

# Complexity Measures for Directed Graphs

Erich Grädel

# Tree-width of undirected graphs

Many graph problems that are algorithmically intractable on general graphs, become simple on **trees** and on **graphs that resemble trees**.

**Simple:** solvable in linear time

# Tree-width of undirected graphs

Many graph problems that are algorithmically intractable on general graphs, become simple on **trees** and on **graphs that resemble trees**.

**Simple:** solvable in linear time

The notion of tree-width makes this precise and gives a measure of how closely a graph resembles a tree.

**Theorem.** (Courcelle) Every graph problem expressible in monadic second-order logic is solvable in linear time on graphs of bounded tree width.

# Different definitions of tree width

Tree width can be defined either in terms of tree decompositions or via the cops-and-robber game.

A **tree-decomposition** of width  $k$  of a graph  $G = (V, E)$  is a tuple  $(T, (X_t)_{t \in T})$  where  $T$  is a tree, and each  $X_t$  is a set of at most  $k + 1$  nodes of  $V$ , such that

- every edge of  $G$  lives as some node of  $T$ :  $(u, v) \in E \implies \{u, v\} \subseteq X_t$  for some  $t \in T$
- for every node  $v \in V$ , the set  $\{t \in T : v \in X_t\}$  is a subtree of  $T$ .

# Different definitions of tree width

Tree width can be defined either in terms of tree decompositions or via the cops-and-robber game.

A **tree-decomposition** of width  $k$  of a graph  $G = (V, E)$  is a tuple  $(T, (X_t)_{t \in T})$  where  $T$  is a tree, and each  $X_t$  is a set of at most  $k + 1$  nodes of  $V$ , such that

- every edge of  $G$  lives as some node of  $T$ :  $(u, v) \in E \implies \{u, v\} \subseteq X_t$  for some  $t \in T$
- for every node  $v \in V$ , the set  $\{t \in T : v \in X_t\}$  is a subtree of  $T$ .

$\text{tw}(G) := \min\{k : G \text{ admits a tree-decomposition of width } k\}$

# The cops-and-robber game

$k$  cops try to catch a robber on  $G = (V, E)$

The robber selects an initial position  $v_0 \in G$

The cops are outside the graph

# The cops-and-robber game

$k$  cops try to catch a robber on  $G = (V, E)$

The robber selects an initial position  $v_0 \in G$

The cops are outside the graph

## Move:

- the cops move from their position  $X \subseteq V$  to a new position  $X' \subseteq V$
- the robber moves from his current position  $v$  along a path in  $G \setminus (X \cap X')$  to a new position  $v' \notin X'$ .

If no such position exists, the robber is caught and the cops have won.

# The cops-and-robber game

$k$  cops try to catch a robber on  $G = (V, E)$

The robber selects an initial position  $v_0 \in G$

The cops are outside the graph

## Move:

- the cops move from their position  $X \subseteq V$  to a new position  $X' \subseteq V$
- the robber moves from his current position  $v$  along a path in  $G \setminus (X \cap X')$  to a new position  $v' \notin X'$ .

If no such position exists, the robber is caught and the cops have won.

$\text{tw}(G) := \min\{k : k + 1 \text{ cops have a winning strategy for the cops-and-robber game on } G\}$

# Tree width of other structures than graphs

Tree-width generalises from graphs to **hypergraphs** and to arbitrary **relational structures**.

The tree width of a relational structure  $\mathfrak{A} = (A, R_1, \dots, R_m)$  is the tree width of its **Gaifman graph**  $(A, E)$  where

$$E = \{(a, b) : a \neq b \text{ and } a, b \text{ coexist in some } \bar{c} \in R_i\}$$

# Tree width of other structures than graphs

Tree-width generalises from graphs to **hypergraphs** and to arbitrary **relational structures**.

The tree width of a relational structure  $\mathfrak{A} = (A, R_1, \dots, R_m)$  is the tree width of its **Gaifman graph**  $(A, E)$  where

$$E = \{(a, b) : a \neq b \text{ and } a, b \text{ coexist in some } \bar{c} \in R_i\}$$

For a **directed graph**, this means that we forget the orientation of the edges, when calculating the tree width.

# Problems where direction matters

Tree width is appropriate also for directed graphs when we study problems that are insensitive to direction of edges, or in settings where the inverse edges are also available.

**Example:** Efficient evaluation of monadic second-order logic  
(Courcelle's Theorem)

# Problems where direction matters

Tree width is appropriate also for directed graphs when we study problems that are insensitive to direction of edges, or in settings where the inverse edges are also available.

**Example:** Efficient evaluation of monadic second-order logic  
(Courcelle's Theorem)

Tree width is problematic for problems where the direction matters and the inverse edges cannot be used:

- Directed graph problems such as **Hamiltonicity**
- Evaluation of modal logics, such as  **$\mu$ -calculus**
- Graph games, such as **parity games**

# Parity games and their complexity

$$G = (V, E, \Omega), \quad V = V_0 \cup V_1, \quad \Omega : V \rightarrow \mathbb{N}$$

Player 0 moves at positions  $v \in V_0$ , Player 1 at positions  $v \in V_1$

$\Omega(v)$  is the **priority** of position  $v$

# Parity games and their complexity

$$G = (V, E, \Omega), \quad V = V_0 \cup V_1, \quad \Omega : V \rightarrow \mathbb{N}$$

Player 0 moves at positions  $v \in V_0$ , Player 1 at positions  $v \in V_1$

$\Omega(v)$  is the **priority** of position  $v$

**Play:** finite or infinite sequence  $\pi = v_0 v_1 v_2 \cdots$  with  $(v_i, v_{i+1}) \in E$

# Parity games and their complexity

$$G = (V, E, \Omega), \quad V = V_0 \cup V_1, \quad \Omega : V \rightarrow \mathbb{N}$$

Player 0 moves at positions  $v \in V_0$ , Player 1 at positions  $v \in V_1$

$\Omega(v)$  is the **priority** of position  $v$

**Play:** finite or infinite sequence  $\pi = v_0 v_1 v_2 \cdots$  with  $(v_i, v_{i+1}) \in E$

**Winning condition:**

- finite plays: who cannot move, loses
- infinite plays: **least priority** seen **infinitely often** determines winner

Player 0 wins  $\pi \iff \min\{k : (\exists^\infty i)\Omega(v_i) = k\}$  is even

# Parity games and their complexity

$$G = (V, E, \Omega), \quad V = V_0 \cup V_1, \quad \Omega : V \rightarrow \mathbb{N}$$

Player 0 moves at positions  $v \in V_0$ , Player 1 at positions  $v \in V_1$

$\Omega(v)$  is the **priority** of position  $v$

**Play:** finite or infinite sequence  $\pi = v_0 v_1 v_2 \cdots$  with  $(v_i, v_{i+1}) \in E$

**Winning condition:**

- finite plays: who cannot move, loses
- infinite plays: **least priority** seen **infinitely often** determines winner

Player 0 wins  $\pi \iff \min\{k : (\exists^\infty i)\Omega(v_i) = k\}$  is even

**Winning regions**  $W_0, W_1$ :

$$W_i = \{v \in V : \text{Player } i \text{ has winning strategy from position } v\}$$

# Significance of parity games

## Enlarge game graphs to simplify winning conditions

many games with complicated winning strategies can be simulated by parity games (over larger game graphs)

- games with winning conditions formulated in temporal logic (LTL) or monadic second-order logic (S1S)
- Muller games: games where the winner only depends on which priorities are seen infinitely often.
- games that model reactive systems

# Significance of parity games

Enlarge game graphs to simplify winning conditions

many games with complicated winning strategies can be simulated by parity games (over larger game graphs)

- games with winning conditions formulated in temporal logic (LTL) or monadic second-order logic (MSO)
- Muller games: games where the winner only depends on which priorities are seen infinitely often.
- games that model reactive systems

Parity games arise as model checking games of fixed-point logics

# Significance of parity games

Enlarge game graphs to simplify winning conditions

many games with complicated winning strategies can be simulated by parity games (over larger game graphs)

- games with winning conditions formulated in temporal logic (LTL) or monadic second-order logic (S1S)
- Muller games: games where the winner only depends on which priorities are seen infinitely often.
- games that model reactive systems

Parity games arise as model checking games of fixed-point logics

Parity games admit positional (i.e., memoryless) winning strategies

# Positional determinacy

In general, a **strategy for Player  $\sigma$**  in an infinite game is a partial function

$$f : V^* V_\sigma \rightarrow V \quad \text{with } f(v_0 \dots v_n) \in v_n E$$

**Positional strategies** depend only on current position, **not on history**

A positional strategy  $f$  for Player 0 which is winning on  $W \subseteq V$

is described by **solitaire subgame**  $G_f = (W, F) \subseteq (V, E)$ :

- $vF = \{f(v)\}$  for  $v \in W \cap V_0$   
 $vF = vE$  for  $v \in W \cap V_1$
- no terminal positions in  $W \cap V_0$   
the **least priority on every cycle** in  $(W, F)$  is **even**.

# Positional determinacy

In general, a **strategy for Player  $\sigma$**  in an infinite game is a partial function

$$f : V^* V_\sigma \rightarrow V \quad \text{with } f(v_0 \dots v_n) \in v_n E$$

**Positional strategies** depend only on current position, **not on history**

A positional strategy  $f$  for Player 0 which is winning on  $W \subseteq V$

is described by **solitaire subgame**  $G_f = (W, F) \subseteq (V, E)$ :

- $vF = \{f(v)\}$  for  $v \in W \cap V_0$   
 $vF = vE$  for  $v \in W \cap V_1$
- no terminal positions in  $W \cap V_0$   
the **least priority on every cycle** in  $(W, F)$  is **even**.

**Proposition** It can be checked in polynomial time whether a subgraph  $(W, F)$  defines indeed a winning strategy on  $W$ .

# Positional Determinacy

**Positional Determinacy Theorem** (Emerson/Jutla, Mostowski)

Parity games are determined, and every player has a positional winning strategy on her winning region.

**Theorem** (Emerson, Jutla, Sistla)

The parity game problem is in  $NP \cap Co-NP$

# Positional Determinacy

**Positional Determinacy Theorem** (Emerson/Jutla, Mostowski)

Parity games are determined, and every player has a positional winning strategy on her winning region.

**Theorem** (Emerson, Jutla, Sistla)

The parity game problem is in  $\text{NP} \cap \text{Co-NP}$

**Proof.** Guess positional strategies  $(W_0, F_0)$  and  $(W_1, F_1)$  for Players 0,1 with  $W_0 \cup W_1 = V$

Check that  $(W_\sigma, F_\sigma)$  is winning strategy for Player  $\sigma$ .

# Positional Determinacy

**Positional Determinacy Theorem** (Emerson/Jutla, Mostowski)

Parity games are determined, and every player has a positional winning strategy on her winning region.

**Theorem** (Emerson, Jutla, Sistla)

The parity game problem is in  $\text{NP} \cap \text{Co-NP}$

**Proof.** Guess positional strategies  $(W_0, F_0)$  and  $(W_1, F_1)$  for Players 0,1 with  $W_0 \cup W_1 = V$

Check that  $(W_\sigma, F_\sigma)$  is winning strategy for Player  $\sigma$ .

Actually the parity game problem is in  $\text{UP} \cap \text{Co-UP}$  (Jurdziński)

**Open problem:** Is it in  $\text{P}_{\text{TIME}}$  ?

# Complexity of parity games

The currently best deterministic algorithms are **polynomial** in the size of the game graph, but **exponential** in the number of priorities.

# Complexity of parity games

The currently best deterministic algorithms are **polynomial** in the size of the game graph, but **exponential** in the number of priorities.

Efficient algorithms for special cases of parity games

**Proposition** (Obdržalek)

Parity games on graphs of bounded tree width are solvable in polynomial time.

# Complexity of parity games

The currently best deterministic algorithms are **polynomial** in the size of the game graph, but **exponential** in the number of priorities.

Efficient algorithms for special cases of parity games

**Proposition** (Obdržalek)

Parity games on graphs of bounded tree width are solvable in polynomial time.

**Challenges:** Define appropriate complexity measures for directed graphs.

Do these measures lead to efficient solutions for parity games of bounded complexity?

# Complexity measures for directed graphs

**Intuition:** Directed acyclic graphs are simple

A graph is complicated if its cycles are highly intertwined.

# Complexity measures for directed graphs

**Intuition:** Directed acyclic graphs are simple

A graph is complicated if its cycles are highly intertwined.

**Observation.** While tree-width is a robust measure, that is “right” for a number of different questions concerning undirected graphs, its heritage for directed graphs is split among several contenders.

# Complexity measures for directed graphs

**Intuition:** Directed acyclic graphs are simple

A graph is complicated if its cycles are highly intertwined.

**Observation.** While tree-width is a robust measure, that is “right” for a number of different questions concerning undirected graphs, its heritage for directed graphs is split among several contenders.

- **Tree width** (of the associated undirected graph)
- **Directed tree width** (Johnson, Robertson, Seymour, Thomas)
- **Directed path width** (Barat)
- **DAG-width** (Berwanger, Dawar, Hunter, Kreutzer, Obdržálek)
- **Entanglement** (Berwanger, Grädel)
- ...

# Directed Tree Width

The success of tree width for divide-and-conquer algorithms has motivated a (not very intuitive) directed generalisation of **tree decomposition** that leads to a notion of **directed tree width**  $\text{dtw}(G)$ .

# Directed Tree Width

The success of tree width for divide-and-conquer algorithms has motivated a (not very intuitive) directed generalisation of **tree decomposition** that leads to a notion of **directed tree width**  $\text{dtw}(G)$ .

Characterisation in terms of the **cops-and-robber game**: When the cops move from a set  $X$  to a set  $X'$ , the robber can escape only by moving in a **strongly connected component** of  $G \setminus (X \cap X')$

# Directed Tree Width

The success of tree width for divide-and-conquer algorithms has motivated a (not very intuitive) directed generalisation of **tree decomposition** that leads to a notion of **directed tree width**  $\text{dtw}(G)$ .

Characterisation in terms of the **cops-and-robber game**: When the cops move from a set  $X$  to a set  $X'$ , the robber can escape only by moving in a **strongly connected component** of  $G \setminus (X \cap X')$

**Theorem.** Hamiltonicity is efficiently decidable on graphs of bounded directed tree width.

# Directed Tree Width

The success of tree width for divide-and-conquer algorithms has motivated a (not very intuitive) directed generalisation of **tree decomposition** that leads to a notion of **directed tree width**  $\text{dtw}(G)$ .

Characterisation in terms of the **cops-and-robber game**: When the cops move from a set  $X$  to a set  $X'$ , the robber can escape only by moving in a **strongly connected component** of  $G \setminus (X \cap X')$

**Theorem.** Hamiltonicity is efficiently decidable on graphs of bounded directed tree width.

Unfortunately, not many algorithmic results for directed tree width.

It could not be settled yet whether **parity games** of bounded directed tree width are solvable in polynomial time.

# DAG Width

DAG: directed acyclic graph

DAG-width measures how closely a graph resembles a DAG.

It can be defined either by a variant of the **cops-and-robber game**, or in terms of **dag-decompositions**.

# DAG Width

DAG: directed acyclic graph

DAG-width measures how closely a graph resembles a DAG.

It can be defined either by a variant of the **cops-and-robber game**, or in terms of **dag-decompositions**.

The definition in terms of the cops and robber game is analogous to the one for tree width, except that the **robber** has to move along **directed paths**, and that **cop-strategies** are required to be **monotone**.

$\text{dag}(G) := \min\{k : k \text{ cops have a monotone strategy to catch the robber on } G\}$

# DAG Decompositions

For two nodes  $d, e$  of a DAG  $D$ , we write  $d \preceq e$  if there is a path from  $d$  to  $e$ .

A **DAG-decomposition** of width  $k$  of a directed graph  $G = (V, E)$  is a tuple  $(D, (X_d)_{d \in D})$  where  $D$  is a dag, and each  $X_d$  is a set of at most  $k$  nodes of  $V$ , such that

- every node of  $v$  lives as some node of  $D$ :  $\bigcup_{d \in D} X_d = V$
- for every arc  $(d, e)$  of  $D$ , all edges of  $G$  that leave  $\bigcup_{e \preceq f} X_f \setminus X_d$  end in  $X_d \cap X_e$
- for every  $\preceq$ -minimal element  $d$  of  $D$ , there is no edge of  $G$  that leaves  $\bigcup_{d \preceq e} X_e$ .
- if  $d \preceq e \preceq f$ , then  $X_d \cap X_f \subseteq X_e$ .

$$\text{dag}(G) := \min\{k : G \text{ admits a DAG-decomposition of width } k\}$$

# Entanglement

**Entanglement game:**  $k$  detectives try to catch a thief on  $G$

The thief selects an initial position  $v_0 \in G$

The detectives are outside the graph

# Entanglement

**Entanglement game:**  $k$  detectives try to catch a thief on  $G$

The thief selects an initial position  $v_0 \in G$

The detectives are outside the graph

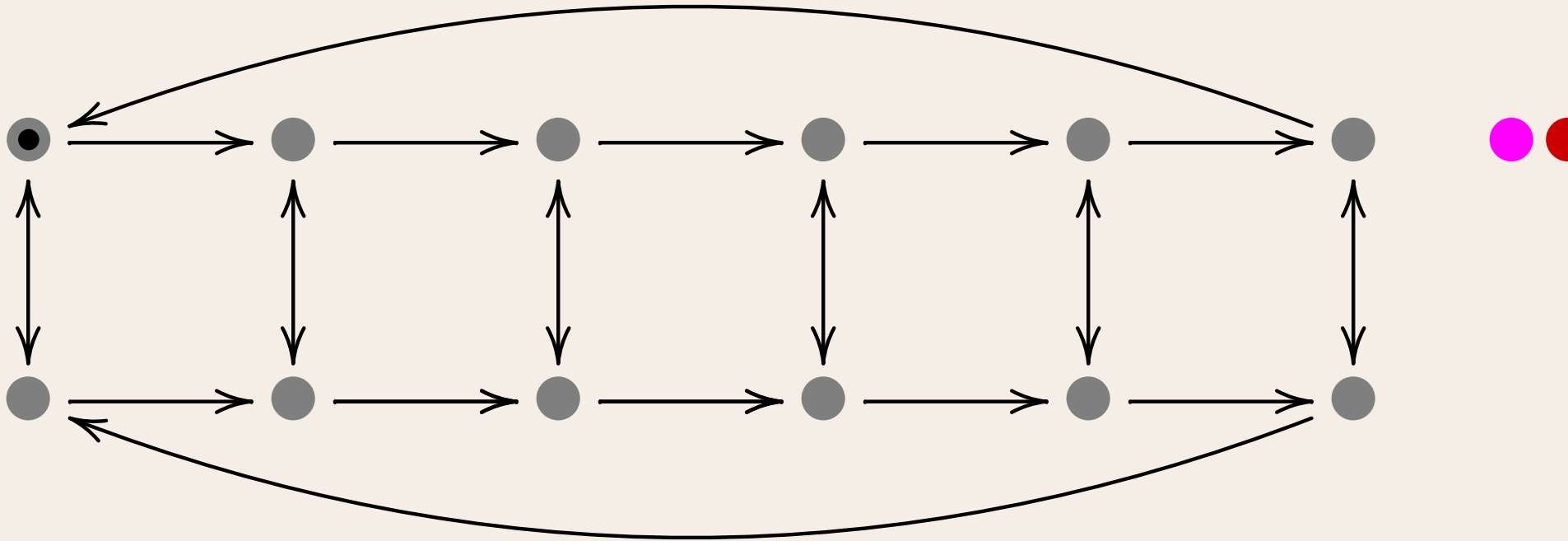
**Move:**

- the detectives stay where they are, or place one of them on the current position  $v$  of the thief.
- the thief moves to a successor  $w \in vE$  not occupied by a detective.  
If no such position exists, the thief is caught and the detectives have won.

$$\text{ent}(G) := \min\{k : k \text{ detectives have a strategy to catch the thief on } G\}$$

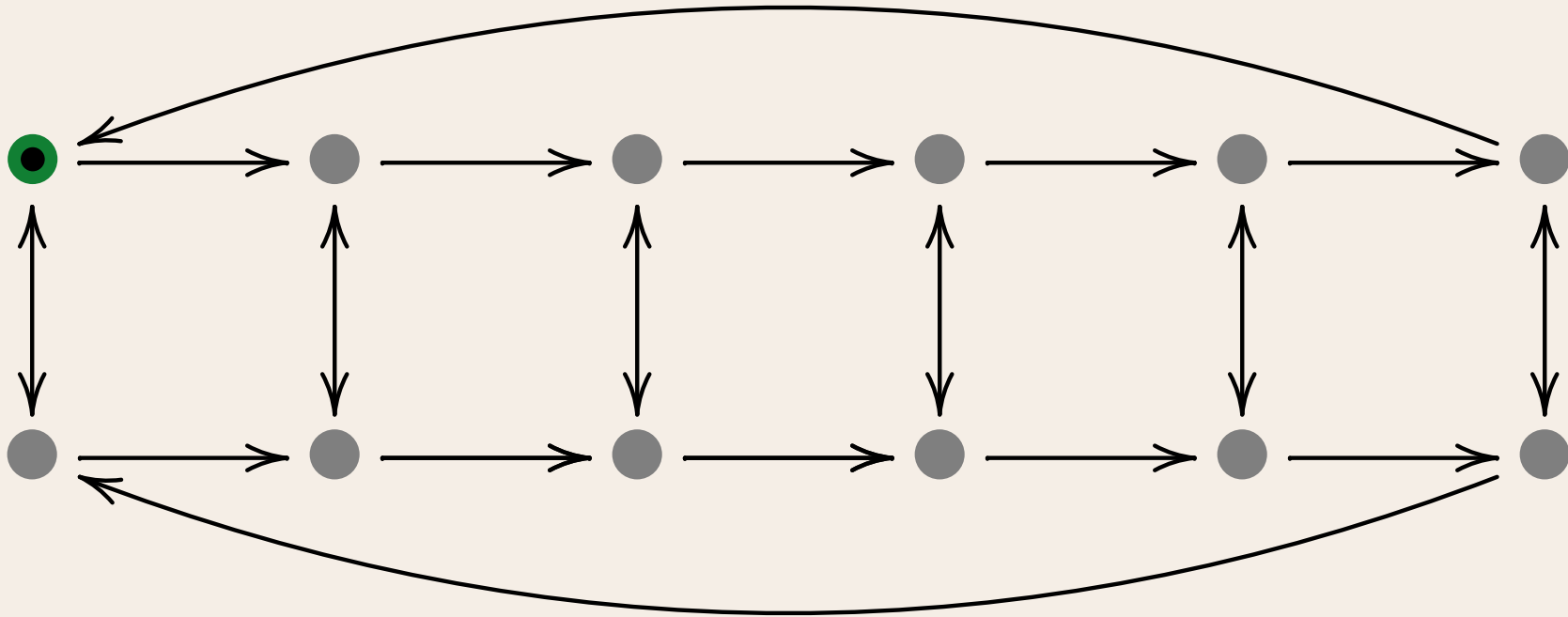
# Entanglement: Examples

$T_{mn}$ :  $(m \times n)$ -torus



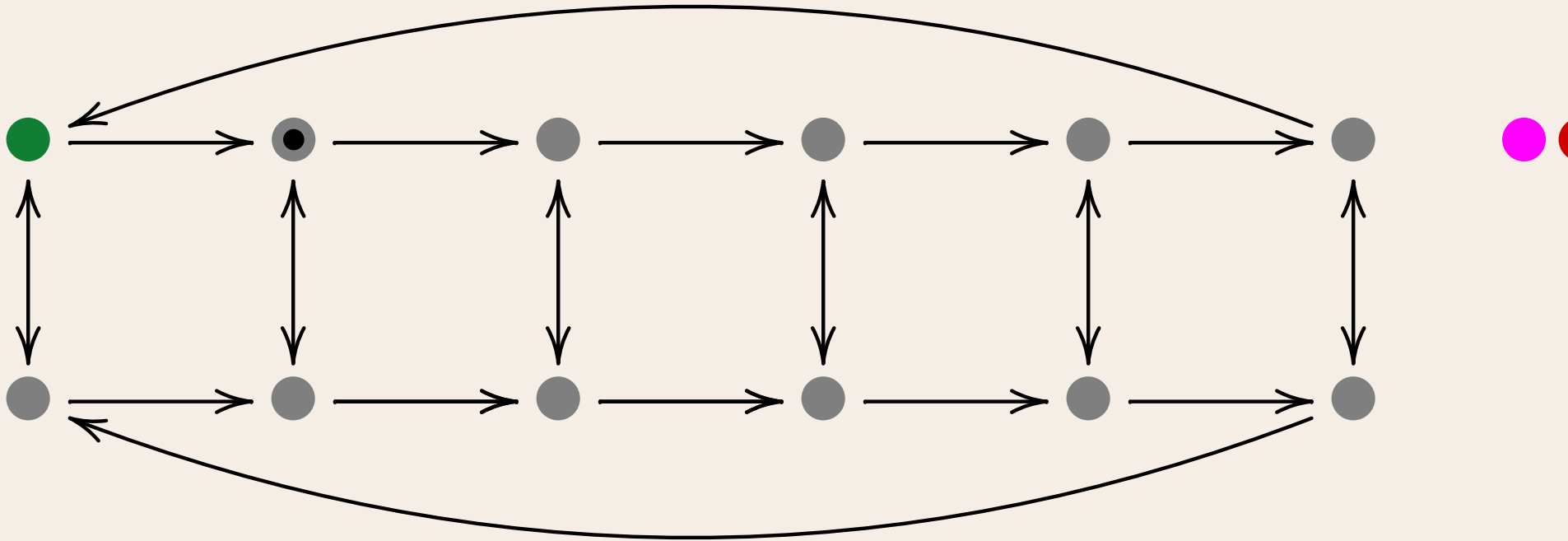
# Entanglement: Examples

$T_{mn}$ :  $(m \times n)$ -torus



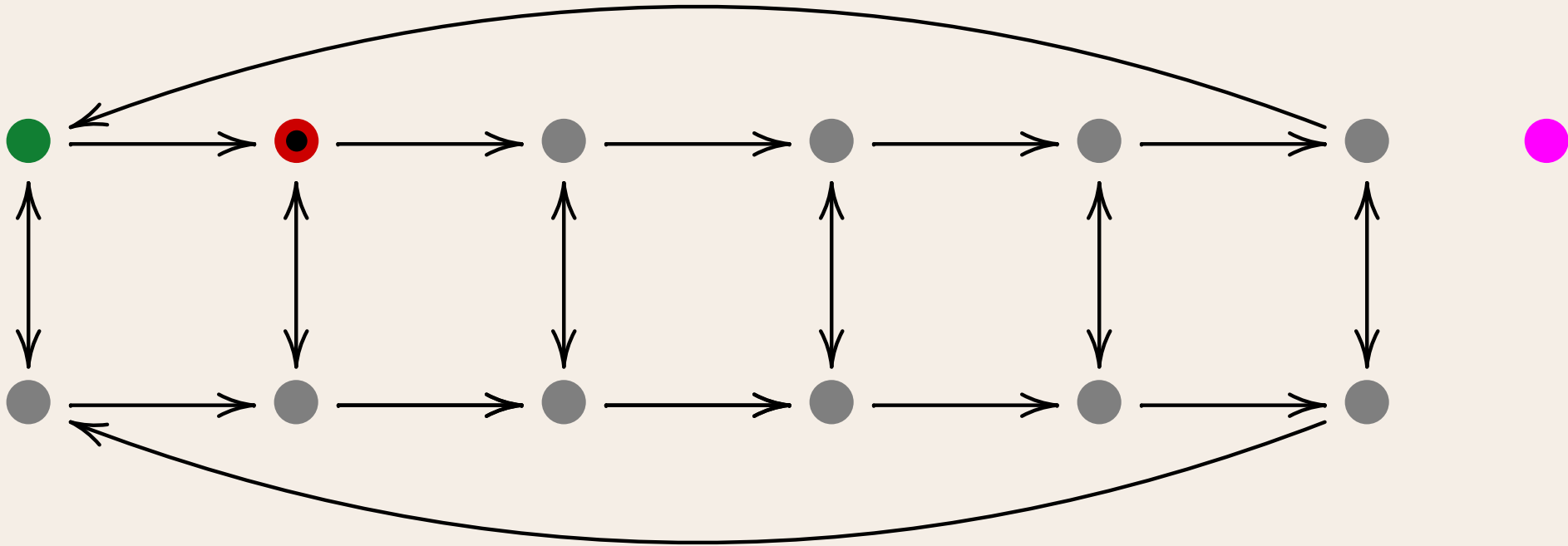
# Entanglement: Examples

$T_{mn}$ :  $(m \times n)$ -torus



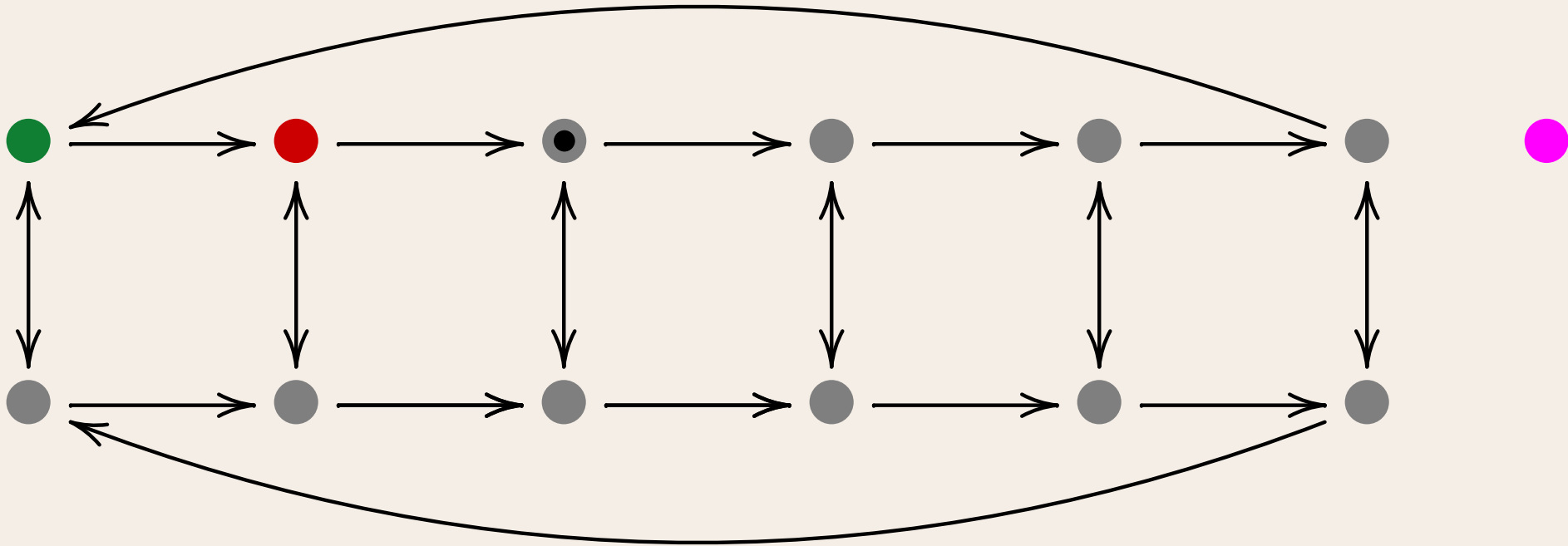
# Entanglement: Examples

$T_{mn}$ :  $(m \times n)$ -torus



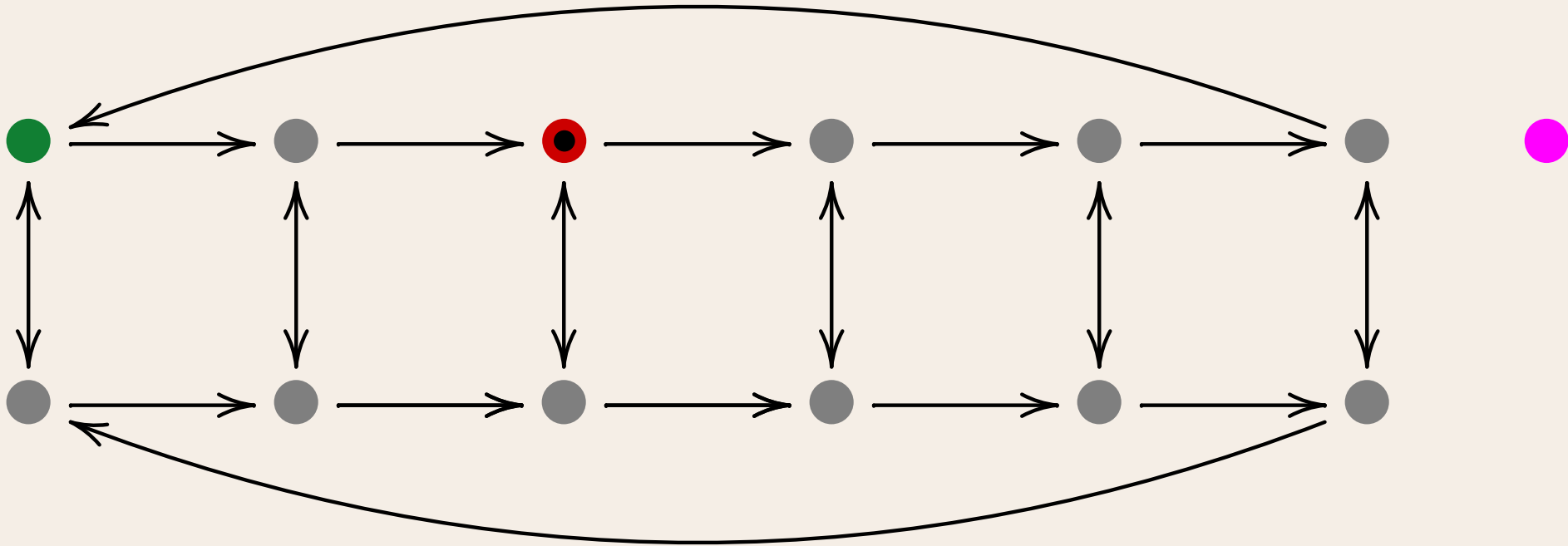
# Entanglement: Examples

$T_{mn}$ :  $(m \times n)$ -torus



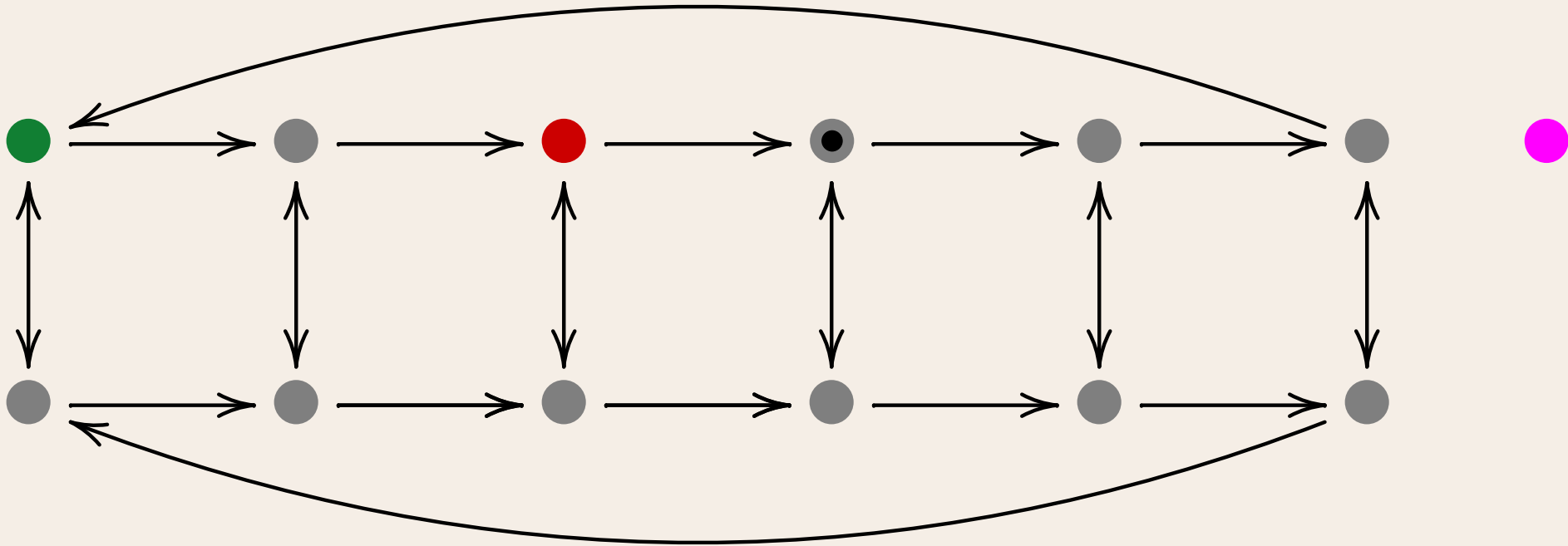
# Entanglement: Examples

$T_{mn}$ :  $(m \times n)$ -torus



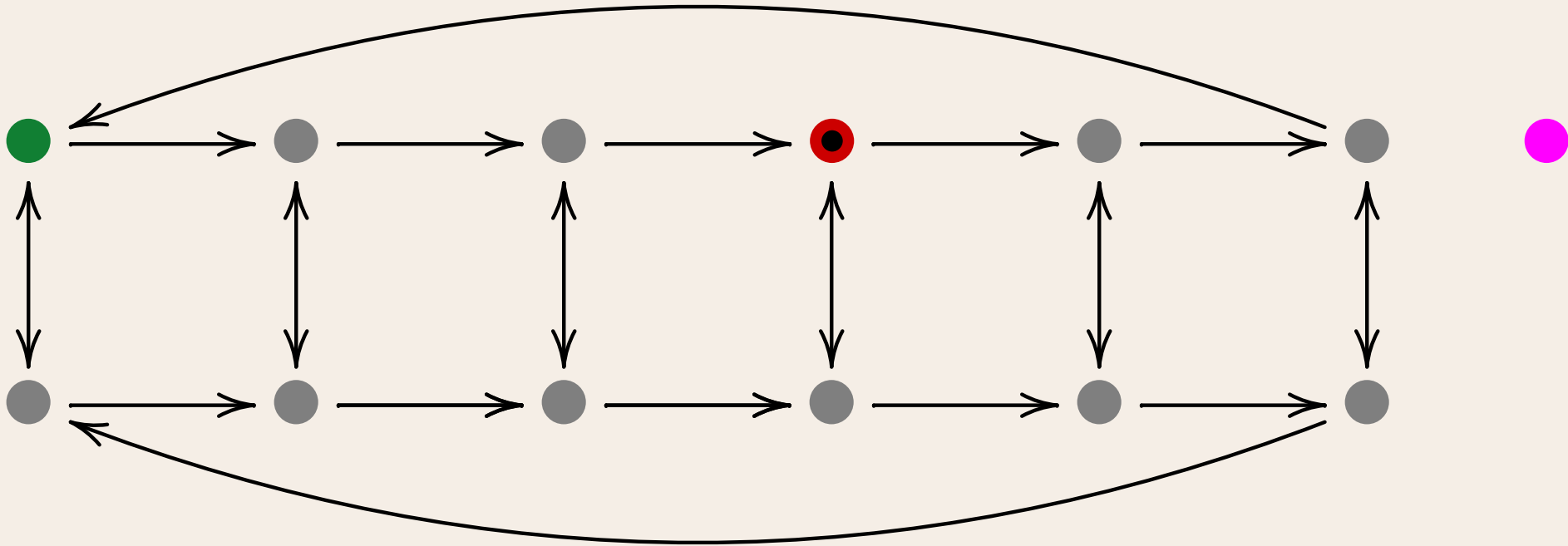
# Entanglement: Examples

$T_{mn}$ :  $(m \times n)$ -torus



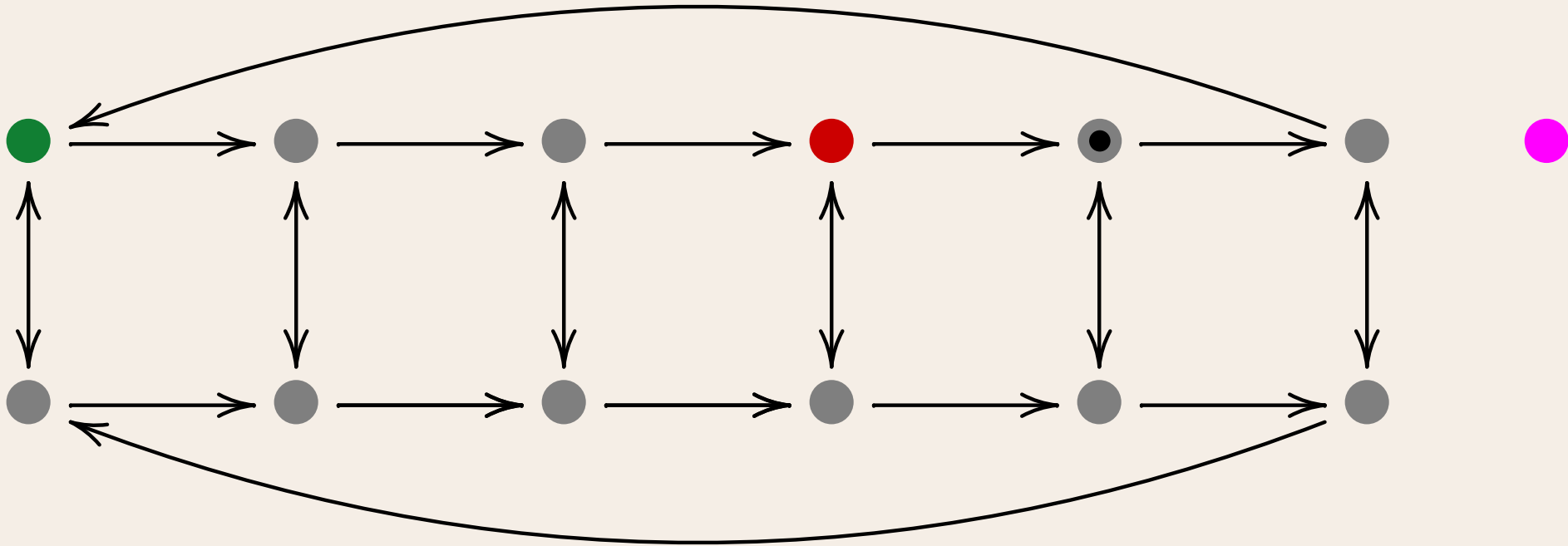
# Entanglement: Examples

$T_{mn}$ :  $(m \times n)$ -torus



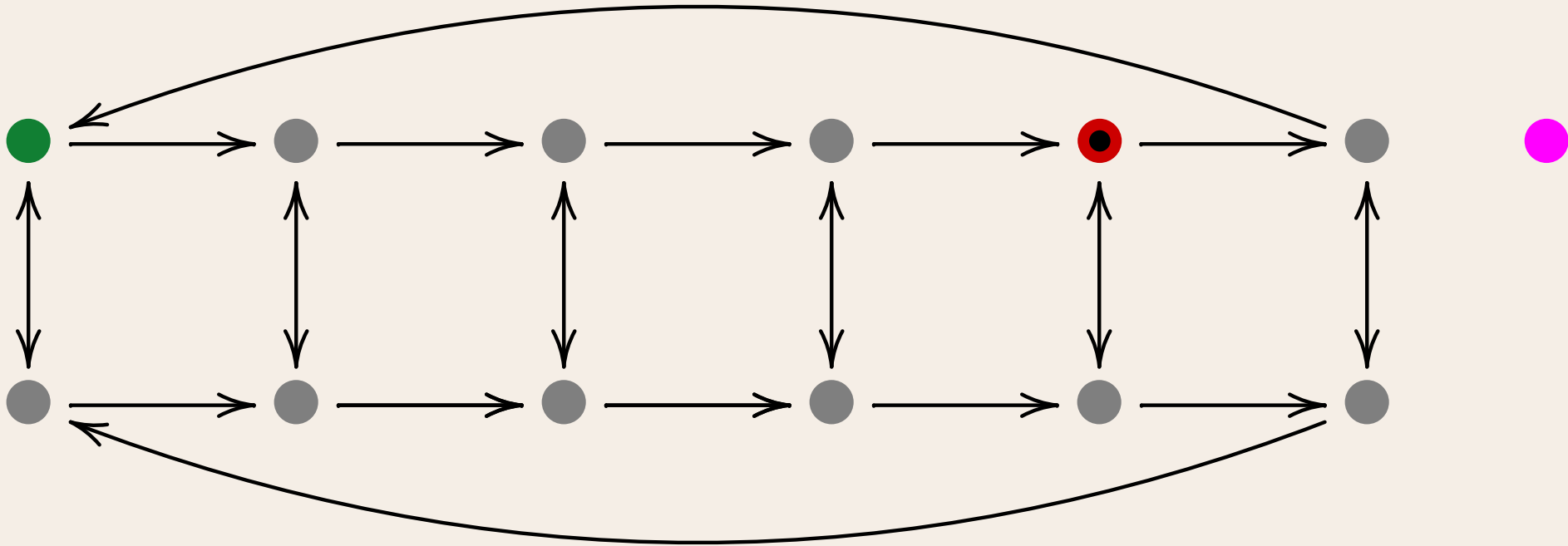
# Entanglement: Examples

$T_{mn}$ :  $(m \times n)$ -torus



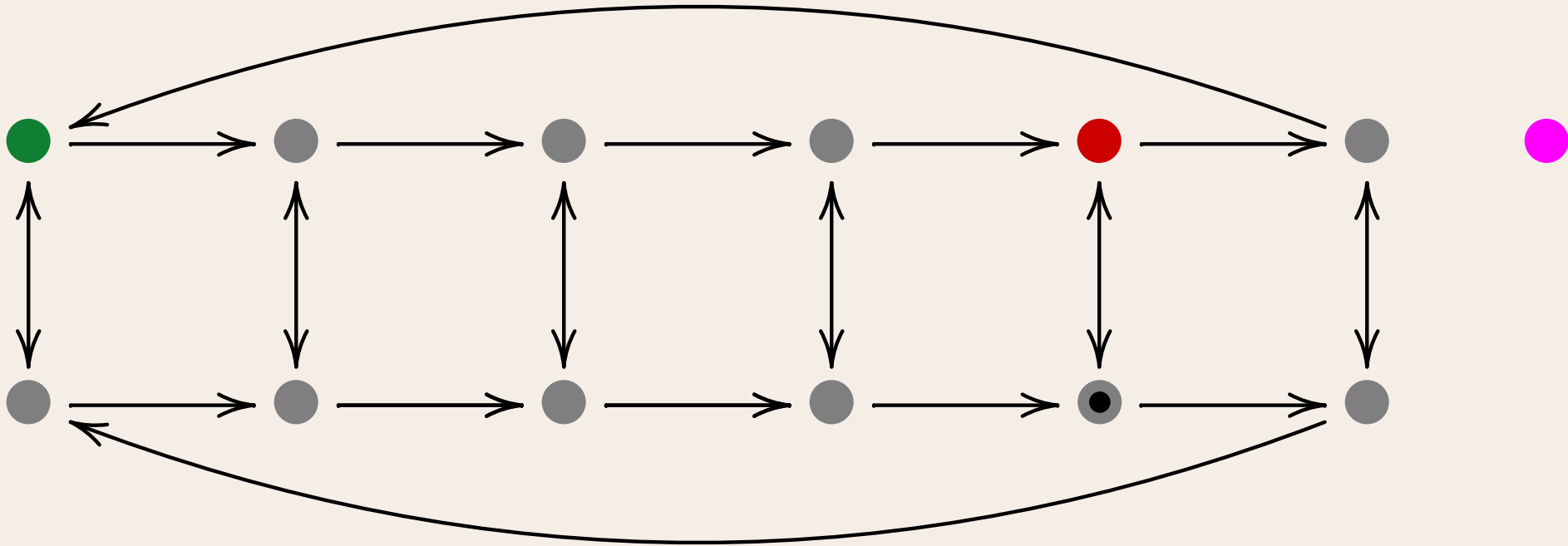
# Entanglement: Examples

$T_{mn}$ :  $(m \times n)$ -torus



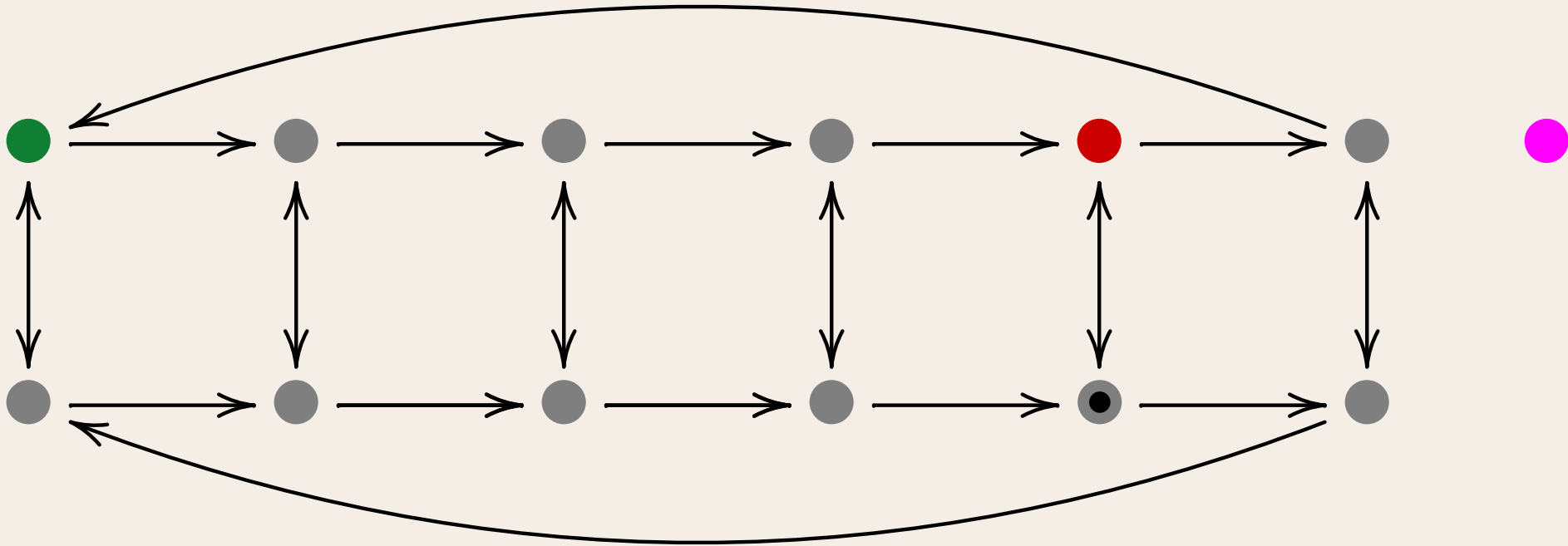
# Entanglement: Examples

$T_{mn}$ :  $(m \times n)$ -torus



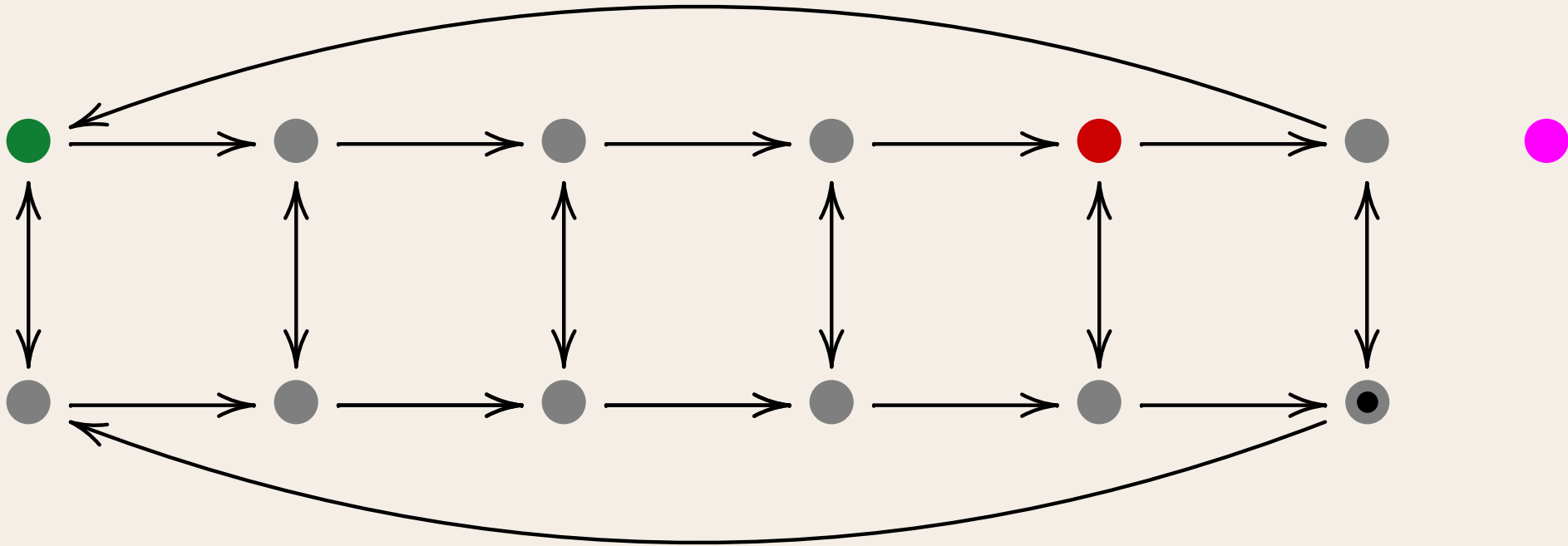
# Entanglement: Examples

$T_{mn}$ :  $(m \times n)$ -torus



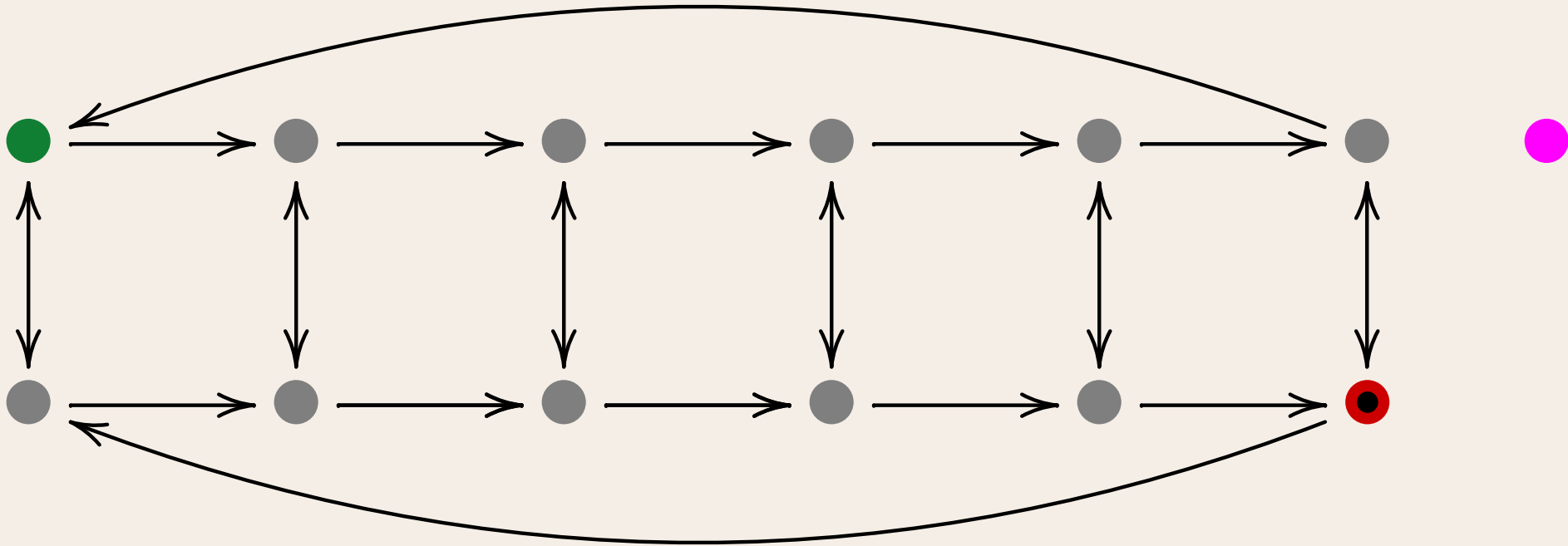
# Entanglement: Examples

$T_{mn}$ :  $(m \times n)$ -torus



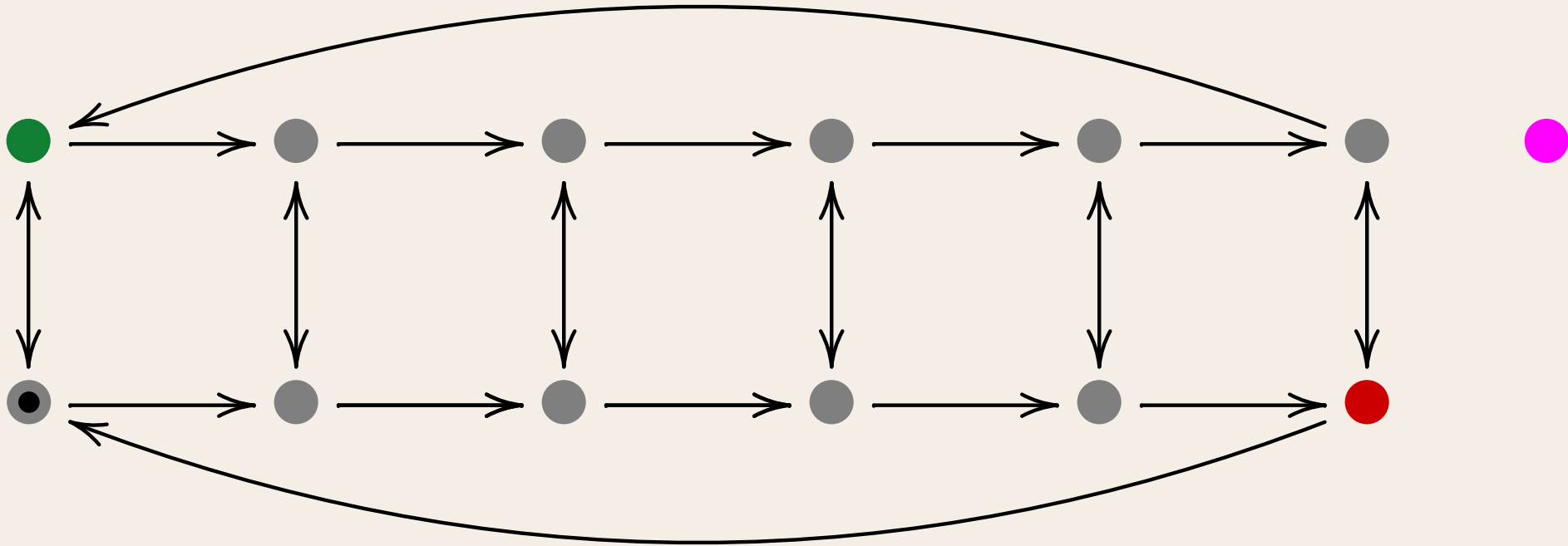
# Entanglement: Examples

$T_{mn}$ :  $(m \times n)$ -torus



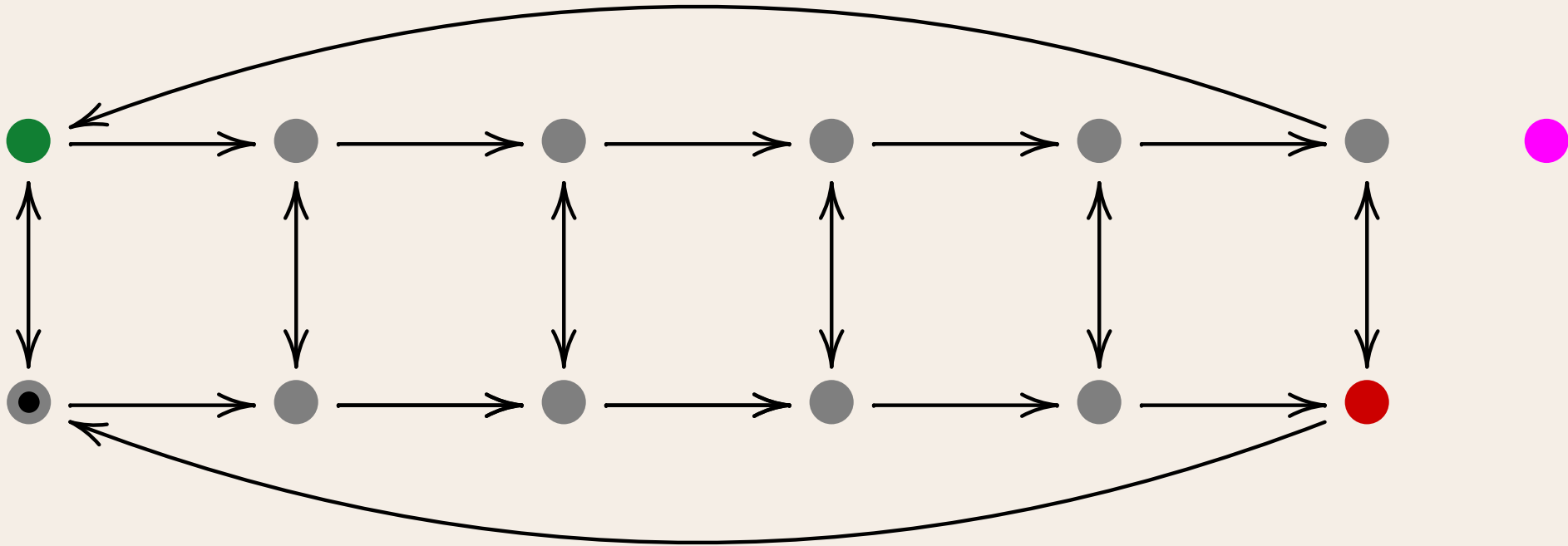
# Entanglement: Examples

$T_{mn}$ :  $(m \times n)$ -torus



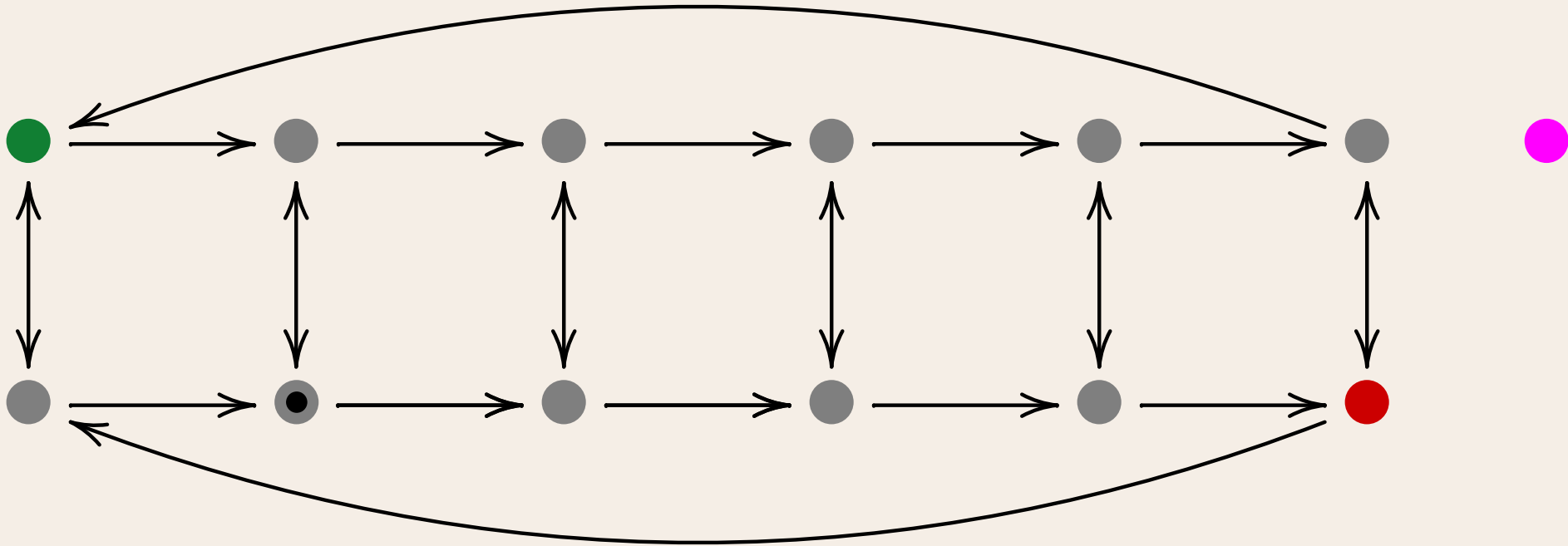
# Entanglement: Examples

$T_{mn}$ :  $(m \times n)$ -torus



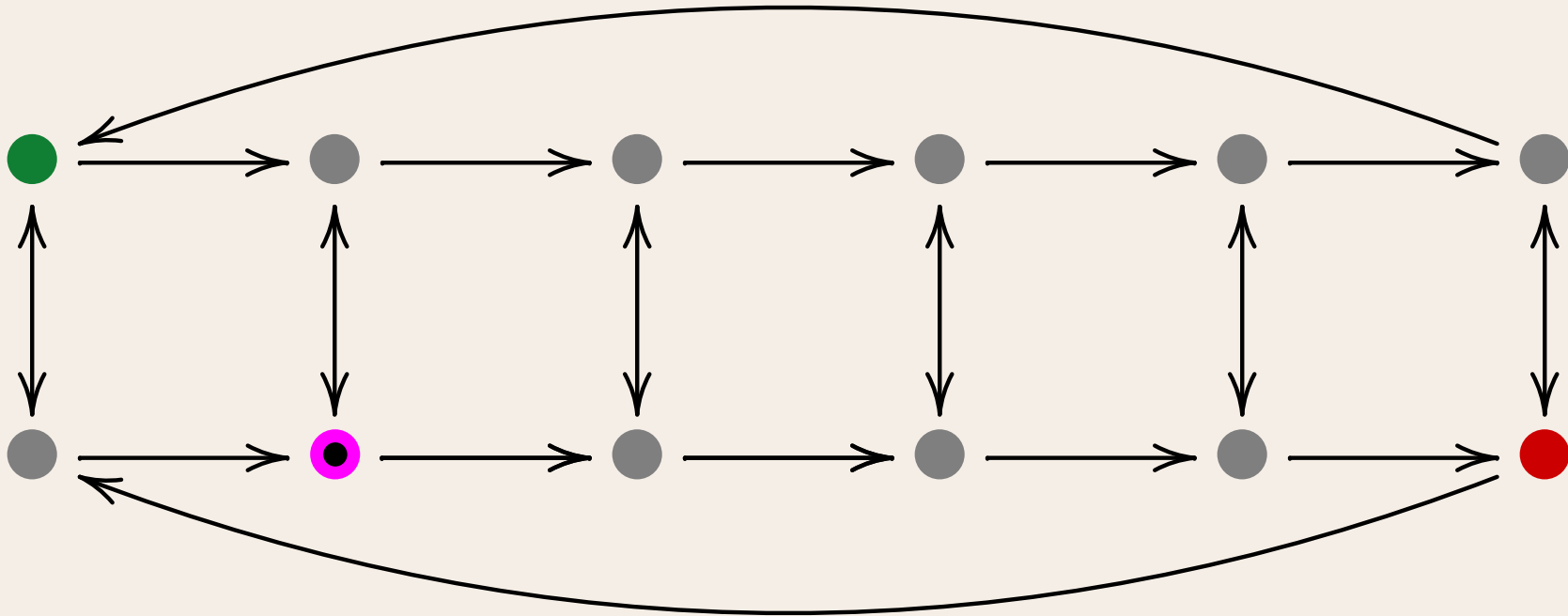
# Entanglement: Examples

$T_{mn}$ :  $(m \times n)$ -torus



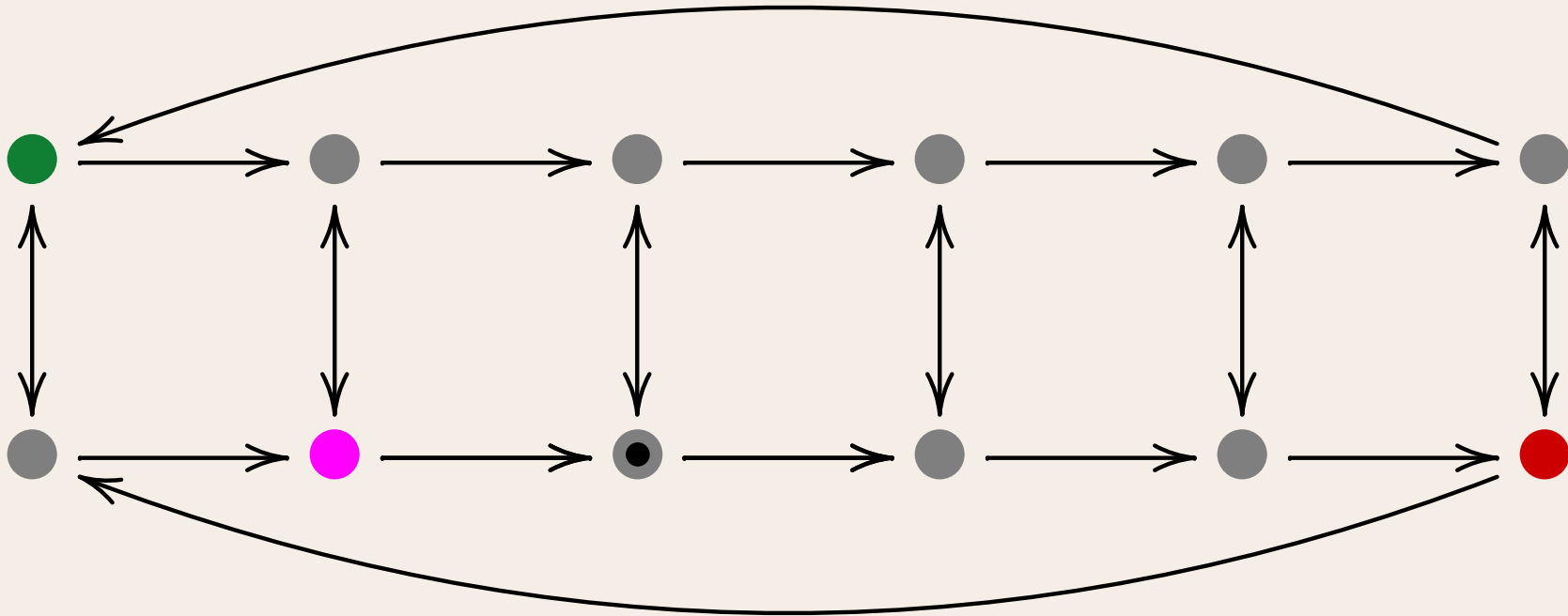
# Entanglement: Examples

$T_{mn}$ :  $(m \times n)$ -torus



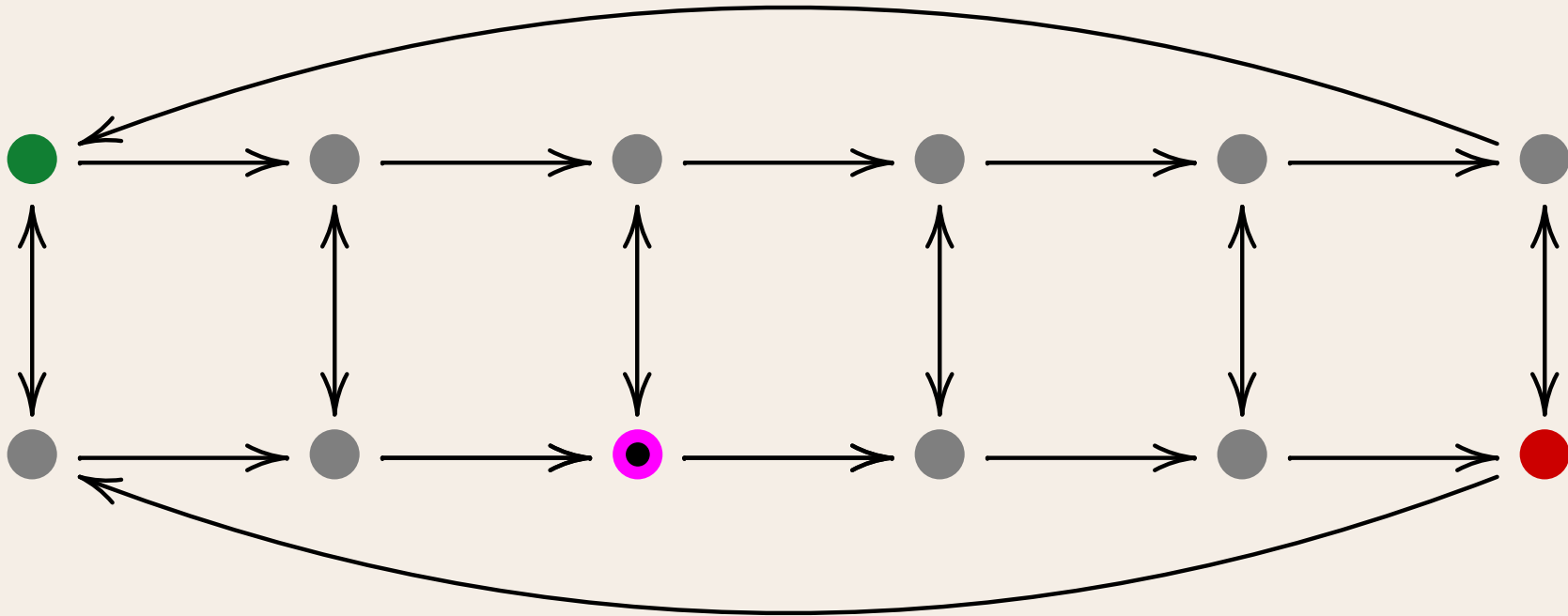
# Entanglement: Examples

$T_{mn}$ :  $(m \times n)$ -torus



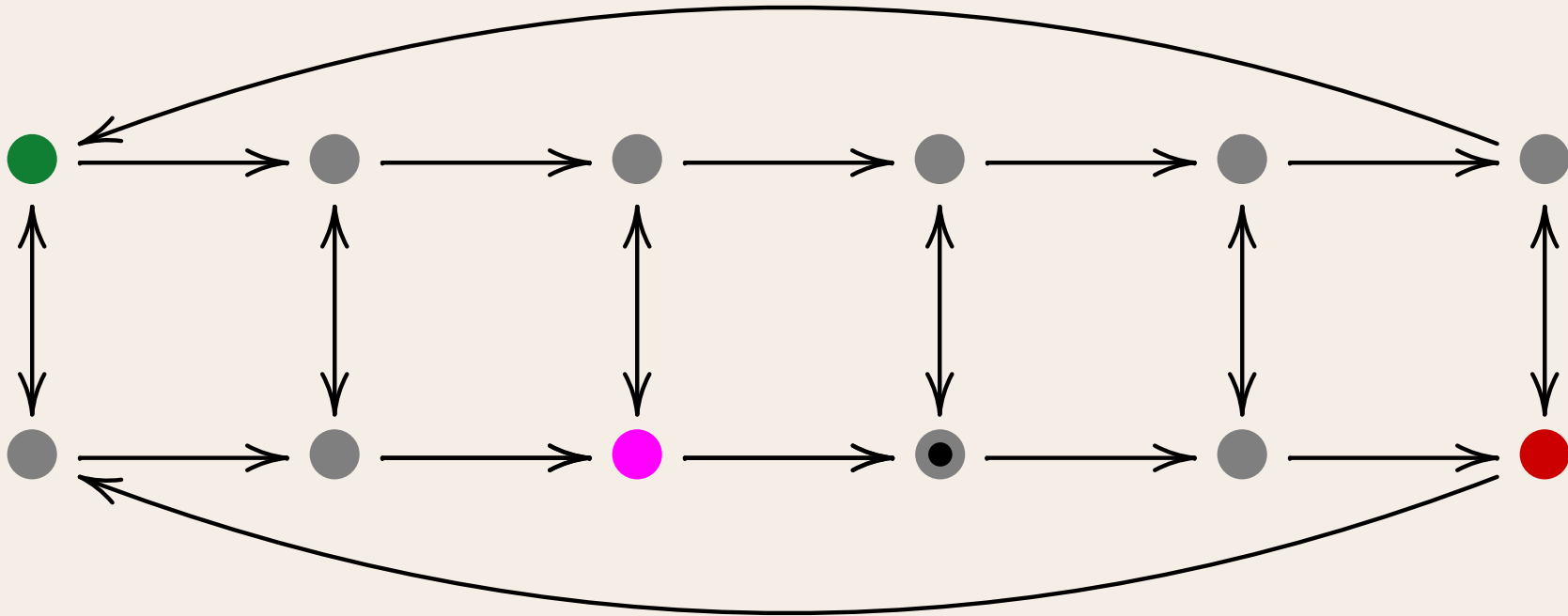
# Entanglement: Examples

$T_{mn}$ :  $(m \times n)$ -torus



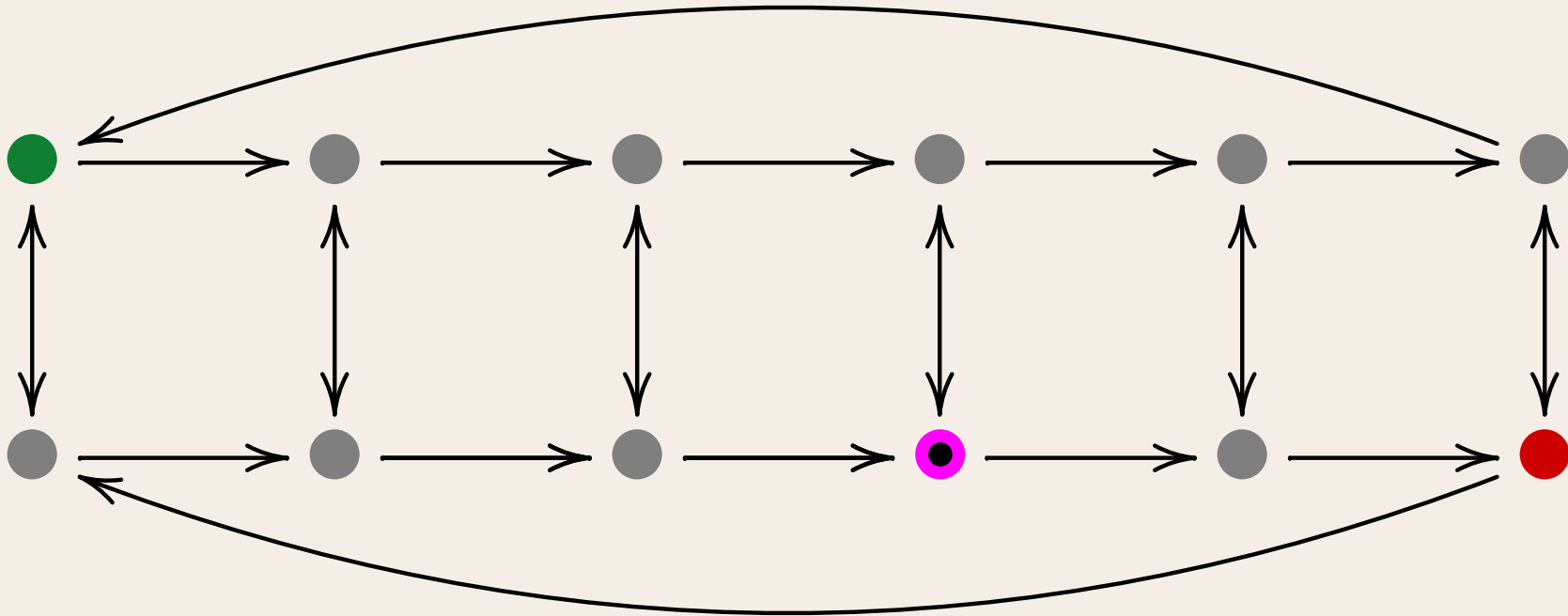
# Entanglement: Examples

$T_{mn}$ :  $(m \times n)$ -torus



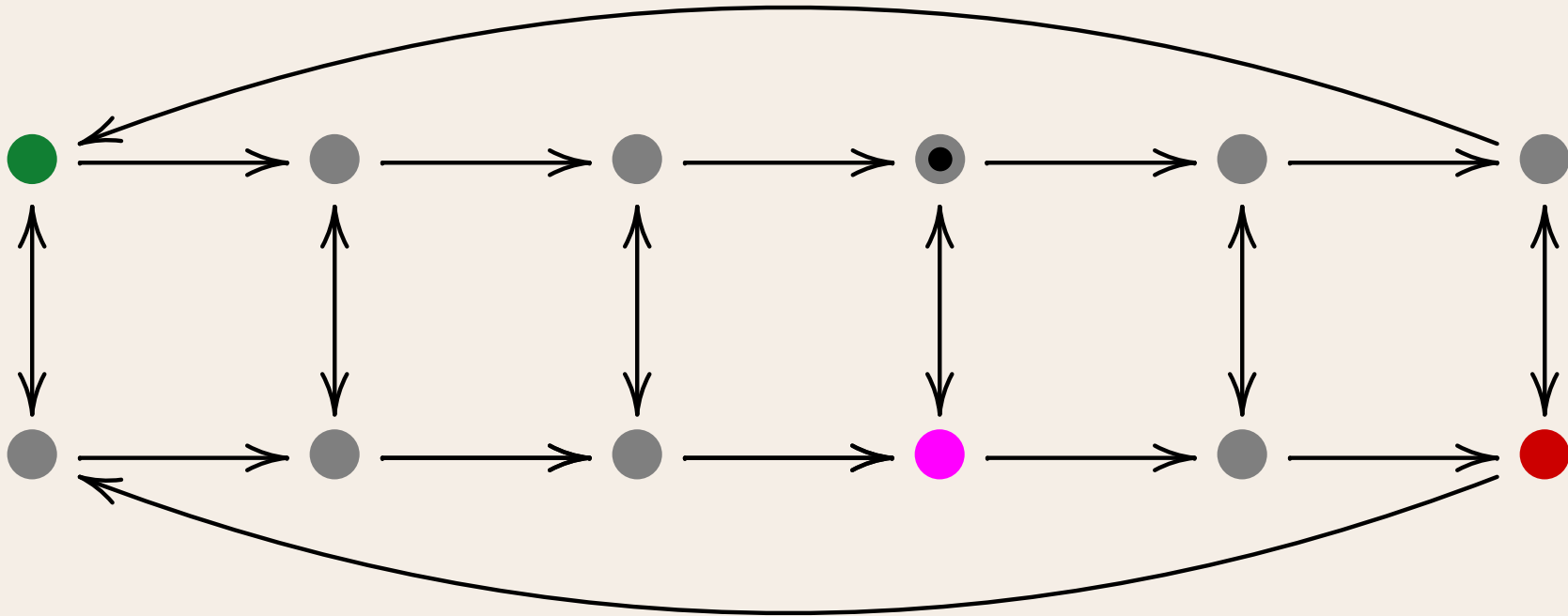
# Entanglement: Examples

$T_{mn}$ :  $(m \times n)$ -torus



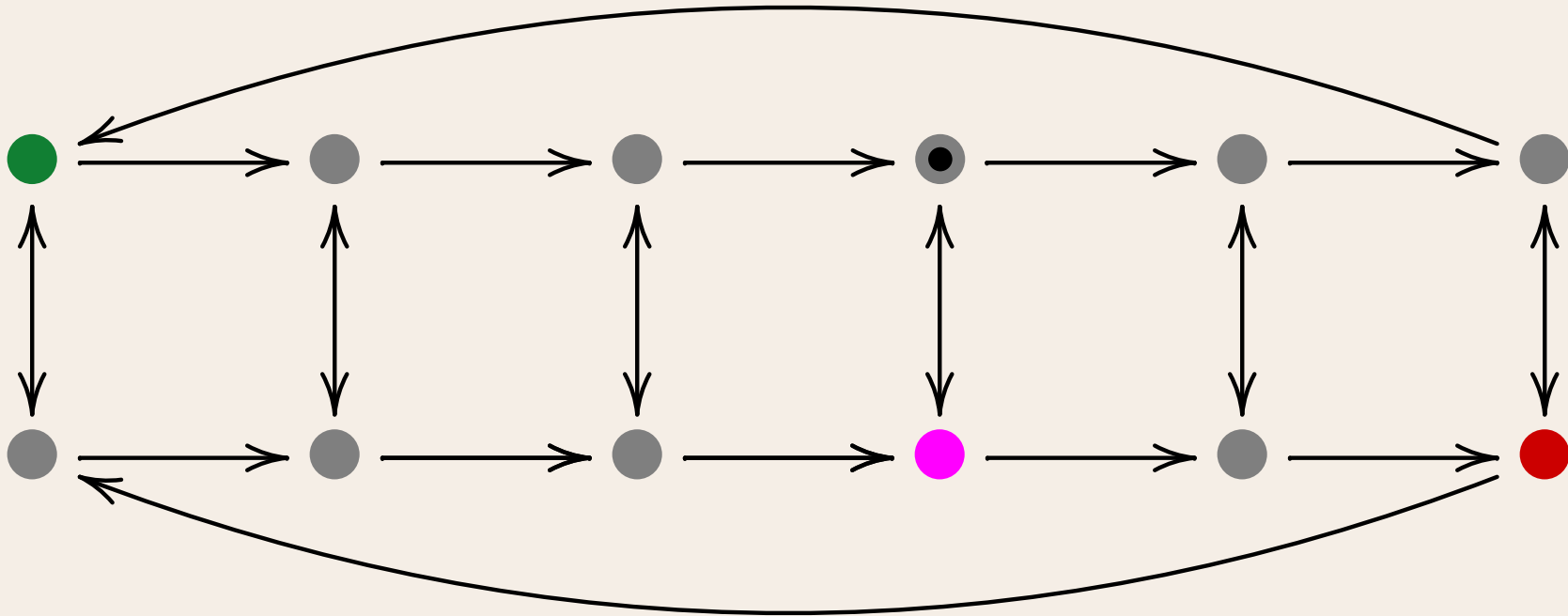
# Entanglement: Examples

$T_{mn}$ :  $(m \times n)$ -torus



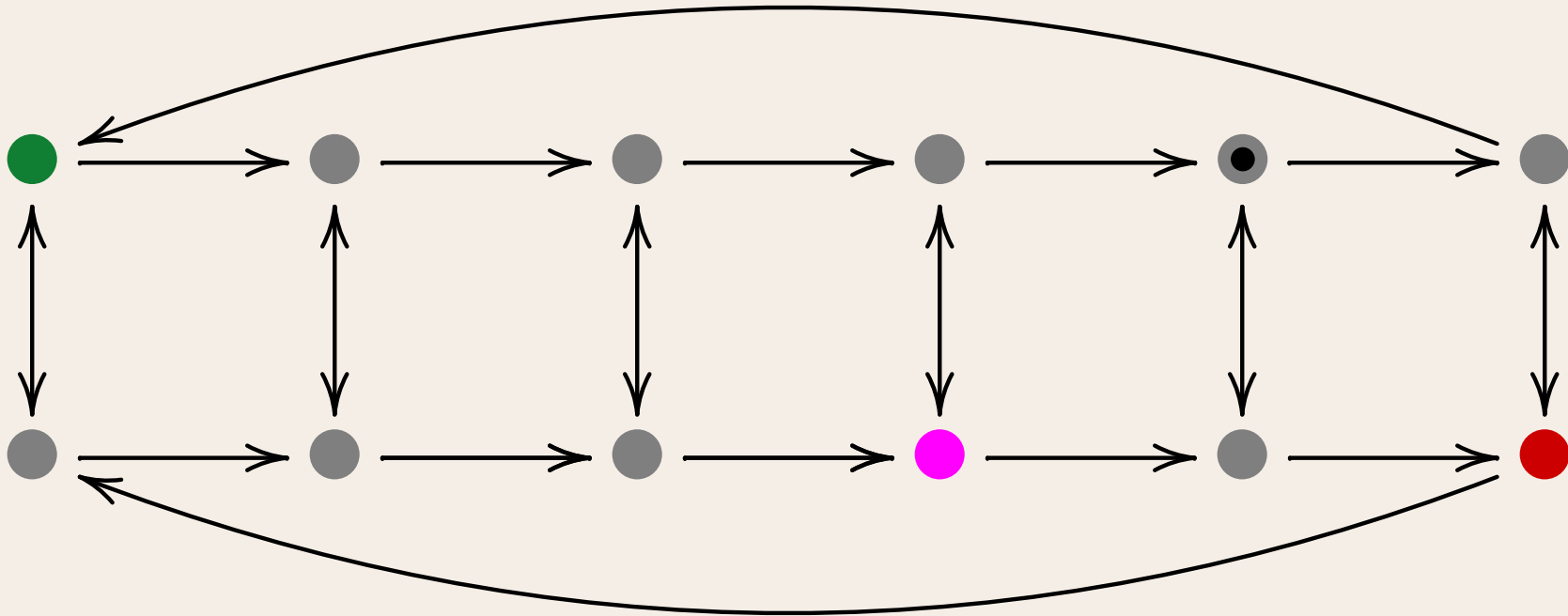
# Entanglement: Examples

$T_{mn}$ :  $(m \times n)$ -torus



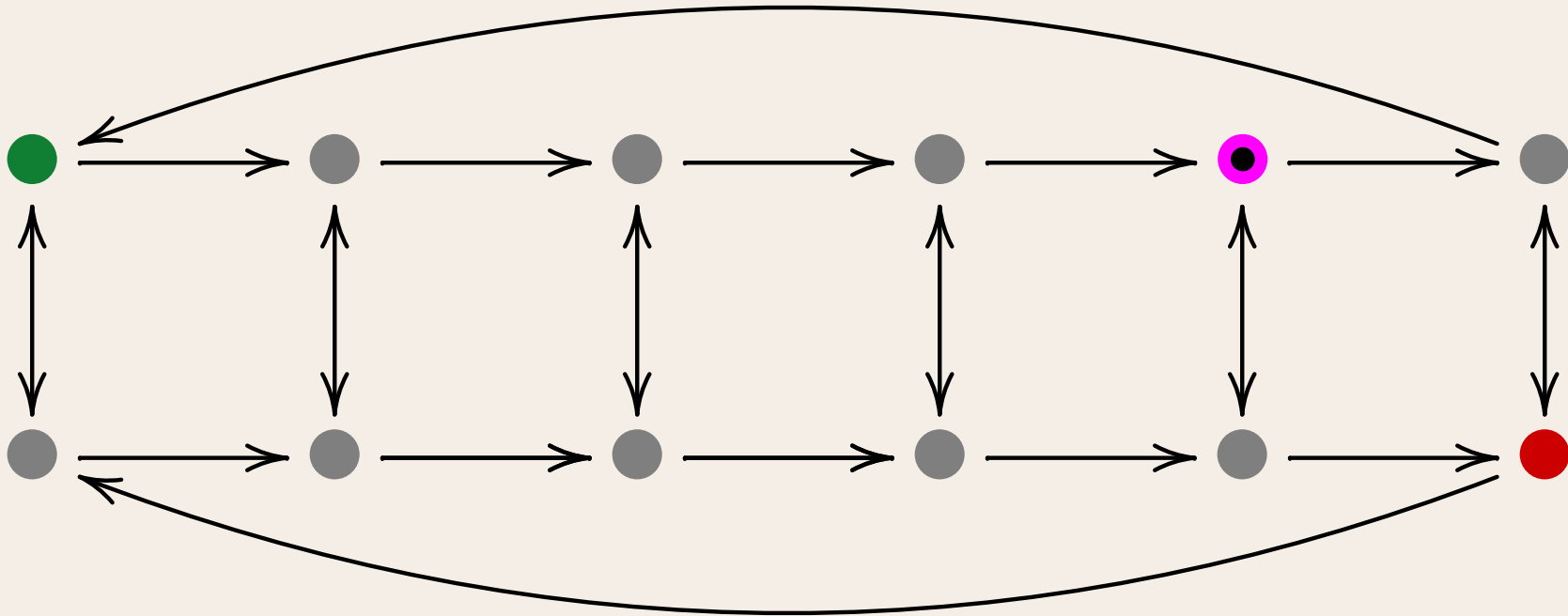
# Entanglement: Examples

$T_{mn}$ :  $(m \times n)$ -torus



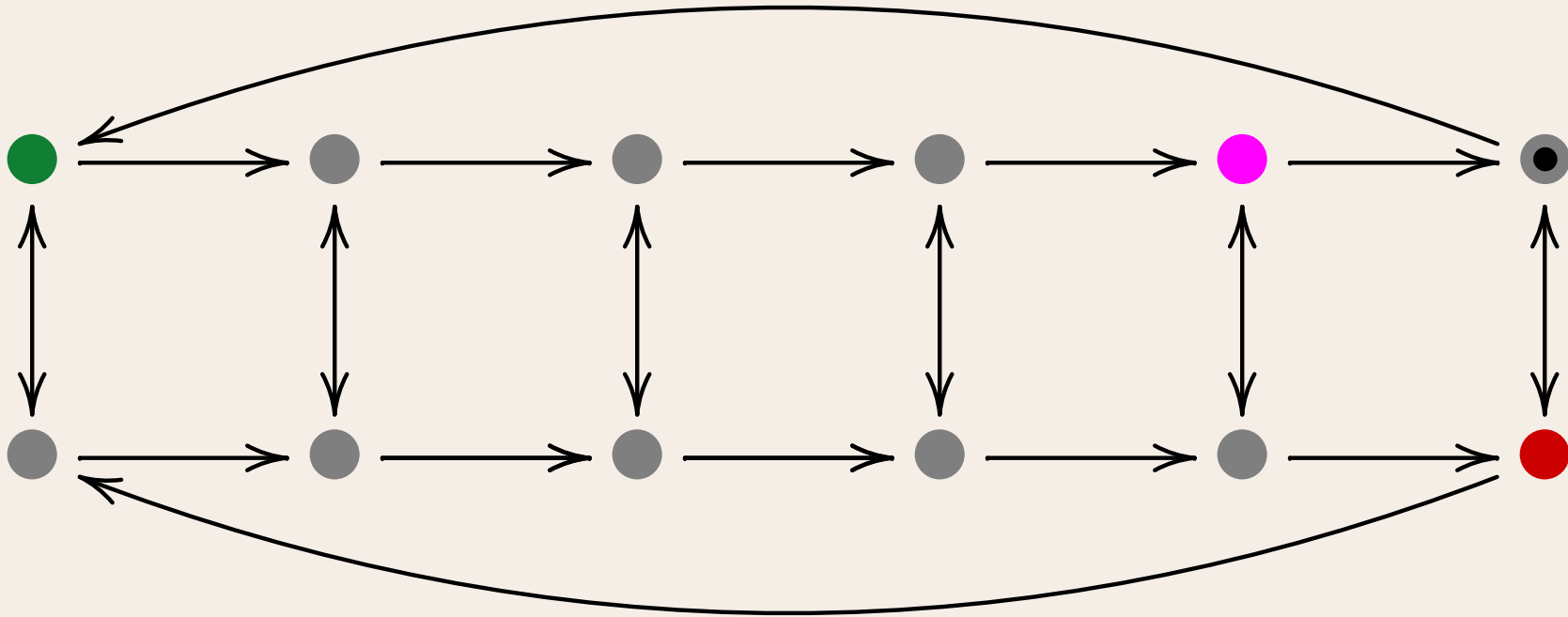
# Entanglement: Examples

$T_{mn}$ :  $(m \times n)$ -torus



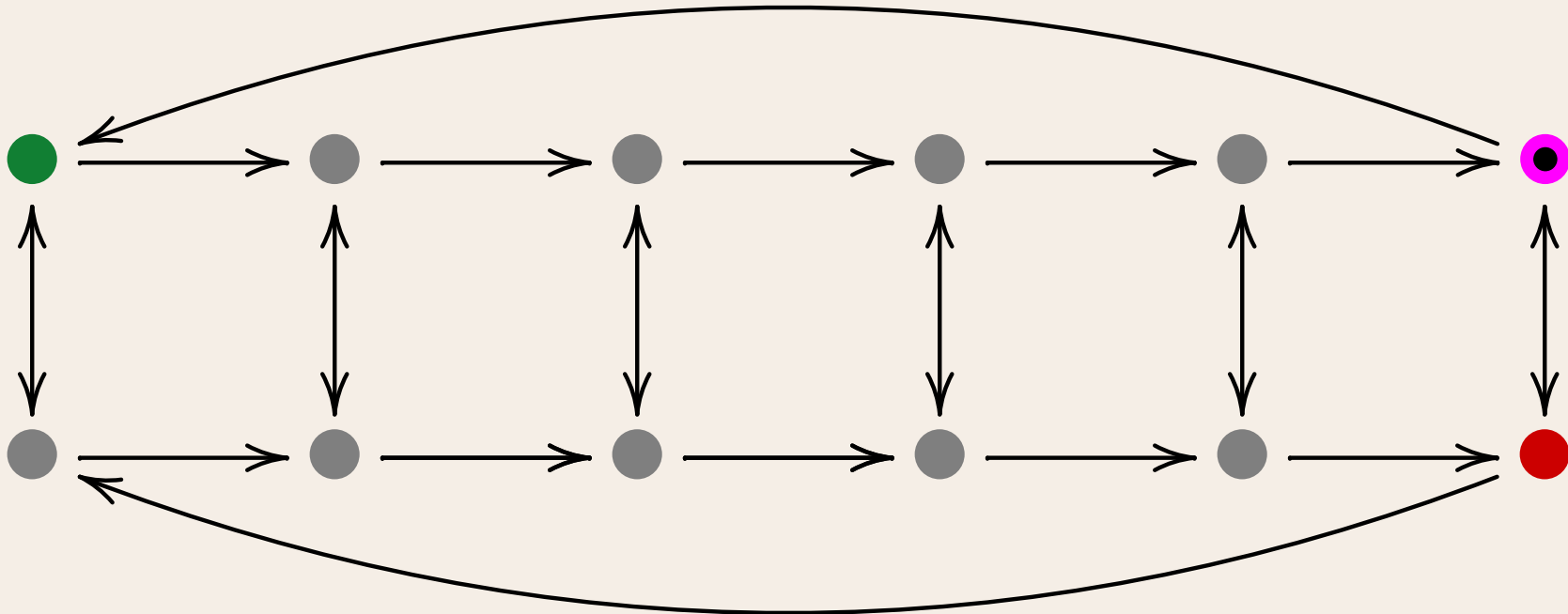
# Entanglement: Examples

$T_{mn}$ :  $(m \times n)$ -torus



# Entanglement: Examples

$T_{mn}$ :  $(m \times n)$ -torus



**Proposition** Let  $m \neq n$

- (1)  $\text{ent}(T_{mn}) = \min(m, n) + 1$  and  $\text{dag}(T_{mn}) = \min(m, n) + 2$ .
- (2)  $\text{ent}(T_{nn}) = n$  and  $\text{dag}(T_{nn}) = n + 1$ .

# Comparison of different complexity measures

## Proposition.

- (1)  $\text{ent}(G) = 0 \iff \text{dag}(G) = 1 \iff G$  is acyclic.
- (2) If  $G$  is the graph of a unary function, then  $\text{ent}(G) = 1$  and  $\text{dag}(G) = 2$ .
- (3) If  $G$  is an undirected (nontrivial) tree, then  $\text{ent}(G) = \text{dag}(G) = 2$ .
- (4) The fully connected directed graph with  $n$  nodes has entanglement  $n$  and DAG-width  $n$ .

# Comparison of different complexity measures

## Proposition.

- (1)  $\text{ent}(G) = 0 \iff \text{dag}(G) = 1 \iff G$  is acyclic.
- (2) If  $G$  is the graph of a unary function, then  $\text{ent}(G) = 1$  and  $\text{dag}(G) = 2$ .
- (3) If  $G$  is an undirected (nontrivial) tree, then  $\text{ent}(G) = \text{dag}(G) = 2$ .
- (4) The fully connected directed graph with  $n$  nodes has entanglement  $n$  and DAG-width  $n$ .

## Theorem. (Berwanger, Dawar, Grädel, Hunter, Kreutzer)

- (1)  $\text{dtw}(G) \leq \text{dag}(G) - 1 \leq \text{ent}(G) \leq (\text{dag}(G) + 1) \log |G|$
- (2) There exist graphs of directed tree-width two, but arbitrarily large DAG-width
- (3) There exist graphs of tree-width two, DAG-width two, but arbitrarily large entanglement
- (4) There exist graphs of entanglement 0, but arbitrarily large tree-width.

# Highly entangled graphs with small dag width and tree width

Construction of graphs  $G(2, k)$ :

$\mathcal{T}_k^\downarrow$  (resp.  $\mathcal{T}_k^\uparrow$ ): full binary tree of depth  $k$ , oriented downwards (upwards)

$G(2, k) := \mathcal{T}_k^\downarrow \cup \mathcal{T}_k^\uparrow$  with appropriate cross-edges:

# Highly entangled graphs with small dag width and tree width

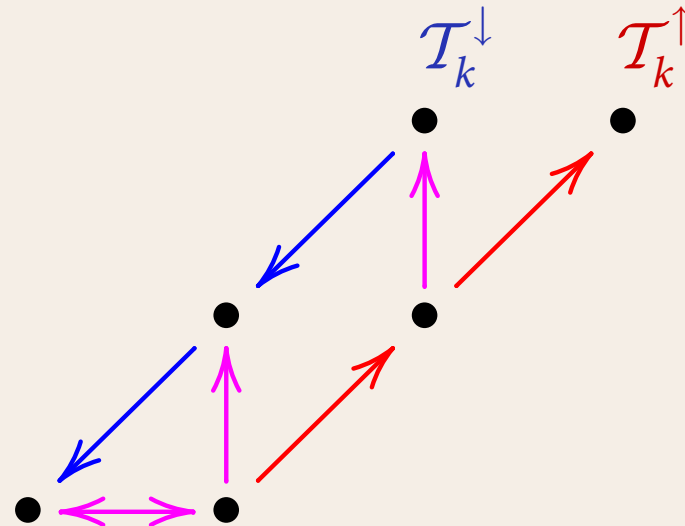
Construction of graphs  $G(2, k)$ :

$\mathcal{T}_k^\downarrow$  (resp.  $\mathcal{T}_k^\uparrow$ ): full binary tree of depth  $k$ , oriented downwards (upwards)

$G(2, k) := \mathcal{T}_k^\downarrow \cup \mathcal{T}_k^\uparrow$  with appropriate cross-edges:

Connect each leaf with its double in both directions.

Connect each node in  $\mathcal{T}_k^\uparrow$  to the double of its father:



# Highly entangled graphs with small dag width and tree width

$G(2, k)$  has tree width 2: Three cops win the cops-and-robber game.

# Highly entangled graphs with small dag width and tree width

$G(2, k)$  has tree width 2: Three cops win the cops-and-robber game.

$G(2, k)$  has dag width 2: Two cops win the directed cops-and-robber game.

# Highly entangled graphs with small dag width and tree width

$G(2, k)$  has tree width 2: Three cops win the cops-and-robber game.

$G(2, k)$  has dag width 2: Two cops win the directed cops-and-robber game.

$G(2, k)$  has entanglement  $> k$ : Call a path **free** if all nodes on the path and all their doubles are unguarded by the detectives. Call a node **blocked** if both the node and its double are guarded. The thief escapes against  $k$  detectives with the following strategy:

From a leaf  $v$  go to an ancestor  $u$  of  $v$  from which there is a free path to a leaf. Go to that leaf.

# Highly entangled graphs with small dag width and tree width

$G(2, k)$  has tree width 2: Three cops win the cops-and-robber game.

$G(2, k)$  has dag width 2: Two cops win the directed cops-and-robber game.

$G(2, k)$  has entanglement  $> k$ : Call a path **free** if all nodes on the path and all their doubles are unguarded by the detectives. Call a node **blocked** if both the node and its double are guarded. The thief escapes against  $k$  detectives with the following strategy:

From a leaf  $v$  go to an ancestor  $u$  of  $v$  from which there is a free path to a leaf. Go to that leaf.

The thief is never below a blocked node. Hence she always can go to an ancestor with a free path to a leaf, and thus escapes forever.

# Complexity of DAG-width and entanglement

**Proposition.** For any fixed  $k \in \mathbb{N}$ , the problem whether a given graph has entanglement  $k$  (resp. DAG-width  $k$ ) is solvable in polynomial time.

# Complexity of DAG-width and entanglement

**Proposition.** For any fixed  $k \in \mathbb{N}$ , the problem whether a given graph has entanglement  $k$  (resp. DAG-width  $k$ ) is solvable in polynomial time.

**Proposition.** Given a graph  $G$  and  $k \in \mathbb{N}$ , deciding whether  $\text{dag}(G) \leq k$  is NP-complete.

**Open problem:** Corresponding question for entanglement

# Applications of DAG-width and entanglement

Hamiltonicity can be decided efficiently on graphs of bounded DAG width or bounded entanglement.

Follows immediately from the fact that  $\text{dtw}(G) \leq \text{dag}(G) \leq \text{ent}(G) + 1$ .

# Applications of DAG-width and entanglement

Hamiltonicity can be decided efficiently on graphs of bounded DAG width or bounded entanglement.

Follows immediately from the fact that  $\text{dtw}(G) \leq \text{dag}(G) \leq \text{ent}(G) + 1$ .

The analogue of Courcelle's Theorem fails for DAG-width and entanglement.

Take any difficult MSO-problem, that does not depend on direction of edges (such as 3-colourability). Redirect edges so that the graph becomes acyclic.

# Applications of DAG-width and entanglement

Hamiltonicity can be decided efficiently on graphs of bounded DAG width or bounded entanglement.

Follows immediately from the fact that  $\text{dtw}(G) \leq \text{dag}(G) \leq \text{ent}(G) + 1$ .

The analogue of Courcelle's Theorem fails for DAG-width and entanglement.

Take any difficult MSO-problem, that does not depend on direction of edges (such as 3-colourability). Redirect edges so that the graph becomes acyclic.

Entanglement is a crucial instrument in the proof of the strictness of the variable hierarchy of the  $\mu$ -calculus

# Applications of DAG-width and entanglement

Hamiltonicity can be decided efficiently on graphs of bounded DAG width or bounded entanglement.

Follows immediately from the fact that  $\text{dtw}(G) \leq \text{dag}(G) \leq \text{ent}(G) + 1$ .

The analogue of Courcelle's Theorem fails for DAG-width and entanglement.

Take any difficult MSO-problem, that does not depend on direction of edges (such as 3-colourability). Redirect edges so that the graph becomes acyclic.

Entanglement is a crucial instrument in the proof of the strictness of the variable hierarchy of the  $\mu$ -calculus

Efficient algorithms for parity games on game graphs with bounded dag width or bounded entanglement

# The variable hierarchy of the $\mu$ -calculus

$L_\mu[k]$ :  $\mu$ -calculus with at most  $k$  different fixed-point variables

- Most of the popular fragments of  $L_\mu$ , such as CTL, PDL, CTL\*, and game logic GL, are actually contained in  $L_\mu[2]$ .
- Syntactically, a formula in  $L_\mu[k]$  is a tree with back-edges with entanglement at most  $k$ .
- Model checking for  $L_\mu[2]$ , and also for game logic GL, is as hard as for the entire  $\mu$ -calculus.

# The variable hierarchy of the $\mu$ -calculus

$L_\mu[k]$ :  $\mu$ -calculus with at most  $k$  different fixed-point variables

- Most of the popular fragments of  $L_\mu$ , such as CTL, PDL, CTL\*, and game logic GL, are actually contained in  $L_\mu[2]$ .
- Syntactically, a formula in  $L_\mu[k]$  is a tree with back-edges with entanglement at most  $k$ .
- Model checking for  $L_\mu[2]$ , and also for game logic GL, is as hard as for the entire  $\mu$ -calculus.

**Problem.** Is the variable hierarchy of  $L_\mu$  strict?

# Entanglement and the variable hierarchy of the $\mu$ -calculus

**Theorem.** Let  $\mathcal{K}$  be a finite transition system with  $\text{ent}(\mathcal{K}) = k$ . For every node  $u$  of  $\mathcal{K}$  there is a formula  $\psi_u \in L_\mu[k]$  that defines  $\mathcal{K}, u$  up to simulation:

$$\mathcal{K}', u' \models \psi_u \iff \mathcal{K}, u \preceq \mathcal{K}', u'$$

# Entanglement and the variable hierarchy of the $\mu$ -calculus

**Theorem.** Let  $\mathcal{K}$  be a finite transition system with  $\text{ent}(\mathcal{K}) = k$ . For every node  $u$  of  $\mathcal{K}$  there is a formula  $\psi_u \in L_\mu[k]$  that defines  $\mathcal{K}, u$  up to simulation:

$$\mathcal{K}', u' \models \psi_u \iff \mathcal{K}, u \preceq \mathcal{K}', u'$$

**Theorem.** Every strongly connected finite graph of entanglement  $k$  can be labelled in such a way that no  $\mu$ -calculus formula with less than  $k$  variables can describe the resulting transition system, up to simulation.

# Entanglement and the variable hierarchy of the $\mu$ -calculus

**Theorem.** Let  $\mathcal{K}$  be a finite transition system with  $\text{ent}(\mathcal{K}) = k$ . For every node  $u$  of  $\mathcal{K}$  there is a formula  $\psi_u \in L_\mu[k]$  that defines  $\mathcal{K}, u$  up to simulation:

$$\mathcal{K}', u' \models \psi_u \iff \mathcal{K}, u \preceq \mathcal{K}', u'$$

**Theorem.** Every strongly connected finite graph of entanglement  $k$  can be labelled in such a way that no  $\mu$ -calculus formula with less than  $k$  variables can describe the resulting transition system, up to simulation.

**Corollary.** Berwanger, G., Lenzi

The variable hierarchy of the  $\mu$ -calculus is strict.

Since game logic GL is contained in  $L_\mu[2]$ , this also proves that  $\text{GL} \subsetneq L_\mu$  and thus solves a problem posed by Parikh in 1985.

# Parity games of bounded directed complexity

**Theorem.** (Berwanger, Grädel 2004)

Parity games of bounded entanglement are solvable in polynomial time.

**Theorem.** (Berwanger, Dawar, Hunter, Kreutzer, Obdržálek 2006)

Parity games of bounded DAG-width are solvable in polynomial time.

# Parity games of bounded directed complexity

**Theorem.** (Berwanger, Grädel 2004)

Parity games of bounded entanglement are solvable in polynomial time.

**Theorem.** (Berwanger, Dawar, Hunter, Kreutzer, Obdržálek 2006)

Parity games of bounded DAG-width are solvable in polynomial time.

Since  $\text{dag}(G) \leq \text{ent}(G) + 1$  the second theorem subsumes the first.

# Parity games of bounded directed complexity

**Theorem.** (Berwanger, Grädel 2004)

Parity games of bounded entanglement are solvable in polynomial time.

**Theorem.** (Berwanger, Dawar, Hunter, Kreutzer, Obdržálek 2006)

Parity games of bounded DAG-width are solvable in polynomial time.

Since  $\text{dag}(G) \leq \text{ent}(G) + 1$  the second theorem subsumes the first.

Nevertheless, entanglement has other advantages for the analysis of parity games:

Transparent proof via game-playing

Actually, only the entanglement of the subgraphs induced by winning strategies need to be considered which may be much smaller than the entanglement (or DAG-width) of the original game graph.

# The superdetective game

For any parity game  $G = (V, V_0, V_1, E, \Omega)$ , Player  $\sigma$ , and any number  $k \leq |V|$ , we define the **Superdetective game**  $G[\sigma, k]$ :

Superdetective wants to show, using  $k$  detectives, that Player  $\sigma$  wins the parity game  $G$ . He controls positions  $v \in V_\sigma$  and can place the  $k$  detectives. Challenger controls positions  $v \in V_{1-\sigma}$  of the opponent.

# The superdetective game

For any parity game  $G = (V, V_0, V_1, E, \Omega)$ , Player  $\sigma$ , and any number  $k \leq |V|$ , we define the **Superdetective game**  $G[\sigma, k]$ :

Superdetective wants to show, using  $k$  detectives, that Player  $\sigma$  wins the parity game  $G$ . He controls positions  $v \in V_\sigma$  and can place the  $k$  detectives. Challenger controls positions  $v \in V_{1-\sigma}$  of the opponent.

**Playing the game:** Superdetective and Challenger play the parity game. In addition, Superdetective may, at any move, transfer one detective to the current position of the parity game.

# The superdetective game

For any parity game  $G = (V, V_0, V_1, E, \Omega)$ , Player  $\sigma$ , and any number  $k \leq |V|$ , we define the **Superdetective game**  $G[\sigma, k]$ :

Superdetective wants to show, using  $k$  detectives, that Player  $\sigma$  wins the parity game  $G$ . He controls positions  $v \in V_\sigma$  and can place the  $k$  detectives. Challenger controls positions  $v \in V_{1-\sigma}$  of the opponent.

**Playing the game:** Superdetective and Challenger play the parity game. In addition, Superdetective may, at any move, transfer one detective to the current position of the parity game.

**Winning condition:** The play ends when the parity game reaches a position occupied by a detective. Superdetective wins if the least position seen since this detective was placed there is even, if  $\sigma = 0$  and odd, if  $\sigma = 1$ . In all other cases, Challenger wins.

# Superdetective game and parity game

## Proposition.

- (1) If Player  $\sigma$  wins the parity game  $G$ , then Superdetective wins  $G[\sigma, k]$  for  $k = \text{ent}(G)$ .

# Superdetective game and parity game

## Proposition.

(1) If Player  $\sigma$  wins the parity game  $G$ , then Superdetective wins  $G[\sigma, k]$  for  $k = \text{ent}(G)$ .

Actually  $k = \text{ent}(G_f)$  suffices, where  $G_f$  is the subgame induced by a positional winning strategy  $f$  for Player  $\sigma$ .

# Superdetective game and parity game

## Proposition.

(1) If Player  $\sigma$  wins the parity game  $G$ , then Superdetective wins  $G[\sigma, k]$  for  $k = \text{ent}(G)$ .

Actually  $k = \text{ent}(G_f)$  suffices, where  $G_f$  is the subgame induced by a positional winning strategy  $f$  for Player  $\sigma$ .

(2) If, for some  $k$ , Superdetective wins  $G[\sigma, k]$ , then Player  $\sigma$  wins the parity game  $G$ .

# Superdetective game and parity game

## Proposition.

(1) If Player  $\sigma$  wins the parity game  $G$ , then Superdetective wins  $G[\sigma, k]$  for  $k = \text{ent}(G)$ .

Actually  $k = \text{ent}(G_f)$  suffices, where  $G_f$  is the subgame induced by a positional winning strategy  $f$  for Player  $\sigma$ .

(2) If, for some  $k$ , Superdetective wins  $G[\sigma, k]$ , then Player  $\sigma$  wins the parity game  $G$ .

**Complexity:** Computing the winner of the Superdetective game  $G[\sigma, k]$  requires alternating space  $((2k + 1) \log |V|)$

# Superdetective game and parity game

## Proposition.

(1) If Player  $\sigma$  wins the parity game  $G$ , then Superdetective wins  $G[\sigma, k]$  for  $k = \text{ent}(G)$ .

Actually  $k = \text{ent}(G_f)$  suffices, where  $G_f$  is the subgame induced by a positional winning strategy  $f$  for Player  $\sigma$ .

(2) If, for some  $k$ , Superdetective wins  $G[\sigma, k]$ , then Player  $\sigma$  wins the parity game  $G$ .

**Complexity:** Computing the winner of the Superdetective game  $G[\sigma, k]$  requires alternating space  $((2k + 1) \log |V|)$

Play the game and record current position of thief and current positions of the  $k$  detectives, along with the minimal priority since he was last posted.

# Parity games of bounded entanglement

**Theorem.** The winner of a parity game  $G$  can be determined in  $\text{ASPACE}((2k + 1) \log |V|)$ , where  $k$  is the minimum entanglement of a subgame  $G_f$  induced by a memoryless winning strategy  $f$ .

# Parity games of bounded entanglement

**Theorem.** The winner of a parity game  $G$  can be determined in  $\text{ASPACE}((2k + 1) \log |V|)$ , where  $k$  is the minimum entanglement of a subgame  $G_f$  induced by a memoryless winning strategy  $f$ .

**Input :** parity game  $G$ , with initial position  $v$

**guess**  $k_0 \leq |V|$

**universally choose**  $k_1 \leq |V|$

**if**  $k_0 \leq k_1$  **then**

**if** Superdetective wins  $G[0, k_0]$  from  $v$  **then** **accept** , **else** **reject**

**if**  $k_1 < k_0$  **then**

**if** Superdetective wins  $G[1, k_1]$  from  $v$  **then** **reject** , **else** **accept**

# Parity games of bounded entanglement

**Theorem.** The winner of a parity game  $G$  can be determined in  $\text{ASPACE}((2k + 1) \log |V|)$ , where  $k$  is the minimum entanglement of a subgame  $G_f$  induced by a memoryless winning strategy  $f$ .

**Input :** parity game  $G$ , with initial position  $v$

**guess**  $k_0 \leq |V|$

**universally choose**  $k_1 \leq |V|$

**if**  $k_0 \leq k_1$  **then**

**if** Superdetective wins  $G[0, k_0]$  from  $v$  **then** **accept** , **else** **reject**

**if**  $k_1 < k_0$  **then**

**if** Superdetective wins  $G[1, k_1]$  from  $v$  **then** **reject** , **else** **accept**

**Corollary.**

Parity games of bounded entanglement are solvable in polynomial time.