



# Playing Games on Graphs

*joint work with  
Martin Hyland, Cambridge*

Andrea Schalk

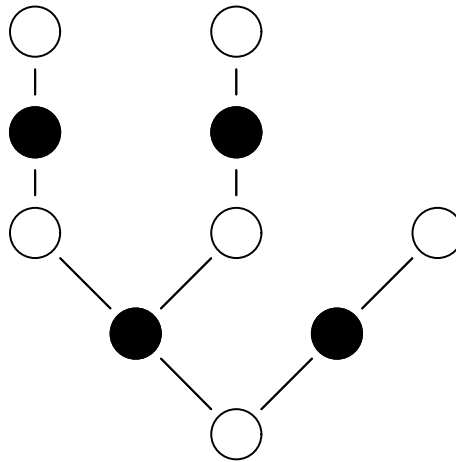
School of Computer Science, The University of Manchester

# Identifying Positions

Games are usually played on **trees**.

# Identifying Positions

Games are usually played on **trees**.



# Identifying Positions

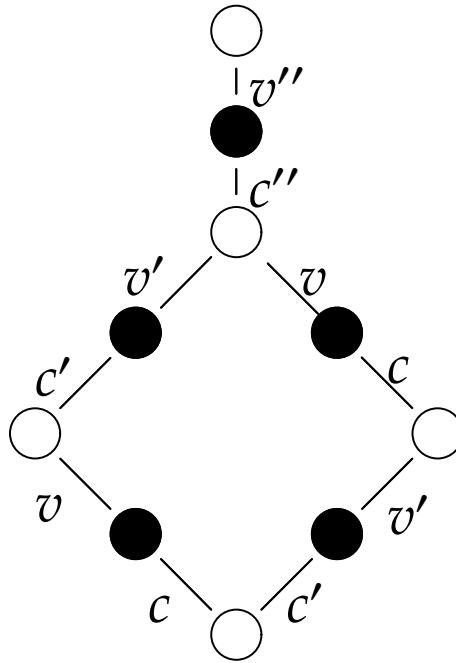
Games are usually played on **trees**.

Sometimes we want to **identify positions** in a game.

# Identifying Positions

Games are usually played on **trees**.

Sometimes we want to **identify positions** in a game.



# Identifying Positions

Games are usually played on **trees**.

Sometimes we want to **identify positions** in a game.

Example:

- $f$ : function with two arguments;
- can use its arguments in either order;
- but higher order functions may not be allowed to tell.

# Graph Games

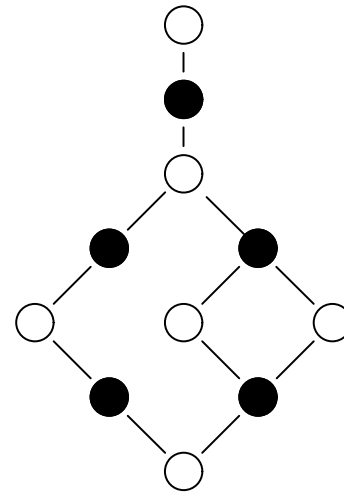
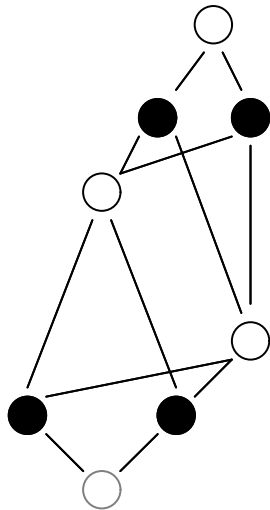
**Definition 1** A *graph game*  $A$  is given by

- a set  $A = A_P + A_O$  of *positions* together with an *initial position*  $*_A \in A_P$ ;
  - ▶  $A_P$  is the set of *Player positions*  $\circ$  (into which Player has moved) and
  - ▶  $A_O$  is the set of *Opponent positions*  $\bullet$  (into which Opponent has moved);
- the structure  $a \longrightarrow a'$  on  $A$  of an acyclic directed  $(A_P, A_O)$ -bipartite graph such that for all  $a \in A$ ,

$$\{a' \in A \mid a' \longrightarrow^* a\}$$

is well-founded with respect to  $(\longrightarrow^*)^{\text{op}}$ .

# Graph Games



# Graph Games: How Are They Played?

Two players: **Player** and **Opponent**.

# Graph Games: How Are They Played?

Two players: **Player** and **Opponent**.

The game starts in the **initial position** and then

- Opponent moves from *P*-positions to *O*-positions  
 $\circ \longrightarrow \bullet$ ;
- Player moves from *O*-positions to *P*-positions  
 $\bullet \longrightarrow \circ$ .

# Graph Games: How Are They Played?

Two players: **Player** and **Opponent**.

The game starts in the **initial position** and then

- Opponent moves from *P*-positions to *O*-positions  
 $\circ \longrightarrow \bullet$ ;
- Player moves from *O*-positions to *P*-positions  
 $\bullet \longrightarrow \circ$ .

Because the graph is **bipartite** Player and Opponent move strictly alternating.

# Graph Games: How Are They Played?

Two players: **Player** and **Opponent**.

The game starts in the **initial position** and then

- Opponent moves from *P*-positions to *O*-positions  
 $\bigcirc \longrightarrow \bullet$ ;
- Player moves from *O*-positions to *P*-positions  
 $\bullet \longrightarrow \bigcirc$ .

Because the graph is **bipartite** Player and Opponent move strictly alternating.

Positions that are not **reachable** from the initial position play no role in the game and may be omitted.

# Graph Games, Example: Tree Games

Every (tree) game can be viewed as a graph game.

- Plays ending in  $P$ -moves give the  $P$ -positions;
- Plays ending in  $O$ -moves give the  $P$ -positions;
- there is an edge from one position to another if the latter extends the former by one move.

# Examples for Graph Games: cdss

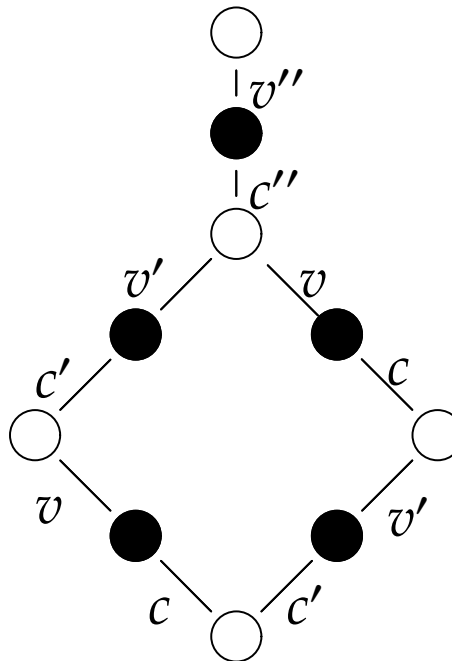
A cds  $R$  defines a **graph game**  $GR$  in an obvious way:

- A  **$P$ -position** is given by a finite states;
- an  **$O$ -position** is a finite state together with a cell that is enabled but not filled in that state.

# Examples for Graph Games: cdss

A *cds*  $R$  defines a **graph game**  $GR$  in an obvious way:

- A  **$P$ -position** is given by a finite states;
- an  **$O$ -position** is a finite state together with a cell that is enabled but not filled in that state.



# Strategies: Definition

Given a position  $a$  in a graph game  $A$  let

$$\uparrow a = \{a' \in A \mid a \longrightarrow^* a'\}.$$

# Strategies: Definition

Given a position  $a$  in a graph game  $A$  let

$$\uparrow a = \{a' \in A \mid a \longrightarrow^* a'\}.$$

**Definition 3** A *strategy*  $\alpha$  for a graph game  $A$  is given by a set of *P-positions* such that

- for every  $a \in \alpha$  there exists  $*_A = a_0, a_1, a_2, \dots, a_{2n} = a$  such that all the  $a_{2i}$  are in  $\alpha$ ;

# Strategies: Definition

Given a position  $a$  in a graph game  $A$  let

$$\uparrow a = \{a' \in A \mid a \longrightarrow^* a'\}.$$

**Definition 3** A *strategy*  $\alpha$  for a graph game  $A$  is given by a set of *P-positions* such that

- for every  $a \in \alpha$  there exists  $*_A = a_0, a_1, a_2, \dots, a_{2n} = a$  such that all the  $a_{2i}$  are in  $\alpha$ ;

‘Every position mentioned in  $\alpha$  is reachable when playing in accordance with  $\alpha$ .’

# Strategies: Definition

Given a position  $a$  in a graph game  $A$  let

$$\uparrow a = \{a' \in A \mid a \longrightarrow^* a'\}.$$

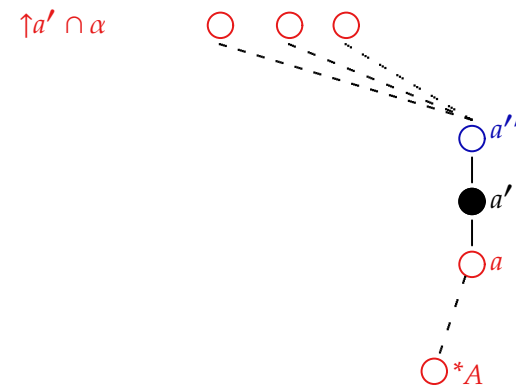
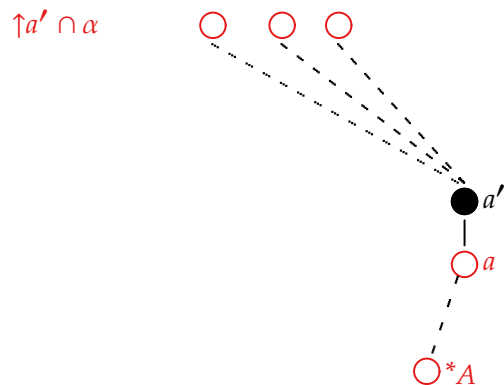
**Definition 3** A *strategy*  $\alpha$  for a graph game  $A$  is given by a set of *P-positions* such that

- for every  $a \in \alpha$  there exists  $*_A = a_0, a_1, a_2, \dots, a_{2n} = a$  such that all the  $a_{2i}$  are in  $\alpha$ ;
- for every  $a \in \alpha$  and  $a \longrightarrow a'$ , if  $\uparrow a' \cap \alpha \neq \emptyset$  then there exists  $a'' \in \alpha$  with  $\uparrow a' \cap \alpha = \uparrow a'' \cap \alpha$ .

# Strategies: Definition

**Definition 3** A *strategy*  $\alpha$  for a graph game  $A$  is given by a set of *P-positions* such that

- for every  $a \in \alpha$  there exists  $*_A = a_0, a_1, a_2, \dots, a_{2n} = a$  such that all the  $a_{2i}$  are in  $\alpha$ ;
- for every  $a \in \alpha$  and  $a \longrightarrow a'$ , if  $\uparrow a' \cap \alpha \neq \emptyset$  then there exists  $a'' \in \alpha$  with  $\uparrow a' \cap \alpha = \uparrow a'' \cap \alpha$ .



# Strategies: Definition

**Definition 3** A *strategy*  $\alpha$  for a graph game  $A$  is given by a set of *P-positions* such that

- for every  $a \in \alpha$  there exists  $*_A = a_0, a_1, a_2, \dots, a_{2n} = a$  such that all the  $a_{2i}$  are in  $\alpha$ ;
- for every  $a \in \alpha$  and  $a \longrightarrow a'$ , if  $\uparrow a' \cap \alpha \neq \emptyset$  then there exists  $a'' \in \alpha$  with  $\uparrow a' \cap \alpha = \uparrow a'' \cap \alpha$ .



Note that  $a''$  is necessarily unique.

# A Category of Graph Games

Can define category of graph games which is

- symmetric monoidal closed and
- has some further categorical structure making it into a **model of intuitionistic linear logic**.

# A Category of Graph Games

Can define category of graph games which is

- symmetric monoidal closed and
- has some further categorical structure making it into a **model of intuitionistic linear logic**.

We define the (linear) **function space** and morphisms are then strategies on it).

# A Category of Graph Games

Can define category of graph games which is

- symmetric monoidal closed and
- has some further categorical structure making it into a **model of intuitionistic linear logic**.

We define the (linear) **function space** and morphisms are then strategies on it).

But we start with the **tensor product**.

# Tensor Product

The tensor unit  $I$  is  $\circ$ .

# Tensor Product

The tensor unit  $\mathbf{I}$  is  $\circ$ .

The tensor  $A \otimes B$  is the game with

- $P$ -positions  $A_P \times B_P$   $\circ\circ$ ;
- $O$ -positions  $(A_P \times B_O) + (A_O \times B_P)$   $\circ\bullet \bullet\circ$ .

The initial position is  $(*_A, *_B)$  and there is a move

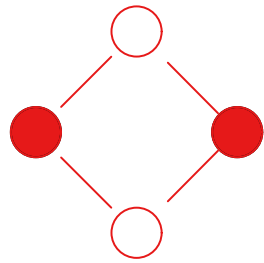
$$(a, b) \longrightarrow (a', b')$$

just when

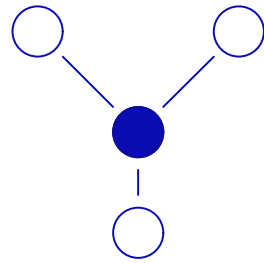
either  $a \longrightarrow a'$  and  $b = b'$  'a move in  $A$ '

or  $a = a'$  and  $b \longrightarrow b'$  'a move in  $B$ '.

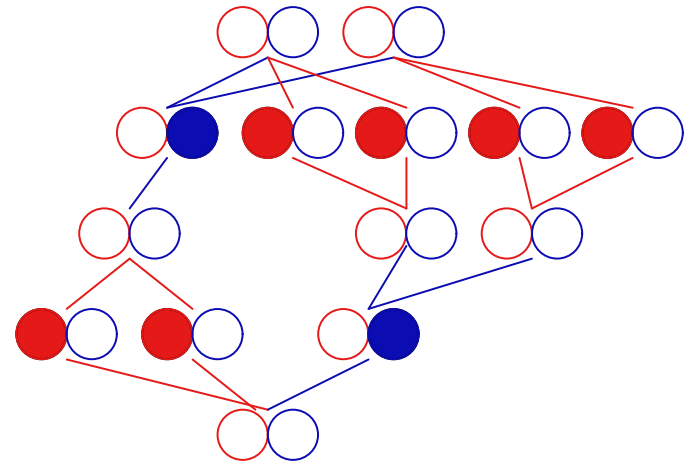
# Tensor Product: Example



$A$



$B$



$A \otimes B$

# Comparison with Simple Tree Games

The tensor product for two tree games:

# Comparison with Simple Tree Games

The tensor product for two tree games:

Opponent owns all  $O$ -moves, Player all  $P$ -moves; a **valid play** is then given by one where

- restrictions to **moves from  $A$**  gives **valid play of  $A$** ;
- restrictions to **moves from  $B$**  gives **valid play of  $B$** ;

# Comparison with Simple Tree Games

The tensor product for two tree games:

Opponent owns all  $O$ -moves, Player all  $P$ -moves; a **valid play** is then given by one where

- restrictions to **moves from  $A$**  gives **valid play of  $A$** ;
- restrictions to **moves from  $B$**  gives **valid play of  $B$** ;

Such plays project onto a position in  $A$  and one in  $B$ .

# Comparison with Simple Tree Games

The tensor product for two tree games:

Opponent owns all  $O$ -moves, Player all  $P$ -moves; a **valid play** is then given by one where

- restrictions to **moves from  $A$**  gives **valid play of  $A$** ;
- restrictions to **moves from  $B$**  gives **valid play of  $B$** ;

Such plays project onto a position in  $A$  and one in  $B$ .

- Take  $A \otimes B$  for tree games;
- identify positions where projections onto  $(A, B)$  agree;
- get  $A \otimes B$  taken as graph games.

# Product of Cdss

To form **product** of two cdss: put them **side by side**.

# Product of Cdss

To form **product** of two cdss: put them **side by side**.

So, form the disjoint union of the cells and similarly for values; enabling of cells is just as before.

# Product of Cdss

To form **product** of two cdss: put them **side by side**.

So, form the disjoint union of the cells and similarly for values; enabling of cells is just as before.

States for the product are (disjoint) unions of states.

# Product of Cdss

To form **product** of two cdss: put them **side by side**.

So, form the disjoint union of the cells and similarly for values; enabling of cells is just as before.

States for the product are (disjoint) unions of states.

The **empty product**, or initial object, **1** is the cds without cells, and

$$G1 = I.$$

# Product of Cdss

To form **product** of two cdss: put them **side by side**.

So, form the disjoint union of the cells and similarly for values; enabling of cells is just as before.

States for the product are (disjoint) unions of states.

The **empty product**, or initial object, **1** is the cds without cells, and

$$G1 = I.$$

Take cdss  $R, S$  then

$$G(R \times S) = GR \otimes GS.$$

# Linear Function Space

The linear function space  $A \multimap B$  is the game with

- $P$ -positions  $(A_P \times B_P) + (A_O \times B_O)$   $\circ\circ \bullet\bullet$ ;
- $O$ -positions  $A_P \times B_O$   $\circ\bullet$ .

The initial position is  $(*_A, *_B)$  and there is a move

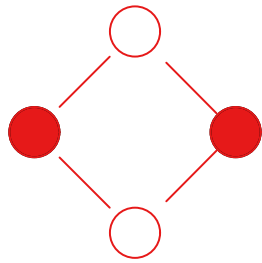
$$(a, b) \longrightarrow (a', b')$$

just when

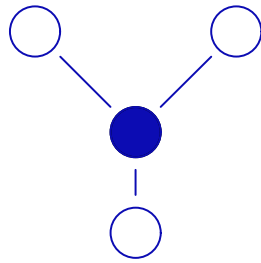
either  $a \longrightarrow a'$  and  $b = b'$  'a move in  $A$ '

or  $a = a'$  and  $b \longrightarrow b'$  'a move in  $B$ '.

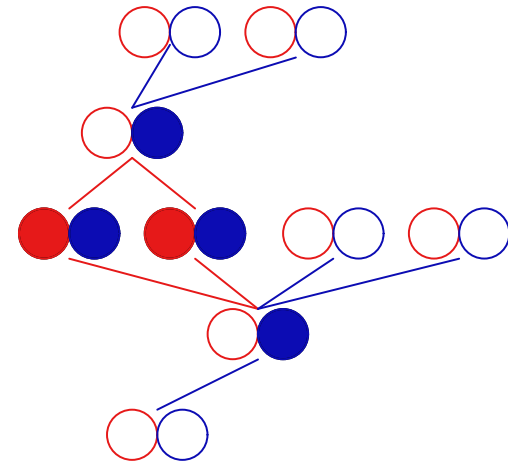
# Linear Function Space: Example



*A*



*B*



*A*  $\rightarrow$  *B*

# Function Space: Play

Can still view the linear function space  $A \multimap B$  as a recipe for ‘playing the games  $A$  and  $B$  in parallel’.

# Function Space: Play

Can still view the linear function space  $A \multimap B$  as a recipe for ‘playing the games  $A$  and  $B$  in parallel’.

The current position in  $A \multimap B$  is given by a pair of positions, one in  $A$  and one in  $B$ .

# Function Space: Play

Can still view the linear function space  $A \multimap B$  as a recipe for ‘playing the games  $A$  and  $B$  in parallel’.

The current position in  $A \multimap B$  is given by a pair of positions, one in  $A$  and one in  $B$ .

Any move is either

- a move in the game  $A$  or
- a move in the game  $B$ .

# Function Space: Play

Can still view the linear function space  $A \multimap B$  as a recipe for ‘playing the games  $A$  and  $B$  in parallel’.

The current position in  $A \multimap B$  is given by a pair of positions, one in  $A$  and one in  $B$ .

Any move is either

- a move in the game  $A$  or
- a move in the game  $B$ .

Again, given tree games  $A$  and  $B$  we can

- take  $A \multimap B$  for tree games and
- identify positions where projections onto  $(A, B)$  agree;
- get  $A \multimap B$  taken as graph games.

# Function Space: Strategies

Strategy on  $A \multimap B$ : set of  $P$ -positions of  $A \multimap B$ .

# Function Space: Strategies

Strategy on  $A \multimap B$ : set of  $P$ -positions of  $A \multimap B$ .

$P$ -positions in  $A \multimap B$  can be of pattern

- $\circ\circ$  (a  $P$ -position in  $A$  and a  $P$ -position in  $B$ );
- $\bullet\bullet$  (an  $O$ -position in  $A$  and an  $O$ -position in  $B$ ).

# Function Space: Strategies

Strategy on  $A \multimap B$ : set of  $P$ -positions of  $A \multimap B$ .

$P$ -positions in  $A \multimap B$  can be of pattern

- $\circ\circ$  (a  $P$ -position in  $A$  and a  $P$ -position in  $B$ );
- $\bullet\bullet$  (an  $O$ -position in  $A$  and an  $O$ -position in  $B$ ).

Thus a strategy  $\rho: A \multimap B$  is a subset of

$$(A_P \times B_P) + (A_O \times B_O).$$

# Function Space: Strategies

Strategy on  $A \multimap B$ : set of  $P$ -positions of  $A \multimap B$ .

$P$ -positions in  $A \multimap B$  can be of pattern

- $\circ\circ$  (a  $P$ -position in  $A$  and a  $P$ -position in  $B$ );
- $\bullet\bullet$  (an  $O$ -position in  $A$  and an  $O$ -position in  $B$ ).

Thus a strategy  $\rho: A \multimap B$  is a subset of

$$(A_P \times B_P) + (A_O \times B_O).$$

So can view  $\rho$  as a relation

$$\begin{array}{ccc} & \rho & \\ A_P & \text{---+---} & B_P \\ A_O & \text{---+---} & B_O. \end{array}$$

# Composition: How

Want to define morphism  $A \longrightarrow B$ : strategy on  $A \multimap B$ .

# Composition: How

Want to define morphism  $A \longrightarrow B$ : strategy on  $A \multimap B$ .

Assume have  $\rho: A \multimap B$ ,  $\sigma: B \multimap C$ , want  $\sigma \circ \rho: A \multimap C$ .

# Composition: How

Want to define morphism  $A \longrightarrow B$ : strategy on  $A \multimap B$ .

Assume have  $\rho: A \multimap B$ ,  $\sigma: B \multimap C$ , want  $\sigma \circ \rho: A \multimap C$ .

$$\begin{array}{ccc} & \rho & \sigma \\ A_P \text{ ---+--- } & B_P & B_P \text{ ---+--- } C_P \\ A_O \text{ ---+--- } & B_O & B_O \text{ ---+--- } C_O. \end{array}$$

# Composition: How

Want to define morphism  $A \longrightarrow B$ : strategy on  $A \multimap B$ .

Assume have  $\rho: A \multimap B$ ,  $\sigma: B \multimap C$ , want  $\sigma \circ \rho: A \multimap C$ .

$$\begin{array}{ccc} & \rho & \sigma \\ A_P \text{ ---+--- } B_P & & B_P \text{ ---+--- } C_P \\ A_O \text{ ---+--- } B_O & & B_O \text{ ---+--- } C_O. \end{array}$$

Try relational composition:

$$\begin{array}{ccc} & \sigma \circ \rho & \\ A_P \text{ ---+--- } B_P \text{ ---+--- } C_P & & \\ A_O \text{ ---+--- } B_O \text{ ---+--- } C_O. & & \end{array}$$

# Composition: It Works!

Can show that composite of two strategies is a strategy.

# Composition: It Works!

Can show that composite of two strategies is a strategy.

This is hard work! But having done it, **composition is easy to handle.**

# Composition: It Works!

Can show that composite of two strategies is a strategy.

Proof works similar to usual composition of strategies for tree games *via* 'composition and hiding'.

# Composition: It Works!

Can show that composite of two strategies is a strategy.

Proof works similar to usual composition of strategies for tree games *via* 'composition and hiding'.

So then category of tree games is embedded in the category of graph games by a full (and faithful) functor.

# Identities

Identities are given by the usual **copycat** strategies.

# Identities

Identities are given by the usual **copycat** strategies.

Here

$$\text{id}_A = \{(a, a) \mid a \text{ is a reachable position of } A\}$$

behaves like the identity for relational composition.

# Identities

Identities are given by the usual **copycat** strategies.

Here

$$\text{id}_A = \{(a, a) \mid a \text{ is a reachable position of } A\}$$

behaves like the identity for relational composition.

# The Category of Graph Games

Now easy to establish that

the category of graph games is **symmetric monoidal closed**.

# Cdss as Graph Games

Assignment  $G$  from cdss to graph games: functor.

# Cdss as Graph Games

Assignment  $G$  from cdss to graph games: **functor**.

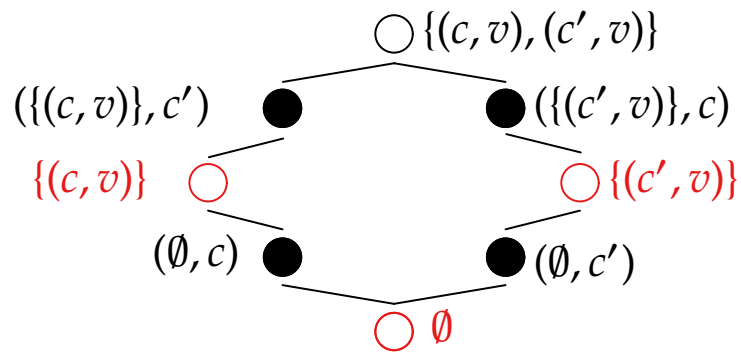
**Faithful** but **not full**; maps products to tensor products.

# Cdss as Graph Games

Assignment  $G$  from cdss to graph games: **functor**.

**Faithful** but **not full**.

For  $R = 1$ ,  $GR \dashv\vdash GS \cong G1 \dashv\vdash GS \cong GS$ .



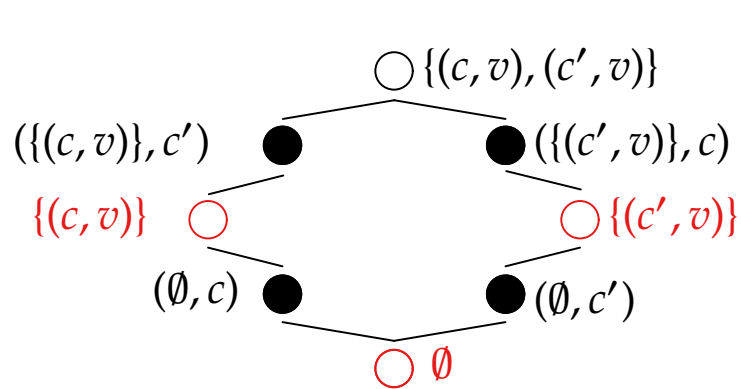
**Strategy** for  $GS$ .

# Cdss as Graph Games

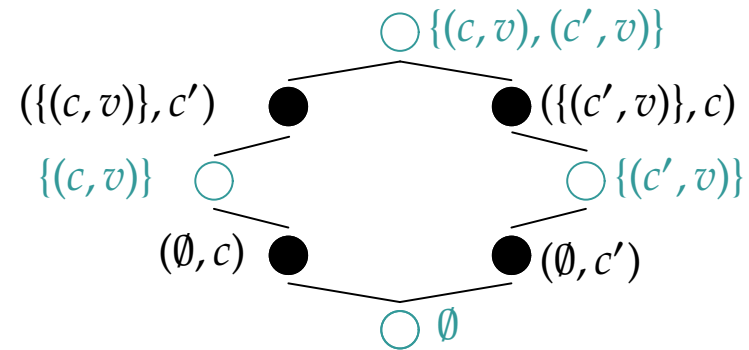
Assignment  $G$  from cdss to graph games: **functor**.

**Faithful** but **not full**.

For  $R = 1$ ,  $GR \dashv\vdash GS \cong G1 \dashv\vdash GS \cong GS$ .



**Strategy** for  $GS$ .



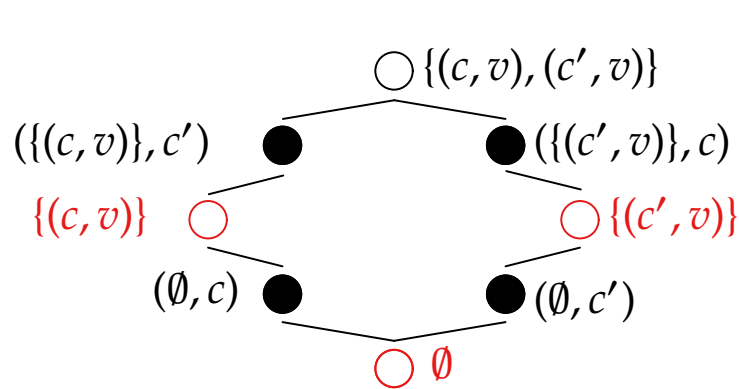
**Strategy** containing  $(c, v)$  and  $(c', v)$  in  $\text{im}G$ .

# Cdss as Graph Games

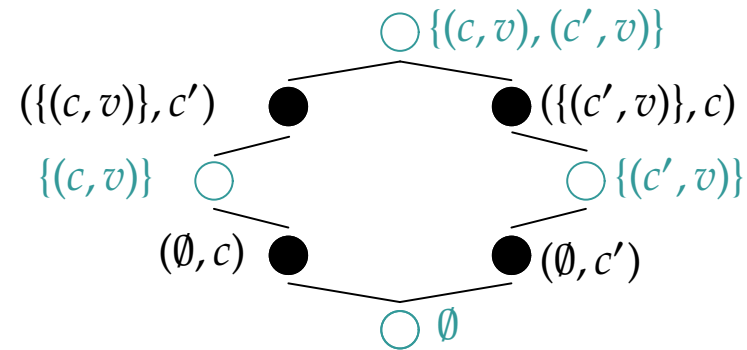
Assignment  $G$  from cdss to graph games: **functor**.

**Faithful** but **not full**.

For  $R = 1$ ,  $GR \dashv\vdash GS \cong G1 \dashv\vdash GS \cong GS$ .



**Strategy** for  $GS$ .



**Strategy** containing  $(c, v)$  and  $(c', v)$  in  $\text{im}G$ .

If  $g: 1 \longrightarrow S$  fills  $c$  and  $c'$  then  $Gg$  contains  $\{(c, v), (c', v)\}$ .

# Additive Structure

The terminal object  $1$  is (again)  $\circ$ .

# Additive Structure

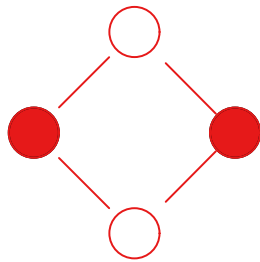
The terminal object  $1$  is (again)  $\circ$ .

Product  $A \times B$ : 'coalesced sum' of  $A$  and  $B$  with  $*_A$  and  $*_B$  identified.

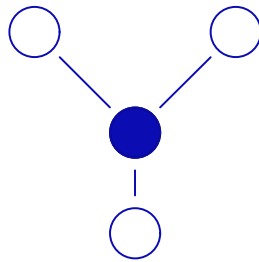
# Additive Structure

The terminal object  $1$  is (again)  $\circ$ .

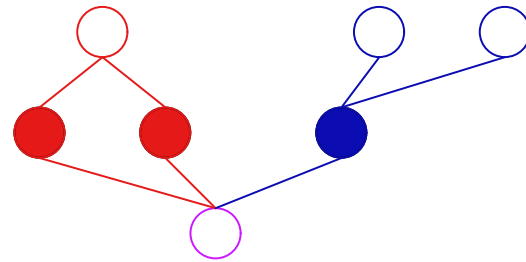
Product  $A \times B$ : 'coalesced sum' of  $A$  and  $B$  with  $*_A$  and  $*_B$  identified.



$A$



$B$

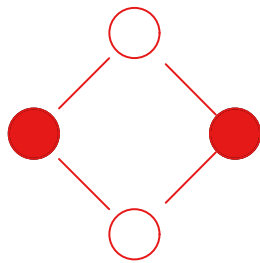


$A \times B$

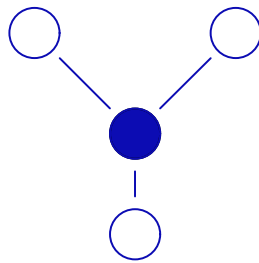
# Additive Structure

The terminal object  $1$  is (again)  $\circ$ .

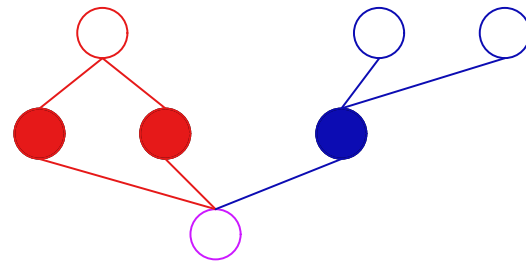
Product  $A \times B$ : 'coalesced sum' of  $A$  and  $B$  with  $*_A$  and  $*_B$  identified.



$A$



$B$



$A \times B$

The category of graph games has arbitrary products.

# A Model for ILL

Can define **linear exponential comonad** on this category.  
(This is really hard...)

# A Model for ILL

Can define **linear exponential comonad** on this category.  
(This is really hard...)

**Theorem 1** *The category **RGGam** of (regulated) graph games is a **model of Intuitionistic Linear Logic**.*

# A Model for ILL

Can define **linear exponential comonad** on this category.  
(This is really hard...)

**Theorem 1** *The category **RGGam** of (regulated) graph games is a **model of Intuitionistic Linear Logic**.*

A delicate result—only one known linear exponential.

# Future and Related Work

Graph games are based on **positions** rather than **moves**.

# Future and Related Work

Graph games are based on **positions** rather than **moves**.

Sometimes want to refer to moves: strategies in  $\text{im}G$  do (also needed for **innocence**).

# Future and Related Work

Graph games are based on **positions** rather than **moves**.

Sometimes want to refer to moves: strategies in  $\text{im}G$  do (also needed for **innocence**).

A version of graph games which allows this was defined by Paul-André Melliès, but his definition has other issues.

# Summary

- Have category of games played on graphs which is symmetric monoidal closed and a **model for intuitionistic linear logic**.

# Summary

- Have category of games played on graphs which is symmetric monoidal closed and a **model for intuitionistic linear logic**.
- Composition of strategies is defined as **relational composition**.

# Summary

- Have category of games played on graphs which is symmetric monoidal closed and a **model for intuitionistic linear logic**.
- Composition of strategies is defined as **relational composition**.
- The category of tree games is embedded into this category by a full functor.

# Summary

- Have category of games played on graphs which is symmetric monoidal closed and a **model for intuitionistic linear logic**.
- Composition of strategies is defined as **relational composition**.
- The category of tree games is embedded into this category by a full functor.
- Strategies defined in this category incorporate some aspects of **innocence**.