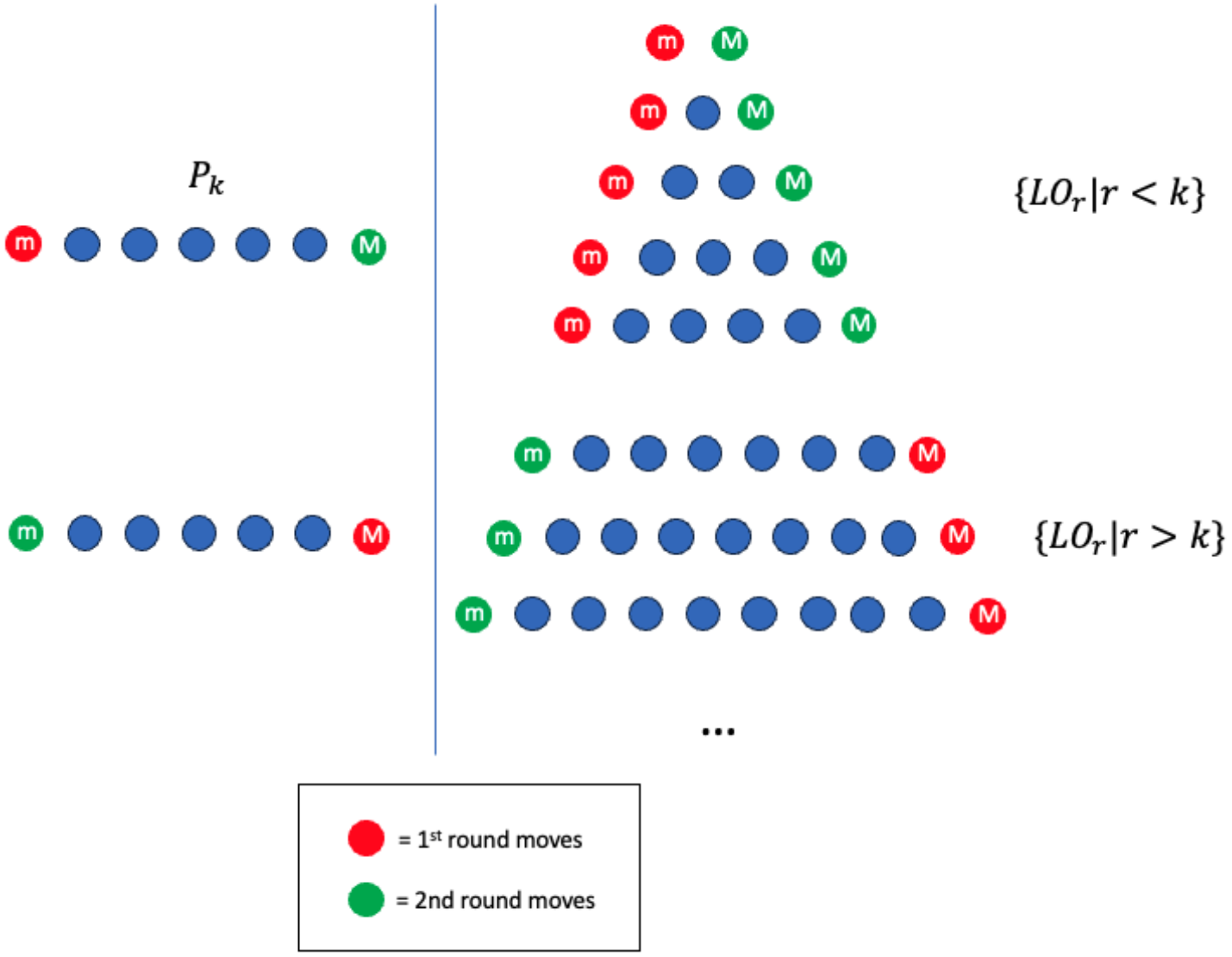


A Result about Linear Orders

- Spoiler uses the following **key fact**:
- The sentences:

  i. Distinguishing the linear order of size n from larger linear orders (σ_n), and
  ii. Distinguishing the linear order of size n from smaller linear orders ($\neg\sigma_{n-1}$)

  both have alternating quantifier signatures, with σ_n possibly having one additional quantifier.



A Result about Linear Orders

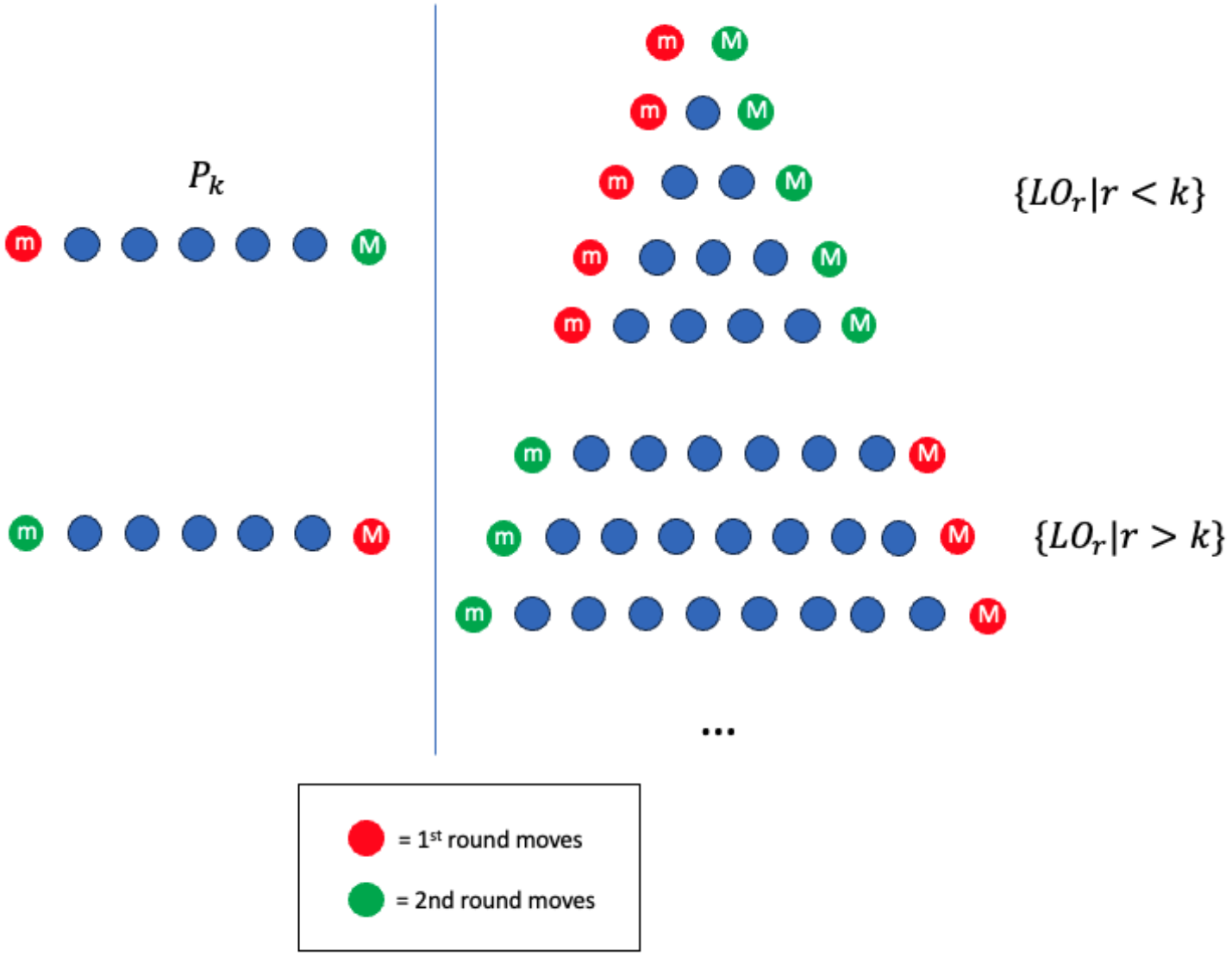
- Spoiler uses the following **key fact**:
- The sentences:

  i. Distinguishing the linear order of size n from larger linear orders (σ_n), and
  ii. Distinguishing the linear order of size n from smaller linear orders ($\neg\sigma_{n-1}$)

  both have alternating quantifier signatures, with σ_n possibly having one additional quantifier.

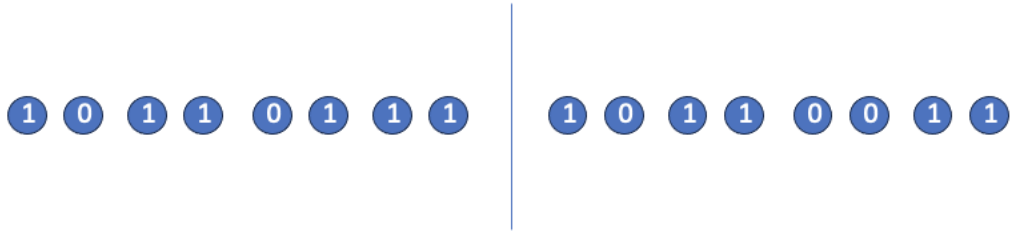
  Thus, tacking on at most an extra (dummy) quantifier at the beginning or end of each they can be given the same quantifier signature.

- This allows for parallel play, still with log(n) + O(1) quantifiers.



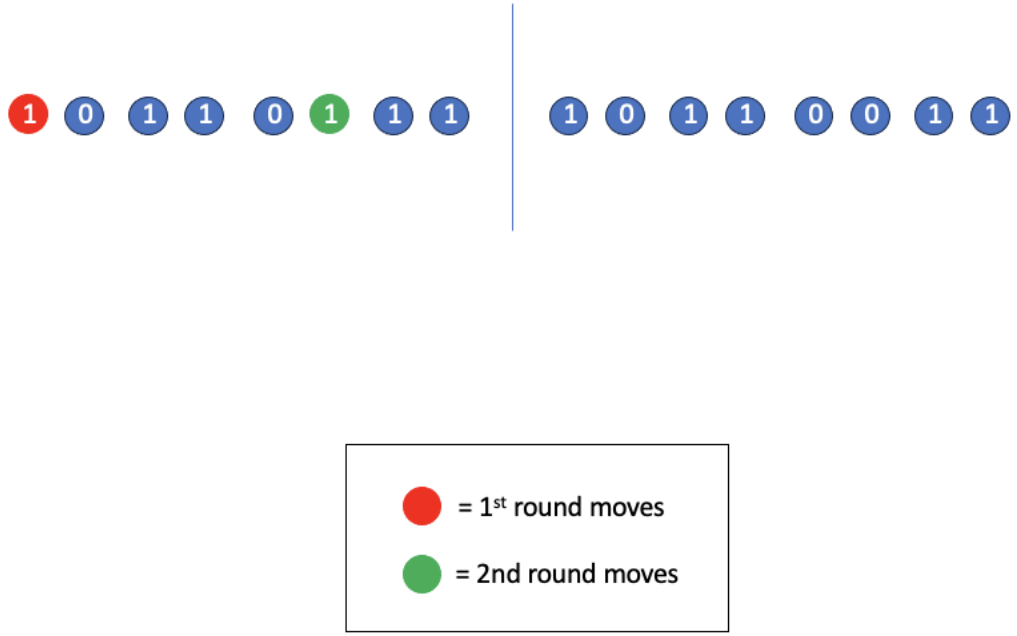
Distinguishing Strings with Few Quantifiers

- How about distinguishing two different **strings** of length n?
- It turns out that we can turn this game into the same parallel play linear order game!



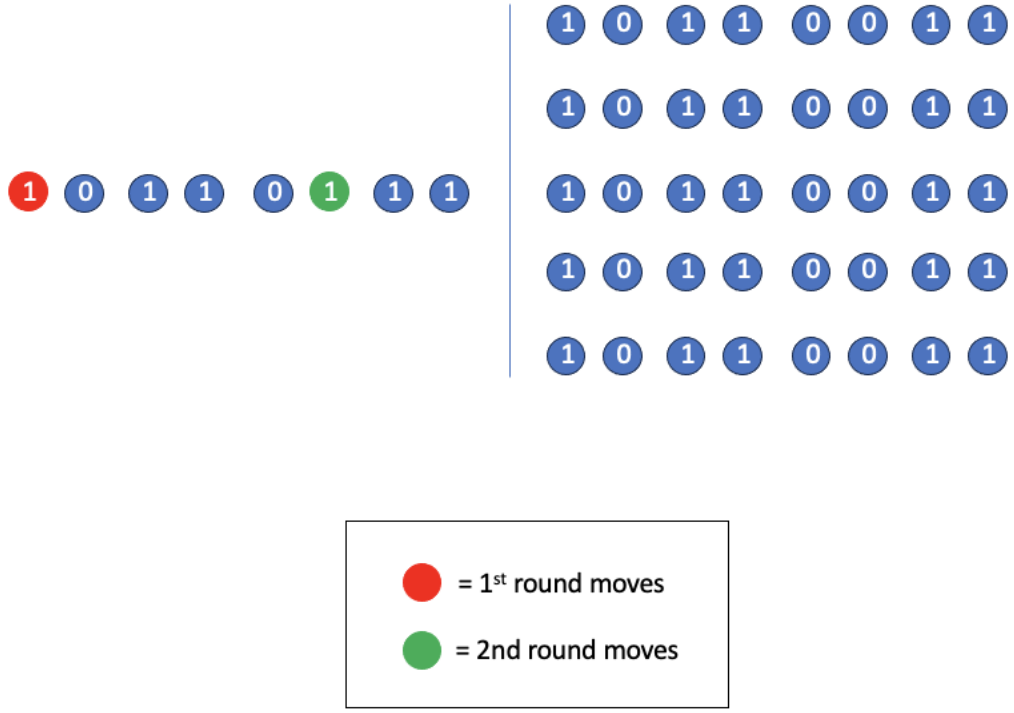
Distinguishing Strings with Few Quantifiers

- How about distinguishing two different **strings** of length n?
- It turns out that we can turn this game into the same parallel play linear order game!
- Spoiler plays his first two moves on the left hand (∃) side. He places his 1st pebble on the 1st element and his second pebble on the first point where the two strings disagree.



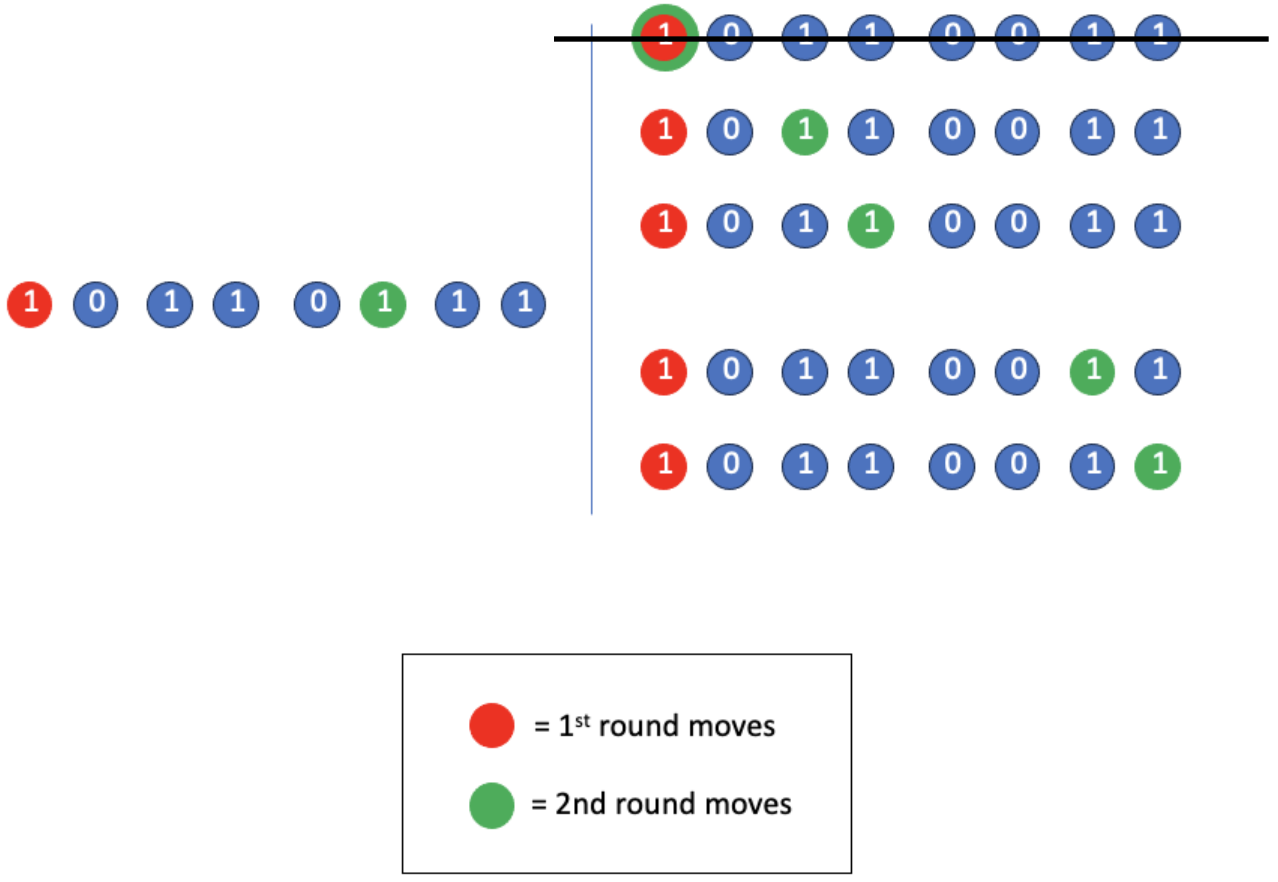
Distinguishing Strings with Few Quantifiers

- How about distinguishing two different **strings** of length n?
- It turns out that we can turn this game into the same parallel play linear order game!
- Spoiler plays his first two moves on the left hand (∃) side. He places his 1st pebble on the 1st element and his second pebble on the first point where the two strings disagree.
- Duplicator must also play her 1st pebble on the 1st element. She will play her 2nd pebble in all possible ways.



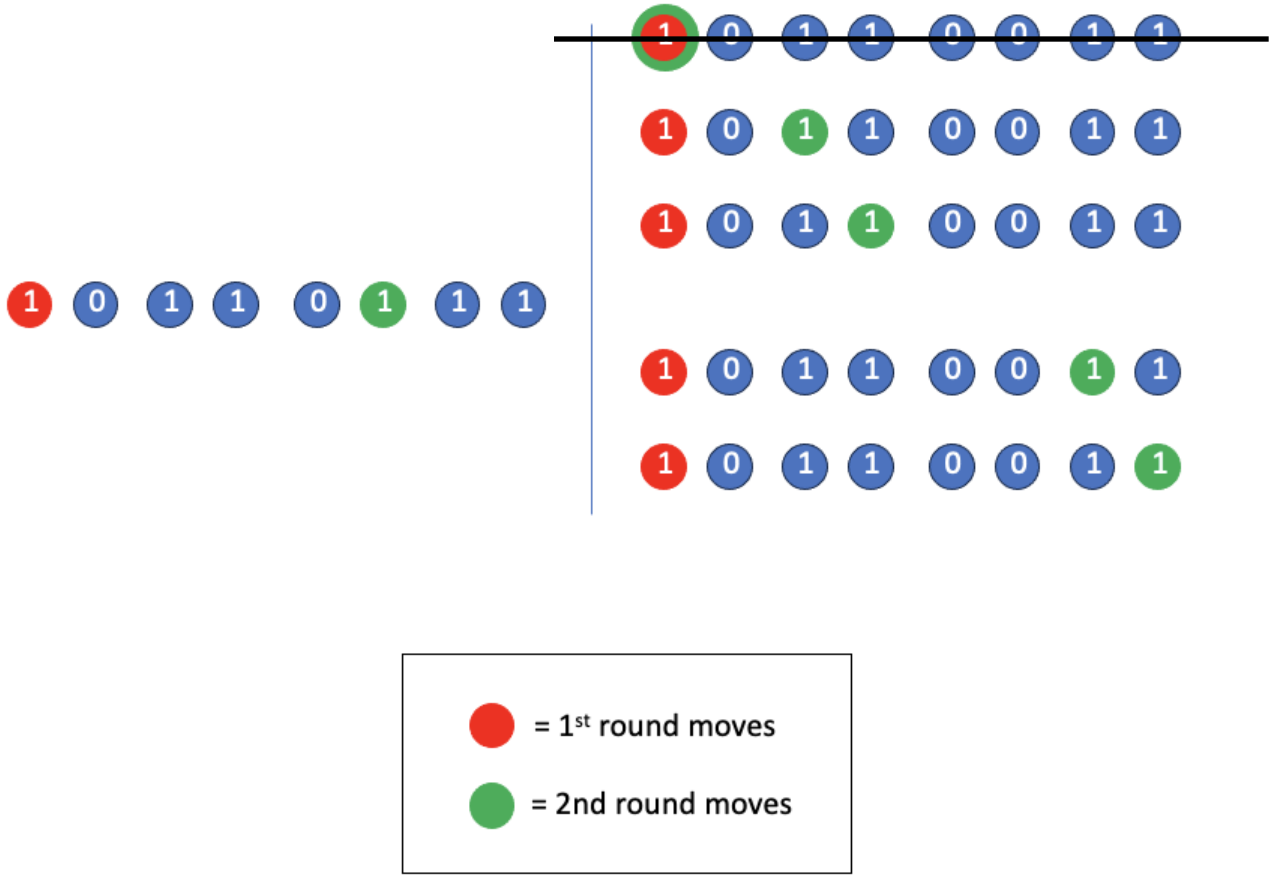
Distinguishing Strings with Few Quantifiers

- How about distinguishing two different **strings** of length n?
- It turns out that we can turn this game into the same parallel play linear order game!
- Spoiler plays his first two moves on the left hand (∃) side. He places his 1st pebble on the 1st element and his second pebble on the first point where the two strings disagree.
- Duplicator must also play her 1st pebble on the 1st element. She will play her 2nd pebble in all possible ways.



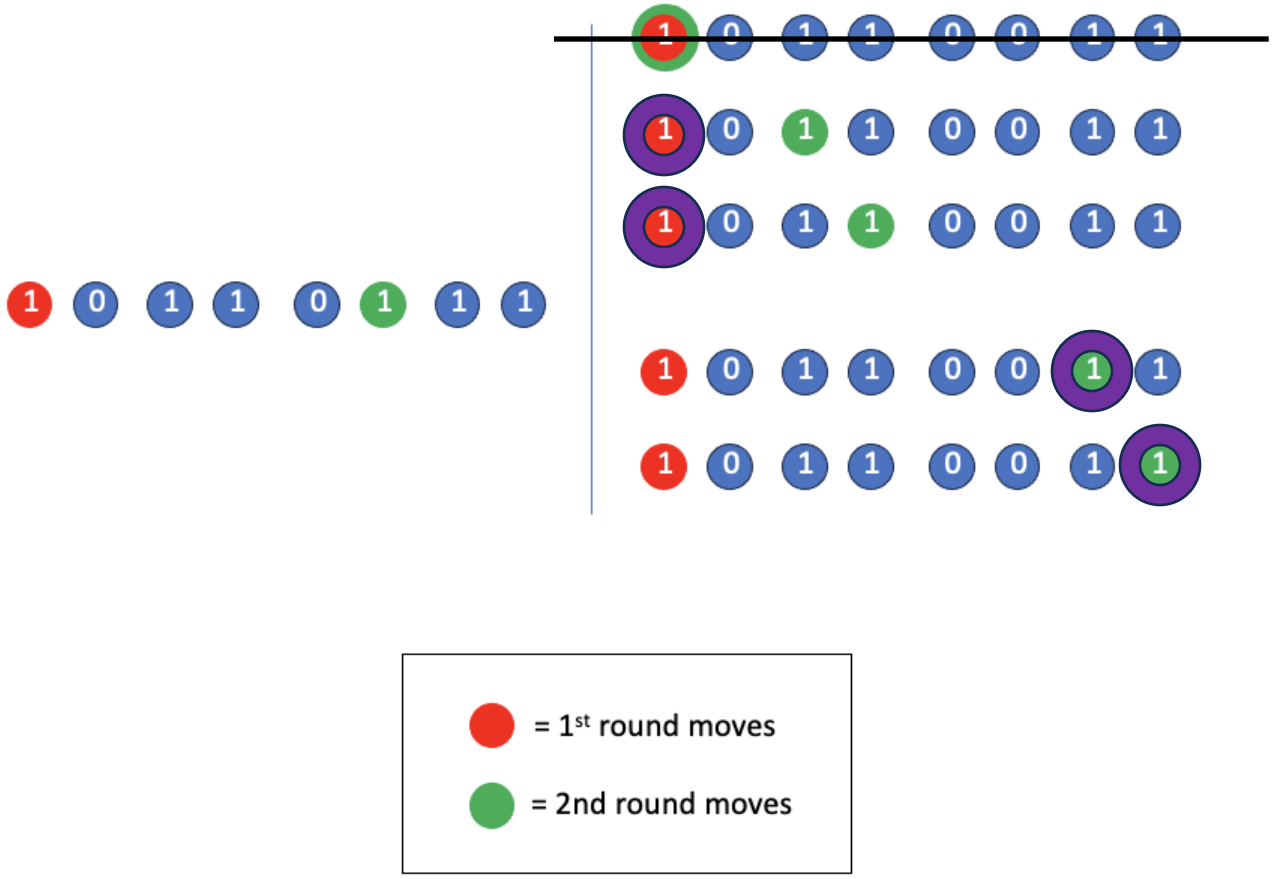
Distinguishing Strings with Few Quantifiers

- **Question**: In one more round, how can Spoiler play in such a way to turn the game into a Parallel Play situation?



Distinguishing Strings with Few Quantifiers

- Spoiler can play his 3rd round moves on top of the red pebbles for the first half of the strings on the right and on top of the green pebbles for the second half of the strings on the right.
- Duplicator just has two meaningful ways to play on the left.
- The game is then reduced to a pair of parallel play linear order games.



Distinguishing Strings with Few Quantifiers

- This argument shows that one can distinguish every two strings of length n from one another with a sentence having $\log n + O(1)$ quantifiers.
- It is also straightforward to show that this many quantifiers may also be necessary.

Distinguishing Strings with Few Quantifiers

- This argument shows that one can distinguish every two strings of length n from one another with a sentence having $\log n + O(1)$ quantifiers.
- It is also straightforward to show that this many quantifiers may also be necessary.
- In fact, it is possible to separate one n-bit string from **all other** n-bit strings with $\log n + O(1)$ quantifiers.
- And recall, we can separate any sparse set of n-bit strings from its complementary set with just $(1 + \epsilon) \log n + O(1)$ quantifiers for arbitrarily small ϵ.

Separating Two Arbitrary Sets of Strings

Recalling our other main result on separating strings:

► **Theorem A.** *Given an arbitrary $\varepsilon > 0$, every Boolean function on n-bit strings can be defined by a FO sentence having $(1+\varepsilon)\frac{n}{\log(n)} + O_\varepsilon(1)$ quantifiers, where the $O_\varepsilon(1)$ additive term depends only on ε and not n. Moreover, there are Boolean functions on n-bit strings that require $\frac{n}{\log(n)} + O(1)$ quantifiers to define.*

The Road Ahead to Realize Neil Immerman's Vision

- We've devised a game that characterizes not just QN, but simultaneously QN and number of variables (VN). We are interested in studying the tradeoff between QN and VN for various properties (extending work of Grohe and Schweikardt). Connected to time vs. space tradeoffs.

- Characterize QN for specific string and graph properties.

- **MILLION $$ PROBLEM:** Can we find a concrete property (that is not known to be outside of NL) that requires more than $O(\log n)$ quantifiers? Such a property would necessarily lie outside of NL by Neil's inclusion relation. We have some ideas….

More Recent Results: How Hard is it to decide WHO WINS an MS-Game?

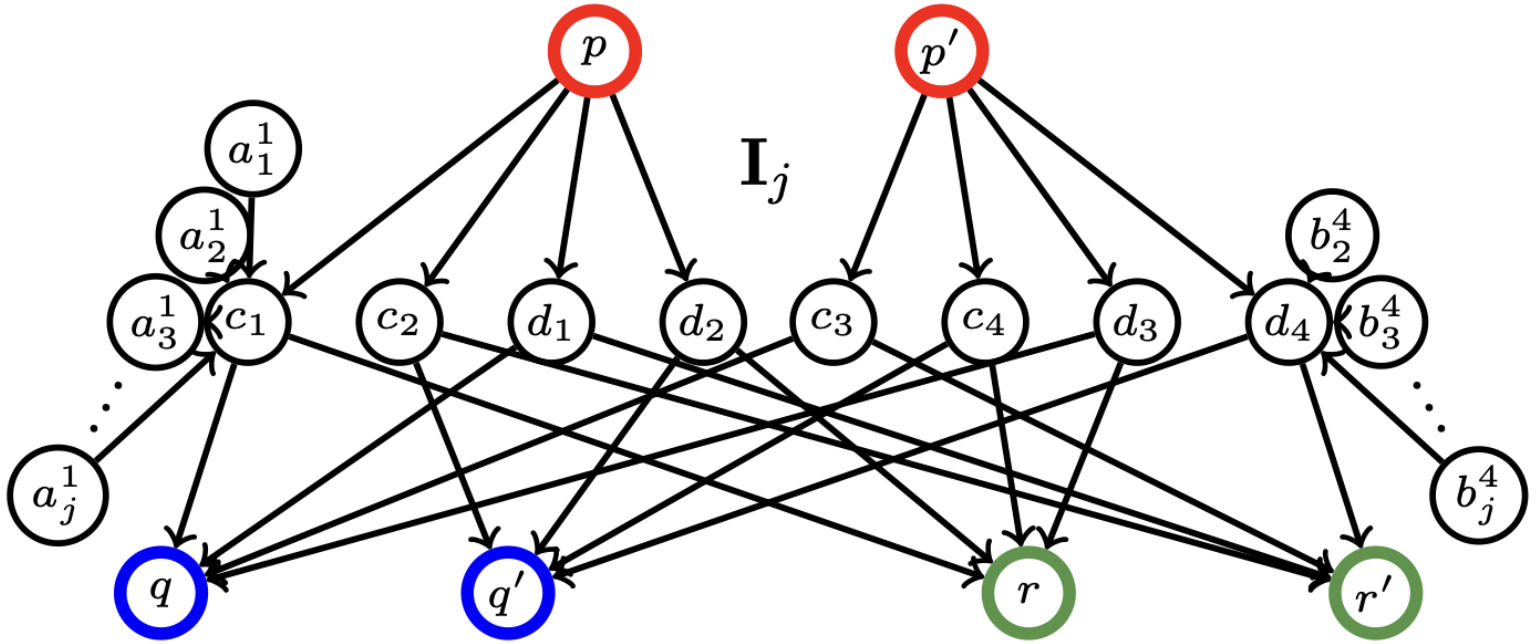
- In 1998 Pazolli showed that, given two **unordered** structures $\{A, B\}$, it is **NP-hard** to decide **WHO WINS** an **Ehrefeucht-Fraïssé game** on these structures.

- In a paper we have just submitted to CSL, we have shown the analogous question for the **MS-game** is **PSPACE-hard**. The reduction is from the PSPACE-hardness of approximating Quantified SAT to within a particular constant c, for $0 < c < 1$.

- The analogous questions for **ordered** structures remain open and appear to be considerably harder (but enticing!).

A key Gadget in these Hardness Arguments

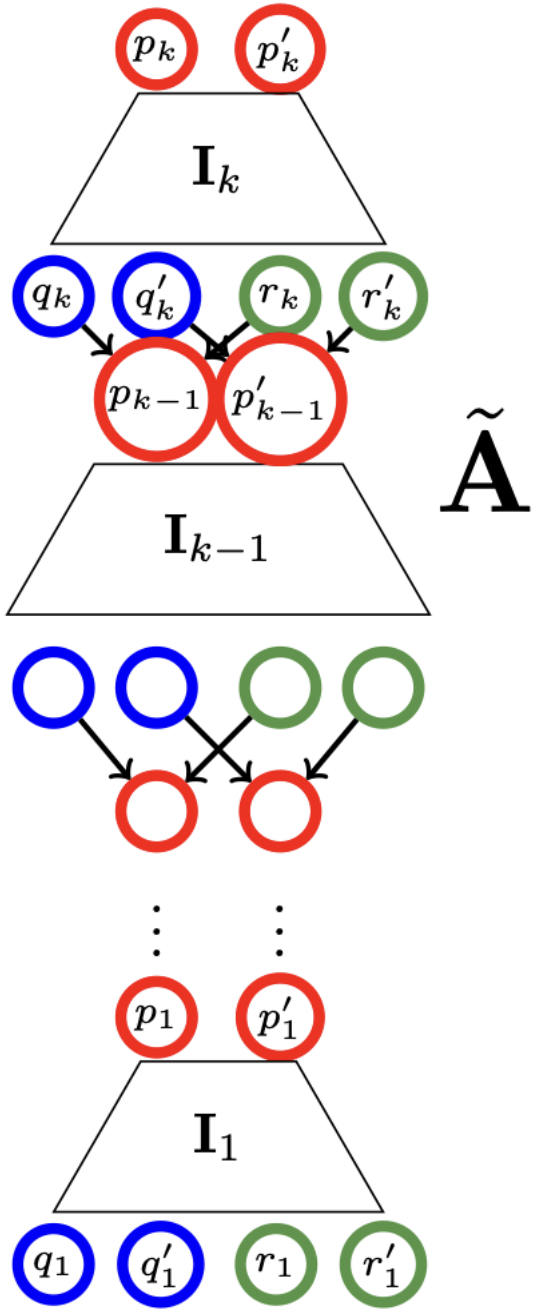
Gadgets by Pazolli and us are closely related to one devised by Cai, Fürer and Immerman to show that Fixed Point Logic + Counting does NOT capture PTIME:



A Stack of Gadgets

Both the Pazolli argument and our arguments (including a refinement of Pazolli's original argument) use a stack of these gadgets.

Can we create software that will verify the game-based arguments we make about such complicated gadgets/stacks of gadgets?

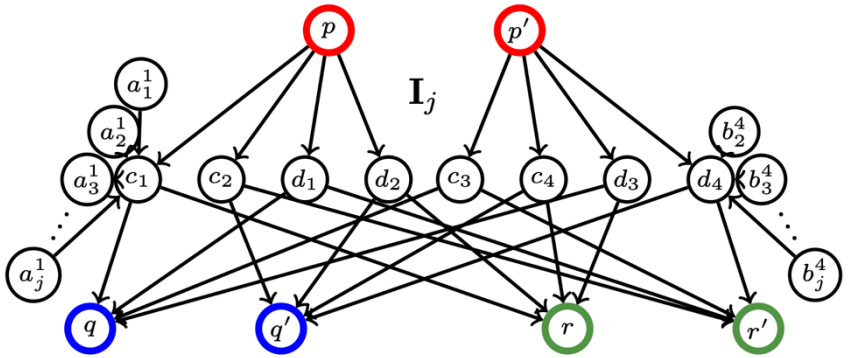


Software to help with the Analysis of the Games – the next frontier?

- We have developed software that can analyze EF-games on reasonably complicated graphs, demonstrating optimal Spoiler and Duplicator play.
- The software works for arbitrary ordered or unordered graphs. Graphs can be directed, undirected, contain loops, constants, and so on.
- Does not yet utilize symmetries (elements with the same k-types) to reduce search trees. Thus, look-ahead analysis only terminates for moderate-sized boards and modest numbers of rounds.

Software to help with the Analysis of the Games – the next frontier?

- We have a notion of an “iconized group”, capturing a subgraph that can be replicated throughout a bigger graph, but we cannot yet quite add the CFI-like gadgets.
- These are really iconized groups with specialized input and output “terminals”
- Once the software is a bit more robust, we will extend it to MS-games, though creating an effective user interface, allowing for management of all the bifurcating boards, is challenging.



Software Demo

Thank you!