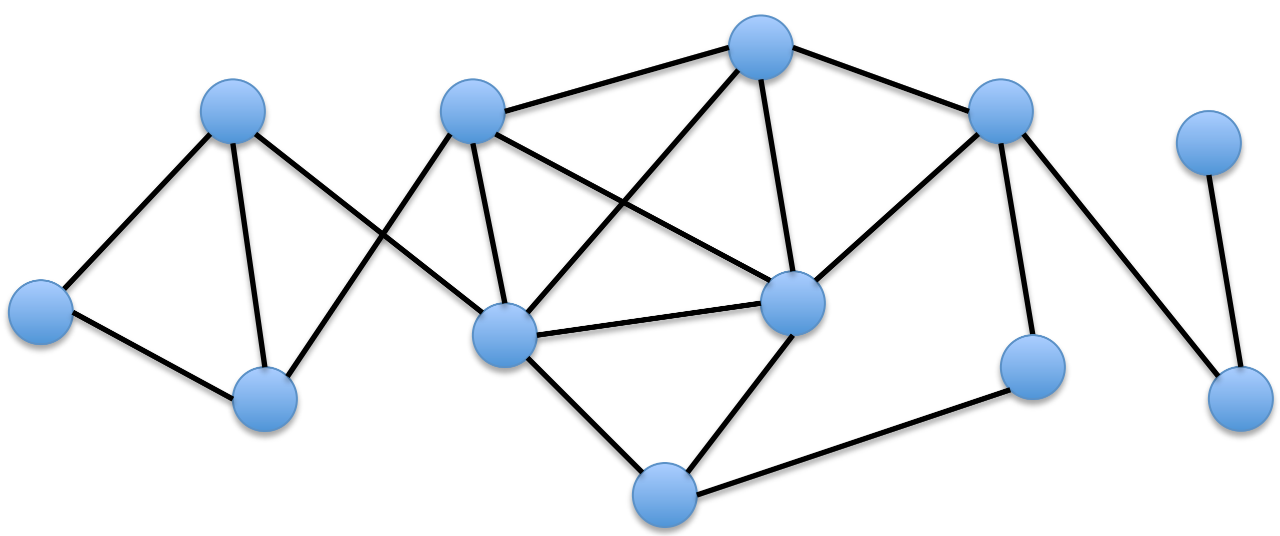
Ehrenfeucht-Fraïssé Games

Relational Structures

- We will consider language consisting of ***relation symbols*** only (no constant or function symbols).
- A **relational structure** $\mathbf{A} = (A, R_1, \ldots, R_t)$ consists of
  – a set A (called the universe of $\mathbf{A}$)
  – a sequence of relations $R_i \subseteq A^{r_i}$
- $\mathbf{A}$ and $\mathbf{B}$ are structures in the same relational language.

Graphs

- A **graph** $G = (V, \sim)$ consists of
  - a set V of vertices
  - a binary relation $\sim \subseteq V^2$ (anti-reflexive and symmetric)



First-Order Logic on Graphs

- **First-order formulas** are built from:

  atomic formulas

  $$x = y \qquad x \sim y$$

  via ***connectives***

  $$\neg\varphi \qquad \varphi \wedge \psi \qquad \varphi \vee \psi \qquad \varphi \rightarrow \psi$$

  and ***quantifiers***

  $$\forall x \;\varphi(x) \qquad \exists x \;\varphi(x)$$

First-O

adjacency relation

“there is an edge between x and y”

- **First-order formulas** …
  atomic formulas

  $$x = y \qquad x \sim y$$

  via ***connectives***

  $$\neg\varphi \qquad \varphi \wedge \psi \qquad \varphi \vee \psi \qquad \varphi \rightarrow \psi$$

  and ***quantifiers***

  $$\forall x\ \varphi(x) \qquad \exists x\ \varphi(x)$$

First-Order Logic on Graphs

- **First-order formulas** are built from:

atomic formulas

$$x = y \qquad x \sim y$$

via ***connectives***

$$\neg\varphi \qquad \varphi \wedge \psi \qquad \varphi \vee \psi \qquad \varphi \rightarrow \psi$$

and ***quantifiers***

$$\forall x\ \varphi(x) \qquad \exists x\ \varphi(x)$$

Variables range over ***vertices***

Definable and Axiomatizable Properties

Definability and Axiomatizability

- Let $\mathcal{C}$ be a class of graphs (i.e. a *graph property*)
- $\mathcal{C}$ is **FO-definable** if there is a single sentence φ such that

$$G \models \varphi \quad \Leftrightarrow \quad G \in \mathcal{C}$$

- $\mathcal{C}$ is **FO-axiomatizable** if there is a set of sentences Σ such that

$$G \models \Sigma \quad \Leftrightarrow \quad G \in \mathcal{C}$$

FO-Definable Graph Properties

- “no isolated vertex” (i.e. every vertex has a neighbor)

$$\forall x \exists y \; x \sim y$$

FO-Definable Graph Properties

- "no isolated vertex" (i.e. every vertex has a neighbor)

$$\forall x \exists y \ x \sim y$$

- "diameter is ≤ 2"

$$\forall x \forall y \big[x = y \vee x \sim y \vee \exists z (x \sim z \wedge z \sim y) \big]$$

FO-Definable Graph Properties

- “no isolated vertex” (i.e. every vertex has a neighbor)

$$\forall x \exists y \; x \sim y$$

- “diameter is ≤ 2”

$$\forall x \forall y \big[x = y \vee x \sim y \vee \exists z (x \sim z \wedge z \sim y) \big]$$

- “3-regular” (every vertex has exact 3 neighbors)

FO-Definable Graph Properties

- “no isolated vertex” (i.e. every vertex has a neighbor)

$$\forall x \exists y \; x \sim y$$

- “diameter is ≤ 2”

$$\forall x \forall y \left[x = y \vee x \sim y \vee \exists z (x \sim z \wedge z \sim y) \right]$$

- “3-regular” (every vertex has exact 3 neighbors)
- “girth > 17” (no induced cycle of length ≤ 17)

FO Definable Properties of Graphs

- But not every property of graphs is FO definable.
- For example, **3-colorability** is naturally expressed in ***second-order logic*** (allowing quantification over sets and relations). But cannot be expressed in first-order logic.

$$\exists R \exists B \exists G \Big(\forall x\, R(x) \vee B(x) \vee G(x) \Big) \wedge$$

$$\forall x \forall y \;\; Edge(x,y) \rightarrow \neg \left(\begin{array}{l} (R(x) \wedge R(y)) \vee \\ (B(x) \wedge B(y)) \vee \\ (G(x) \wedge G(y)) \end{array} \right)$$

FO Definable Properties of Graphs

- To show that a class $\mathcal{C}$ is first-order definable: simply write down a first-order formula that defines it.
- How can we show that a class $\mathcal{C}$ is *not* first-order definable?

Quantifier-Rank & k-Equivalence

Quantifier-rank

- **Quantifier-rank** of a formula is the maximum nesting depth of quantifiers

Quantifier-rank

- **Quantifier-rank** of a formula is the maximum nesting depth of quantifiers

- Rank-3 formula:

$$\exists x\,[\forall y\,[E(x,y) \vee \exists z\,[E(y,z) \wedge \neg(x = z)]]]$$

Quantifier-rank

- **Quantifier-rank** of a formula is the maximum nesting depth of quantifiers

- Inductive definition:

  $\text{rank}(x = y) = \text{rank}(R(x_1,\ldots,x_r)) = 0,$

  $\text{rank}(\neg\varphi) = \text{rank}(\varphi),$

  $\text{rank}(\varphi \wedge \psi) = \text{rank}(\varphi \vee \psi) = \max\{\text{rank}(\varphi), \text{rank}(\psi)\},$

  $\text{rank}(\forall x\ \varphi(x)) = \text{rank}(\exists x\ \varphi(x)) = \text{qr}(\varphi) + 1$

k-Equivalence

- Structures **A** and **B** are **k-equivalent** (denoted **A** $\equiv_k$ **B**) if they satisfy the same sentences of quantifier-rank k.
- In other words, **A** $\equiv_k$ **B** iff **A** and **B** cannot be distinguished by any first-order sentence of quantifier-rank k.

The Ehrenfeucht-Fraïssé Game

EF Game

- The **k-round Ehrenfeucht-Fraisse game** on structures **A** and **B** has two players, ***Spoiler*** and ***Duplicator***

EF Game

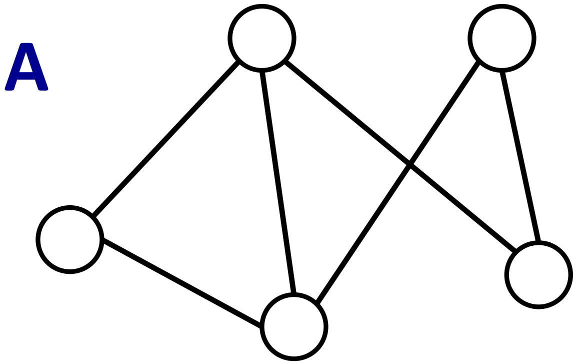
- The **k-round Ehrenfeucht-Fraisse game** on structures **A** and **B** has two players, ***Spoiler*** and ***Duplicator***
- The game captures the *quantifier-rank* needed to distinguish **A** and **B** in first-order logic

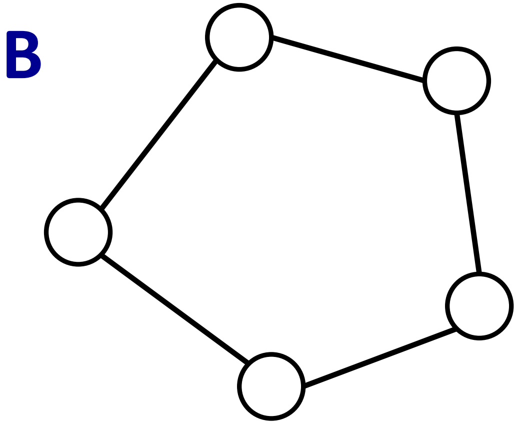
Duplicator's goal: prove $\mathbf{A} \equiv_k \mathbf{B}$

Spoiler's goal: refute $\mathbf{A} \equiv_k \mathbf{B}$

EF Game

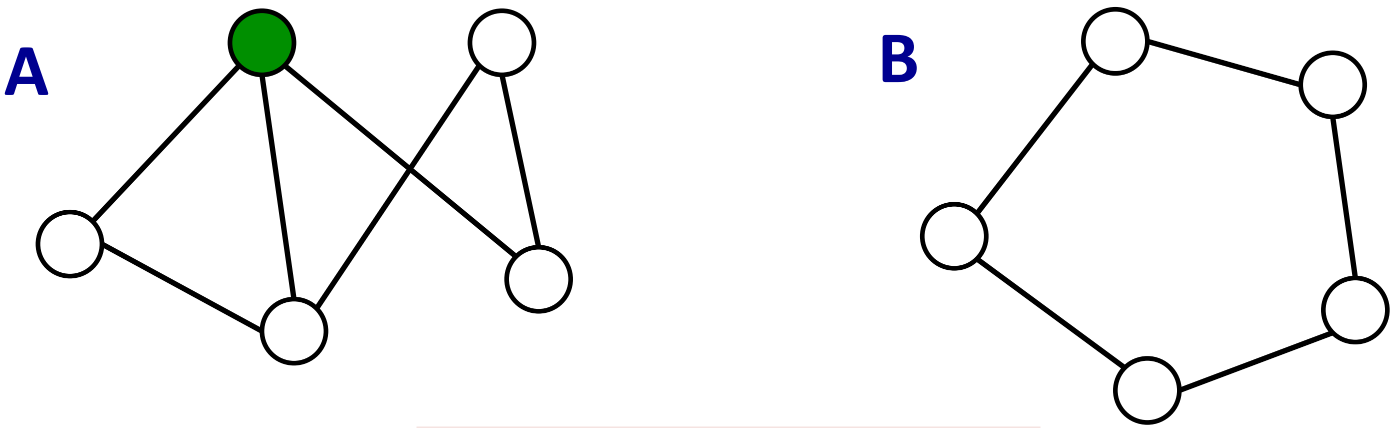
- <u>In each of k rounds:</u>
  1. ***Spoiler*** picks an element in either **A** or **B**,
  2. ***Duplicator*** picks an element in the other structure





EF Game

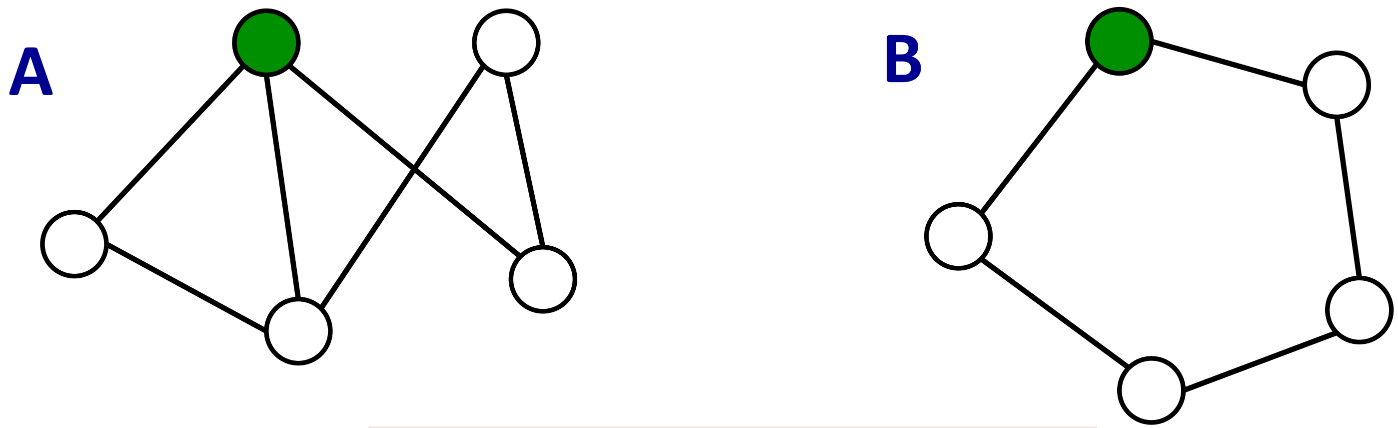
- <u>In each of k rounds:</u>
  1. ***Spoiler*** picks an element in either **A** or **B**,
  2. ***Duplicator*** picks an element in the other structure



Round 1: *Spoiler*

EF Game

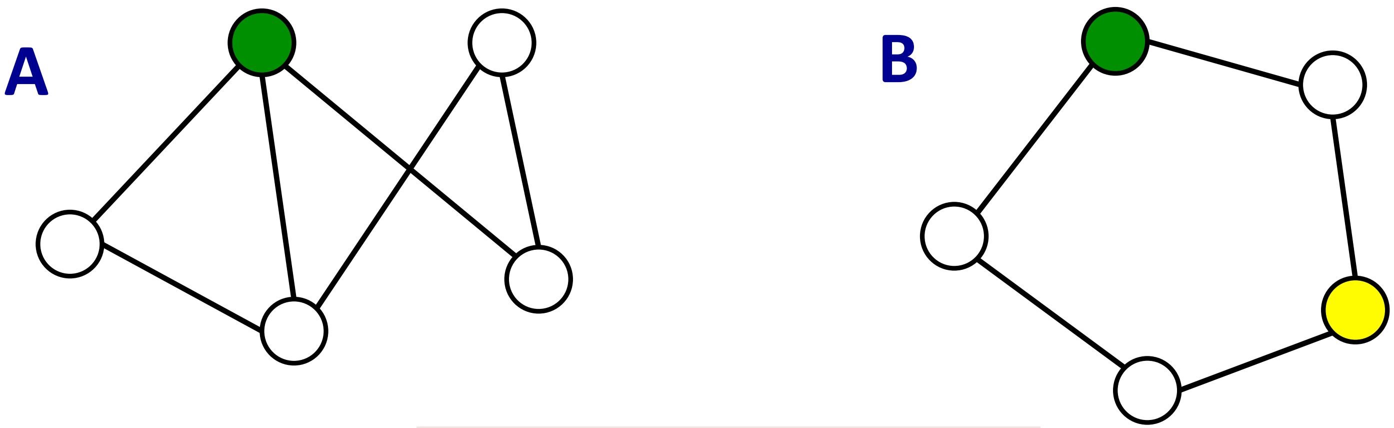
- <u>In each of k rounds:</u>
  1. ***Spoiler*** picks an element in either **A** or **B**,
  2. ***Duplicator*** picks an element in the other structure



Round 1: *Duplicator*

EF Game

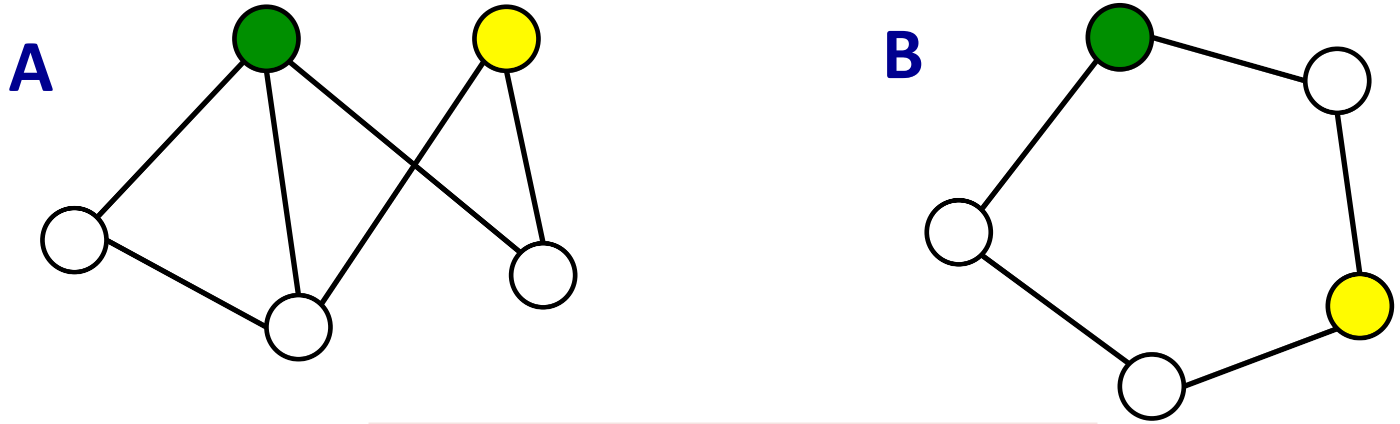
- <u>In each of k rounds:</u>
  1. ***Spoiler*** picks an element in either **A** or **B**,
  2. ***Duplicator*** picks an element in the other structure



Round 2: *Spoiler*

EF Game

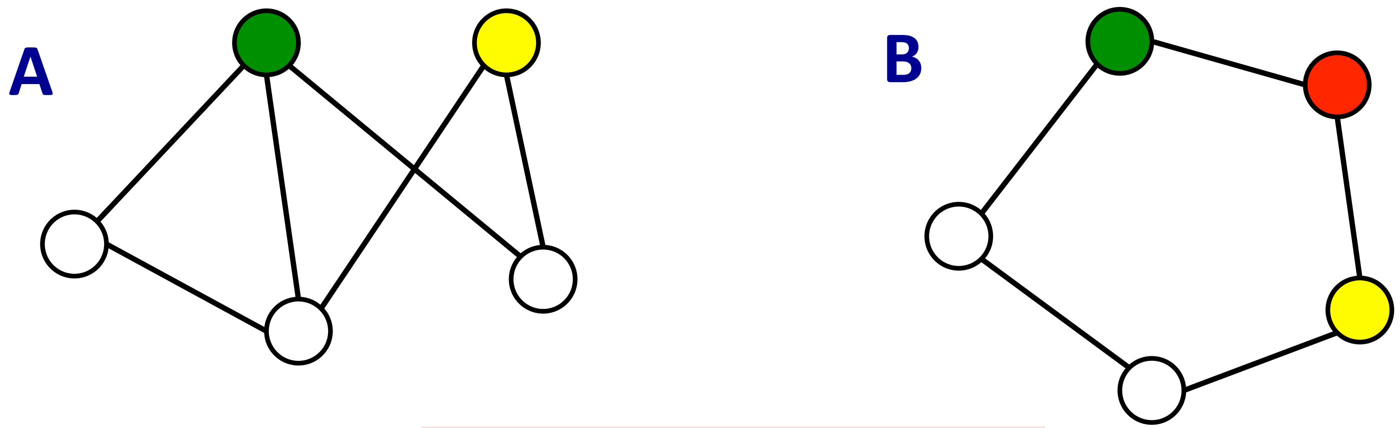
- <u>In each of k rounds:</u>
  1. ***Spoiler*** picks an element in either **A** or **B**,
  2. ***Duplicator*** picks an element in the other structure



Round 2: ***Duplicator***

EF Game

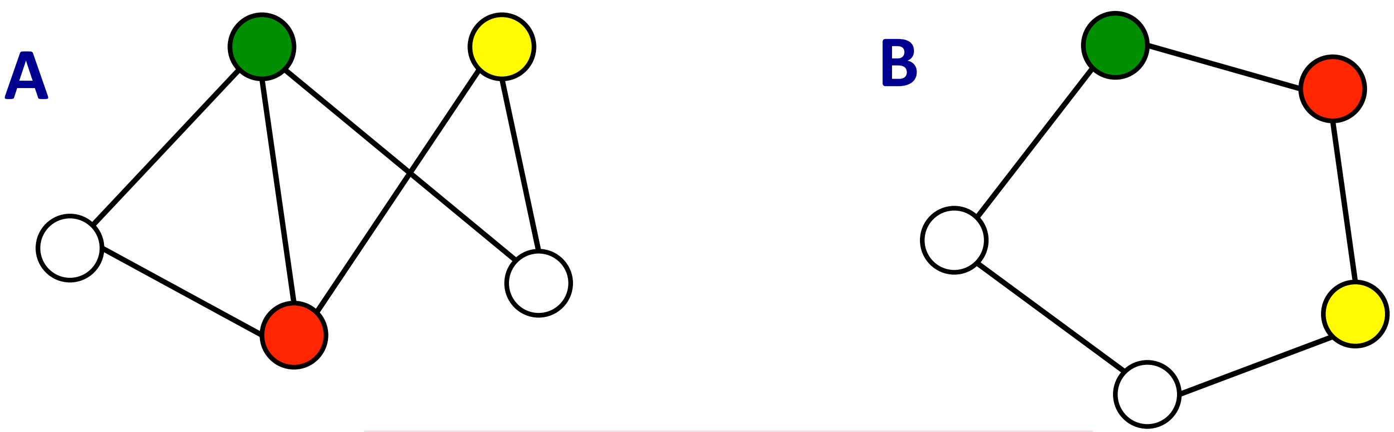
- <u>In each of k rounds:</u>
  1. ***Spoiler*** picks an element in either **A** or **B**,
  2. ***Duplicator*** picks an element in the other structure



Round 3: *Spoiler*

EF Game

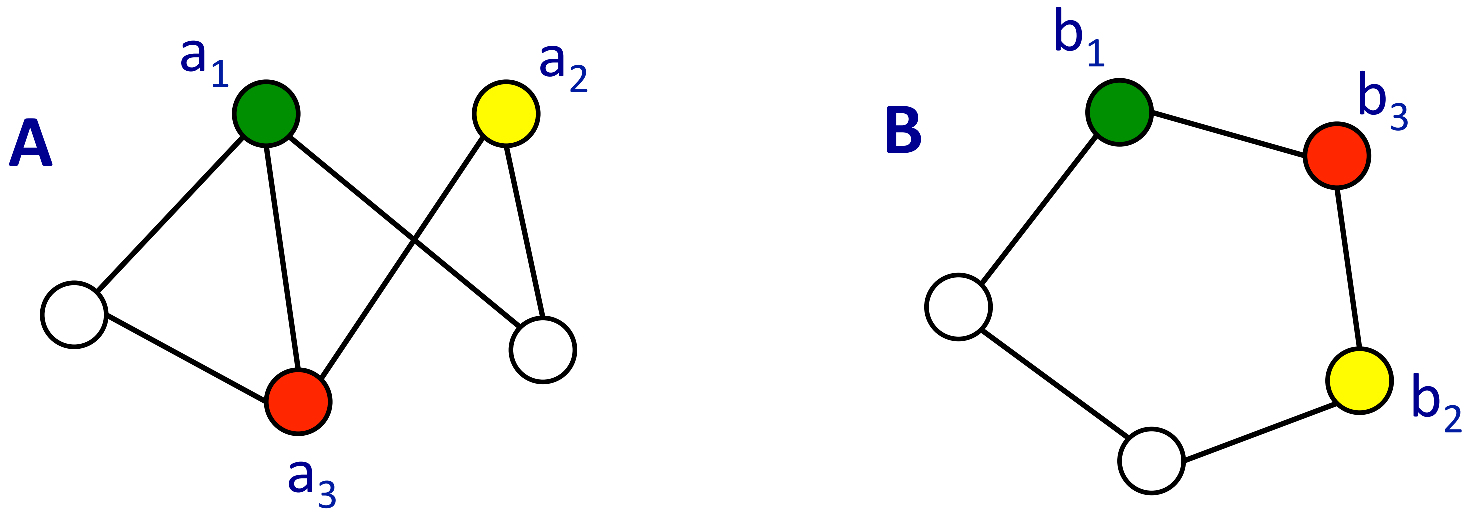
- <u>In each of k rounds:</u>
  1. ***Spoiler*** picks an element in either **A** or **B**,
  2. ***Duplicator*** picks an element in the other structure



Round 3: *Duplicator*

EF Game

- After k rounds: There are distinguished elements $a_1, \ldots, a_k$ in **A** and $b_1, \ldots, b_k$ in **B**

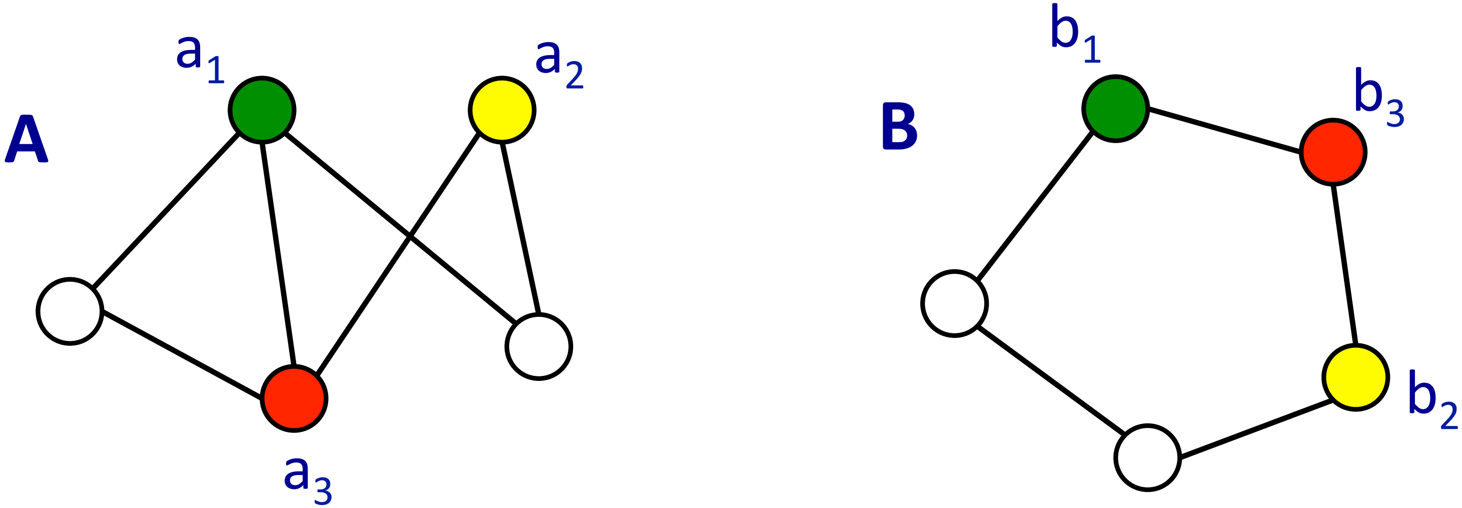


EF Game

- ***Duplicator*** is declared the winner iff

$$\{a_1 \mapsto b_1, \ldots, a_k \mapsto b_k\}$$

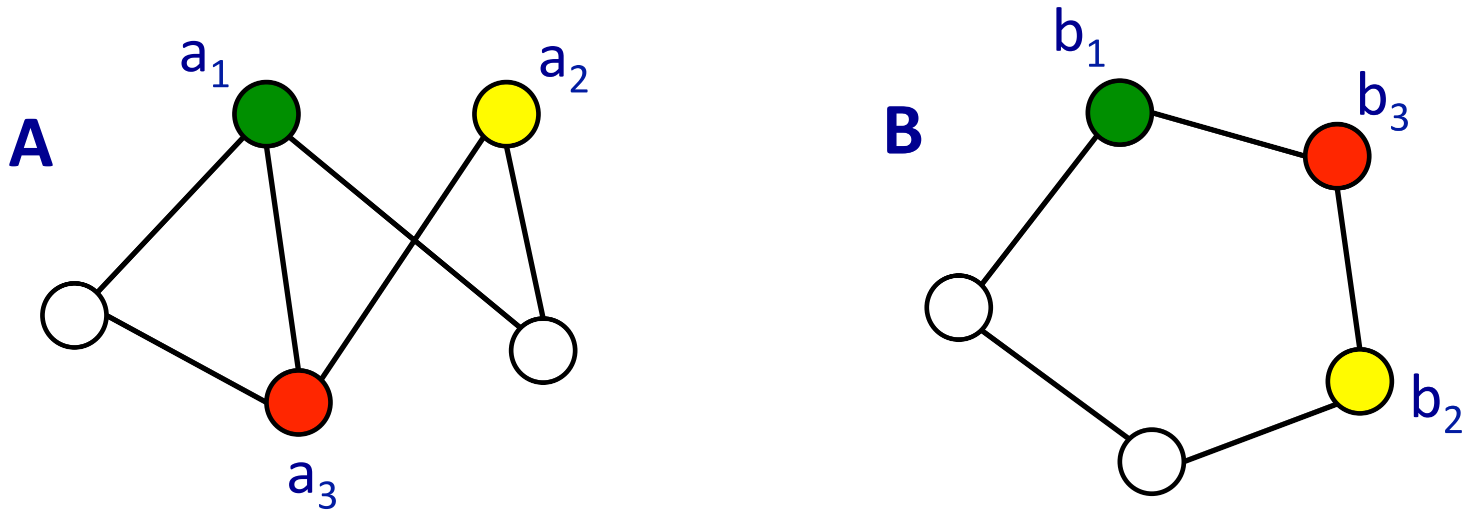
constitutes a **partial isomorphism** between **A** and **B**



EF Game

If **A** and **B** are graphs, ***partial isomorphism*** means:

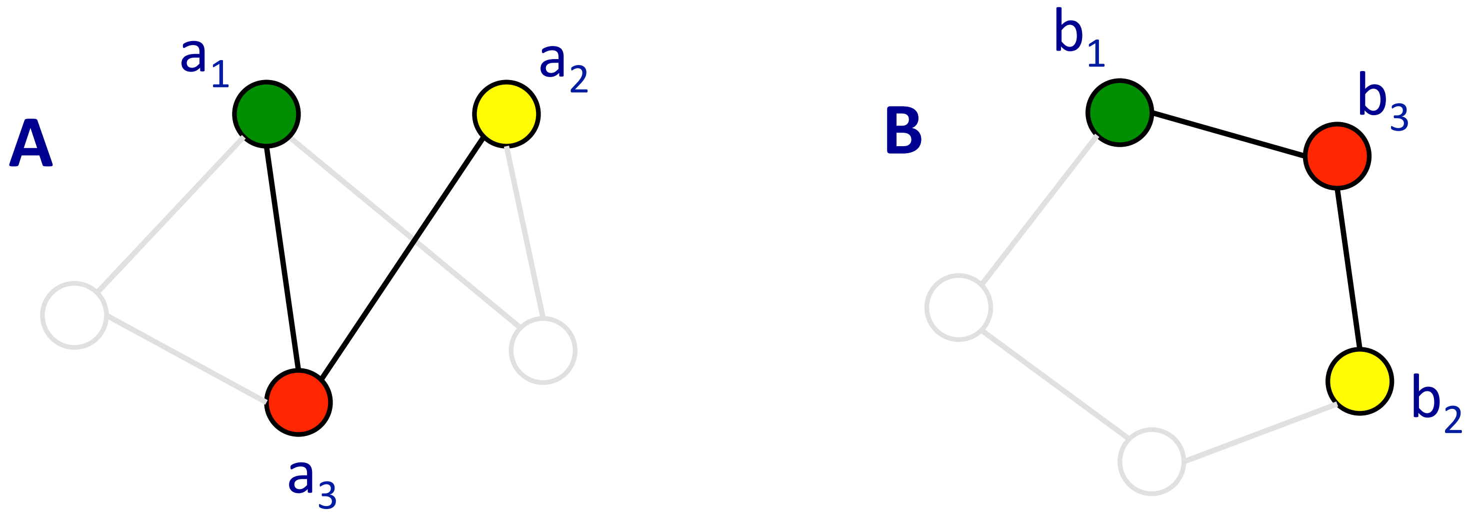
1. $a_i = a_j \iff b_i = b_j$
2. (a_i, a_j) is an edge of **A** $\iff$ (b_i, b_j) is an edge in **B**



EF Game

If **A** and **B** are graphs, ***partial isomorphism*** means:

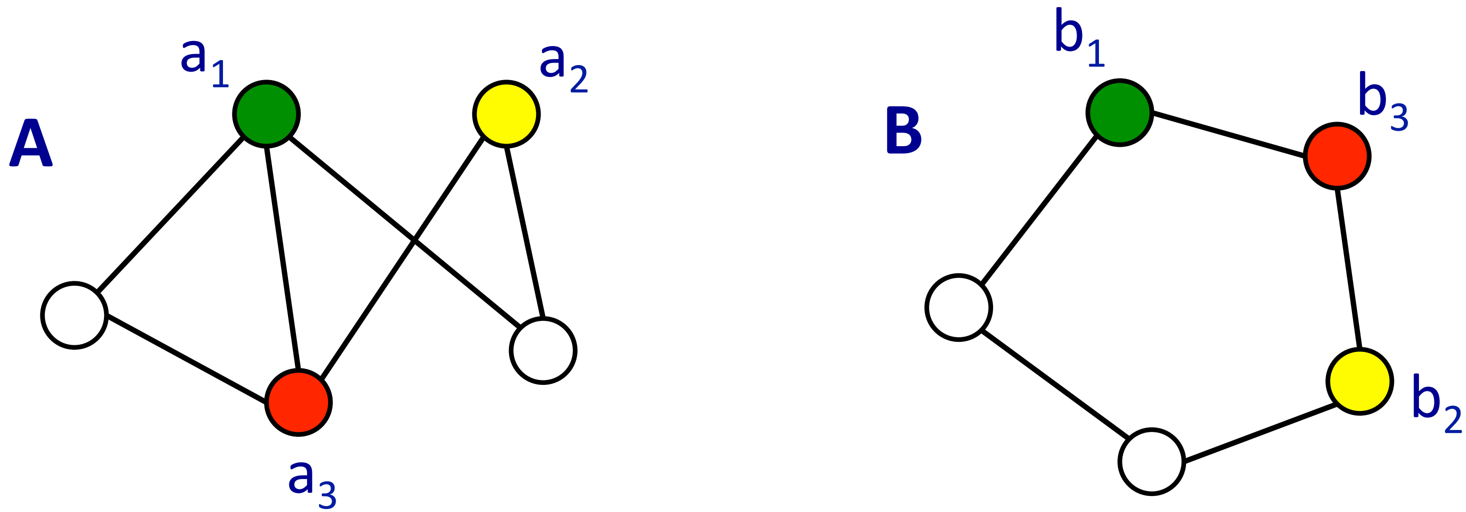
1. $a_i = a_j \iff b_i = b_j$
2. (a_i, a_j) is an edge of **A** $\iff$ (b_i, b_j) is an edge in **B**



EF Game

If **A** and **B** are graphs, ***partial isomorphism*** means:

1. $a_i = a_j \iff b_i = b_j$
2. (a_i, a_j) is an edge of **A** $\iff$ (b_i, b_j) is an edge in **B**

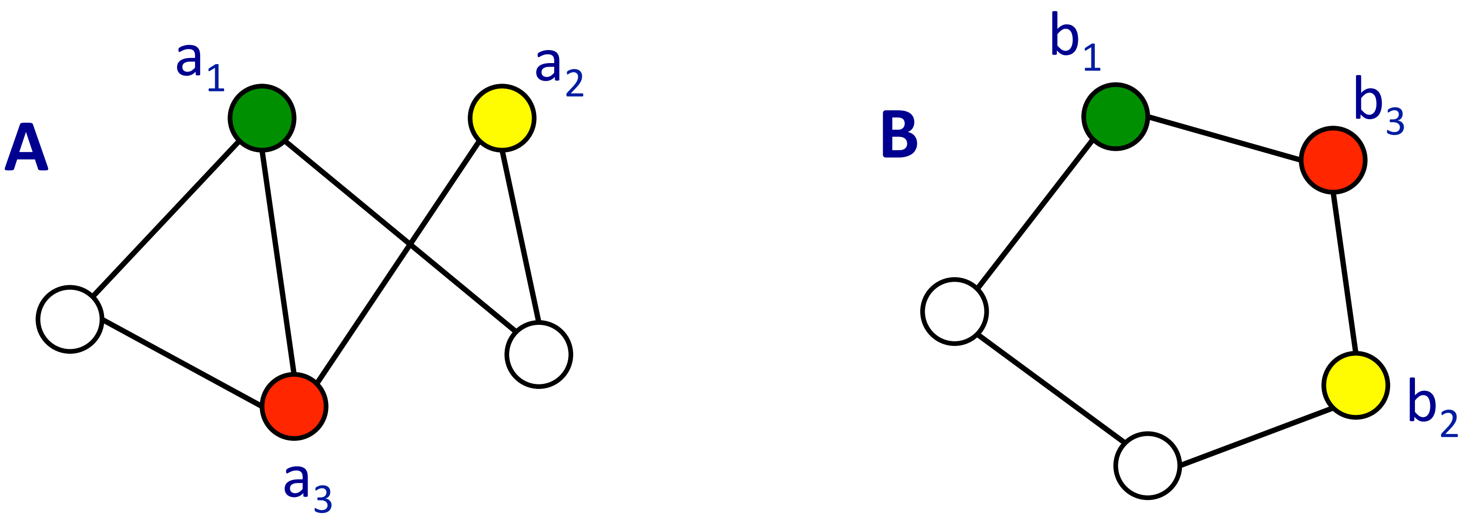


EF Game

- ***Duplicator*** is declared the winner iff

$$\{a_1 \mapsto b_1, \ldots, a_k \mapsto b_k\}$$

constitutes a **partial isomorphism** between **A** and **B**



EF Game

- ***Duplicator*** is declared the winner iff

  $$\{a_1 \mapsto b_1, \ldots, a_k \mapsto b_k\}$$

  constitutes a **partial isomorphism** between **A** and **B**

- ***Spoiler*** wins otherwise (iff $\{a_1 \mapsto b_1, \ldots, a_k \mapsto b_k\}$ is not a partial isomorphism)

EF Game

- ***Duplicator*** is declared the winner iff

  $\{a_1 \mapsto b_1, \ldots, a_k \mapsto b_k\}$

  constitutes a **partial isomorphism** between **A** and **B**

- ***Spoiler*** wins otherwise (iff $\{a_1 \mapsto b_1, \ldots, a_k \mapsto b_k\}$ is not a partial isomorphism)

- Fact: There exists a ***winning strategy*** for one of the two players.

EF Game

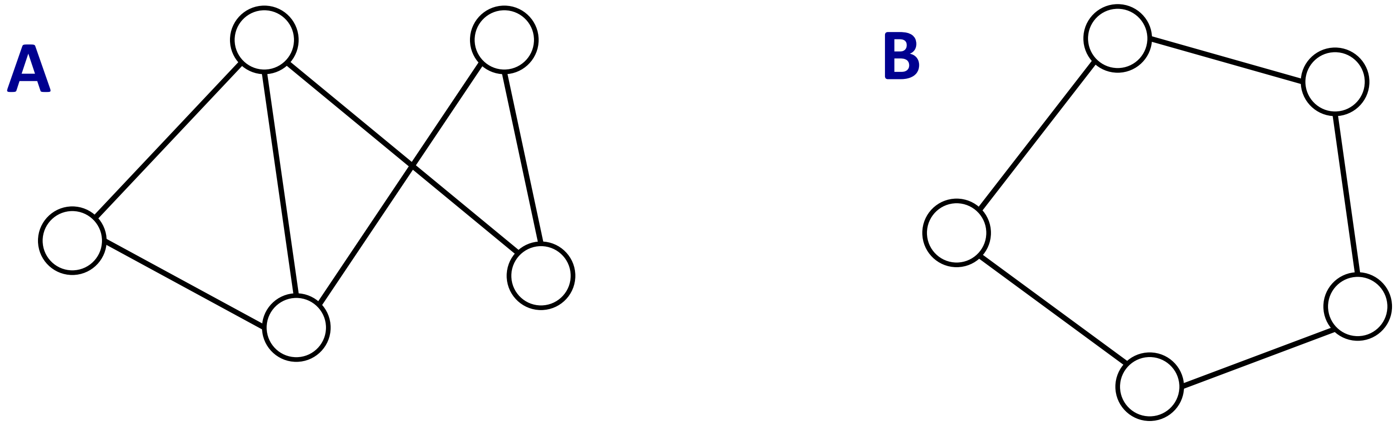
- ***Duplicator*** is de

> True of any deterministic zero-sum game of finite length.

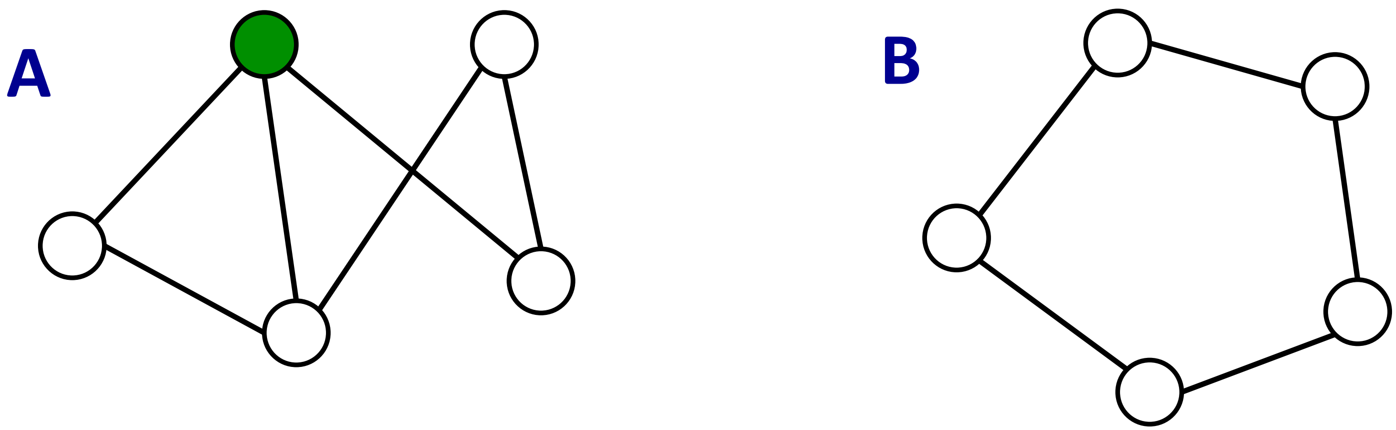
- ***Spo*** $b_k\}$ is not a partial i morphism)

- <u>Fact</u>: There exists a ***winning strategy*** for one of the two players.

- In our example, Spoiler has a winning strategy in the 3-round game:

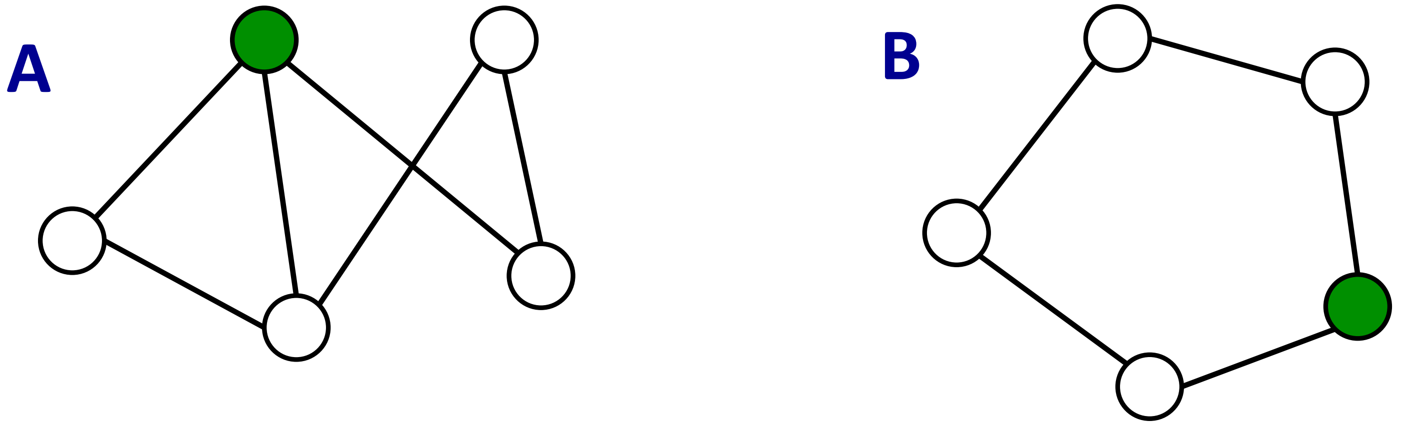


- In our example, Spoiler has a winning strategy in the 3-round game:



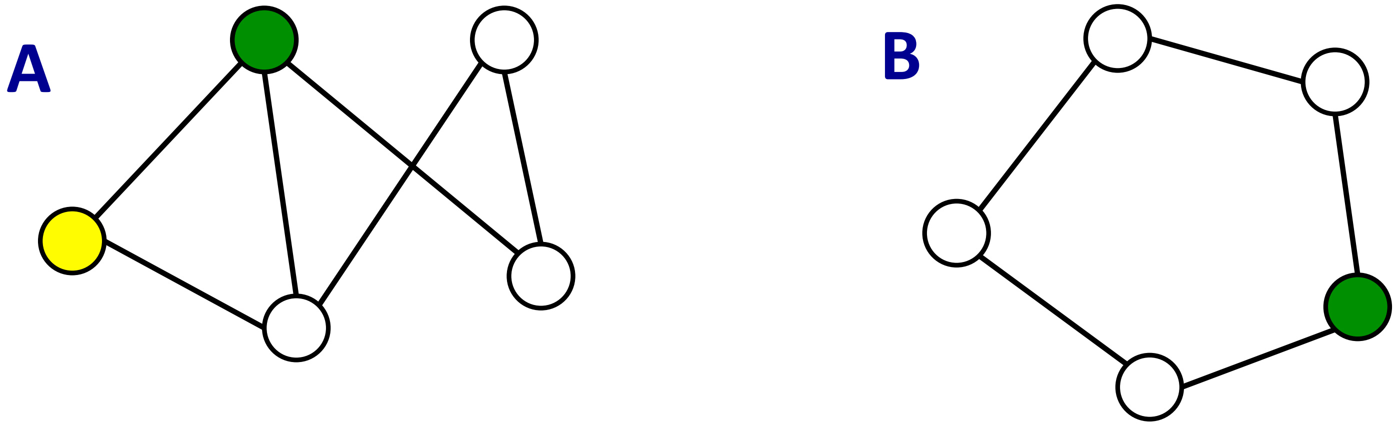
Round 1: ***Spoiler***

- In our example, Spoiler has a winning strategy in the 3-round game:



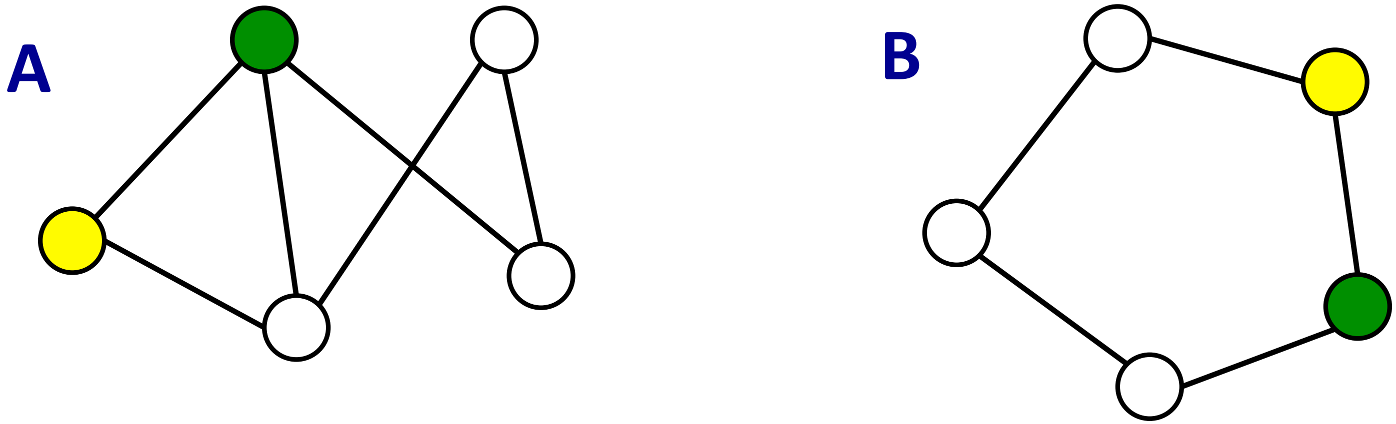
Round 1: ***Duplicator***

- In our example, Spoiler has a winning strategy in the 3-round game:



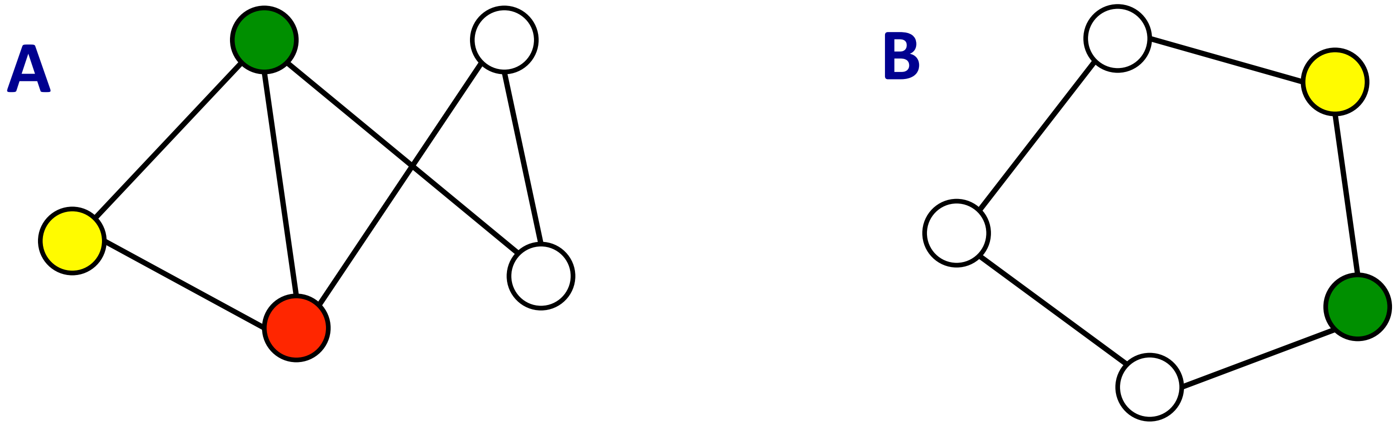
Round 2: *Spoiler*

- In our example, Spoiler has a winning strategy in the 3-round game:



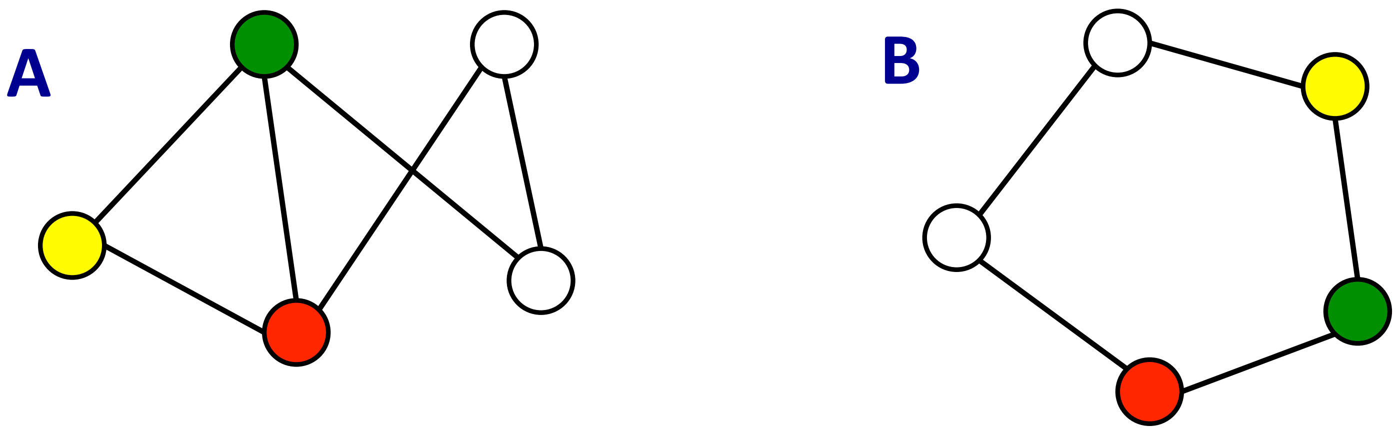
Round 2: ***Duplicator***

- In our example, Spoiler has a winning strategy in the 3-round game:



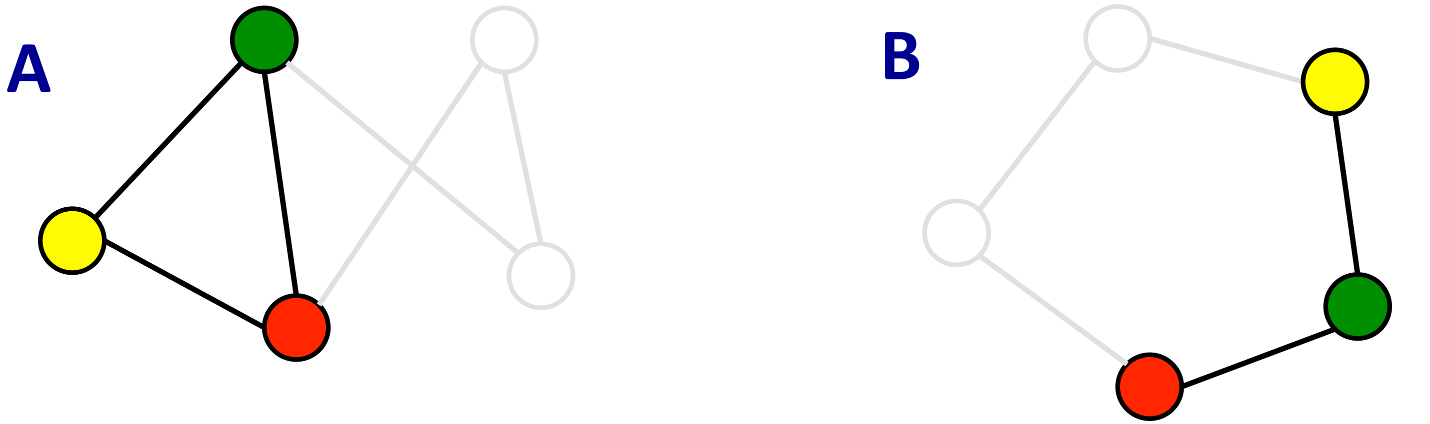
Round 2: ***Spoiler***

- In our example, Spoiler has a winning strategy in the 3-round game:



Round 3: ***Duplicator***

- In our example, Spoiler has a winning strategy in the 3-round game:



<u>not</u> a partial isomorphism: ***Spoiler*** wins

Fundamental Theorem of EF Games

Theorem

Duplicator has a winning strategy in the k-round EF game on **A** and **B** if, and only if, **A** $\equiv_k$ **B**.

Fundamental Theorem of EF Games

Theorem

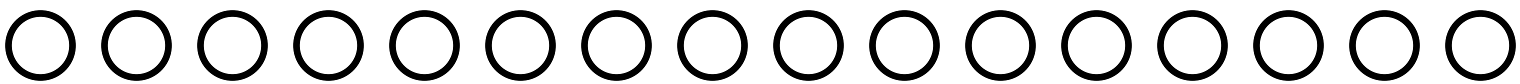
Duplicator has a winning strategy in the k-round EF game on **A** and **B** if, and only if, **A** $\equiv_k$ **B**.

- Proof by induction on k.

Repertoire of Winning Strategies

Linear orders

- Finite structures **A** with universe {1, …, n} and binary relation $<$



- We say **A** is EVEN if n is even

Linear orders

<u>Theorem</u>

The class of EVEN linear orders is *not* FO definable.

Linear orders

Theorem

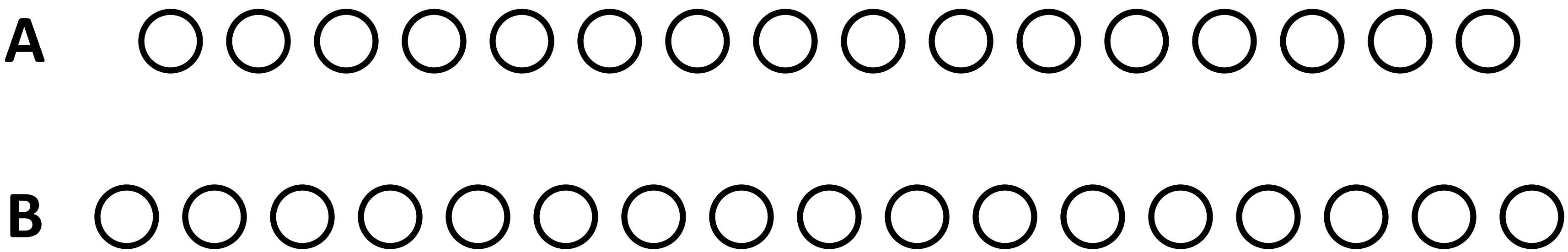
The class of EVEN linear orders is *not* FO definable.

Proof

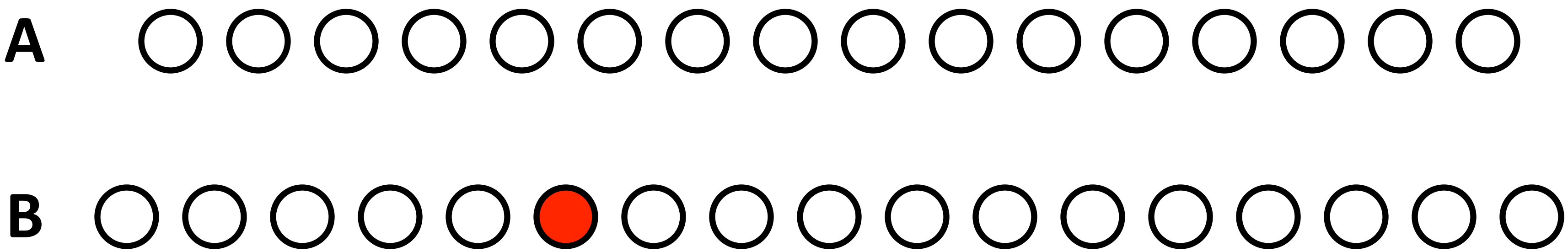
For arbitrary k, we show that **A** $\equiv_k$ **B** where **A** is a linear order of even size 2^k and **B** is a linear order of odd size $2^k + 1$.

We prove **A** $\equiv_k$ **B** by giving a winning strategy for ***Duplicator*** in the k-round EF game on **A** and **B**.

Winning strategy (by picture)

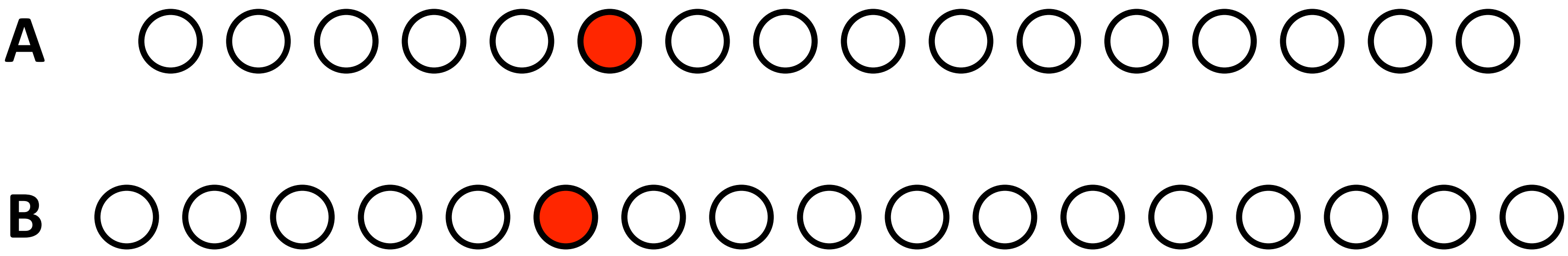


Winning strategy (by picture)



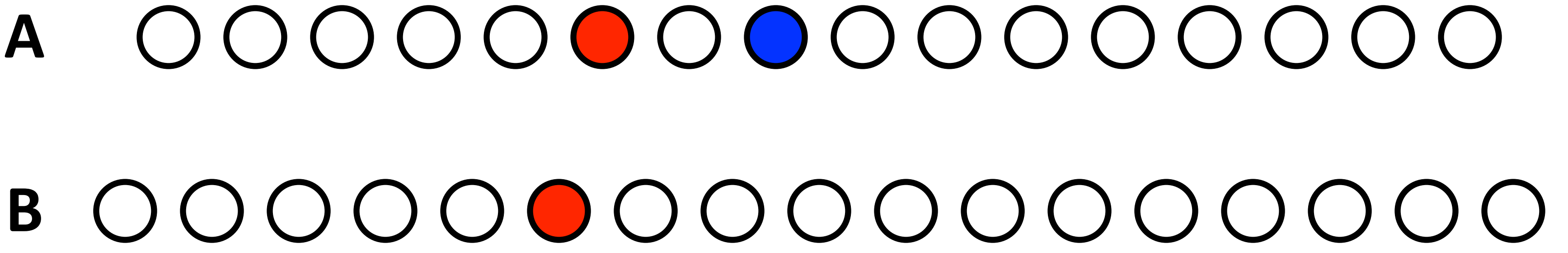
Round 1: ***Spoiler***

Winning strategy (by picture)



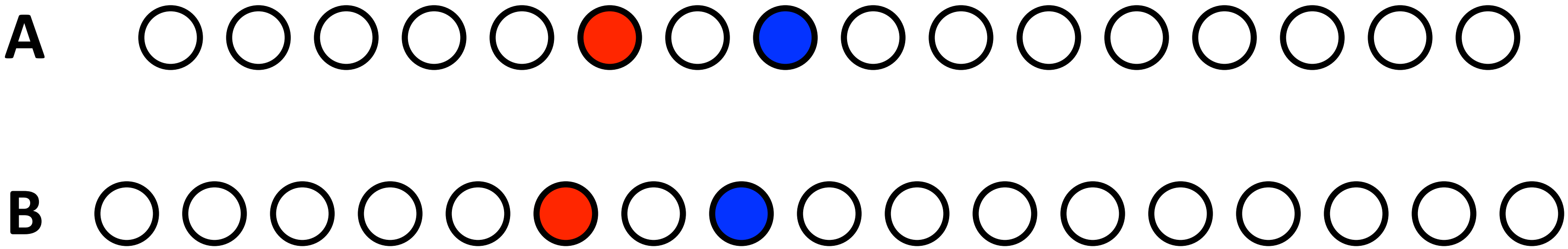
Round 1: ***Duplicator***

Winning strategy (by picture)



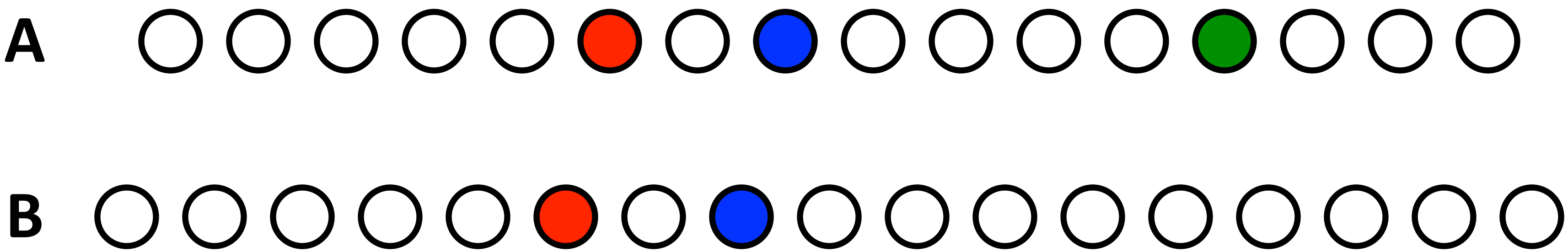
Round 2: ***Spoiler***

Winning strategy (by picture)



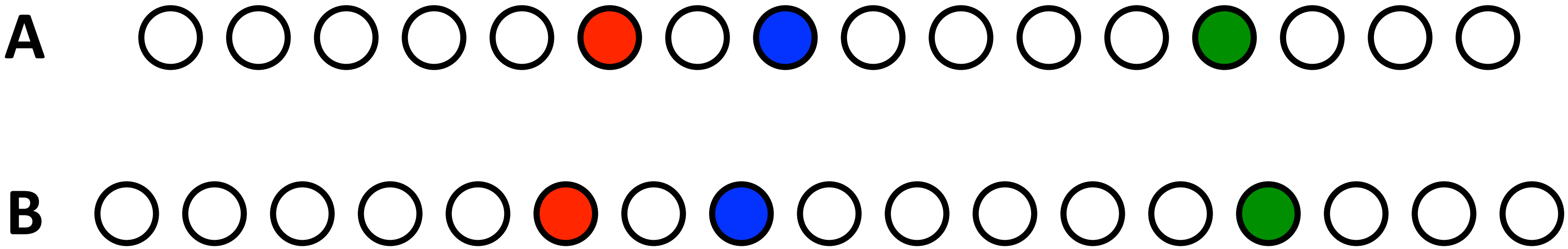
Round 2: ***Duplicator***

Winning strategy (by picture)



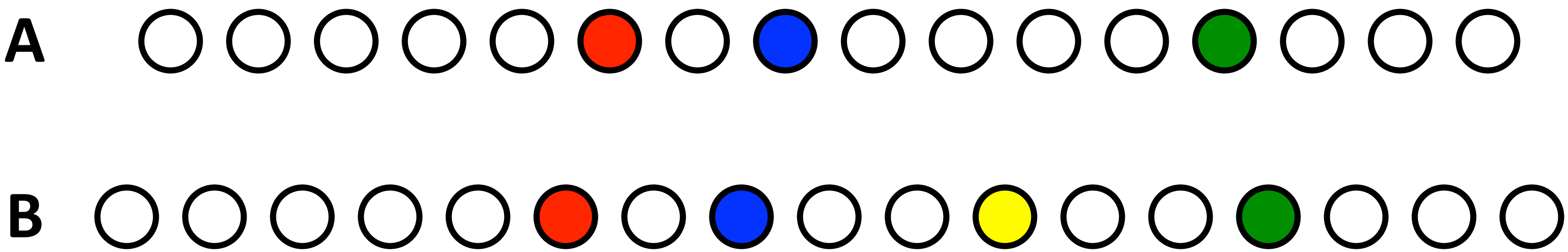
Round 3: ***Spoiler***

Winning strategy (by picture)



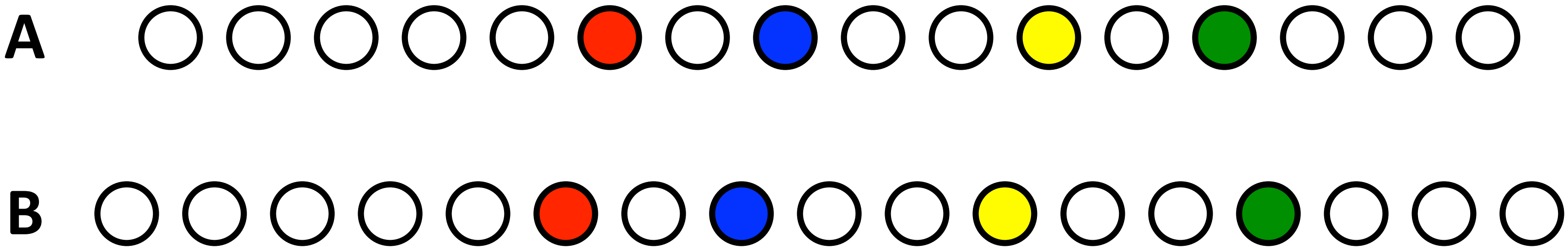
Round 3: ***Duplicator***

Winning strategy (by picture)



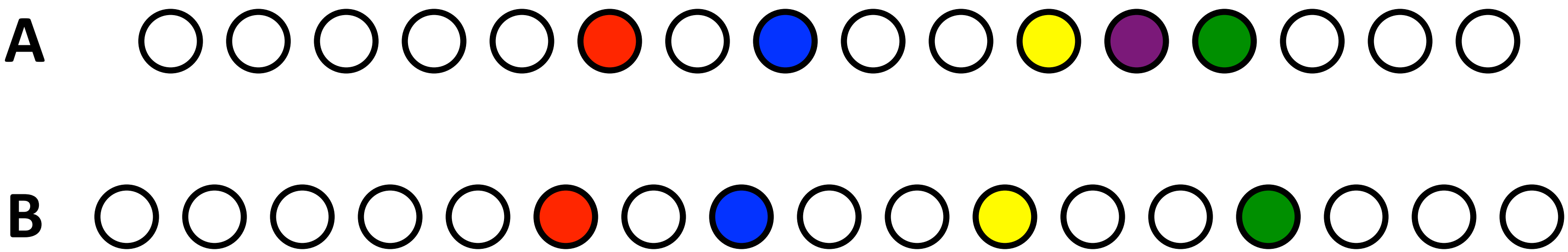
Round 4: ***Spoiler***

Winning strategy (by picture)



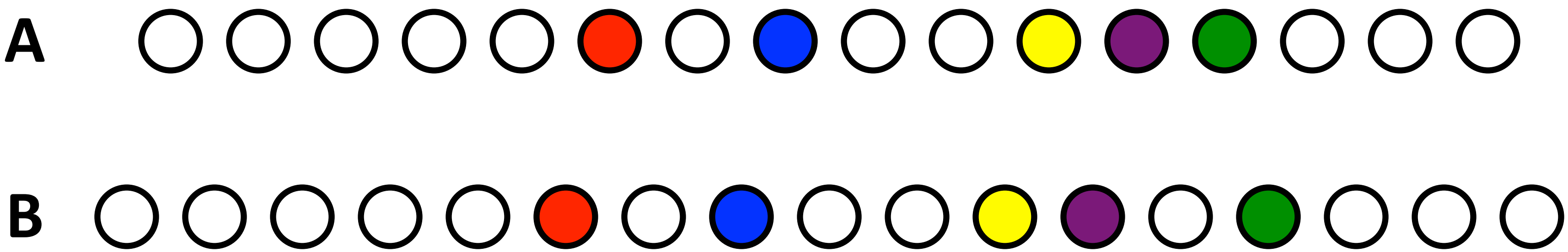
Round 4: ***Duplicator***

Winning strategy (by picture)



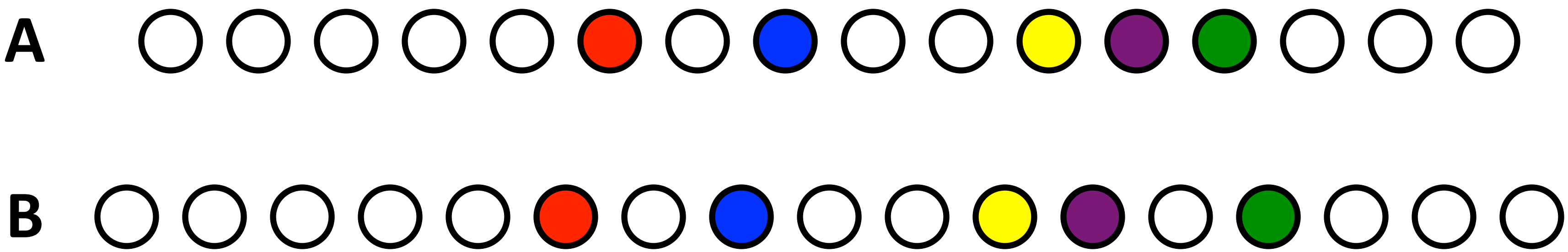
Round 5: ***Spoiler***

Winning strategy (by picture)



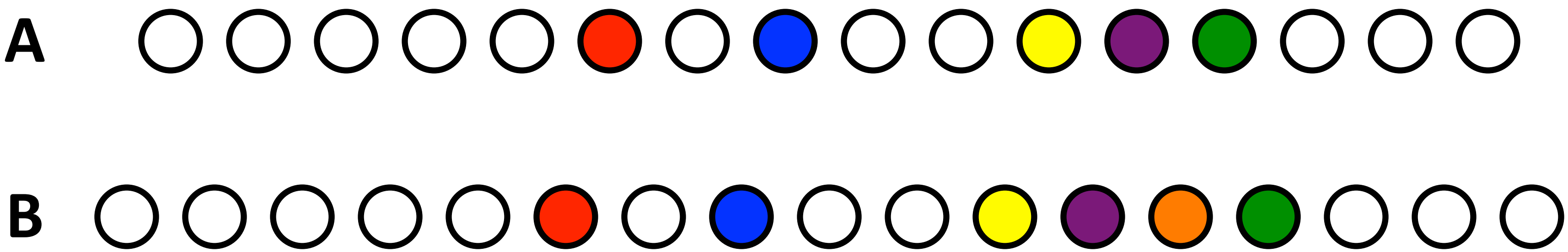
Round 5: ***Duplicator***

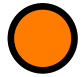
Winning strategy (by picture)



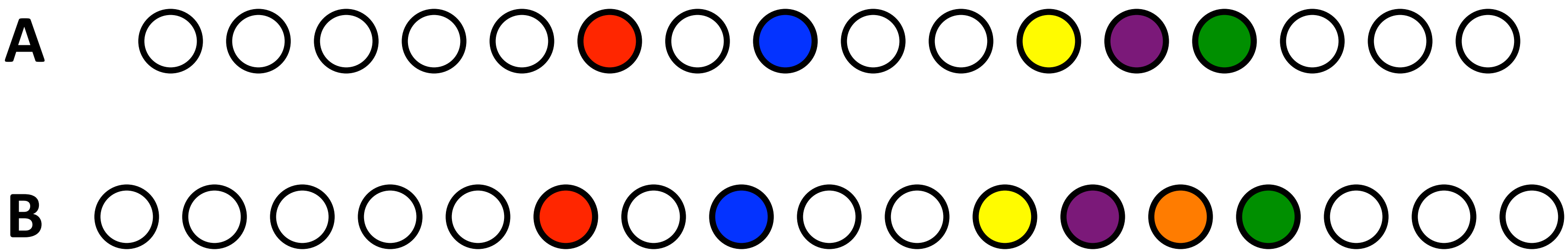
- So far, ***Duplicator*** is *winning* (i.e., $\{a_1 \mapsto b_1, \ldots, a_k \mapsto b_k\}$ is a partial isomorphism).

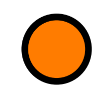
Winning strategy (by picture)



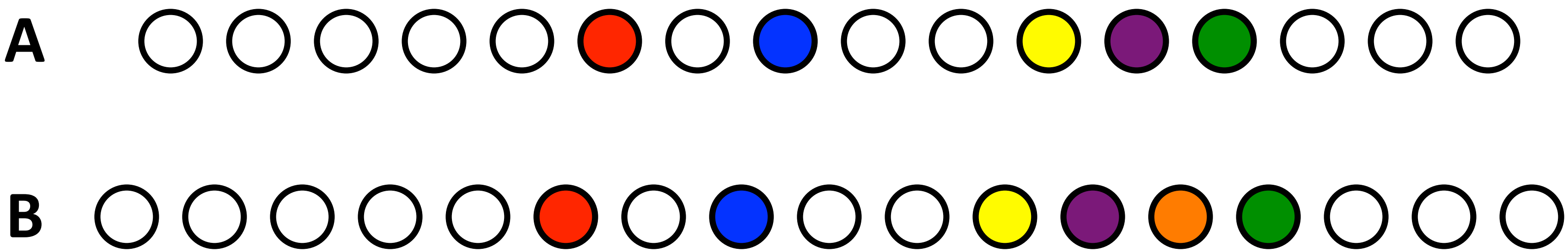
- So far, ***Duplicator*** is *winning* (i.e., $\{a_1 \mapsto b_1, \ldots, a_k \mapsto b_k\}$ is a partial isomorphism).
- However, ***Duplicator*** *loses* after ***Spoiler*** plays .

Winning strategy (by picture)



- So far, ***Duplicator*** is *winning* (i.e., $\{a_1 \mapsto b_1, \ldots, a_k \mapsto b_k\}$ is a partial isomorphism).
- However, ***Duplicator*** *loses* after ***Spoiler*** plays .

Winning strategy (by picture)



Duplicator's winning strategy:

in round j, preserve all distances between chosen elements up to 2^{k-j}

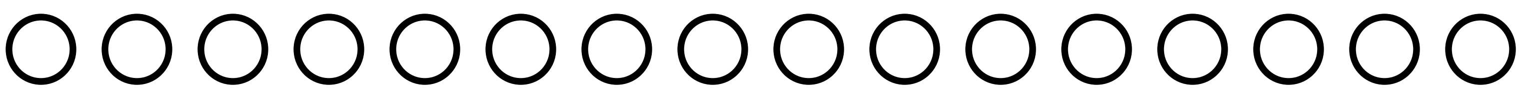
Connectivity

Corollary. GRAPH CONNECTIVITY is *not* FO definable

Connectivity

Corollary. GRAPH CONNECTIVITY is *not* FO definable

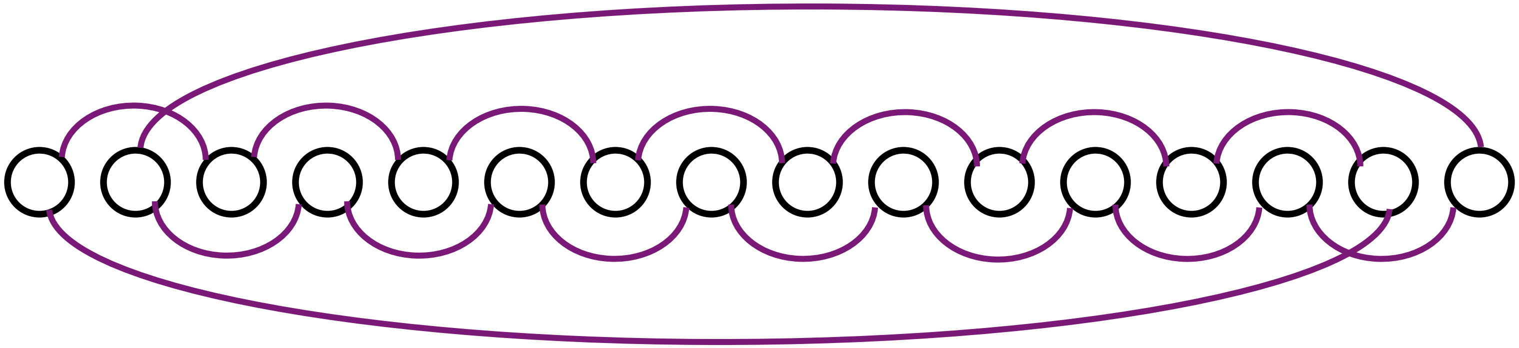
- If **A** is a linear order of size n, let G(**A**) be the graph with edges { i, i+2 mod n } for all a ∈ **A**



Connectivity

Corollary. GRAPH CONNECTIVITY is *not* FO definable

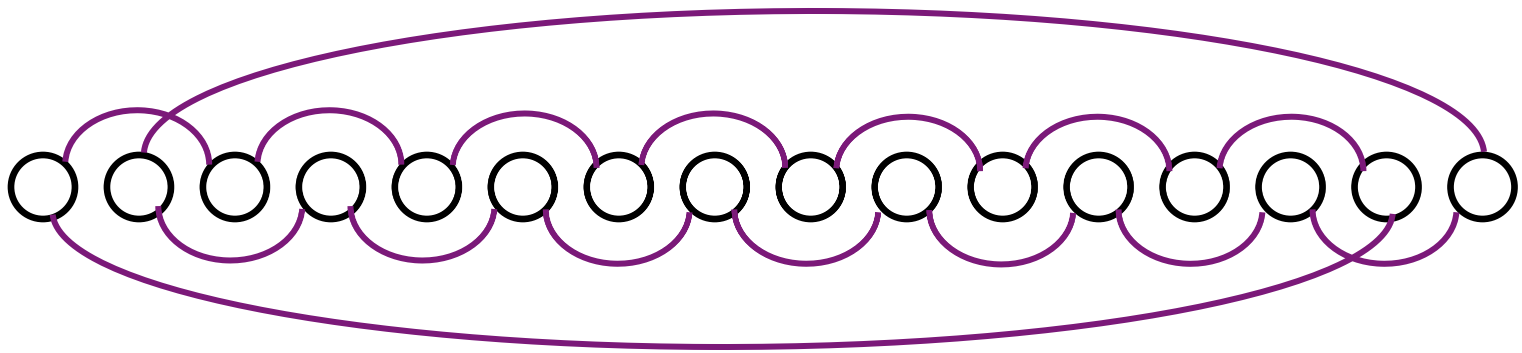
- If **A** is a linear order of size n, let G(**A**) be the graph with edges { i, i+2 mod n } for all a ∈ **A**



Obs 1: G(**A**) is connected if and only if n is odd

Corollary: … *not* FO definable

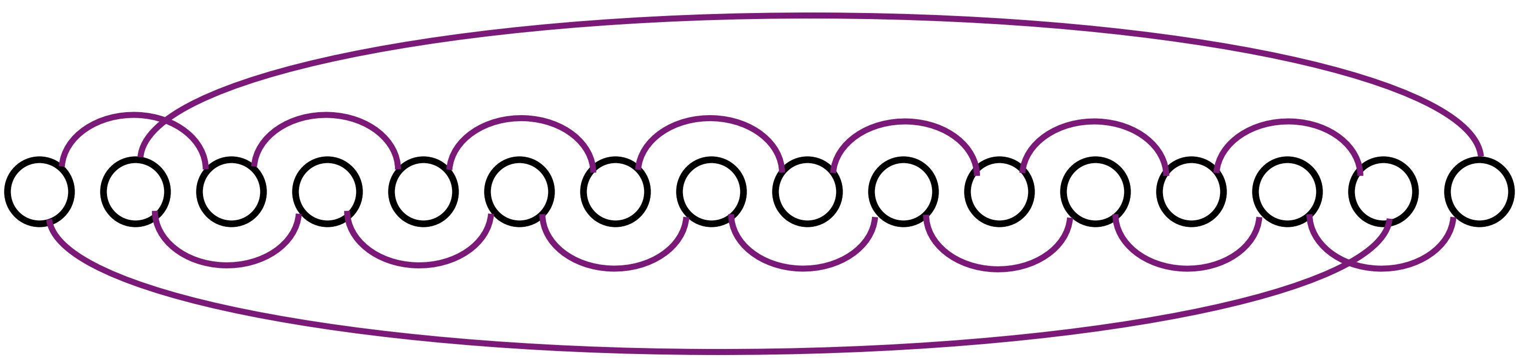
- If **A** is a linear order of size n, let G(**A**) be the graph with edges { i, i+2 mod n } for all a ∈ **A**



Obs 2: G(**A**) is first-order definable from **A**

Corollary: *not* FO definable

- If **A** is a linear order of size n, let G(**A**) be the graph with edges { i, i+2 mod n } for all a ∈ **A**



Connectivity

Corollary. GRAPH CONNECTIVITY is *not* FO definable

- If **A** is a linear order of size n, let G(**A**) be the graph with edges { i, i+2 mod n } for all a ∈ **A**

- If φ were a first-order formula defining GRAPH CONNECTIVITY, then by replacing each sub-formula E(x,y) with a formula "x and y have cyclic distance 2 in the linear order **A**", we could define EVENNESS of **A** (which we showed is impossible by the EF game).

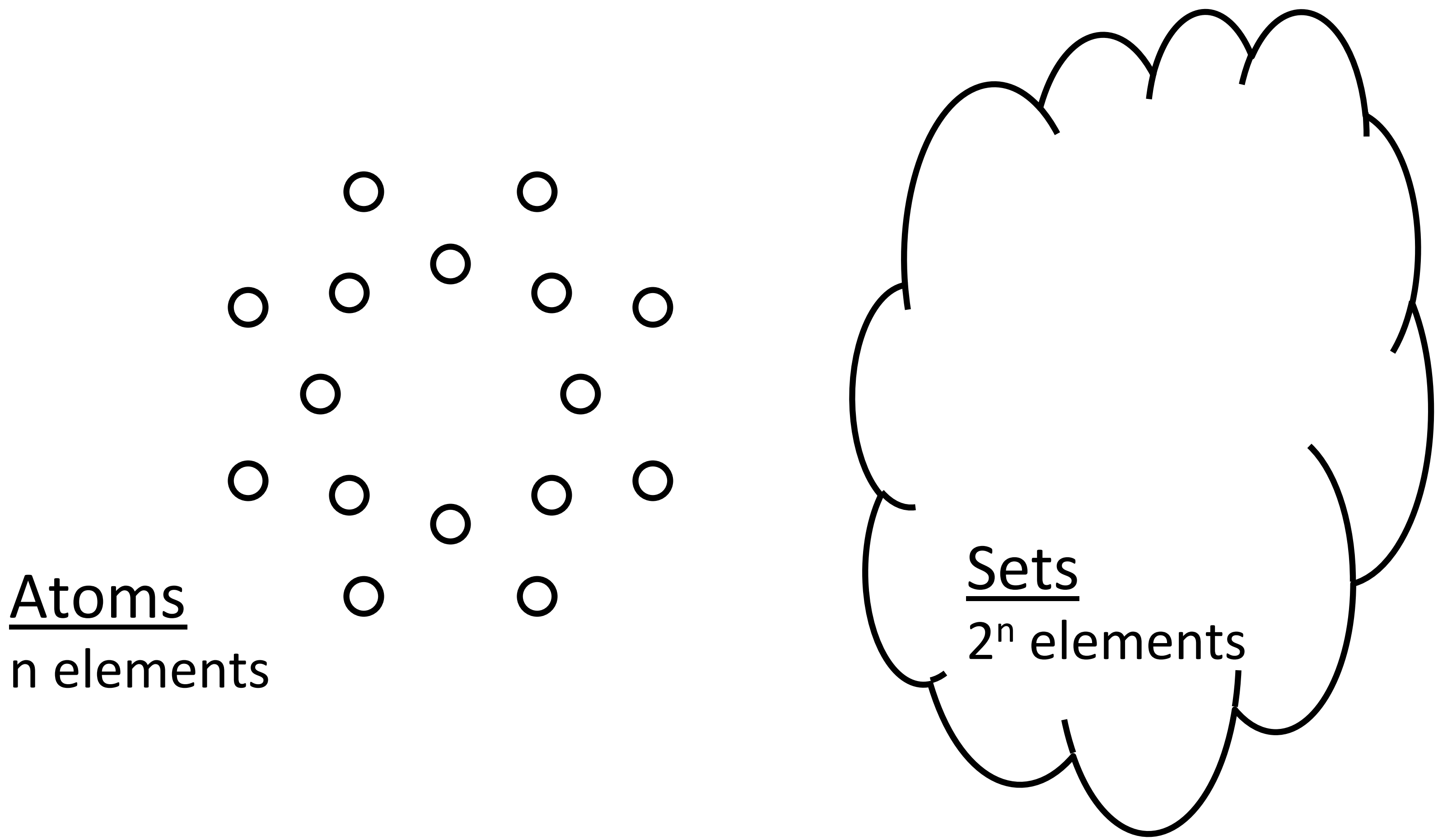
Connectivity

Corollary. GRAPH CONNECTIVITY is *not* FO definable

- This result can be proved directly by playing the EF game e.g. on graphs C_n and $C_n + C_n$
- The reduction to EVENNESS of linear orders illustrates the technique of a *first-order interpretations*.

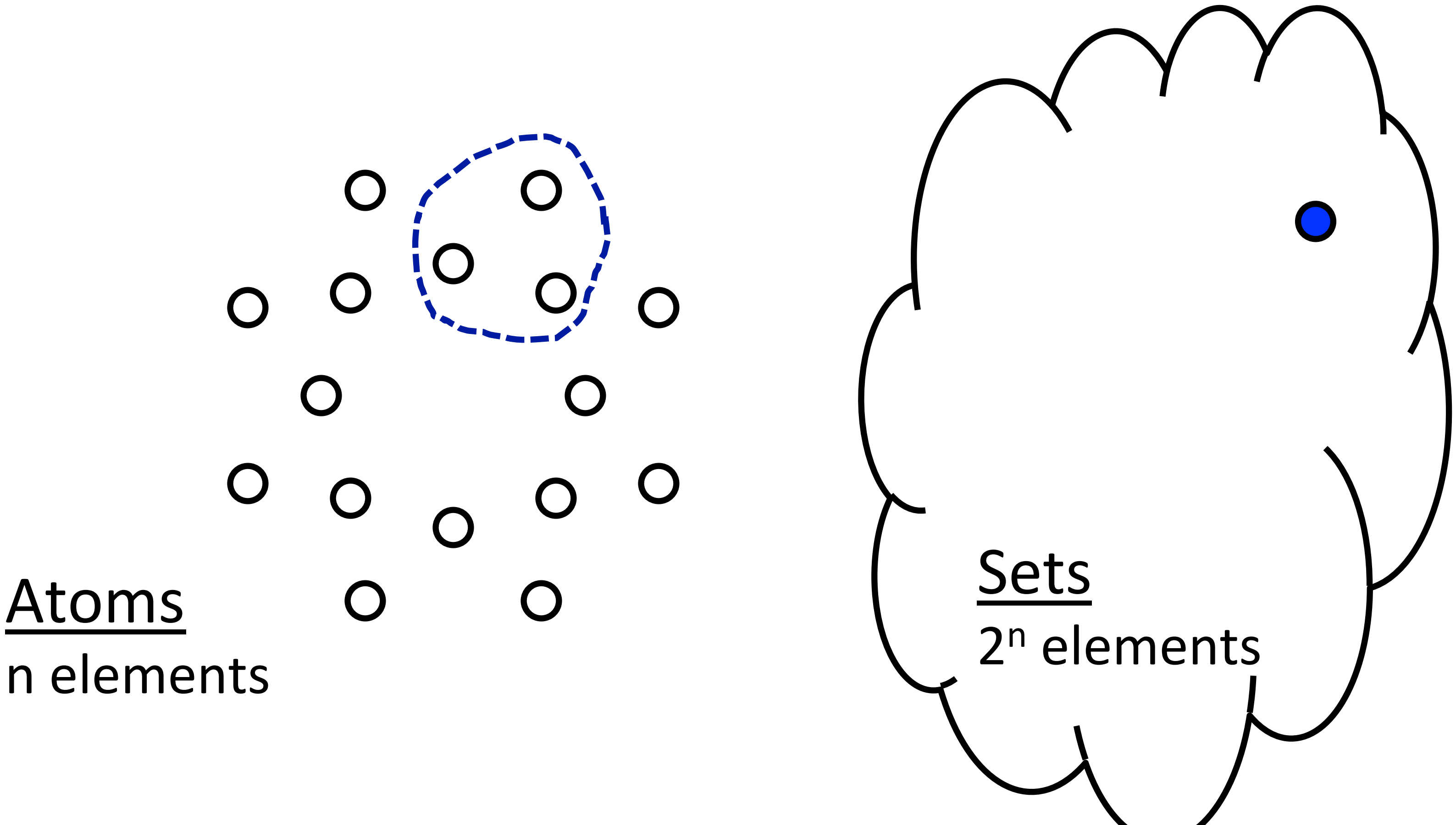
Set-powersets

- SetPow_n is the structure $([n] \cup 2^{[n]}, \underline{\text{Atoms}}, \underline{\text{Sets}}, \underline{\text{In}})$



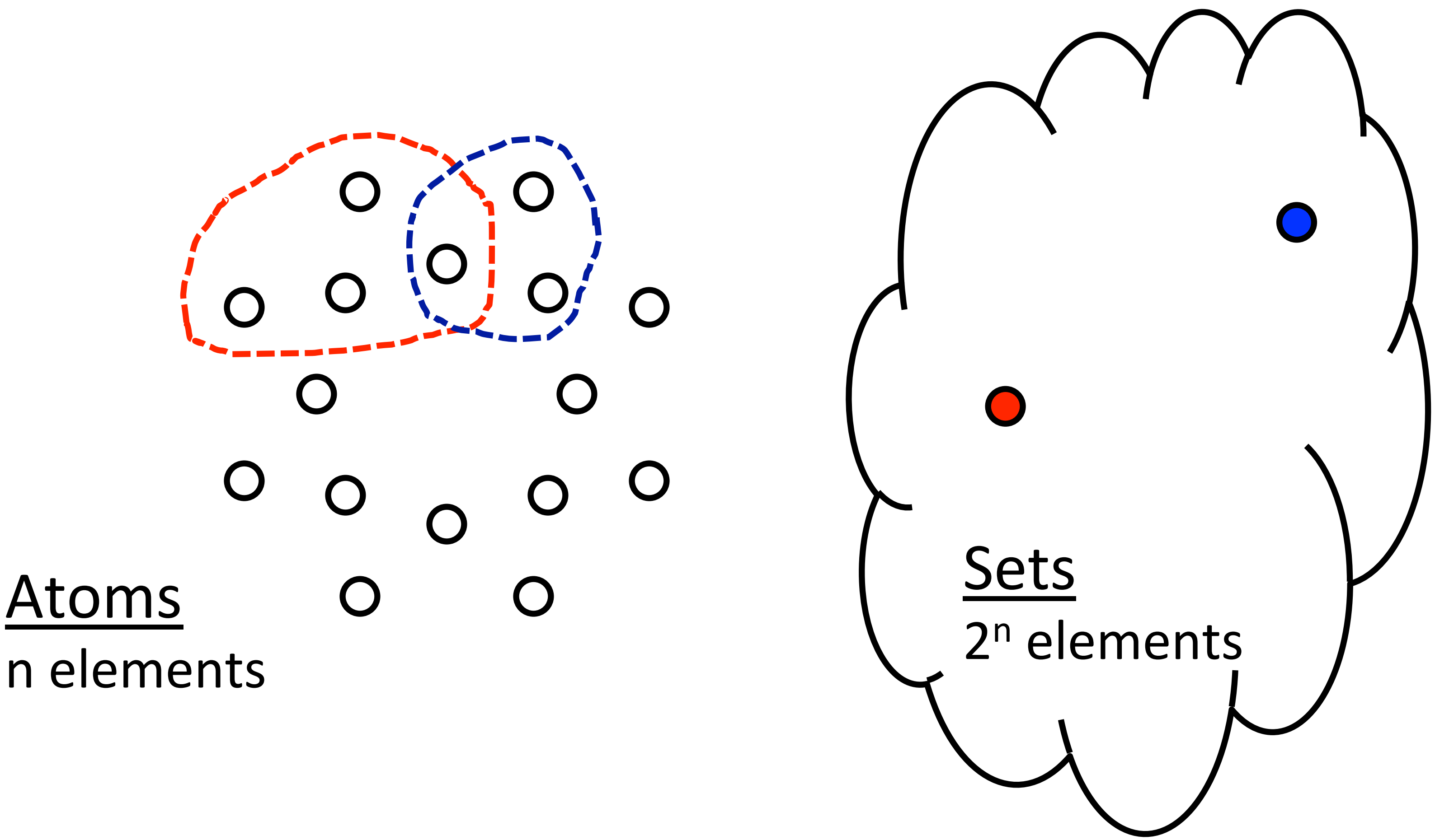
Set-powersets

- SetPow$_n$ is the structure $([n] \cup 2^{[n]}$, Atoms, Sets, In$)$



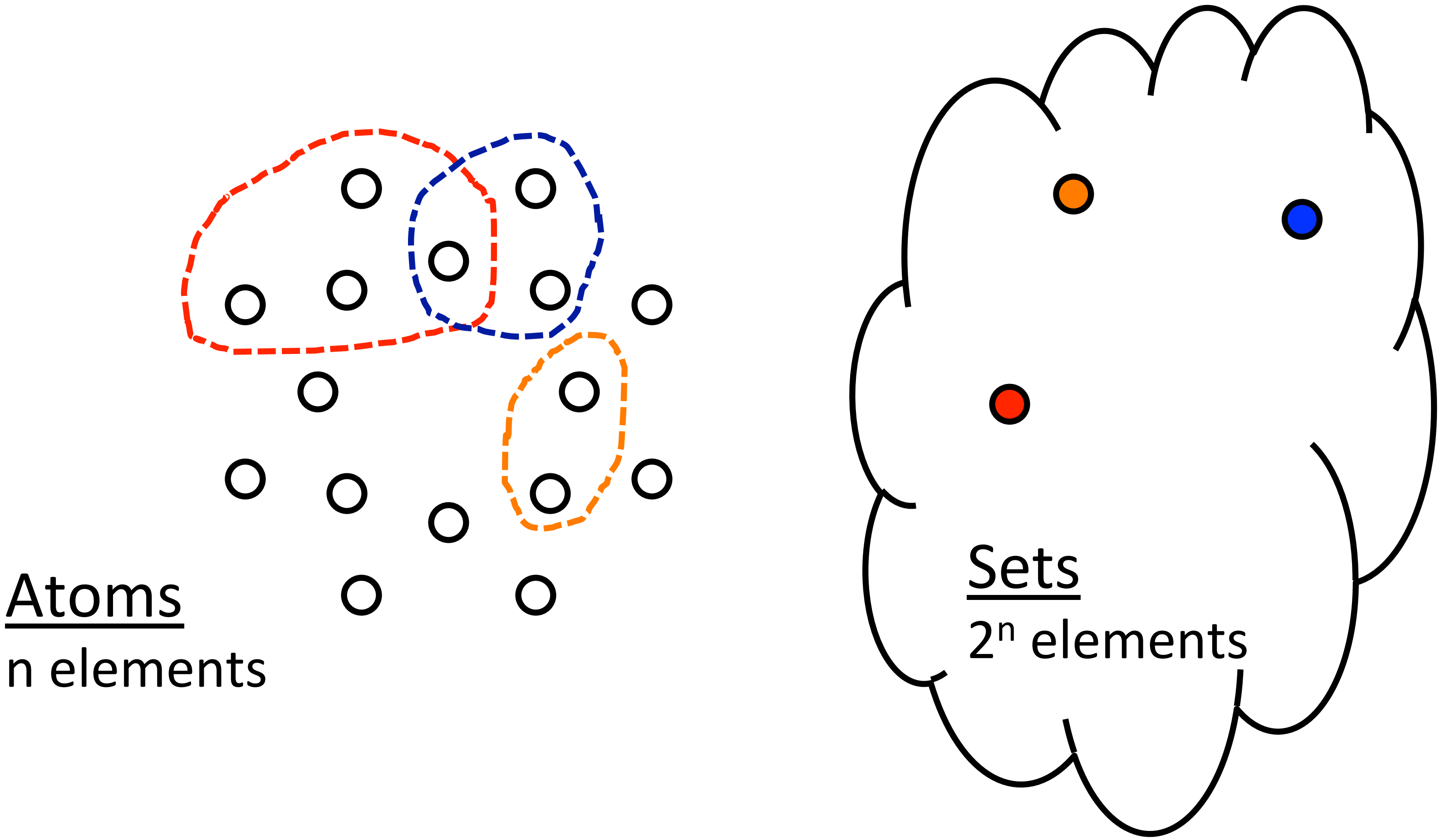
Set-powersets

- SetPow$_n$ is the structure ($[n] \cup 2^{[n]}$, Atoms, Sets, In)



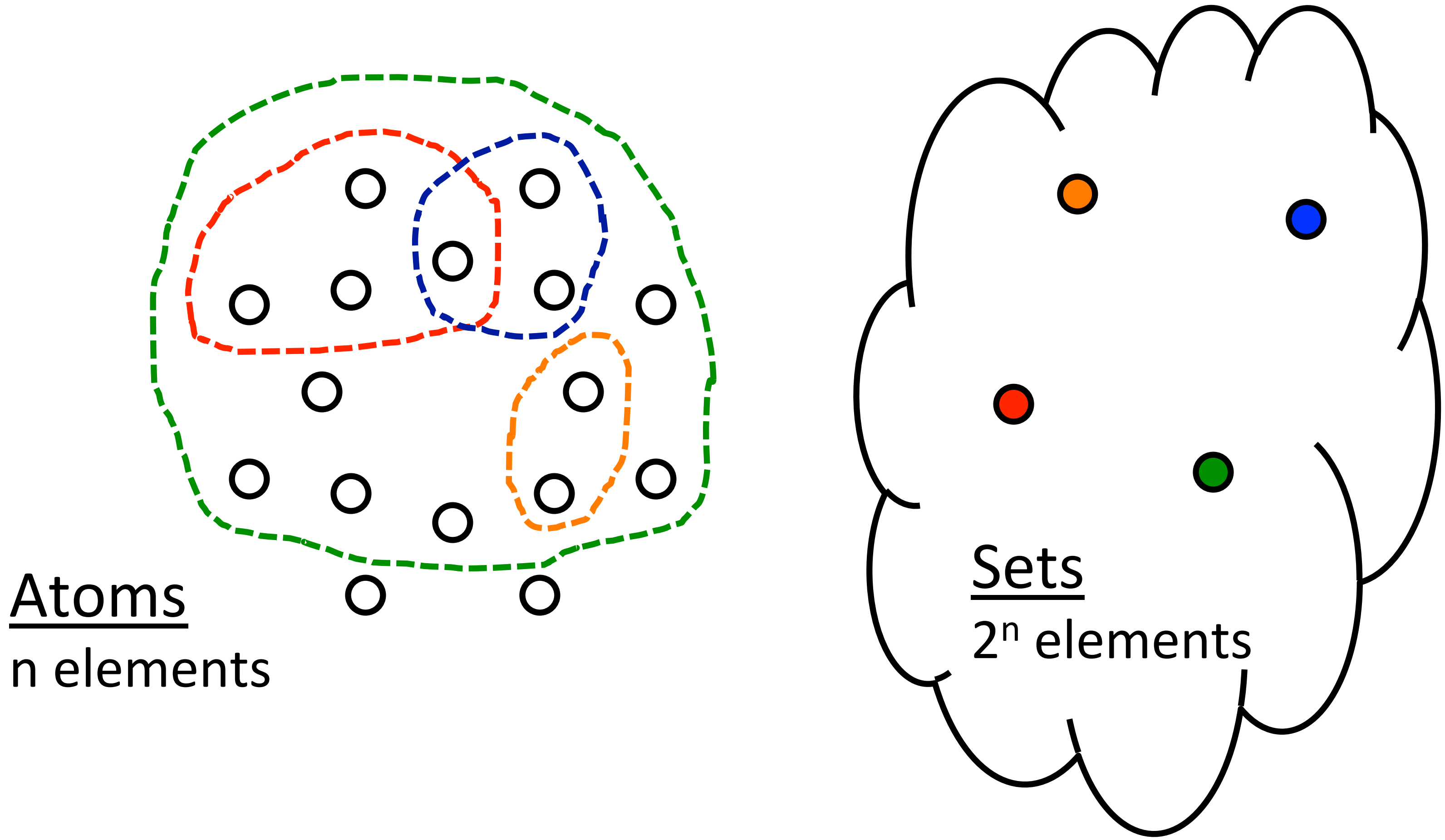
Set-powersets

- SetPow_n is the structure $([n] \cup 2^{[n]}$, <u>Atoms</u>, <u>Sets</u>, <u>In</u>$)$



Set-powersets

- SetPow_n is the structure $([n] \cup 2^{[n]}$, Atoms, Sets, In$)$



Set-powersets

- SetPow$_n$ is the structure $([n] \cup 2^{[n]}, \underline{\text{Atoms}}, \underline{\text{Sets}}, \underline{\text{In}})$ where

$$\underline{\text{Atoms}} = [n] = \{1,\ldots,n\},$$

$$\underline{\text{Sets}} = \text{powerset of } \underline{\text{Atoms}},$$

$$\underline{\text{In}} = \{(i,X) \in \underline{\text{Atoms}} \times \underline{\text{Sets}} \mid i \in X\}.$$

Set-powersets

- SetPow_n is the structure $([n] \cup 2^{[n]}, \underline{\text{Atoms}}, \underline{\text{Sets}}, \underline{\text{In}})$ where

$$\underline{\text{Atoms}} = [n] = \{1,\ldots,n\},$$
$$\underline{\text{Sets}} = \text{powerset of } \underline{\text{Atoms}},$$
$$\underline{\text{In}} = \{(i,X) \in \underline{\text{Atoms}} \times \underline{\text{Sets}} \mid i \in X\}.$$

- A **set-powerset** is any structure **A** with relations $\{\underline{\text{Atoms}}, \underline{\text{Sets}}, \underline{\text{In}}\}$ which is isomorphic to SetPow_n for some $n > 0$. It is said to be EVEN/ODD according to the parity of n.

Set-powersets

Obs. The class of set-powersets is FO definable.

- We *cannot* say (in first-order logic):

$\forall X \subseteq \underline{\text{Atoms}}\ \exists S \in \underline{\text{Sets}}\ \forall x \in \underline{\text{Atoms}},\ x \in X \Leftrightarrow \underline{\text{In}}(x,S)$

- Instead, we say:

"$\emptyset \in \underline{\text{Sets}}$" $\wedge$ $\forall S \in \underline{\text{Sets}}\ \forall x \in \underline{\text{Atoms}}$ "$S \cup \{x\} \in \underline{\text{Sets}}$"

- This formula exploits *finiteness* in an essential way.

Set-powersets

Theorem

The class of EVEN set-powersets is not FO definable.

Set-powersets

Theorem

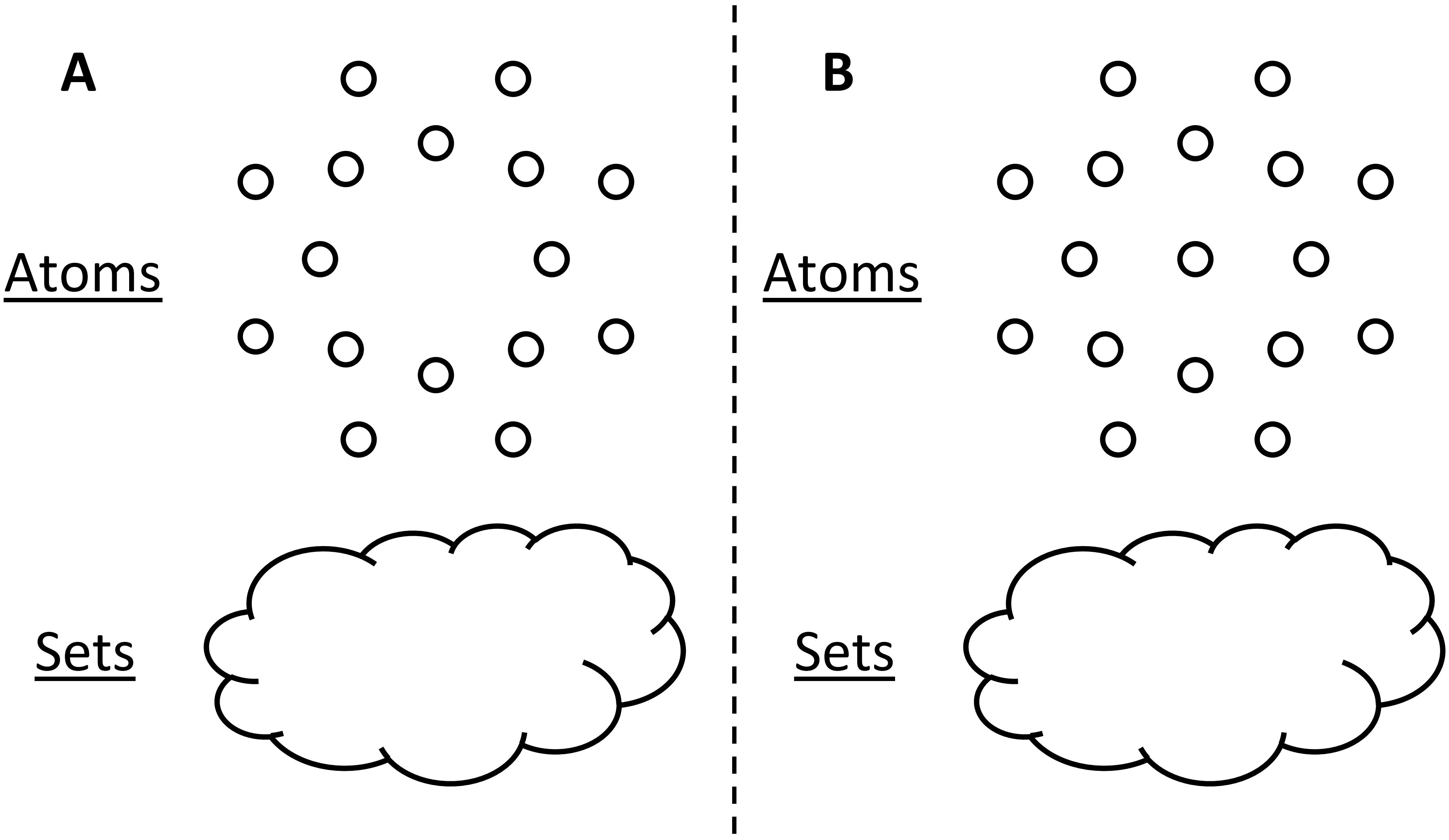
The class of EVEN set-powersets is not FO definable.

Proof

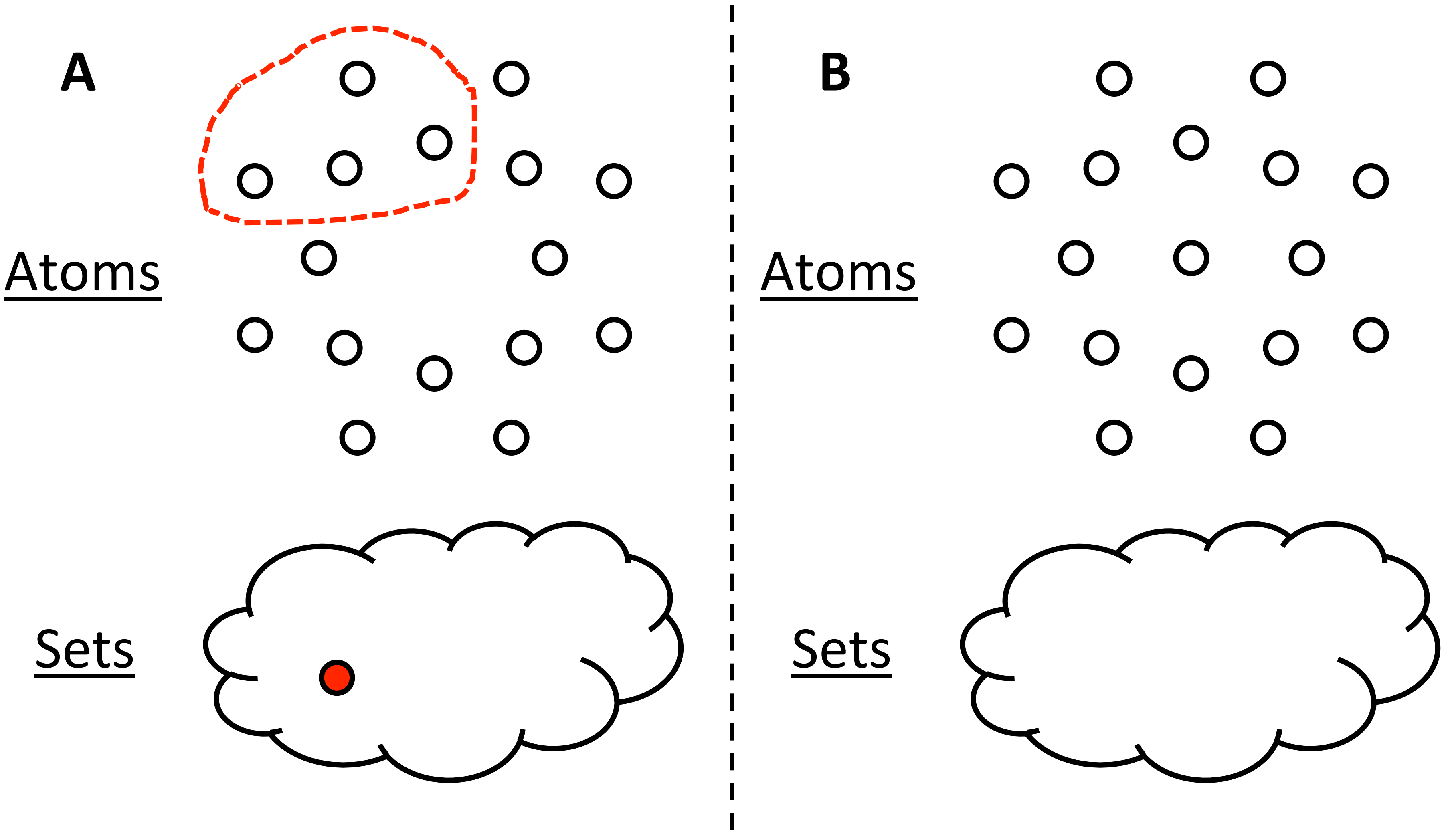
For every k, we show that ***Duplicator*** has a winning strategy in the k-round Ehrenfeucht-Fraisse game on

$$\mathbf{A} = \text{SetPow}_{2\text{^}k} \quad \text{and} \quad \mathbf{B} = \text{SetPow}_{2\text{^}k+1}$$

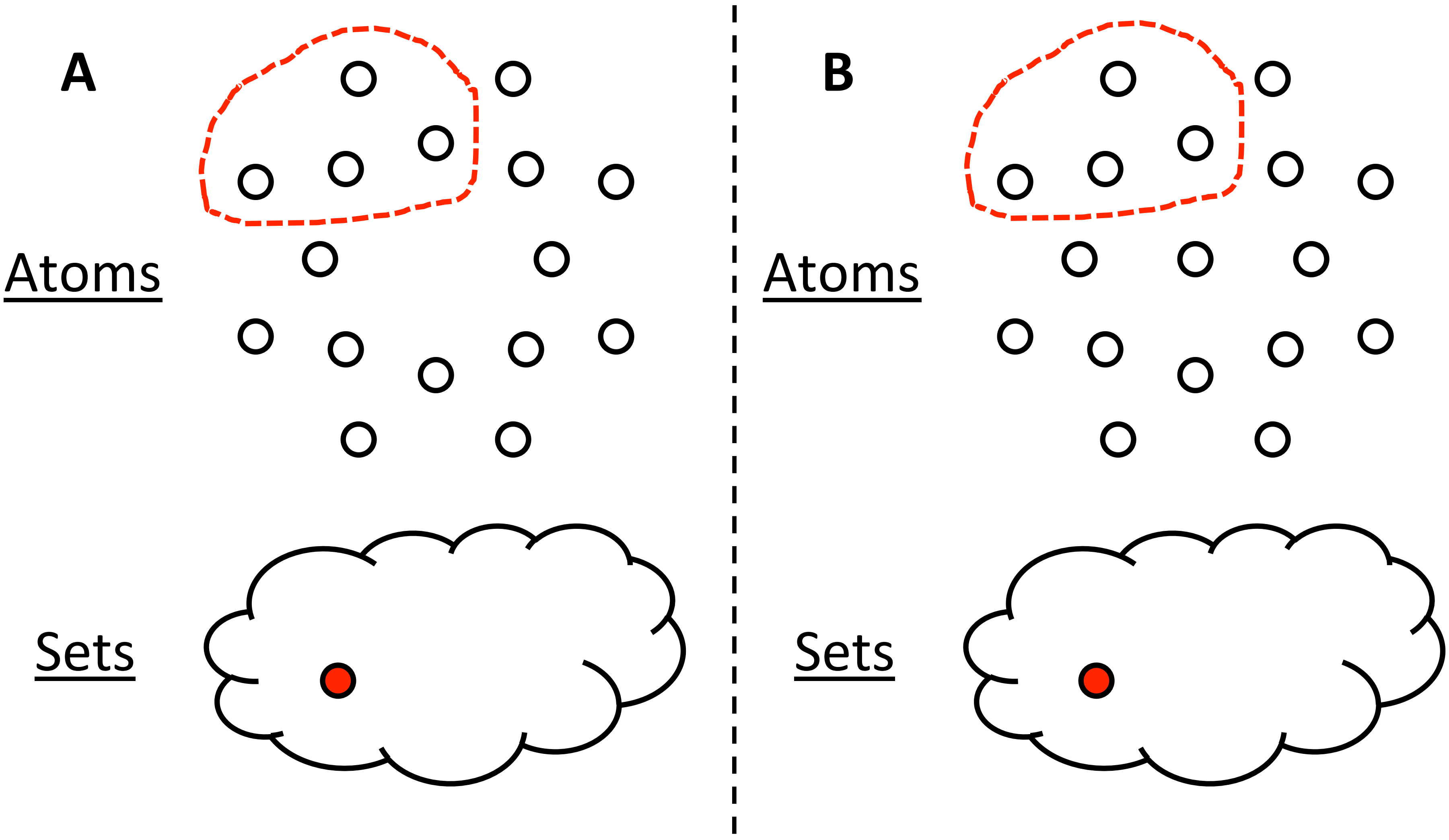
Winning strategy (by picture)



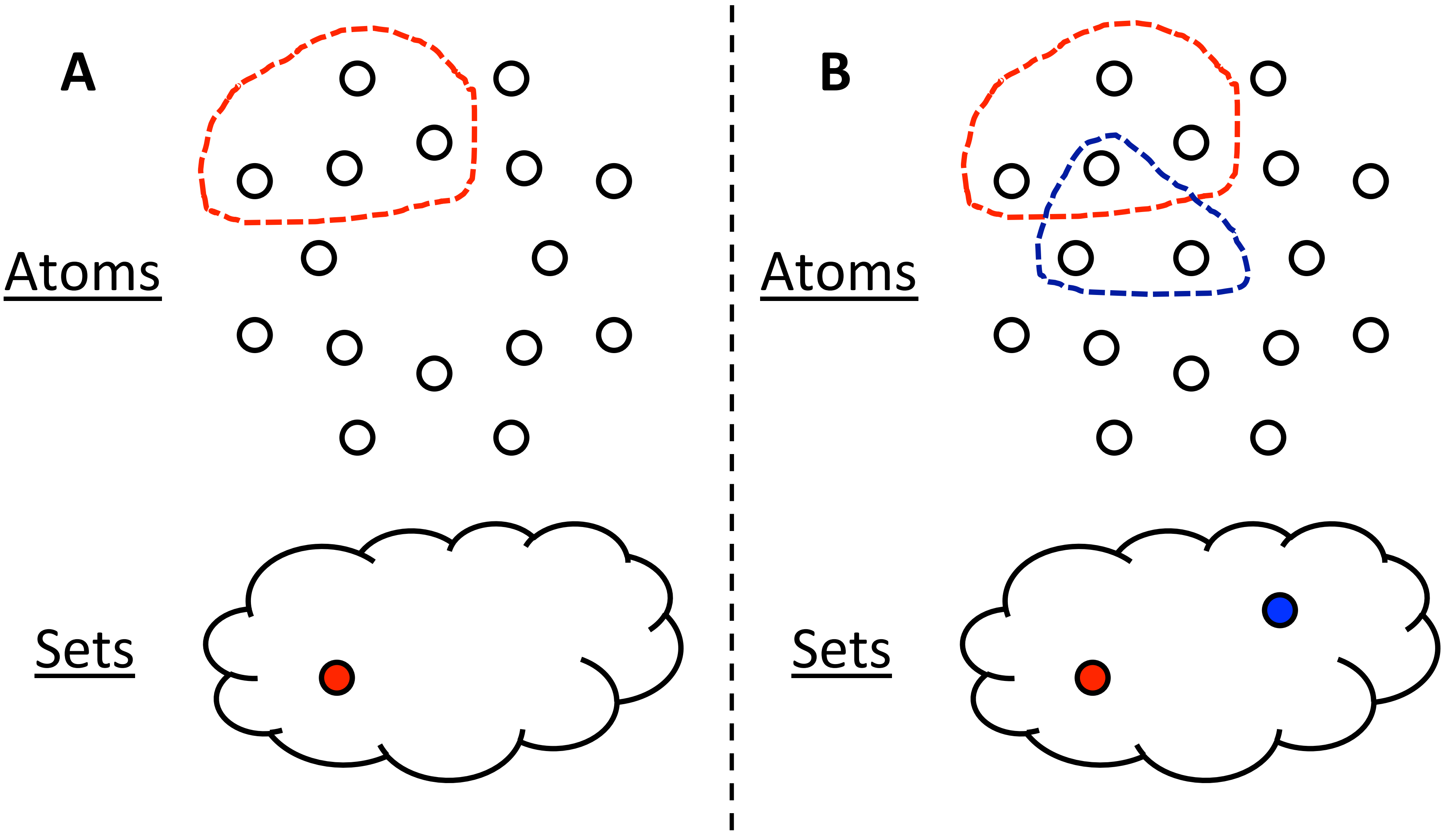
Winning strategy (by picture)



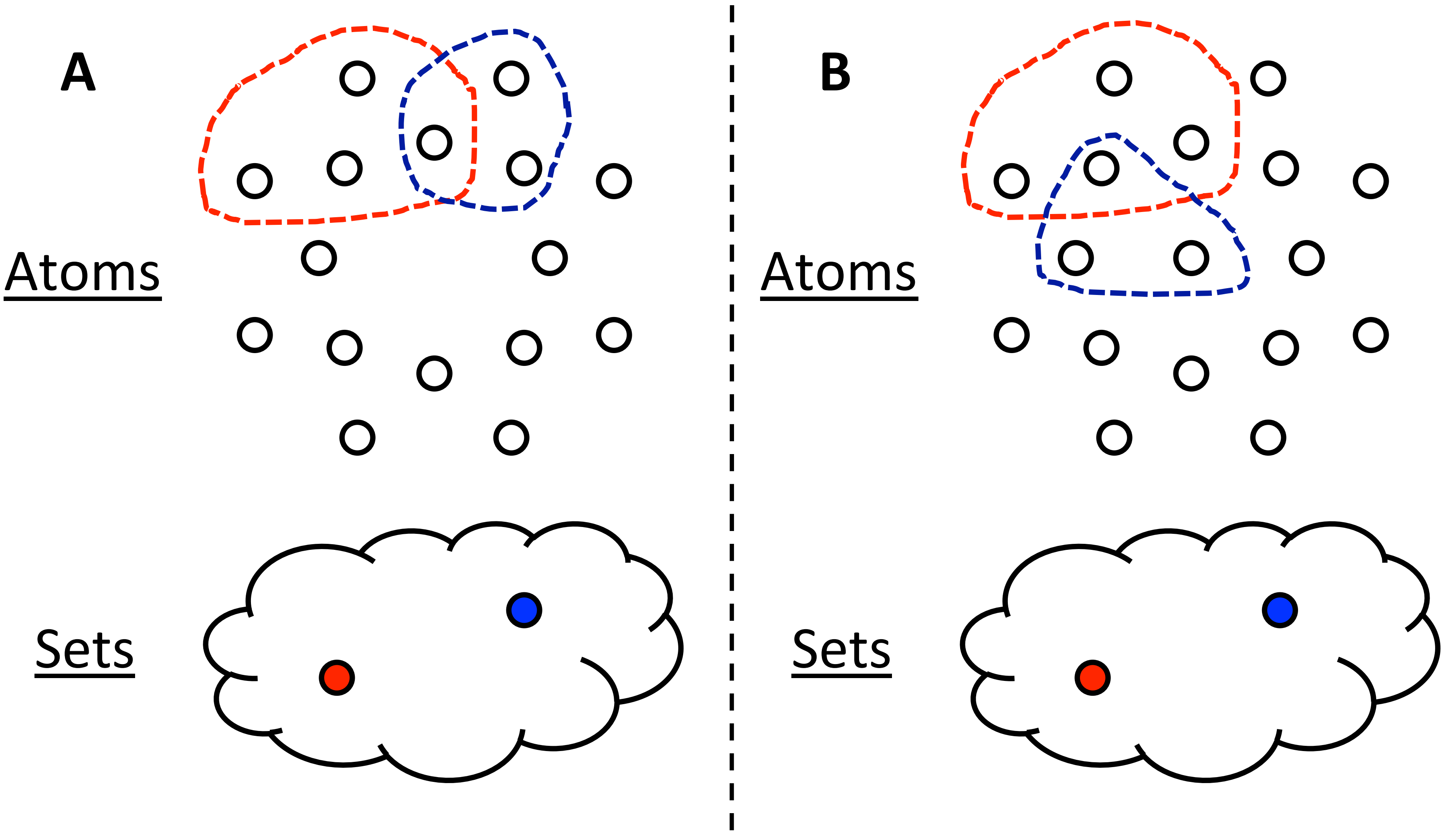
Winning strategy (by picture)



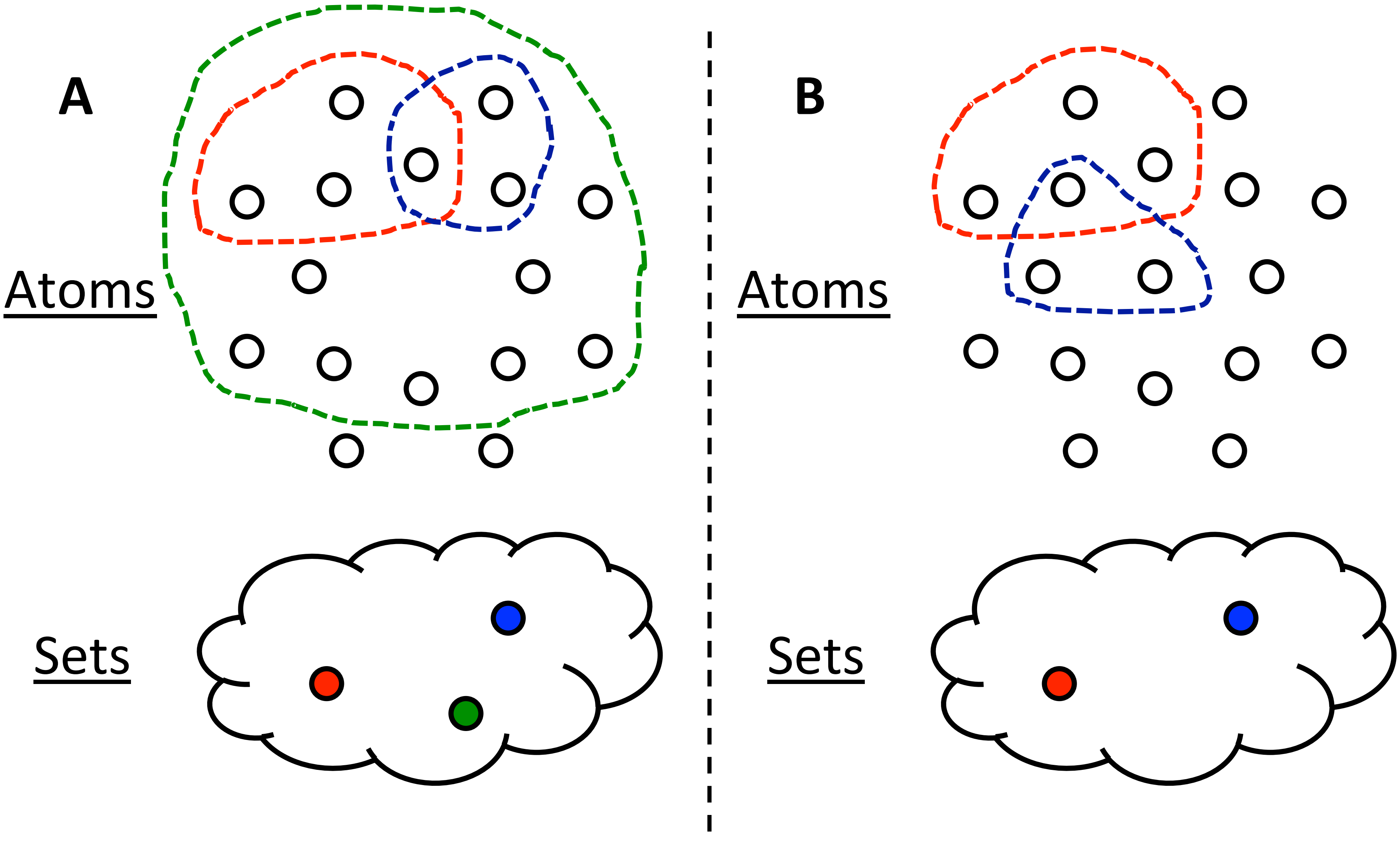
Winning strategy (by picture)



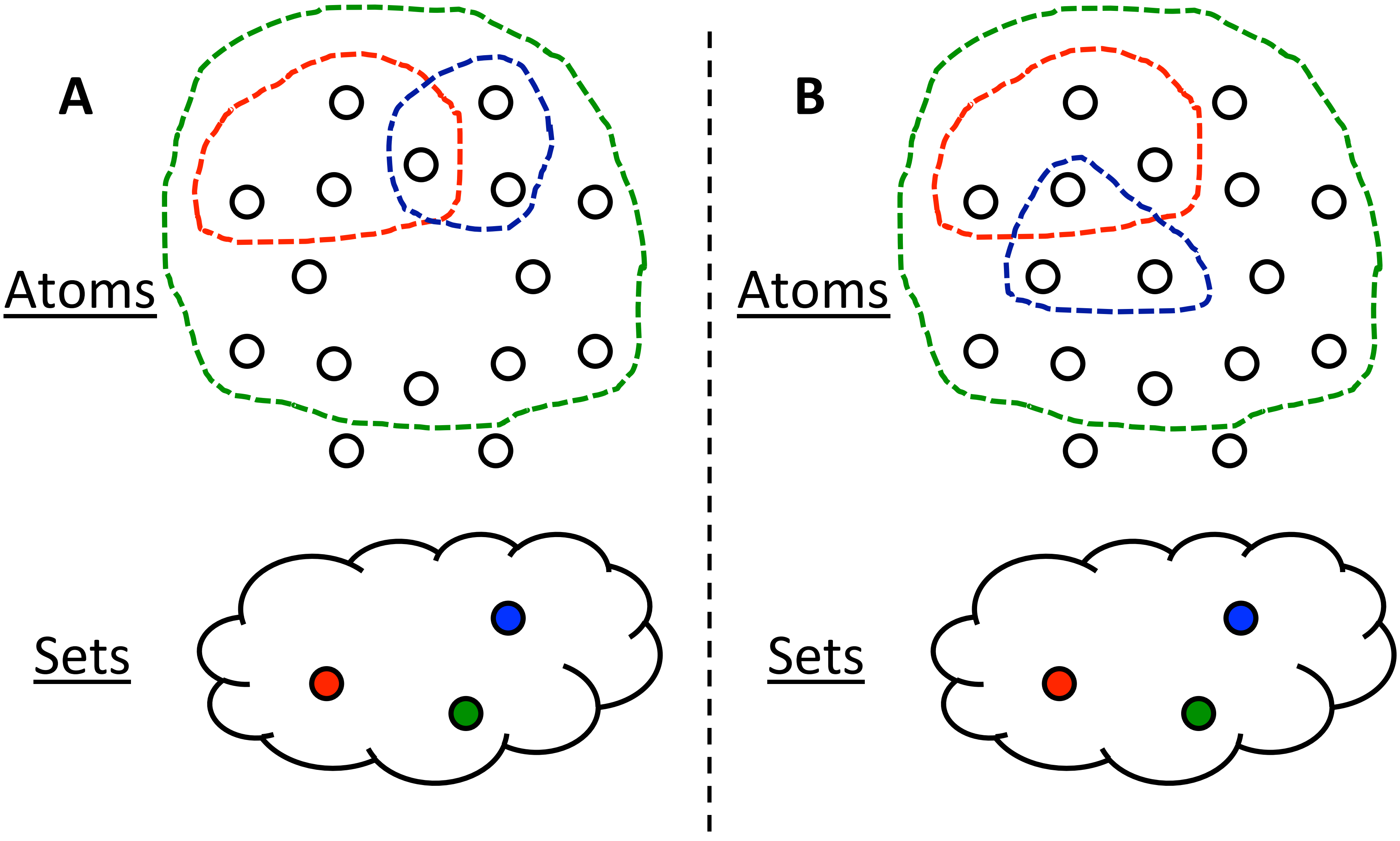
Winning strategy (by picture)



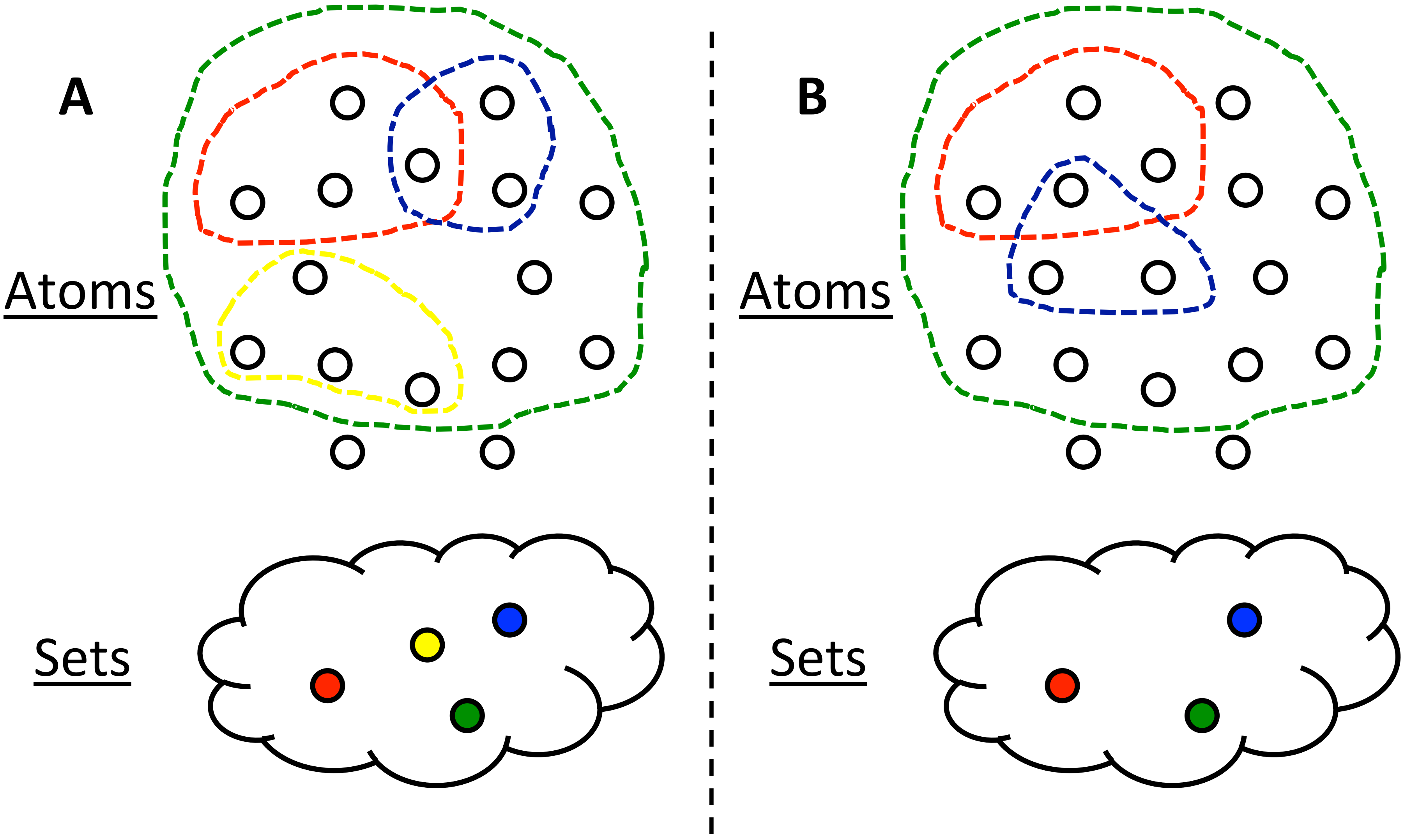
Winning strategy (by picture)



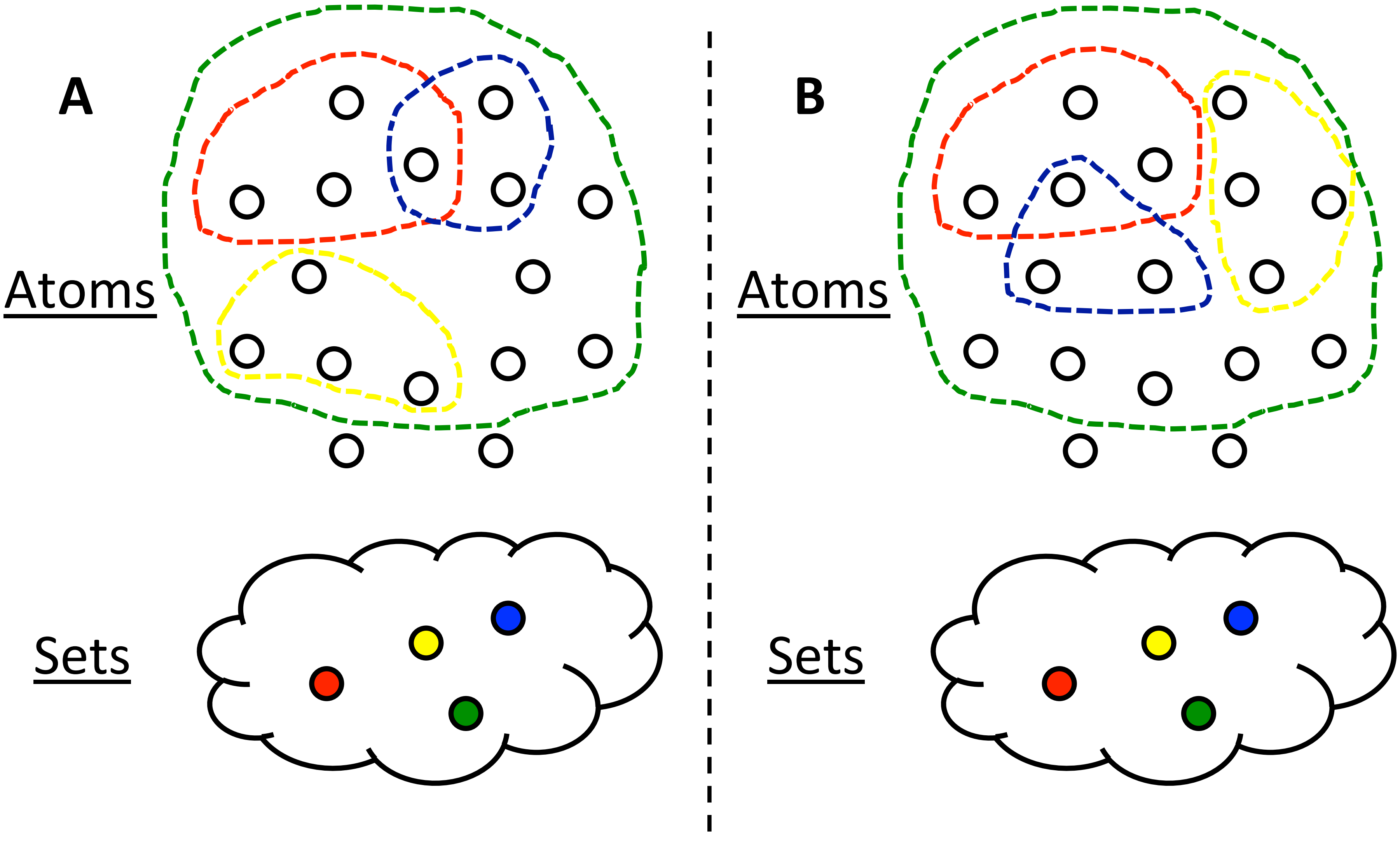
Winning strategy (by picture)



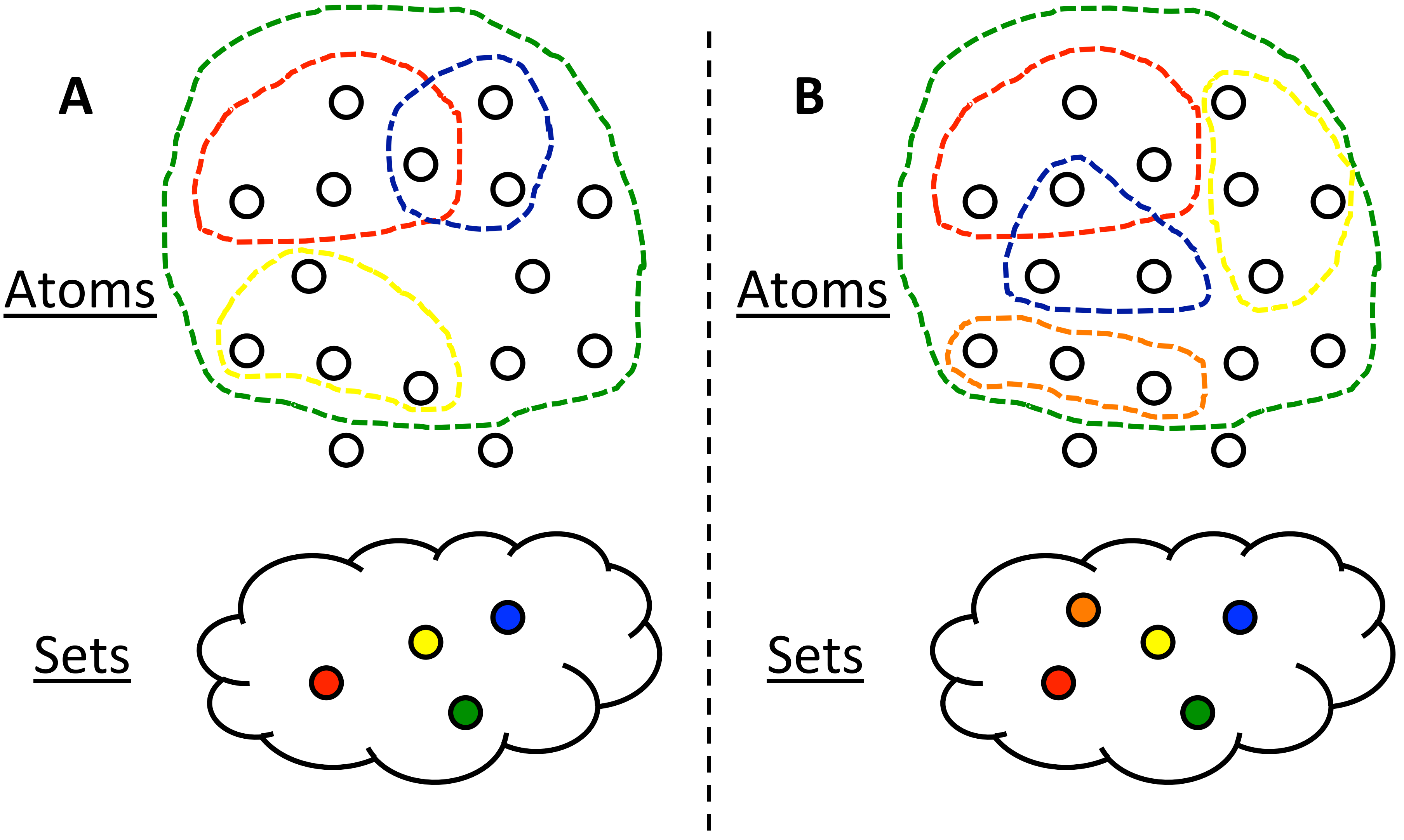
Winning strategy (by picture)



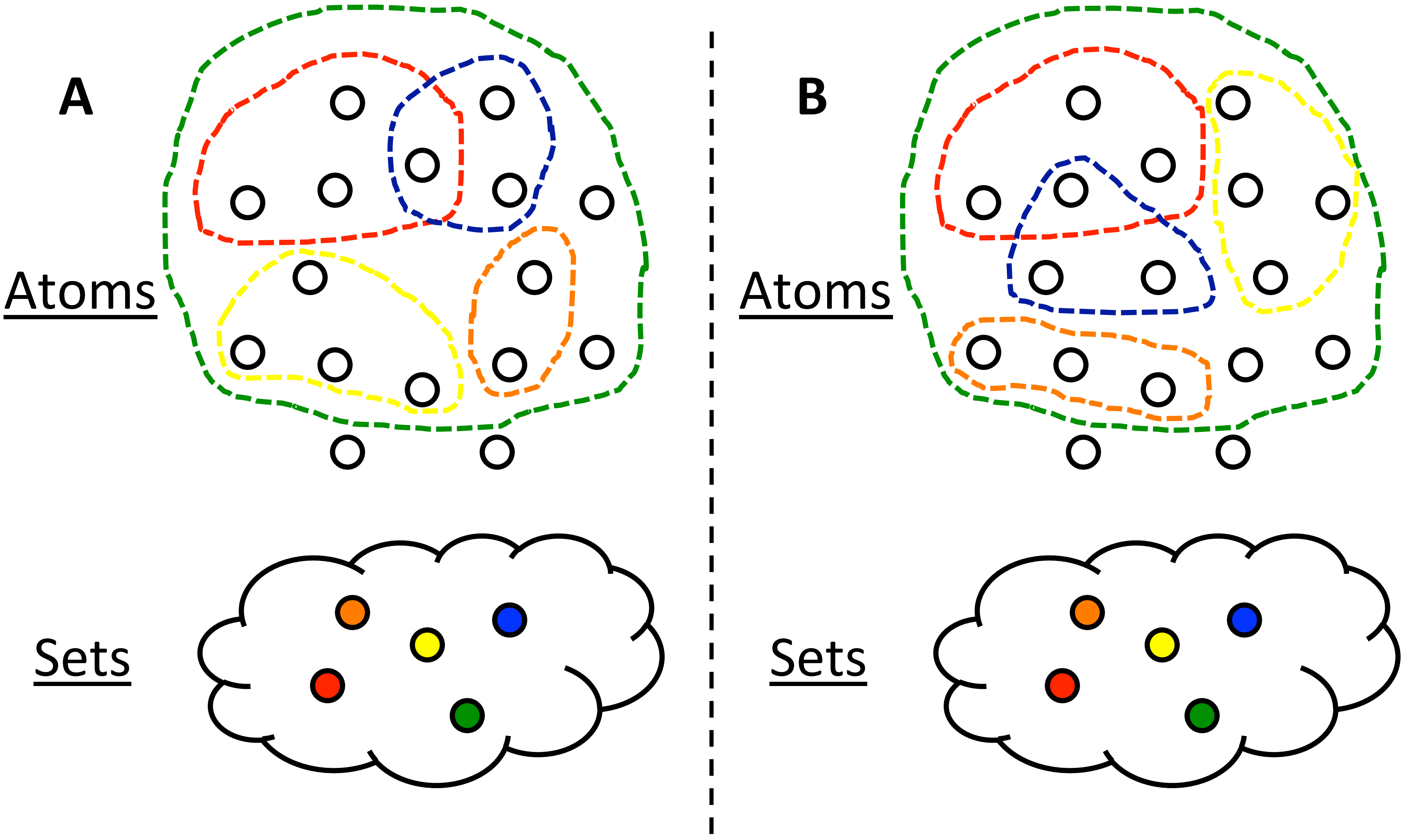
Winning strategy (by picture)



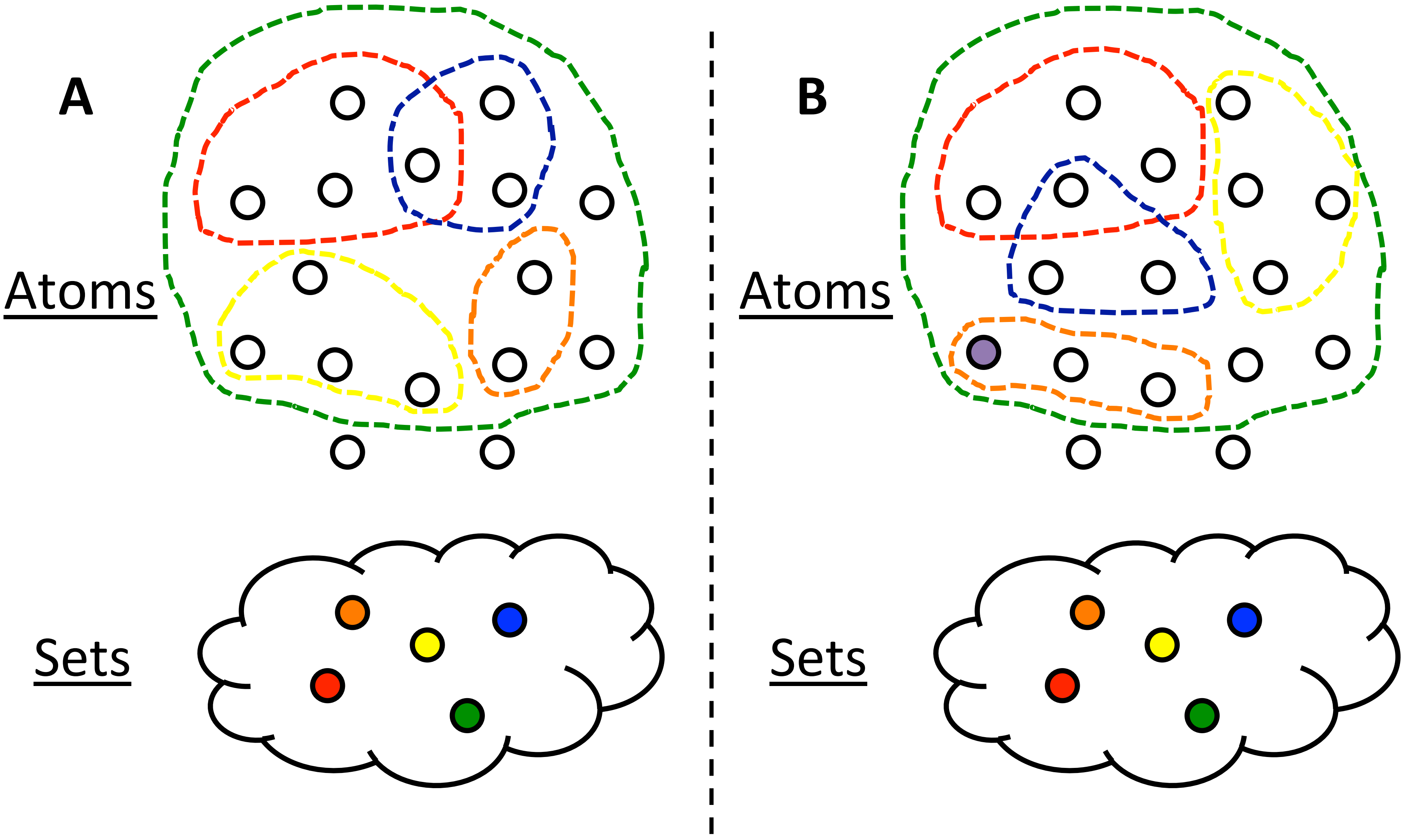
Winning strategy (by picture)



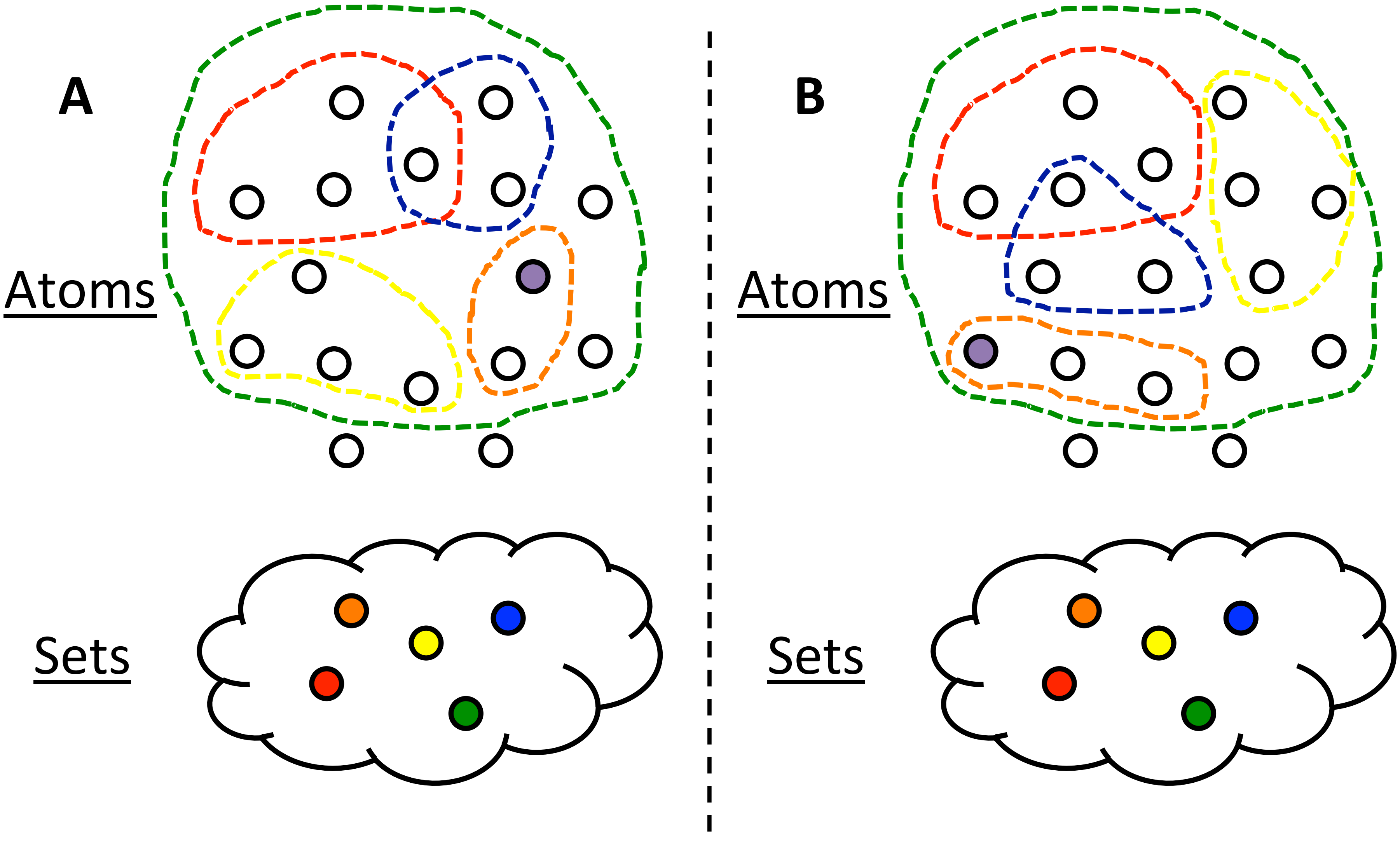
Winning strategy (by picture)



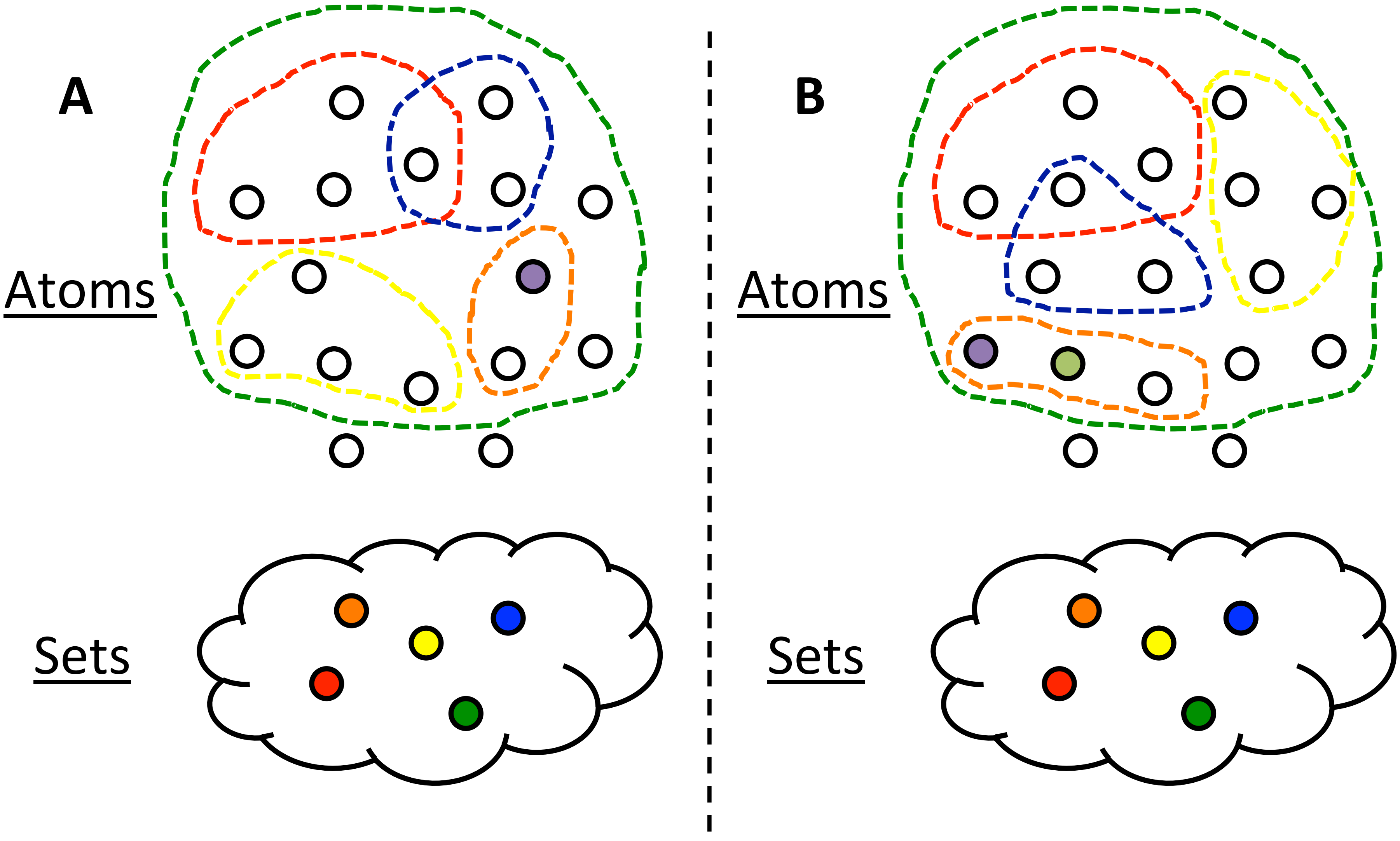
Winning strategy (by picture)



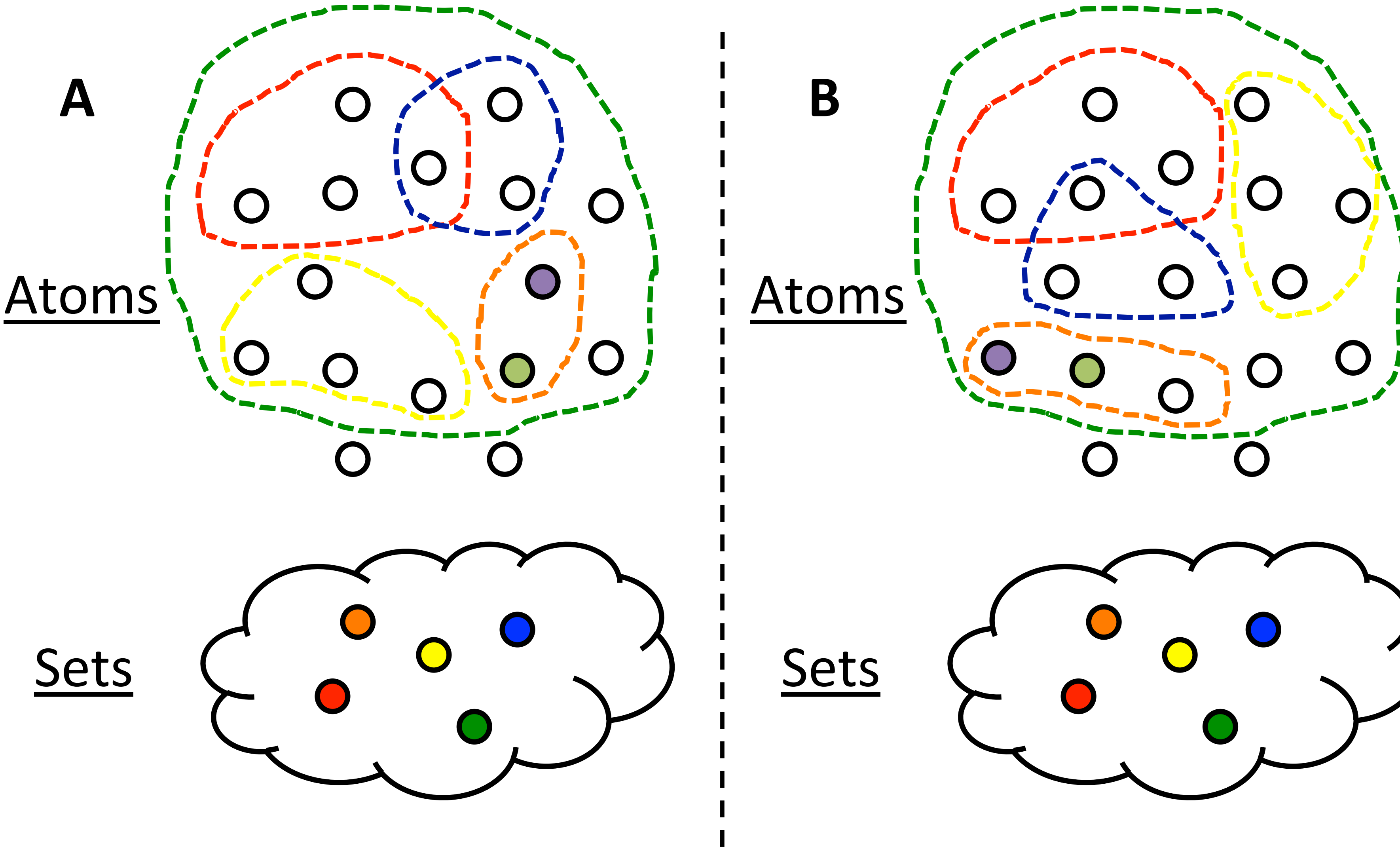
Winning strategy (by picture)



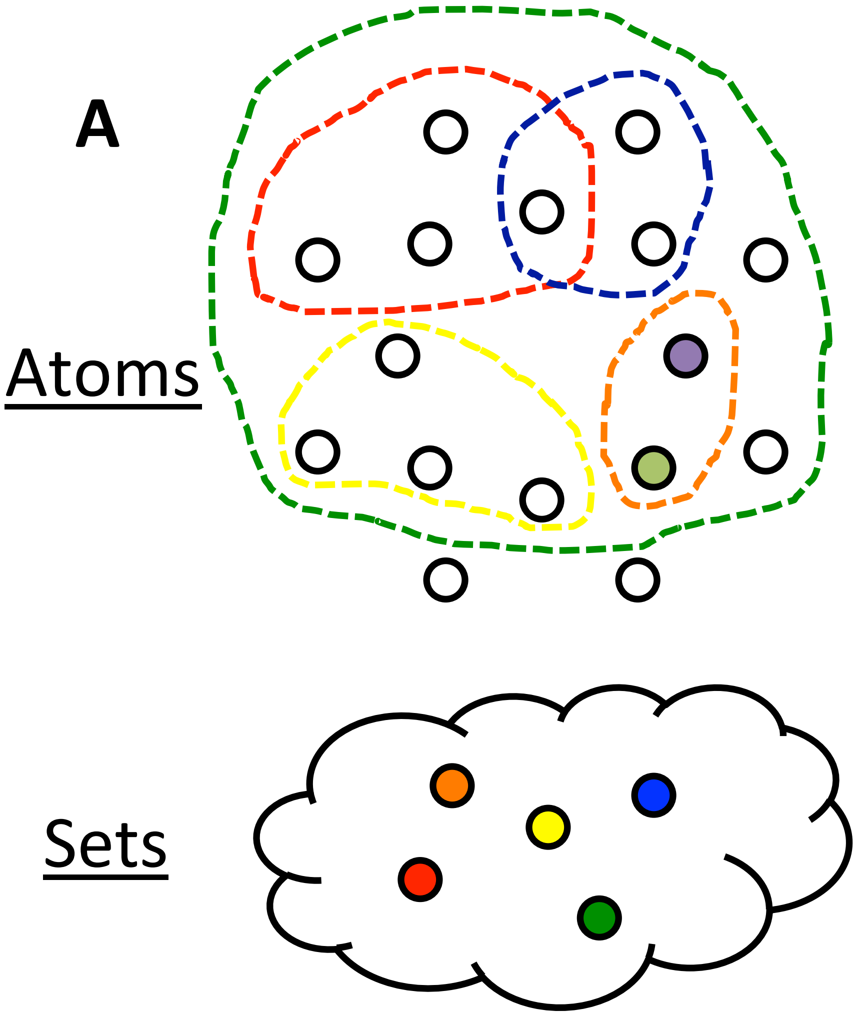
Winning strategy (by picture)

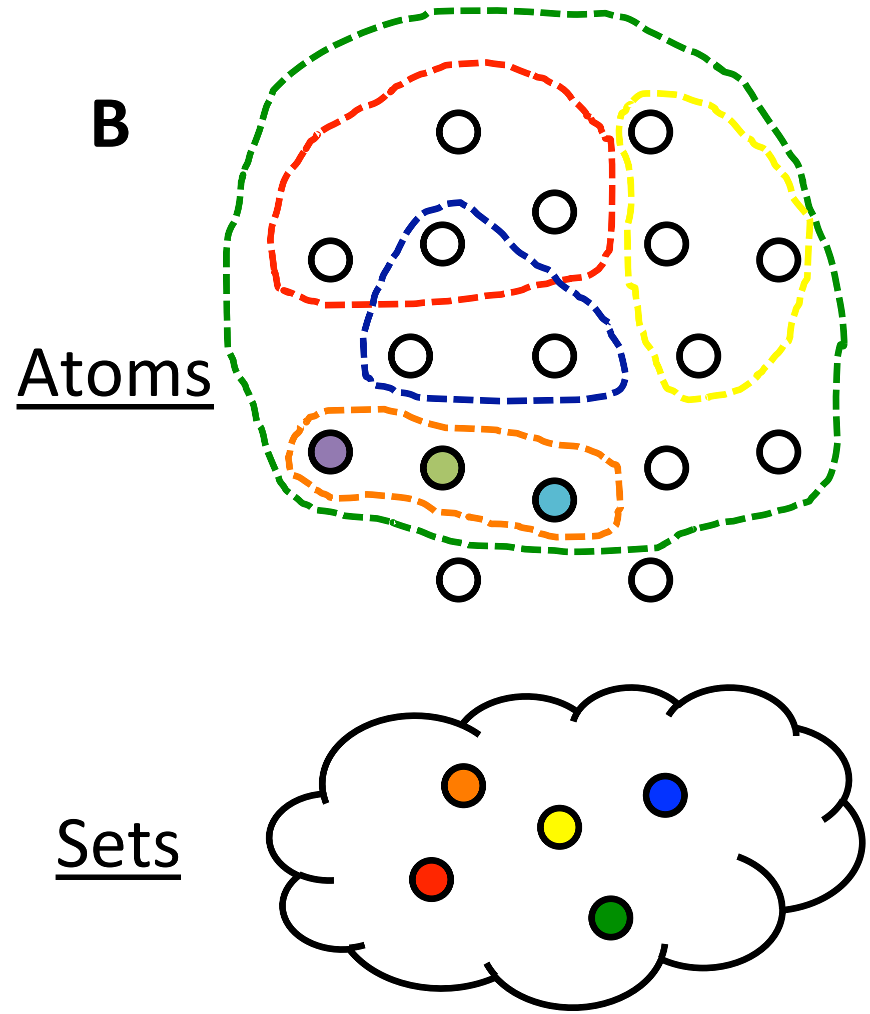


Winning strategy (by picture)

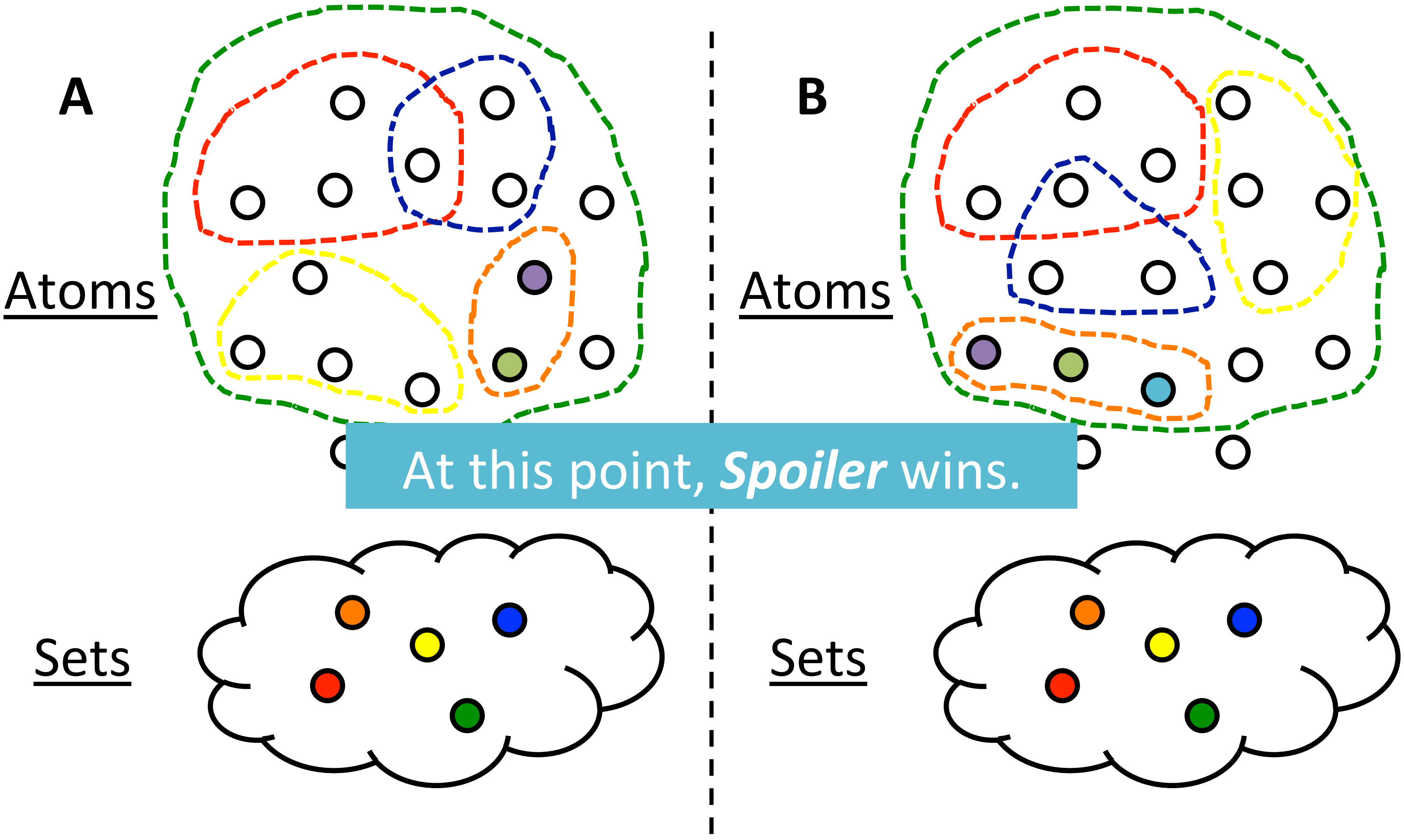


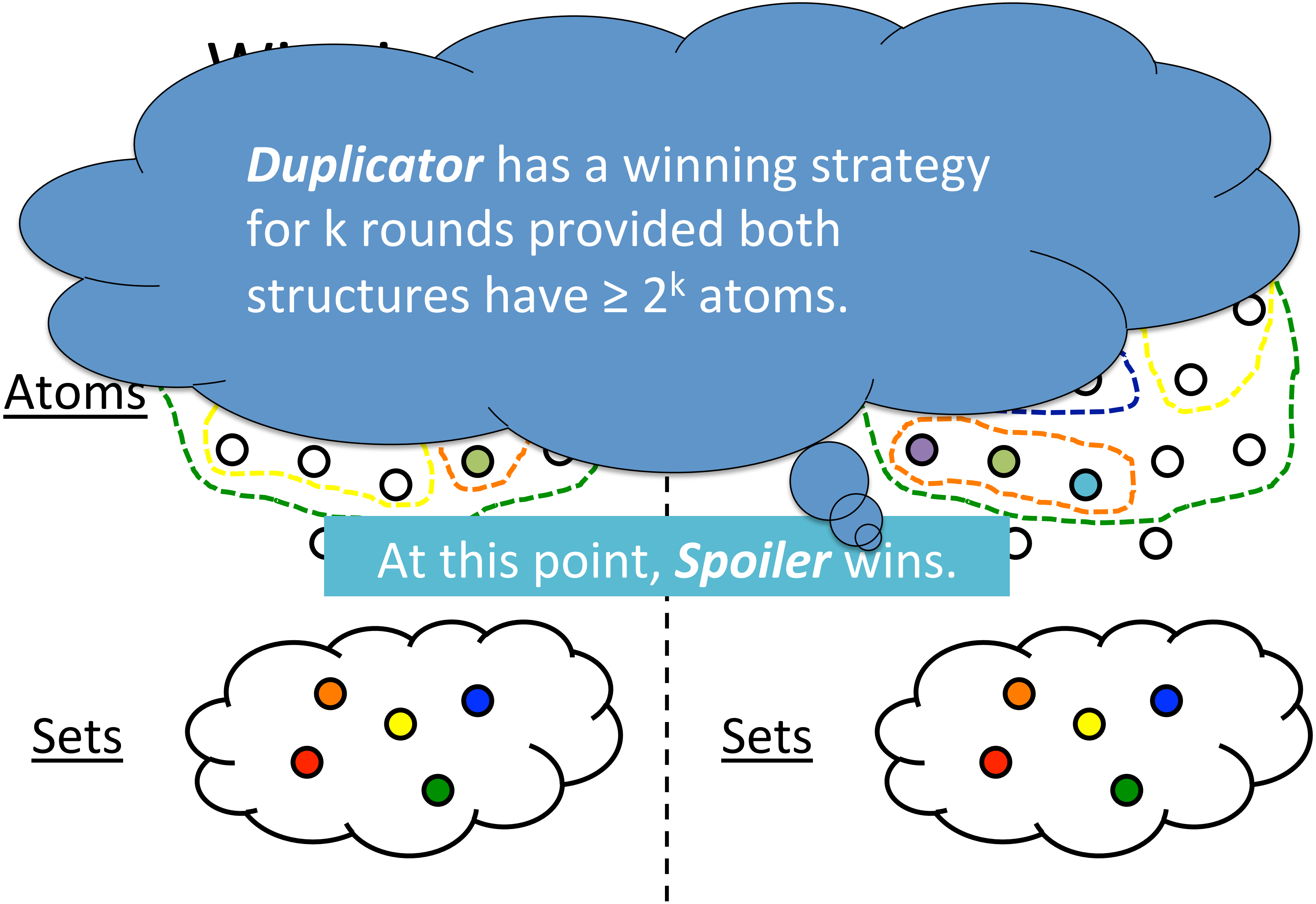
Winning strategy (by picture)





Winning strategy (by picture)




Duplicator has a winning strategy for k rounds provided both structures have ≥ 2^k atoms.
Atoms
At this point, *Spoiler* wins.
Sets
Sets

Duplicator has a winning strategy for k rounds provided both structures have $\geq 2^k$ atoms.

Atoms

***Duplicator's* winning strategy**:

in round j, preserve the cardinality up to 2^{k-j} of every Boolean combination of the chosen sets and atoms