

# Dependent Types and Finite Limits in Games

Norihiro Yamada

yamad041@umn.edu  
University of Minnesota

Logic and Computer Science Seminar  
Research Institute for Mathematical Sciences  
Kyoto University  
June 3, 2021

# Goal and plan of the talk

# Goal and plan of the talk

The goal of this talk is to convey:

# Goal and plan of the talk

The goal of this talk is to convey:

- 1 Introduction to game semantics and why it matters
- 2 Challenges in game semantics of dependent types
- 3 Main ideas in my solution
- 4 Ongoing and future research

# Game semantics: why does it matter?

## Game semantics: why does it matter?

*Game semantics* is a particular class of mathematical semantics of logic and computation that interprets types and terms by *games* and *strategies*, respectively.

## Game semantics: why does it matter?

*Game semantics* is a particular class of mathematical semantics of logic and computation that interprets types and terms by *games* and *strategies*, respectively. Its notable advantages are:

## Game semantics: why does it matter?

*Game semantics* is a particular class of mathematical semantics of logic and computation that interprets types and terms by *games* and *strategies*, respectively. Its notable advantages are:

- Naturality: terms as *interactive processes* (e.g.,  $\phi : \forall x \in \mathbb{N}. P(x)$ );



## Game semantics: why does it matter?

*Game semantics* is a particular class of mathematical semantics of logic and computation that interprets types and terms by *games* and *strategies*, respectively. Its notable advantages are:

- Naturality: terms as *interactive processes* (e.g.,  $\phi : \forall x \in \mathbb{N}. P(x)$ );
- Precision due to *intensionality*: *full completeness/abstraction*;

## Game semantics: why does it matter?

*Game semantics* is a particular class of mathematical semantics of logic and computation that interprets types and terms by *games* and *strategies*, respectively. Its notable advantages are:

- Naturality: terms as *interactive processes* (e.g.,  $\phi : \forall x \in \mathbb{N}. P(x)$ );
- Precision due to *intensionality*: *full completeness/abstraction*;
- Semantics: *syntax-independent*, *analytic* and *non-inductive*;

## Game semantics: why does it matter?

*Game semantics* is a particular class of mathematical semantics of logic and computation that interprets types and terms by *games* and *strategies*, respectively. Its notable advantages are:

- Naturality: terms as *interactive processes* (e.g.,  $\phi : \forall x \in \mathbb{N}. P(x)$ );
- Precision due to *intensionality*: *full completeness/abstraction*;
- Semantics: *syntax-independent*, *analytic* and *non-inductive*;
- *Uniform* interpretation: *effects* and *linear logic*.

## Game semantics: why does it matter?

*Game semantics* is a particular class of mathematical semantics of logic and computation that interprets types and terms by *games* and *strategies*, respectively. Its notable advantages are:

- Naturality: terms as *interactive processes* (e.g.,  $\phi : \forall x \in \mathbb{N}. P(x)$ );
- Precision due to *intensionality*: *full completeness/abstraction*;
- Semantics: *syntax-independent*, *analytic* and *non-inductive*;
- *Uniform* interpretation: *effects* and *linear logic*.

Why does it matter?

## Game semantics: why does it matter?

*Game semantics* is a particular class of mathematical semantics of logic and computation that interprets types and terms by *games* and *strategies*, respectively. Its notable advantages are:

- Naturality: terms as *interactive processes* (e.g.,  $\phi : \forall x \in \mathbb{N}. P(x)$ );
- Precision due to *intensionality*: *full completeness/abstraction*;
- Semantics: *syntax-independent*, *analytic* and *non-inductive*;
- *Uniform* interpretation: *effects* and *linear logic*.

Why does it matter?

- As an inspiration for new categories and types;

## Game semantics: why does it matter?

*Game semantics* is a particular class of mathematical semantics of logic and computation that interprets types and terms by *games* and *strategies*, respectively. Its notable advantages are:

- Naturality: terms as *interactive processes* (e.g.,  $\phi : \forall x \in \mathbb{N}. P(x)$ );
- Precision due to *intensionality*: *full completeness/abstraction*;
- Semantics: *syntax-independent*, *analytic* and *non-inductive*;
- *Uniform* interpretation: *effects* and *linear logic*.

Why does it matter?

- As an inspiration for new categories and types;
- For the meta-theoretic study of type theory (e.g., independence of Markov's principle);

## Game semantics: why does it matter?

*Game semantics* is a particular class of mathematical semantics of logic and computation that interprets types and terms by *games* and *strategies*, respectively. Its notable advantages are:

- Naturality: terms as *interactive processes* (e.g.,  $\phi : \forall x \in \mathbb{N}. P(x)$ );
- Precision due to *intensionality*: *full completeness/abstraction*;
- Semantics: *syntax-independent*, *analytic* and *non-inductive*;
- *Uniform* interpretation: *effects* and *linear logic*.

Why does it matter?

- As an inspiration for new categories and types;
- For the meta-theoretic study of type theory (e.g., independence of Markov's principle);
- As pure mathematics of logic and computation in its own right: *algorithms*, *normalisation*, *higher-order computability*, etc.

# A challenge: game semantics of dependent types



## A challenge: game semantics of dependent types

One of the most difficult problems in game semantics is to interpret dependent types.

## A challenge: game semantics of dependent types

One of the most difficult problems in game semantics is to interpret dependent types. No established solution for more than 25 years.

## A challenge: game semantics of dependent types

One of the most difficult problems in game semantics is to interpret dependent types. No established solution for more than 25 years.

- This is notable since game semantics has been highly successful in modelling a wide range of logics and computations;

## A challenge: game semantics of dependent types

One of the most difficult problems in game semantics is to interpret dependent types. No established solution for more than 25 years.

- This is notable since game semantics has been highly successful in modelling a wide range of logics and computations;
- Even more than one game semantics of System F was established since 2005.

## A challenge: game semantics of dependent types

One of the most difficult problems in game semantics is to interpret dependent types. No established solution for more than 25 years.

- This is notable since game semantics has been highly successful in modelling a wide range of logics and computations;
- Even more than one game semantics of System F was established since 2005.

Why does game semantics of dependent types matter?

## A challenge: game semantics of dependent types

One of the most difficult problems in game semantics is to interpret dependent types. No established solution for more than 25 years.

- This is notable since game semantics has been highly successful in modelling a wide range of logics and computations;
- Even more than one game semantics of System F was established since 2005.

Why does game semantics of dependent types matter?

- For the meta-theoretic study of dependent type theories;

## A challenge: game semantics of dependent types

One of the most difficult problems in game semantics is to interpret dependent types. No established solution for more than 25 years.

- This is notable since game semantics has been highly successful in modelling a wide range of logics and computations;
- Even more than one game semantics of System F was established since 2005.

Why does game semantics of dependent types matter?

- For the meta-theoretic study of dependent type theories;
- For combining dependent types and effects/linear logic;

## A challenge: game semantics of dependent types

One of the most difficult problems in game semantics is to interpret dependent types. No established solution for more than 25 years.

- This is notable since game semantics has been highly successful in modelling a wide range of logics and computations;
- Even more than one game semantics of System F was established since 2005.

Why does game semantics of dependent types matter?

- For the meta-theoretic study of dependent type theories;
- For combining dependent types and effects/linear logic;
- For designing new dependent types and their categorical semantics;



## A challenge: game semantics of dependent types

One of the most difficult problems in game semantics is to interpret dependent types. No established solution for more than 25 years.

- This is notable since game semantics has been highly successful in modelling a wide range of logics and computations;
- Even more than one game semantics of System F was established since 2005.

Why does game semantics of dependent types matter?

- For the meta-theoretic study of dependent type theories;
- For combining dependent types and effects/linear logic;
- For designing new dependent types and their categorical semantics;
- For computational understanding of quantifications.

## A challenge: game semantics of dependent types

One of the most difficult problems in game semantics is to interpret dependent types. No established solution for more than 25 years.

- This is notable since game semantics has been highly successful in modelling a wide range of logics and computations;
- Even more than one game semantics of System F was established since 2005.

Why does game semantics of dependent types matter?

- For the meta-theoretic study of dependent type theories;
- For combining dependent types and effects/linear logic;
- For designing new dependent types and their categorical semantics;
- For computational understanding of quantifications.

If you are working on these topics, we should collaborate!

# Why difficult?

## Why difficult?

The main challenge in game semantics of dependent types is to model quantifications  $\Pi(A, B)$  and  $\Sigma(A, B)$  in terms of *processes*.

## Why difficult?

The main challenge in game semantics of dependent types is to model quantifications  $\Pi(A, B)$  and  $\Sigma(A, B)$  in terms of *processes*.

- The set-theoretic interpretations,  $f : A \rightarrow \bigcup_{x \in A} B(x)$  and  $(a, b) \in A \times \bigcup_{x \in A} B(x)$ , where  $\forall x \in A. f(x) \in B(x)$  and  $b \in B(a)$ , are simple since the type dependency is on *total, static* objects;

## Why difficult?

The main challenge in game semantics of dependent types is to model quantifications  $\Pi(A, B)$  and  $\Sigma(A, B)$  in terms of *processes*.

- The set-theoretic interpretations,  $f : A \rightarrow \bigcup_{x \in A} B(x)$  and  $(a, b) \in A \times \bigcup_{x \in A} B(x)$ , where  $\forall x \in A. f(x) \in B(x)$  and  $b \in B(a)$ , are simple since the type dependency is on *total, static* objects;
- In contrast,  $x, f(x), a$  and  $b$  are often *partial, dynamic* processes in game semantics, and it is far from obvious how to impose the type dependency on *ever-changing stages* of processes.

## Why difficult?

The main challenge in game semantics of dependent types is to model quantifications  $\Pi(A, B)$  and  $\Sigma(A, B)$  in terms of *processes*.

- The set-theoretic interpretations,  $f : A \rightarrow \bigcup_{x \in A} B(x)$  and  $(a, b) \in A \times \bigcup_{x \in A} B(x)$ , where  $\forall x \in A. f(x) \in B(x)$  and  $b \in B(a)$ , are simple since the type dependency is on *total, static* objects;
- In contrast,  $x, f(x), a$  and  $b$  are often *partial, dynamic* processes in game semantics, and it is far from obvious how to impose the type dependency on *ever-changing stages* of processes.

There is one proposal on game semantics of dependent types by [Abramsky et al., 2015], but it is not completely satisfactory:

## Why difficult?

The main challenge in game semantics of dependent types is to model quantifications  $\Pi(A, B)$  and  $\Sigma(A, B)$  in terms of *processes*.

- The set-theoretic interpretations,  $f : A \rightarrow \bigcup_{x \in A} B(x)$  and  $(a, b) \in A \times \bigcup_{x \in A} B(x)$ , where  $\forall x \in A. f(x) \in B(x)$  and  $b \in B(a)$ , are simple since the type dependency is on *total, static* objects;
- In contrast,  $x, f(x), a$  and  $b$  are often *partial, dynamic* processes in game semantics, and it is far from obvious how to impose the type dependency on *ever-changing stages* of processes.

There is one proposal on game semantics of dependent types by [Abramsky et al., 2015], but it is not completely satisfactory:

- **Ad-hoc:** It is limited to a very specific class of dependent types;



## Why difficult?

The main challenge in game semantics of dependent types is to model quantifications  $\Pi(A, B)$  and  $\Sigma(A, B)$  in terms of *processes*.

- The set-theoretic interpretations,  $f : A \rightarrow \bigcup_{x \in A} B(x)$  and  $(a, b) \in A \times \bigcup_{x \in A} B(x)$ , where  $\forall x \in A. f(x) \in B(x)$  and  $b \in B(a)$ , are simple since the type dependency is on *total, static* objects;
- In contrast,  $x, f(x), a$  and  $b$  are often *partial, dynamic* processes in game semantics, and it is far from obvious how to impose the type dependency on *ever-changing stages* of processes.

There is one proposal on game semantics of dependent types by [Abramsky et al., 2015], but it is not completely satisfactory:

- **Ad-hoc:** It is limited to a very specific class of dependent types;
- **No games for Sigma-types:**  $\Pi(\Sigma(A, B), C)$  as  $\Pi(A, \Pi(B, C))$ , and  $\Pi(A, \Sigma(B, C))$  as  $(\phi : \Pi(A, B), \psi : \Pi(A, C\{\phi\}))$ ;

## Why difficult?

The main challenge in game semantics of dependent types is to model quantifications  $\Pi(A, B)$  and  $\Sigma(A, B)$  in terms of *processes*.

- The set-theoretic interpretations,  $f : A \rightarrow \bigcup_{x \in A} B(x)$  and  $(a, b) \in A \times \bigcup_{x \in A} B(x)$ , where  $\forall x \in A. f(x) \in B(x)$  and  $b \in B(a)$ , are simple since the type dependency is on *total, static* objects;
- In contrast,  $x, f(x), a$  and  $b$  are often *partial, dynamic* processes in game semantics, and it is far from obvious how to impose the type dependency on *ever-changing stages* of processes.

There is one proposal on game semantics of dependent types by [Abramsky et al., 2015], but it is not completely satisfactory:

- **Ad-hoc:** It is limited to a very specific class of dependent types;
- **No games for Sigma-types:**  $\Pi(\Sigma(A, B), C)$  as  $\Pi(A, \Pi(B, C))$ , and  $\Pi(A, \Sigma(B, C))$  as  $(\phi : \Pi(A, B), \psi : \Pi(A, C\{\phi\}))$ ;
- **Syntactic and inductive:** It models types and terms by *lists* of games and strategies, respectively.

# Main results

# Main results

Theorem (game semantics of dependent types)

---

## Main results

Theorem (game semantics of dependent types)

*There is game semantics of Martin-Löf type theory with One-, Zero-, N-, Pi-, Sigma- and Id-types as well as universes.*

# Main results

## Theorem (game semantics of dependent types)

*There is game semantics of Martin-Löf type theory with One-, Zero-, N-, Pi-, Sigma- and Id-types as well as universes.*

- It interprets a standard class of dependent types;

## Main results

### Theorem (game semantics of dependent types)

*There is game semantics of Martin-Löf type theory with One-, Zero-, N-, Pi-, Sigma- and Id-types as well as universes.*

- It interprets a standard class of dependent types;
- It has no list constructions (i.e., not inductive or syntactic);

## Main results

### Theorem (game semantics of dependent types)

*There is game semantics of Martin-Löf type theory with One-, Zero-, N-, Pi-, Sigma- and Id-types as well as universes.*

- It interprets a standard class of dependent types;
- It has no list constructions (i.e., not inductive or syntactic);
- It interprets Sigma-types directly by games.



## Main results

### Theorem (game semantics of dependent types)

*There is game semantics of Martin-Löf type theory with One-, Zero-, N-, Pi-, Sigma- and Id-types as well as universes.*

- It interprets a standard class of dependent types;
- It has no list constructions (i.e., not inductive or syntactic);
- It interprets Sigma-types directly by games.

### Theorem (game-semantic finite limits)

*The game semantics has all finite limits.*

## Main results

### Theorem (game semantics of dependent types)

*There is game semantics of Martin-Löf type theory with One-, Zero-, N-, Pi-, Sigma- and Id-types as well as universes.*

- It interprets a standard class of dependent types;
- It has no list constructions (i.e., not inductive or syntactic);
- It interprets Sigma-types directly by games.

### Theorem (game-semantic finite limits)

*The game semantics has all finite limits.*

### Corollary (game semantics of homotopy type theory)

*There is game semantics of homotopy type theory.*

# Games and strategies (1/2)

# Games and strategies (1/2)

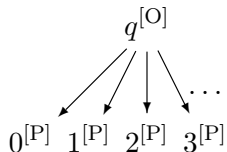
## Definition (games)

A *game* is a rooted DAG whose vertices (or *moves*) have parity O/P, and paths from a root (or *positions*) have parity OPOP...

# Games and strategies (1/2)

## Definition (games)

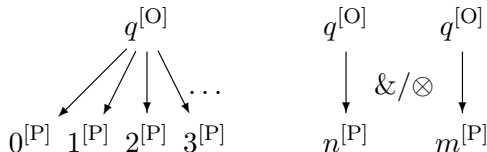
A **game** is a rooted DAG whose vertices (or **moves**) have parity O/P, and paths from a root (or **positions**) have parity OPOP...



## Games and strategies (1/2)

## Definition (games)

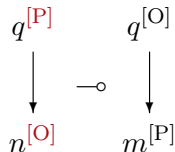
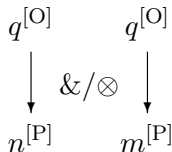
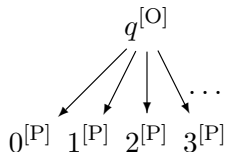
A **game** is a rooted DAG whose vertices (or **moves**) have parity O/P, and paths from a root (or **positions**) have parity OPOP...



## Games and strategies (1/2)

## Definition (games)

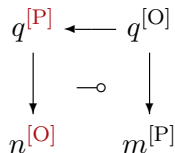
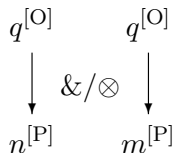
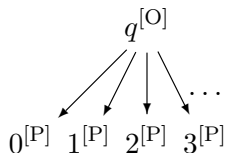
A **game** is a rooted DAG whose vertices (or **moves**) have parity O/P, and paths from a root (or **positions**) have parity OPOP...



## Games and strategies (1/2)

## Definition (games)

A **game** is a rooted DAG whose vertices (or **moves**) have parity O/P, and paths from a root (or **positions**) have parity OPOP...

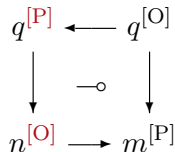
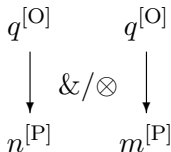
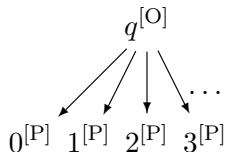




## Games and strategies (1/2)

## Definition (games)

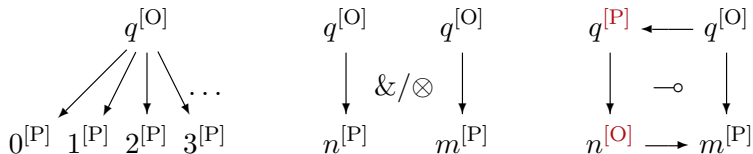
A **game** is a rooted DAG whose vertices (or **moves**) have parity O/P, and paths from a root (or **positions**) have parity OPOP...



## Games and strategies (1/2)

## Definition (games)

A **game** is a rooted DAG whose vertices (or **moves**) have parity O/P, and paths from a root (or **positions**) have parity OPOP...

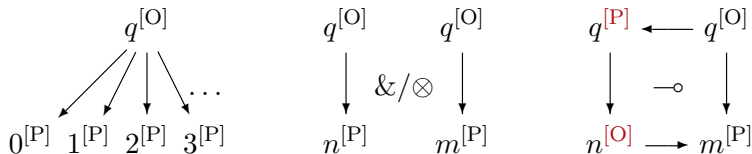


## Definition (product, tensor and linear implication on games)

## Games and strategies (1/2)

## Definition (games)

A **game** is a rooted DAG whose vertices (or **moves**) have parity O/P, and paths from a root (or **positions**) have parity OPOP...



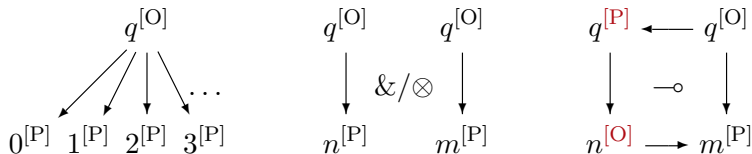
## Definition (product, tensor and linear implication on games)

**Product:**  $G \& H := G \uplus H$ ;

## Games and strategies (1/2)

## Definition (games)

A **game** is a rooted DAG whose vertices (or **moves**) have parity O/P, and paths from a root (or **positions**) have parity OPOP...



## Definition (product, tensor and linear implication on games)

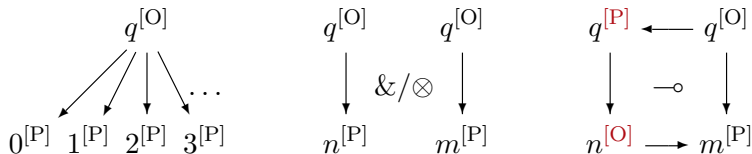
**Product:**  $G \& H := G \uplus H$ ;

**Tensor:**  $G \otimes H := \{ \mathbf{s} \in (M_G \uplus M_H)^{\text{alt}} \mid \mathbf{s} \upharpoonright G \in G, \mathbf{s} \upharpoonright H \in H \}$ ;

## Games and strategies (1/2)

## Definition (games)

A **game** is a rooted DAG whose vertices (or **moves**) have parity O/P, and paths from a root (or **positions**) have parity OPOP...



## Definition (product, tensor and linear implication on games)

**Product:**  $G \& H := G \uplus H$ ;

**Tensor:**  $G \otimes H := \{ \mathbf{s} \in (M_G \uplus M_H)^{\text{alt}} \mid \mathbf{s} \upharpoonright G \in G, \mathbf{s} \upharpoonright H \in H \}$ ;

**Linear implication:**

$G \multimap H := \{ \mathbf{s} \in (M_G^{\text{flip}} \uplus M_H)^{\text{alt}} \mid \mathbf{s} \upharpoonright G \in G, \mathbf{s} \upharpoonright H \in H \}$ .

## Games and strategies (2/2)

## Games and strategies (2/2)

Example (games as sets of positions)

## Games and strategies (2/2)

Example (games as sets of positions)

*Terminal game*  $T := \{\epsilon\}$ ;



## Games and strategies (2/2)

Example (games as sets of positions)

*Terminal game*  $T := \{\epsilon\}$ ; *empty game*  $\mathbf{0} := \{\epsilon, q\}$ ;

## Games and strategies (2/2)

Example (games as sets of positions)

*Terminal game*  $T := \{\epsilon\}$ ; *empty game*  $\mathbf{0} := \{\epsilon, q\}$ ; *boolean game*  $\Omega := \{\epsilon, q\} \cup \{qb \mid b \in \mathbb{B}\}$ ;

## Games and strategies (2/2)

Example (games as sets of positions)

*Terminal game*  $T := \{\epsilon\}$ ; *empty game*  $\mathbf{0} := \{\epsilon, q\}$ ; *boolean game*  $\Omega := \{\epsilon, q\} \cup \{qb \mid b \in \mathbb{B}\}$ ; *N-game*  $N := \{\epsilon, q\} \cup \{qn \mid n \in \mathbb{N}\}$ .

## Games and strategies (2/2)

### Example (games as sets of positions)

*Terminal game*  $T := \{\epsilon\}$ ; *empty game*  $\mathbf{0} := \{\epsilon, q\}$ ; *boolean game*  $\Omega := \{\epsilon, q\} \cup \{qb \mid b \in \mathbb{B}\}$ ; *N-game*  $N := \{\epsilon, q\} \cup \{qn \mid n \in \mathbb{N}\}$ .

### Definition (strategies)

## Games and strategies (2/2)

## Example (games as sets of positions)

**Terminal game**  $T := \{\epsilon\}$ ; **empty game**  $\mathbf{0} := \{\epsilon, q\}$ ; **boolean game**  $\Omega := \{\epsilon, q\} \cup \{qb \mid b \in \mathbb{B}\}$ ; **N-game**  $N := \{\epsilon, q\} \cup \{qn \mid n \in \mathbb{N}\}$ .

## Definition (strategies)

A **strategy**  $\sigma$  on a game  $G$ , written  $\sigma : G$ , is a partial map

$$\{ \text{odd-length positions } m_1 m_2 \dots m_{2i+1} \text{ in } G \} \rightarrow \{ \text{P-moves } m \text{ in } G \}$$

s.t.  $m_1 m_2 \dots m_{2i+1} m$  is a position in  $G$ .

## Games and strategies (2/2)

## Example (games as sets of positions)

**Terminal game**  $T := \{\epsilon\}$ ; **empty game**  $\mathbf{0} := \{\epsilon, q\}$ ; **boolean game**  $\Omega := \{\epsilon, q\} \cup \{qb \mid b \in \mathbb{B}\}$ ; **N-game**  $N := \{\epsilon, q\} \cup \{qn \mid n \in \mathbb{N}\}$ .

## Definition (strategies)

A **strategy**  $\sigma$  on a game  $G$ , written  $\sigma : G$ , is a partial map

$$\{ \text{odd-length positions } m_1 m_2 \dots m_{2i+1} \text{ in } G \} \rightarrow \{ \text{P-moves } m \text{ in } G \}$$

s.t.  $m_1 m_2 \dots m_{2i+1} m$  is a position in  $G$ .

It is **total** if it always responds:  $\forall sm \in G^{\text{Odd}}. s \in \sigma \Rightarrow \exists smn \in \sigma$ .

## Games and strategies (2/2)

## Example (games as sets of positions)

**Terminal game**  $T := \{\epsilon\}$ ; **empty game**  $0 := \{\epsilon, q\}$ ; **boolean game**  $\Omega := \{\epsilon, q\} \cup \{qb \mid b \in \mathbb{B}\}$ ; **N-game**  $N := \{\epsilon, q\} \cup \{qn \mid n \in \mathbb{N}\}$ .

## Definition (strategies)

A **strategy**  $\sigma$  on a game  $G$ , written  $\sigma : G$ , is a partial map

$$\{ \text{odd-length positions } m_1 m_2 \dots m_{2i+1} \text{ in } G \} \rightarrow \{ \text{P-moves } m \text{ in } G \}$$

s.t.  $m_1 m_2 \dots m_{2i+1} m$  is a position in  $G$ .

It is **total** if it always responds:  $\forall sm \in G^{\text{Odd}}. s \in \sigma \Rightarrow \exists smn \in \sigma$ .

## Example (strategies as certain subsets of even-length positions)

## Games and strategies (2/2)

## Example (games as sets of positions)

**Terminal game**  $T := \{\epsilon\}$ ; **empty game**  $0 := \{\epsilon, q\}$ ; **boolean game**  $\Omega := \{\epsilon, q\} \cup \{qb \mid b \in \mathbb{B}\}$ ; **N-game**  $N := \{\epsilon, q\} \cup \{qn \mid n \in \mathbb{N}\}$ .

## Definition (strategies)

A **strategy**  $\sigma$  on a game  $G$ , written  $\sigma : G$ , is a partial map

$$\{ \text{odd-length positions } m_1 m_2 \dots m_{2i+1} \text{ in } G \} \rightarrow \{ \text{P-moves } m \text{ in } G \}$$

s.t.  $m_1 m_2 \dots m_{2i+1} m$  is a position in  $G$ .

It is **total** if it always responds:  $\forall sm \in G^{\text{Odd}}. s \in \sigma \Rightarrow \exists smn \in \sigma$ .

## Example (strategies as certain subsets of even-length positions)

$$\top := \{\epsilon\} : T;$$



## Games and strategies (2/2)

## Example (games as sets of positions)

**Terminal game**  $T := \{\epsilon\}$ ; **empty game**  $\mathbf{0} := \{\epsilon, q\}$ ; **boolean game**  $\Omega := \{\epsilon, q\} \cup \{qb \mid b \in \mathbb{B}\}$ ; **N-game**  $N := \{\epsilon, q\} \cup \{qn \mid n \in \mathbb{N}\}$ .

## Definition (strategies)

A **strategy**  $\sigma$  on a game  $G$ , written  $\sigma : G$ , is a partial map

$$\{ \text{odd-length positions } m_1 m_2 \dots m_{2i+1} \text{ in } G \} \rightarrow \{ \text{P-moves } m \text{ in } G \}$$

s.t.  $m_1 m_2 \dots m_{2i+1} m$  is a position in  $G$ .

It is **total** if it always responds:  $\forall sm \in G^{\text{Odd}}. s \in \sigma \Rightarrow \exists smn \in \sigma$ .

## Example (strategies as certain subsets of even-length positions)

$$\top := \{\epsilon\} : T; \perp := \{\epsilon\} : \mathbf{0};$$

## Games and strategies (2/2)

## Example (games as sets of positions)

**Terminal game**  $T := \{\epsilon\}$ ; **empty game**  $\mathbf{0} := \{\epsilon, q\}$ ; **boolean game**  $\Omega := \{\epsilon, q\} \cup \{qb \mid b \in \mathbb{B}\}$ ; **N-game**  $N := \{\epsilon, q\} \cup \{qn \mid n \in \mathbb{N}\}$ .

## Definition (strategies)

A **strategy**  $\sigma$  on a game  $G$ , written  $\sigma : G$ , is a partial map

$$\{ \text{odd-length positions } m_1 m_2 \dots m_{2i+1} \text{ in } G \} \rightarrow \{ \text{P-moves } m \text{ in } G \}$$

s.t.  $m_1 m_2 \dots m_{2i+1} m$  is a position in  $G$ .

It is **total** if it always responds:  $\forall sm \in G^{\text{Odd}}. s \in \sigma \Rightarrow \exists smn \in \sigma$ .

## Example (strategies as certain subsets of even-length positions)

$$\top := \{\epsilon\} : T; \perp := \{\epsilon\} : \mathbf{0}; \underline{n} := \text{Pref}(\{qn\})^{\text{Even}} : N;$$

## Games and strategies (2/2)

## Example (games as sets of positions)

**Terminal game**  $T := \{\epsilon\}$ ; **empty game**  $\mathbf{0} := \{\epsilon, q\}$ ; **boolean game**  $\Omega := \{\epsilon, q\} \cup \{qb \mid b \in \mathbb{B}\}$ ; **N-game**  $N := \{\epsilon, q\} \cup \{qn \mid n \in \mathbb{N}\}$ .

## Definition (strategies)

A **strategy**  $\sigma$  on a game  $G$ , written  $\sigma : G$ , is a partial map

$$\{ \text{odd-length positions } m_1 m_2 \dots m_{2i+1} \text{ in } G \} \rightarrow \{ \text{P-moves } m \text{ in } G \}$$

s.t.  $m_1 m_2 \dots m_{2i+1} m$  is a position in  $G$ .

It is **total** if it always responds:  $\forall sm \in G^{\text{Odd}}. s \in \sigma \Rightarrow \exists smn \in \sigma$ .

## Example (strategies as certain subsets of even-length positions)

$\top := \{\epsilon\} : T$ ;  $\perp := \{\epsilon\} : \mathbf{0}$ ;  $\underline{n} := \text{Pref}(\{qn\})^{\text{Even}} : N$ ;  $\langle \underline{0}, \underline{1} \rangle : N \& N$ ;

## Games and strategies (2/2)

## Example (games as sets of positions)

**Terminal game**  $T := \{\epsilon\}$ ; **empty game**  $\mathbf{0} := \{\epsilon, q\}$ ; **boolean game**  $\Omega := \{\epsilon, q\} \cup \{qb \mid b \in \mathbb{B}\}$ ; **N-game**  $N := \{\epsilon, q\} \cup \{qn \mid n \in \mathbb{N}\}$ .

## Definition (strategies)

A **strategy**  $\sigma$  on a game  $G$ , written  $\sigma : G$ , is a partial map

$$\{ \text{odd-length positions } m_1 m_2 \dots m_{2i+1} \text{ in } G \} \rightarrow \{ \text{P-moves } m \text{ in } G \}$$

s.t.  $m_1 m_2 \dots m_{2i+1} m$  is a position in  $G$ .

It is **total** if it always responds:  $\forall sm \in G^{\text{Odd}}. s \in \sigma \Rightarrow \exists smn \in \sigma$ .

## Example (strategies as certain subsets of even-length positions)

$\top := \{\epsilon\} : T$ ;  $\perp := \{\epsilon\} : \mathbf{0}$ ;  $\underline{n} := \text{Pref}(\{qn\})^{\text{Even}} : N$ ;  $\langle \underline{0}, \underline{1} \rangle : N \& N$ ;  
 $\Delta := \langle \text{id}_\Omega, \text{id}_\Omega \rangle : \Omega \multimap \Omega \& \Omega$ ;

## Games and strategies (2/2)

## Example (games as sets of positions)

**Terminal game**  $T := \{\epsilon\}$ ; **empty game**  $\mathbf{0} := \{\epsilon, q\}$ ; **boolean game**  $\Omega := \{\epsilon, q\} \cup \{qb \mid b \in \mathbb{B}\}$ ; **N-game**  $N := \{\epsilon, q\} \cup \{qn \mid n \in \mathbb{N}\}$ .

## Definition (strategies)

A **strategy**  $\sigma$  on a game  $G$ , written  $\sigma : G$ , is a partial map

$$\{ \text{odd-length positions } m_1 m_2 \dots m_{2i+1} \text{ in } G \} \rightarrow \{ \text{P-moves } m \text{ in } G \}$$

s.t.  $m_1 m_2 \dots m_{2i+1} m$  is a position in  $G$ .

It is **total** if it always responds:  $\forall sm \in G^{\text{Odd}}. s \in \sigma \Rightarrow \exists smn \in \sigma$ .

## Example (strategies as certain subsets of even-length positions)

$\top := \{\epsilon\} : T$ ;  $\perp := \{\epsilon\} : \mathbf{0}$ ;  $\underline{n} := \text{Pref}(\{qn\})^{\text{Even}} : N$ ;  $\langle \underline{0}, \underline{1} \rangle : N \& N$ ;  
 $\Delta := \langle \text{id}_\Omega, \text{id}_\Omega \rangle : \Omega \multimap \Omega \& \Omega$ ;  $\Delta : \Omega \not\multimap \Omega \otimes \Omega$ .

# Existing games cannot interpret Sigma-types (1/4)

# Existing games cannot interpret Sigma-types (1/4)

The main challenge in game semantics of dependent types:

# Existing games cannot interpret Sigma-types (1/4)

The main challenge in game semantics of dependent types:

*The **intensionality** prohibits games from modelling Sigma-types.*



# Existing games cannot interpret Sigma-types (1/4)

The main challenge in game semantics of dependent types:

*The **intensionality** prohibits games from modelling Sigma-types.*

**Remark:** We should not discard the intensionality since it makes game semantics a highly powerful approach to logic and computation.

## Existing games cannot interpret Sigma-types (1/4)

The main challenge in game semantics of dependent types:

*The **intensionality** prohibits games from modelling Sigma-types.*

**Remark:** We should not discard the intensionality since it makes game semantics a highly powerful approach to logic and computation.

To explain it, let us first model dependent types  $x : C \vdash D(x)$  Type by families  $D = (D(x))_{x:C}$  of games  $D(x)$ , where  $C$  models the type  $C$ .

## Existing games cannot interpret Sigma-types (1/4)

The main challenge in game semantics of dependent types:

*The intensionality prohibits games from modelling Sigma-types.*

**Remark:** We should not discard the intensionality since it makes game semantics a highly powerful approach to logic and computation.

To explain it, let us first model dependent types  $x : C \vdash D(x)$  Type by families  $D = (D(x))_{x:C}$  of games  $D(x)$ , where  $C$  models the type  $C$ .

Then, in light of product  $\&$ , a natural idea is to model the Sigma-type  $\Sigma_{x:C} D(x)$  by a subgame  $\Sigma(C, D) \subseteq C \& \bigcup_{x:C} D(x)$  such that strategies on  $\Sigma(C, D)$  are the pairings  $\langle \sigma, \tau \rangle$  of  $\sigma : C$  and  $\tau : D(\sigma)$ .

## Existing games cannot interpret Sigma-types (1/4)

The main challenge in game semantics of dependent types:

*The intensionality prohibits games from modelling Sigma-types.*

**Remark:** We should not discard the intensionality since it makes game semantics a highly powerful approach to logic and computation.

To explain it, let us first model dependent types  $x : C \vdash D(x)$  Type by families  $D = (D(x))_{x:C}$  of games  $D(x)$ , where  $C$  models the type  $C$ .

Then, in light of product  $\&$ , a natural idea is to model the Sigma-type  $\Sigma_{x:C} D(x)$  by a subgame  $\Sigma(C, D) \subseteq C \& \bigcup_{x:C} D(x)$  such that strategies on  $\Sigma(C, D)$  are the pairings  $\langle \sigma, \tau \rangle$  of  $\sigma : C$  and  $\tau : D(\sigma)$ .

However, this idea *does not work* for the following two problems:

## Existing games cannot interpret Sigma-types (1/4)

The main challenge in game semantics of dependent types:

*The intensionality prohibits games from modelling Sigma-types.*

**Remark:** We should not discard the intensionality since it makes game semantics a highly powerful approach to logic and computation.

To explain it, let us first model dependent types  $x : C \vdash D(x)$  Type by families  $D = (D(x))_{x:C}$  of games  $D(x)$ , where  $C$  models the type  $C$ .

Then, in light of product  $\&$ , a natural idea is to model the Sigma-type  $\Sigma_{x:C} D(x)$  by a subgame  $\Sigma(C, D) \subseteq C \& \bigcup_{x:C} D(x)$  such that strategies on  $\Sigma(C, D)$  are the pairings  $\langle \sigma, \tau \rangle$  of  $\sigma : C$  and  $\tau : D(\sigma)$ .

However, this idea *does not work* for the following two problems:

- ① Each game  $G$  determines strategies on  $G$  (i.e., opposite to sets);

## Existing games cannot interpret Sigma-types (1/4)

The main challenge in game semantics of dependent types:

*The intensionality prohibits games from modelling Sigma-types.*

**Remark:** We should not discard the intensionality since it makes game semantics a highly powerful approach to logic and computation.

To explain it, let us first model dependent types  $x : C \vdash D(x)$  Type by families  $D = (D(x))_{x:C}$  of games  $D(x)$ , where  $C$  models the type  $C$ .

Then, in light of product  $\&$ , a natural idea is to model the Sigma-type  $\Sigma_{x:C} D(x)$  by a subgame  $\Sigma(C, D) \subseteq C \& \bigcup_{x:C} D(x)$  such that strategies on  $\Sigma(C, D)$  are the pairings  $\langle \sigma, \tau \rangle$  of  $\sigma : C$  and  $\tau : D(\sigma)$ .

However, this idea *does not work* for the following two problems:

- ① Each game  $G$  determines strategies on  $G$  (i.e., opposite to sets);
- ② Strategies  $\sigma : G$  cannot change the game  $G$ .

# Existing games cannot interpret Sigma-types (2/4)

# Existing games cannot interpret Sigma-types (2/4)

First example:



## Existing games cannot interpret Sigma-types (2/4)

**First example:** Consider the dependent type  $x : \mathbb{N} \vdash \mathbb{N}_b(x)$  Type such that the canonical terms of the simple type  $\mathbb{N}_b(\underline{k})$  for each  $k \in \mathbb{N}$  are the numerals  $\underline{n}$  that satisfy  $n \leq k$ .

## Existing games cannot interpret Sigma-types (2/4)

**First example:** Consider the dependent type  $x : \mathbb{N} \vdash \mathbf{N}_b(x)$  Type such that the canonical terms of the simple type  $\mathbf{N}_b(\underline{k})$  for each  $k \in \mathbb{N}$  are the numerals  $\underline{n}$  that satisfy  $n \leq k$ .

We model  $\mathbf{N}_b$  by the family  $N_b$  of games  $N_b(\underline{k}) := \text{Pref}(\{qn \mid n \leq k\})$ .

## Existing games cannot interpret Sigma-types (2/4)

**First example:** Consider the dependent type  $x : \mathbb{N} \vdash \mathbf{N}_b(x)$  Type such that the canonical terms of the simple type  $\mathbf{N}_b(\underline{k})$  for each  $k \in \mathbb{N}$  are the numerals  $\underline{n}$  that satisfy  $n \leq k$ .

We model  $\mathbf{N}_b$  by the family  $N_b$  of games  $N_b(\underline{k}) := \text{Pref}(\{qn \mid n \leq k\})$ . Then, a subgame  $\Sigma(N, N_b) \subseteq N \ \& \ N$  that models the Sigma-type  $\Sigma_{x:\mathbb{N}}\mathbf{N}_b(x)$  must satisfy  $\langle \underline{k}, \underline{n} \rangle : \Sigma(N, N_b) \Leftrightarrow k \geq n$  for all  $k, n \in \mathbb{N}$ .

## Existing games cannot interpret Sigma-types (2/4)

**First example:** Consider the dependent type  $x : \mathbb{N} \vdash \mathbf{N}_b(x)$  Type such that the canonical terms of the simple type  $\mathbf{N}_b(\underline{k})$  for each  $k \in \mathbb{N}$  are the numerals  $\underline{n}$  that satisfy  $n \leq k$ .

We model  $\mathbf{N}_b$  by the family  $N_b$  of games  $N_b(\underline{k}) := \text{Pref}(\{qn \mid n \leq k\})$ .

Then, a subgame  $\Sigma(N, N_b) \subseteq N \& N$  that models the Sigma-type

$\Sigma_{x:\mathbb{N}}\mathbf{N}_b(x)$  must satisfy  $\langle \underline{k}, \underline{n} \rangle : \Sigma(N, N_b) \Leftrightarrow k \geq n$  for all  $k, n \in \mathbb{N}$ .

However, such  $\Sigma(N, N_b)$  does not exist since  $\langle \underline{0}, \underline{0} \rangle, \langle \underline{1}, \underline{1} \rangle : \Sigma(N, N_b)$

implies  $\langle \underline{1}, \underline{0} \rangle : \Sigma(N, N_b)$  by the definition of strategies on a game.

## Existing games cannot interpret Sigma-types (2/4)

**First example:** Consider the dependent type  $x : \mathbb{N} \vdash \mathbb{N}_b(x)$  Type such that the canonical terms of the simple type  $\mathbb{N}_b(\underline{k})$  for each  $k \in \mathbb{N}$  are the numerals  $\underline{n}$  that satisfy  $n \leq k$ .

We model  $\mathbb{N}_b$  by the family  $N_b$  of games  $N_b(\underline{k}) := \text{Pref}(\{qn \mid n \leq k\})$ .

Then, a subgame  $\Sigma(N, N_b) \subseteq N \& N$  that models the Sigma-type

$\Sigma_{x:\mathbb{N}}\mathbb{N}_b(x)$  must satisfy  $\langle \underline{k}, \underline{n} \rangle : \Sigma(N, N_b) \Leftrightarrow k \geq n$  for all  $k, n \in \mathbb{N}$ .

However, such  $\Sigma(N, N_b)$  does not exist since  $\langle \underline{0}, \underline{0} \rangle, \langle \underline{1}, \underline{1} \rangle : \Sigma(N, N_b)$  implies  $\langle \underline{1}, \underline{0} \rangle : \Sigma(N, N_b)$  by the definition of strategies on a game.



## Existing games cannot interpret Sigma-types (2/4)

**First example:** Consider the dependent type  $x : \mathbb{N} \vdash \mathbb{N}_b(x)$  Type such that the canonical terms of the simple type  $\mathbb{N}_b(\underline{k})$  for each  $k \in \mathbb{N}$  are the numerals  $\underline{n}$  that satisfy  $n \leq k$ .

We model  $\mathbb{N}_b$  by the family  $N_b$  of games  $N_b(\underline{k}) := \text{Pref}(\{qn \mid n \leq k\})$ .

Then, a subgame  $\Sigma(N, N_b) \subseteq N \ \& \ N$  that models the Sigma-type

$\Sigma_{x:\mathbb{N}}\mathbb{N}_b(x)$  must satisfy  $\langle \underline{k}, \underline{n} \rangle : \Sigma(N, N_b) \Leftrightarrow k \geq n$  for all  $k, n \in \mathbb{N}$ .

However, such  $\Sigma(N, N_b)$  does not exist since  $\langle \underline{0}, \underline{0} \rangle, \langle \underline{1}, \underline{1} \rangle : \Sigma(N, N_b)$  implies  $\langle \underline{1}, \underline{0} \rangle : \Sigma(N, N_b)$  by the definition of strategies on a game.



*First problem:* Each game  $G$  determines strategies on  $G$ .

# Existing games cannot interpret Sigma-types (3/4)

# Existing games cannot interpret Sigma-types (3/4)

Second example:



## Existing games cannot interpret Sigma-types (3/4)

**Second example:** Consider the dependent type  $x : \mathbb{N} \vdash \text{List}_{\mathbb{N}}(x)$  Type such that the canonical terms of the simple type  $\text{List}_{\mathbb{N}}(\underline{k})$  for each  $k \in \mathbb{N}$  are  $k$ -lists  $(\underline{n}_1, \underline{n}_2, \dots, \underline{n}_k)$  of numerals.

## Existing games cannot interpret Sigma-types (3/4)

**Second example:** Consider the dependent type  $x : \mathbb{N} \vdash \text{List}_{\mathbb{N}}(x)$  Type such that the canonical terms of the simple type  $\text{List}_{\mathbb{N}}(\underline{k})$  for each  $k \in \mathbb{N}$  are  $k$ -lists  $(\underline{n}_1, \underline{n}_2, \dots, \underline{n}_k)$  of numerals.

We interpret  $\text{List}_{\mathbb{N}}$  by the family  $\text{List}_N$  of games  $\text{List}_N(\underline{k})$  ( $k \in \mathbb{N}$ ), which are the  $k$ -times iteration of tensor  $\otimes$  on  $N$  (n.b.,  $\text{List}_N(\underline{0}) = T$ ).

## Existing games cannot interpret Sigma-types (3/4)

**Second example:** Consider the dependent type  $x : \mathbb{N} \vdash \text{List}_{\mathbb{N}}(x)$  Type such that the canonical terms of the simple type  $\text{List}_{\mathbb{N}}(\underline{k})$  for each  $k \in \mathbb{N}$  are  $k$ -lists  $(\underline{n}_1, \underline{n}_2, \dots, \underline{n}_k)$  of numerals.

We interpret  $\text{List}_{\mathbb{N}}$  by the family  $\text{List}_N$  of games  $\text{List}_N(\underline{k})$  ( $k \in \mathbb{N}$ ), which are the  $k$ -times iteration of tensor  $\otimes$  on  $N$  (n.b.,  $\text{List}_N(\underline{0}) = T$ ). If a subgame  $\Sigma(N, \text{List}_N) \subseteq N \ \& \ \bigcup_{k \in \mathbb{N}} \text{List}_N(\underline{k})$  models the Sigma-type  $\Sigma_{x:\mathbb{N}} \text{List}_{\mathbb{N}}(x)$ , then  $\langle \underline{0}, \top \rangle$  and  $\langle \underline{1}, \underline{0} \rangle$  are *total* on  $\Sigma(N, \text{List}_N)$ .

## Existing games cannot interpret Sigma-types (3/4)

**Second example:** Consider the dependent type  $x : \mathbb{N} \vdash \text{List}_{\mathbb{N}}(x)$  Type such that the canonical terms of the simple type  $\text{List}_{\mathbb{N}}(\underline{k})$  for each  $k \in \mathbb{N}$  are  $k$ -lists  $(\underline{n}_1, \underline{n}_2, \dots, \underline{n}_k)$  of numerals.

We interpret  $\text{List}_{\mathbb{N}}$  by the family  $\text{List}_N$  of games  $\text{List}_N(\underline{k})$  ( $k \in \mathbb{N}$ ), which are the  $k$ -times iteration of tensor  $\otimes$  on  $N$  (n.b.,  $\text{List}_N(\underline{0}) = T$ ). If a subgame  $\Sigma(N, \text{List}_N) \subseteq N \ \& \ \bigcup_{k \in \mathbb{N}} \text{List}_N(\underline{k})$  models the Sigma-type  $\Sigma_{x:\mathbb{N}} \text{List}_{\mathbb{N}}(x)$ , then  $\langle \underline{0}, \top \rangle$  and  $\langle \underline{1}, \underline{0} \rangle$  are *total* on  $\Sigma(N, \text{List}_N)$ . However, the totality of  $\langle \underline{0}, \top \rangle$  contradicts  $\langle \underline{1}, \underline{0} \rangle : \Sigma(N, \text{List}_N)$ .

## Existing games cannot interpret Sigma-types (3/4)

**Second example:** Consider the dependent type  $x : \mathbb{N} \vdash \text{List}_{\mathbb{N}}(x)$  Type such that the canonical terms of the simple type  $\text{List}_{\mathbb{N}}(\underline{k})$  for each  $k \in \mathbb{N}$  are  $k$ -lists  $(\underline{n}_1, \underline{n}_2, \dots, \underline{n}_k)$  of numerals.

We interpret  $\text{List}_{\mathbb{N}}$  by the family  $\text{List}_N$  of games  $\text{List}_N(\underline{k})$  ( $k \in \mathbb{N}$ ), which are the  $k$ -times iteration of tensor  $\otimes$  on  $N$  (n.b.,  $\text{List}_N(\underline{0}) = T$ ). If a subgame  $\Sigma(N, \text{List}_N) \subseteq N \ \& \ \bigcup_{k \in \mathbb{N}} \text{List}_N(\underline{k})$  models the Sigma-type  $\Sigma_{x:\mathbb{N}} \text{List}_{\mathbb{N}}(x)$ , then  $\langle \underline{0}, \top \rangle$  and  $\langle \underline{1}, \underline{0} \rangle$  are *total* on  $\Sigma(N, \text{List}_N)$ . However, the totality of  $\langle \underline{0}, \top \rangle$  contradicts  $\langle \underline{1}, \underline{0} \rangle : \Sigma(N, \text{List}_N)$ .



## Existing games cannot interpret Sigma-types (3/4)

**Second example:** Consider the dependent type  $x : \mathbb{N} \vdash \text{List}_N(x)$  Type such that the canonical terms of the simple type  $\text{List}_N(\underline{k})$  for each  $k \in \mathbb{N}$  are  $k$ -lists  $(\underline{n}_1, \underline{n}_2, \dots, \underline{n}_k)$  of numerals.

We interpret  $\text{List}_N$  by the family  $\text{List}_N$  of games  $\text{List}_N(\underline{k})$  ( $k \in \mathbb{N}$ ), which are the  $k$ -times iteration of tensor  $\otimes$  on  $N$  (n.b.,  $\text{List}_N(\underline{0}) = T$ ). If a subgame  $\Sigma(N, \text{List}_N) \subseteq N \ \& \ \bigcup_{k \in \mathbb{N}} \text{List}_N(\underline{k})$  models the Sigma-type  $\Sigma_{x:\mathbb{N}} \text{List}_N(x)$ , then  $\langle \underline{0}, \top \rangle$  and  $\langle \underline{1}, \underline{0} \rangle$  are *total* on  $\Sigma(N, \text{List}_N)$ . However, the totality of  $\langle \underline{0}, \top \rangle$  contradicts  $\langle \underline{1}, \underline{0} \rangle : \Sigma(N, \text{List}_N)$ .



*Second problem:* Strategies  $\sigma : G$  cannot change the game  $G$ .

# Existing games cannot interpret Sigma-types (4/4)

# Existing games cannot interpret Sigma-types (4/4)

Existing method:



## Existing games cannot interpret Sigma-types (4/4)

**Existing method:** Abramsky et al. interpret the Sigma-type  $\Sigma_{x:N} N_b(x)$  by the set of all *lists*  $(\underline{k} : N, \underline{n} : N_b(\underline{k}))$ , and its terms by these *lists*.

## Existing games cannot interpret Sigma-types (4/4)

**Existing method:** Abramsky et al. interpret the Sigma-type  $\Sigma_{x:N} N_b(x)$  by the set of all *lists*  $(\underline{k} : N, \underline{n} : N_b(\underline{k}))$ , and its terms by these *lists*. More generally, they adopt the *syntactic, inductive* method of *contexts*  $\Gamma = A_1.A_2 \dots A_n$  and *context morphisms*

$$(\Delta \xrightarrow{\phi_1} A_1, \Delta \xrightarrow{\phi_2} A_2\{\phi_1\}, \dots, \Delta \xrightarrow{\phi_n} A_n\{(\phi_1, \phi_2, \dots, \phi_{n-1})\}) : \Delta \rightarrow \Gamma.$$

## Existing games cannot interpret Sigma-types (4/4)

**Existing method:** Abramsky et al. interpret the Sigma-type  $\Sigma_{x:N} N_b(x)$  by the set of all *lists*  $(\underline{k} : N, \underline{n} : N_b(\underline{k}))$ , and its terms by these *lists*. More generally, they adopt the *syntactic, inductive* method of *contexts*  $\Gamma = A_1.A_2 \dots A_n$  and *context morphisms*

$$(\Delta \xrightarrow{\phi_1} A_1, \Delta \xrightarrow{\phi_2} A_2\{\phi_1\}, \dots, \Delta \xrightarrow{\phi_n} A_n\{(\phi_1, \phi_2, \dots, \phi_{n-1})\}) : \Delta \rightarrow \Gamma.$$

*This list construction is nothing about game semantics.*

## Existing games cannot interpret Sigma-types (4/4)

**Existing method:** Abramsky et al. interpret the Sigma-type  $\Sigma_{x:N} N_b(x)$  by the set of all *lists* ( $\underline{k} : N, \underline{n} : N_b(\underline{k})$ ), and its terms by these *lists*. More generally, they adopt the *syntactic, inductive* method of *contexts*  $\Gamma = A_1.A_2 \dots A_n$  and *context morphisms*

$$(\Delta \xrightarrow{\phi_1} A_1, \Delta \xrightarrow{\phi_2} A_2\{\phi_1\}, \dots, \Delta \xrightarrow{\phi_n} A_n\{(\phi_1, \phi_2, \dots, \phi_{n-1})\}) : \Delta \rightarrow \Gamma.$$

*This list construction is nothing about game semantics.*

Also, in their approach, *total* morphisms on the game  $\Sigma(N, \text{List}_N)$  are

$$(\underline{k}, \underline{n}_1 \otimes \underline{n}_2 \otimes \dots) \quad (\text{whatever } k \in \mathbb{N} \text{ is})$$

where the second component is an *infinite* iteration of tensor  $\otimes$ .

## Existing games cannot interpret Sigma-types (4/4)

**Existing method:** Abramsky et al. interpret the Sigma-type  $\Sigma_{x:N} N_b(x)$  by the set of all *lists* ( $\underline{k} : N, \underline{n} : N_b(\underline{k})$ ), and its terms by these *lists*. More generally, they adopt the *syntactic, inductive* method of *contexts*  $\Gamma = A_1.A_2 \dots A_n$  and *context morphisms*

$$(\Delta \xrightarrow{\phi_1} A_1, \Delta \xrightarrow{\phi_2} A_2\{\phi_1\}, \dots, \Delta \xrightarrow{\phi_n} A_n\{(\phi_1, \phi_2, \dots, \phi_{n-1})\}) : \Delta \rightarrow \Gamma.$$

*This list construction is nothing about game semantics.*

Also, in their approach, *total* morphisms on the game  $\Sigma(N, \text{List}_N)$  are

$$(\underline{k}, \underline{n}_1 \otimes \underline{n}_2 \otimes \dots) \quad (\text{whatever } k \in \mathbb{N} \text{ is})$$

where the second component is an *infinite* iteration of tensor  $\otimes$ . They focus on a limited form of dependent types to circumvent this problem.

## Existing games cannot interpret Sigma-types (4/4)

**Existing method:** Abramsky et al. interpret the Sigma-type  $\Sigma_{x:N} N_b(x)$  by the set of all *lists* ( $\underline{k} : N, \underline{n} : N_b(\underline{k})$ ), and its terms by these *lists*. More generally, they adopt the *syntactic, inductive* method of *contexts*  $\Gamma = A_1.A_2 \dots A_n$  and *context morphisms*

$$(\Delta \xrightarrow{\phi_1} A_1, \Delta \xrightarrow{\phi_2} A_2\{\phi_1\}, \dots, \Delta \xrightarrow{\phi_n} A_n\{(\phi_1, \phi_2, \dots, \phi_{n-1})\}) : \Delta \rightarrow \Gamma.$$

*This list construction is nothing about game semantics.*

Also, in their approach, *total* morphisms on the game  $\Sigma(N, \text{List}_N)$  are

$$(\underline{k}, \underline{n}_1 \otimes \underline{n}_2 \otimes \dots) \quad (\text{whatever } k \in \mathbb{N} \text{ is})$$

where the second component is an *infinite* iteration of tensor  $\otimes$ . They focus on a limited form of dependent types to circumvent this problem.

*No games interpret more standard dependent types (e.g.,  $\text{List}_N$ ).*

# No equalisers in games

# No equalisers in games

This problem in game semantics of Sigma-types is also related to:



## No equalisers in games

This problem in game semantics of Sigma-types is also related to:

*The category of games and strategies is **not finitely complete**.*

## No equalisers in games

This problem in game semantics of Sigma-types is also related to:

*The category of games and strategies is **not finitely complete**.*

In fact, there is no equaliser of

$$N \& N \begin{array}{c} \xrightarrow{\underline{tt}} \\ \xrightarrow{\geq} \end{array} \Omega$$

since such an equaliser would serve as the impossible game  $\Sigma(N, N_b)$ .

## No equalisers in games

This problem in game semantics of Sigma-types is also related to:

*The category of games and strategies is **not finitely complete**.*

In fact, there is no equaliser of

$$N \& N \begin{array}{c} \xrightarrow{\text{tt}} \\ \xrightarrow{\quad} \\ \cong \end{array} \Omega$$

since such an equaliser would serve as the impossible game  $\Sigma(N, N_b)$ .

Why does it matter?

## No equalisers in games

This problem in game semantics of Sigma-types is also related to:

*The category of games and strategies is **not finitely complete**.*

In fact, there is no equaliser of

$$N \& N \begin{array}{c} \xrightarrow{\text{tt}} \\ \xrightarrow{\cong} \end{array} \Omega$$

since such an equaliser would serve as the impossible game  $\Sigma(N, N_b)$ .

Why does it matter?

- To internalise  $\infty$ -groupoids in the category of games and strategies for game semantics of *homotopy type theory*;

## No equalisers in games

This problem in game semantics of Sigma-types is also related to:

*The category of games and strategies is **not finitely complete**.*

In fact, there is no equaliser of

$$N \& N \begin{array}{c} \xrightarrow{\text{tt}} \\ \xrightarrow{\geq} \end{array} \Omega$$

since such an equaliser would serve as the impossible game  $\Sigma(N, N_b)$ .

Why does it matter?

- To internalise  $\infty$ -groupoids in the category of games and strategies for game semantics of *homotopy type theory*;
- To study *predicative topos* by games.

# My solution: strategy filtering and game changing (1/3)

# My solution: strategy filtering and game changing (1/3)

Observation:

# My solution: strategy filtering and game changing (1/3)

**Observation:** Given a game  $G$ , every position  $s \in G$  is contained in some strategy  $\sigma : G$ .



# My solution: strategy filtering and game changing (1/3)

**Observation:** Given a game  $G$ , every position  $\mathbf{s} \in G$  is contained in some strategy  $\sigma : G$ . Specifically, define  $\sigma := \text{Pref}(\{\mathbf{s}\})^{\text{Even}}$ .

# My solution: strategy filtering and game changing (1/3)

**Observation:** Given a game  $G$ , every position  $\mathbf{s} \in G$  is contained in some strategy  $\sigma : G$ . Specifically, define  $\sigma := \text{Pref}(\{\mathbf{s}\})^{\text{Even}}$ .

*Player does not lose anything by fixing a strategy before a play.*

# My solution: strategy filtering and game changing (1/3)

**Observation:** Given a game  $G$ , every position  $\mathbf{s} \in G$  is contained in some strategy  $\sigma : G$ . Specifically, define  $\sigma := \text{Pref}(\{\mathbf{s}\})^{\text{Even}}$ .

*Player does not lose anything by fixing a strategy before a play.*

Idea:

## My solution: strategy filtering and game changing (1/3)

**Observation:** Given a game  $G$ , every position  $\mathbf{s} \in G$  is contained in some strategy  $\sigma : G$ . Specifically, define  $\sigma := \text{Pref}(\{\mathbf{s}\})^{\text{Even}}$ .

*Player does not lose anything by fixing a strategy before a play.*

**Idea:** To equip each game  $G$  with a map  $f_G : \text{st}(G) \rightarrow \text{sub}(G)$ , and *only permit  $\sigma : G$  whose restriction to  $f_G(\sigma)$  follows the rules of  $f_G(\sigma)$ .*

## My solution: strategy filtering and game changing (1/3)

**Observation:** Given a game  $G$ , every position  $\mathbf{s} \in G$  is contained in some strategy  $\sigma : G$ . Specifically, define  $\sigma := \text{Pref}(\{\mathbf{s}\})^{\text{Even}}$ .

*Player does not lose anything by fixing a strategy before a play.*

**Idea:** To equip each game  $G$  with a map  $f_G : \text{st}(G) \rightarrow \text{sub}(G)$ , and *only permit  $\sigma : G$  whose restriction to  $f_G(\sigma)$  follows the rules of  $f_G(\sigma)$ .*

$$\forall \mathbf{smn} \in \sigma. \mathbf{sm} \in f_G(\sigma) \Rightarrow \mathbf{smn} \in f_G(\sigma). \quad (1)$$

## My solution: strategy filtering and game changing (1/3)

**Observation:** Given a game  $G$ , every position  $\mathbf{s} \in G$  is contained in some strategy  $\sigma : G$ . Specifically, define  $\sigma := \text{Pref}(\{\mathbf{s}\})^{\text{Even}}$ .

*Player does not lose anything by fixing a strategy before a play.*

**Idea:** To equip each game  $G$  with a map  $f_G : \text{st}(G) \rightarrow \text{sub}(G)$ , and *only permit  $\sigma : G$  whose restriction to  $f_G(\sigma)$  follows the rules of  $f_G(\sigma)$ .*

$$\forall \mathbf{smn} \in \sigma. \mathbf{sm} \in f_G(\sigma) \Rightarrow \mathbf{smn} \in f_G(\sigma). \quad (1)$$

A play in such a generalised game  $G = (G, f_G)$  proceeds as follows:

## My solution: strategy filtering and game changing (1/3)

**Observation:** Given a game  $G$ , every position  $\mathbf{s} \in G$  is contained in some strategy  $\sigma : G$ . Specifically, define  $\sigma := \text{Pref}(\{\mathbf{s}\})^{\text{Even}}$ .

*Player does not lose anything by fixing a strategy before a play.*

**Idea:** To equip each game  $G$  with a map  $f_G : \text{st}(G) \rightarrow \text{sub}(G)$ , and *only permit  $\sigma : G$  whose restriction to  $f_G(\sigma)$  follows the rules of  $f_G(\sigma)$ .*

$$\forall \mathbf{smn} \in \sigma. \mathbf{sm} \in f_G(\sigma) \Rightarrow \mathbf{smn} \in f_G(\sigma). \quad (1)$$

A play in such a generalised game  $G = (G, f_G)$  proceeds as follows:

$G$

## My solution: strategy filtering and game changing (1/3)

**Observation:** Given a game  $G$ , every position  $\mathbf{s} \in G$  is contained in some strategy  $\sigma : G$ . Specifically, define  $\sigma := \text{Pref}(\{\mathbf{s}\})^{\text{Even}}$ .

*Player does not lose anything by fixing a strategy before a play.*

**Idea:** To equip each game  $G$  with a map  $f_G : \text{st}(G) \rightarrow \text{sub}(G)$ , and *only permit  $\sigma : G$  whose restriction to  $f_G(\sigma)$  follows the rules of  $f_G(\sigma)$ .*

$$\forall \mathbf{smn} \in \sigma. \mathbf{sm} \in f_G(\sigma) \Rightarrow \mathbf{smn} \in f_G(\sigma). \quad (1)$$

A play in such a generalised game  $G = (G, f_G)$  proceeds as follows:

$$\frac{G}{q_G}$$

What is your strategy?



## My solution: strategy filtering and game changing (1/3)

**Observation:** Given a game  $G$ , every position  $\mathbf{s} \in G$  is contained in some strategy  $\sigma : G$ . Specifically, define  $\sigma := \text{Pref}(\{\mathbf{s}\})^{\text{Even}}$ .

*Player does not lose anything by fixing a strategy before a play.*

**Idea:** To equip each game  $G$  with a map  $f_G : \text{st}(G) \rightarrow \text{sub}(G)$ , and *only permit  $\sigma : G$  whose restriction to  $f_G(\sigma)$  follows the rules of  $f_G(\sigma)$ .*

$$\forall \mathbf{smn} \in \sigma. \mathbf{sm} \in f_G(\sigma) \Rightarrow \mathbf{smn} \in f_G(\sigma). \quad (1)$$

A play in such a generalised game  $G = (G, f_G)$  proceeds as follows:

$$\frac{G}{qG} \quad \text{What is your strategy?}$$

$\sigma$  It is  $\sigma : G$ , which satisfies the axiom (1)!

## My solution: strategy filtering and game changing (1/3)

**Observation:** Given a game  $G$ , every position  $\mathbf{s} \in G$  is contained in some strategy  $\sigma : G$ . Specifically, define  $\sigma := \text{Pref}(\{\mathbf{s}\})^{\text{Even}}$ .

*Player does not lose anything by fixing a strategy before a play.*

**Idea:** To equip each game  $G$  with a map  $f_G : \text{st}(G) \rightarrow \text{sub}(G)$ , and *only permit  $\sigma : G$  whose restriction to  $f_G(\sigma)$  follows the rules of  $f_G(\sigma)$ .*

$$\forall \mathbf{smn} \in \sigma. \mathbf{sm} \in f_G(\sigma) \Rightarrow \mathbf{smn} \in f_G(\sigma). \quad (1)$$

A play in such a generalised game  $G = (G, f_G)$  proceeds as follows:

$$\frac{G}{q_G} \quad \text{What is your strategy?}$$

$$\sigma \quad \text{It is } \sigma : G, \text{ which satisfies the axiom (1)!}$$

$$m_1$$

## My solution: strategy filtering and game changing (1/3)

**Observation:** Given a game  $G$ , every position  $\mathbf{s} \in G$  is contained in some strategy  $\sigma : G$ . Specifically, define  $\sigma := \text{Pref}(\{\mathbf{s}\})^{\text{Even}}$ .

*Player does not lose anything by fixing a strategy before a play.*

**Idea:** To equip each game  $G$  with a map  $f_G : \text{st}(G) \rightarrow \text{sub}(G)$ , and *only permit  $\sigma : G$  whose restriction to  $f_G(\sigma)$  follows the rules of  $f_G(\sigma)$ .*

$$\forall \mathbf{smn} \in \sigma. \mathbf{sm} \in f_G(\sigma) \Rightarrow \mathbf{smn} \in f_G(\sigma). \quad (1)$$

A play in such a generalised game  $G = (G, f_G)$  proceeds as follows:

$$\frac{G}{\begin{array}{l} qG \\ \sigma \\ m_1 \\ m_2 \end{array}} \quad \begin{array}{l} \text{What is your strategy?} \\ \text{It is } \sigma : G, \text{ which satisfies the axiom (1)!} \end{array}$$

## My solution: strategy filtering and game changing (1/3)

**Observation:** Given a game  $G$ , every position  $\mathbf{s} \in G$  is contained in some strategy  $\sigma : G$ . Specifically, define  $\sigma := \text{Pref}(\{\mathbf{s}\})^{\text{Even}}$ .

*Player does not lose anything by fixing a strategy before a play.*

**Idea:** To equip each game  $G$  with a map  $f_G : \text{st}(G) \rightarrow \text{sub}(G)$ , and *only permit  $\sigma : G$  whose restriction to  $f_G(\sigma)$  follows the rules of  $f_G(\sigma)$ .*

$$\forall \mathbf{smn} \in \sigma. \mathbf{sm} \in f_G(\sigma) \Rightarrow \mathbf{smn} \in f_G(\sigma). \quad (1)$$

A play in such a generalised game  $G = (G, f_G)$  proceeds as follows:

$$\frac{G}{\begin{array}{l} qG \\ \sigma \\ m_1 \\ m_2 \\ \vdots \end{array}} \quad \begin{array}{l} \text{What is your strategy?} \\ \text{It is } \sigma : G, \text{ which satisfies the axiom (1)!} \end{array}$$

## My solution: strategy filtering and game changing (1/3)

**Observation:** Given a game  $G$ , every position  $\mathbf{s} \in G$  is contained in some strategy  $\sigma : G$ . Specifically, define  $\sigma := \text{Pref}(\{\mathbf{s}\})^{\text{Even}}$ .

*Player does not lose anything by fixing a strategy before a play.*

**Idea:** To equip each game  $G$  with a map  $f_G : \text{st}(G) \rightarrow \text{sub}(G)$ , and *only permit  $\sigma : G$  whose restriction to  $f_G(\sigma)$  follows the rules of  $f_G(\sigma)$ .*

$$\forall \mathbf{smn} \in \sigma. \mathbf{sm} \in f_G(\sigma) \Rightarrow \mathbf{smn} \in f_G(\sigma). \quad (1)$$

A play in such a generalised game  $G = (G, f_G)$  proceeds as follows:

$$\frac{G}{\begin{array}{l} qG \\ \sigma \\ m_1 \\ m_2 \\ \vdots \end{array}} \quad \begin{array}{l} \text{What is your strategy?} \\ \text{It is } \sigma : G, \text{ which satisfies the axiom (1)!} \\ \\ \\ (m_1 m_2 \dots \text{ is played by } \sigma \text{ in } f_G(\sigma) \subseteq G) \end{array}$$

# My solution: strategy filtering and game changing (2/3)

## My solution: strategy filtering and game changing (2/3)

For instance, we can interpret the Sigma-type  $\Sigma_{x:N} N_b(x)$  by the pair  $\Sigma(N, N_b) = (N \& N, f_{\Sigma(N, N_b)})$ , where  $f_{\Sigma(N, N_b)} : \langle \underline{k}, \underline{n} \rangle \mapsto N \& N_b(\underline{k})$ .

## My solution: strategy filtering and game changing (2/3)

For instance, we can interpret the Sigma-type  $\Sigma_{x:N} N_b(x)$  by the pair  $\Sigma(N, N_b) = (N \& N, f_{\Sigma(N, N_b)})$ , where  $f_{\Sigma(N, N_b)} : \langle \underline{k}, \underline{n} \rangle \mapsto N \& N_b(\underline{k})$ .

$$\underline{\Sigma(N \quad , \quad N_b)}$$



## My solution: strategy filtering and game changing (2/3)

For instance, we can interpret the Sigma-type  $\Sigma_{x:N} N_b(x)$  by the pair  $\Sigma(N, N_b) = (N \& N, f_{\Sigma(N, N_b)})$ , where  $f_{\Sigma(N, N_b)} : \langle \underline{k}, \underline{n} \rangle \mapsto N \& N_b(\underline{k})$ .

$$\frac{\Sigma(N \quad , \quad N_b)}{q_{\Sigma(N, N_b)}}$$

## My solution: strategy filtering and game changing (2/3)

For instance, we can interpret the Sigma-type  $\Sigma_{x:N} N_b(x)$  by the pair  $\Sigma(N, N_b) = (N \& N, f_{\Sigma(N, N_b)})$ , where  $f_{\Sigma(N, N_b)} : \langle \underline{k}, \underline{n} \rangle \mapsto N \& N_b(\underline{k})$ .

$$\frac{\Sigma(N \quad , \quad N_b)}{q_{\Sigma(N, N_b)} \langle \underline{1}, \underline{0} \rangle}$$

## My solution: strategy filtering and game changing (2/3)

For instance, we can interpret the Sigma-type  $\Sigma_{x:N} N_b(x)$  by the pair  $\Sigma(N, N_b) = (N \& N, f_{\Sigma(N, N_b)})$ , where  $f_{\Sigma(N, N_b)} : \langle \underline{k}, \underline{n} \rangle \mapsto N \& N_b(\underline{k})$ .

$$\frac{\Sigma(N \quad , \quad N_b)}{q_{\Sigma(N, N_b)} \langle \underline{1}, \underline{0} \rangle}$$

$q$

## My solution: strategy filtering and game changing (2/3)

For instance, we can interpret the Sigma-type  $\Sigma_{x:N} N_b(x)$  by the pair  $\Sigma(N, N_b) = (N \& N, f_{\Sigma(N, N_b)})$ , where  $f_{\Sigma(N, N_b)} : \langle \underline{k}, \underline{n} \rangle \mapsto N \& N_b(\underline{k})$ .

$$\frac{\Sigma(N \quad , \quad N_b)}{q_{\Sigma(N, N_b)} \langle \underline{1}, \underline{0} \rangle}$$

$q$

$1$

## My solution: strategy filtering and game changing (2/3)

For instance, we can interpret the Sigma-type  $\Sigma_{x:N} N_b(x)$  by the pair  $\Sigma(N, N_b) = (N \& N, f_{\Sigma(N, N_b)})$ , where  $f_{\Sigma(N, N_b)} : \langle \underline{k}, \underline{n} \rangle \mapsto N \& N_b(\underline{k})$ .

$$\frac{\Sigma(N \quad , \quad N_b)}{q_{\Sigma(N, N_b)} \langle \underline{1}, \underline{0} \rangle} \quad \frac{\Sigma(N \quad , \quad N_b)}{1}$$

## My solution: strategy filtering and game changing (2/3)

For instance, we can interpret the Sigma-type  $\Sigma_{x:N} N_b(x)$  by the pair  $\Sigma(N, N_b) = (N \& N, f_{\Sigma(N, N_b)})$ , where  $f_{\Sigma(N, N_b)} : \langle \underline{k}, \underline{n} \rangle \mapsto N \& N_b(\underline{k})$ .

$$\frac{\Sigma(N \quad , \quad N_b)}{q_{\Sigma(N, N_b)} \langle \underline{1}, \underline{0} \rangle} \quad \frac{\Sigma(N \quad , \quad N_b)}{q_{\Sigma(N, N_b)}}$$

$q$   
 $1$

## My solution: strategy filtering and game changing (2/3)

For instance, we can interpret the Sigma-type  $\Sigma_{x:N} N_b(x)$  by the pair  $\Sigma(N, N_b) = (N \& N, f_{\Sigma(N, N_b)})$ , where  $f_{\Sigma(N, N_b)} : \langle \underline{k}, \underline{n} \rangle \mapsto N \& N_b(\underline{k})$ .

$$\begin{array}{c}
 \frac{\Sigma(N \quad , \quad N_b)}{q_{\Sigma(N, N_b)} \langle \underline{1}, \underline{0} \rangle} \\
 q \\
 1
 \end{array}
 \qquad
 \frac{\Sigma(N \quad , \quad N_b)}{q_{\Sigma(N, N_b)} \langle \underline{1}, \underline{0} \rangle}$$

## My solution: strategy filtering and game changing (2/3)

For instance, we can interpret the Sigma-type  $\Sigma_{x:N} N_b(x)$  by the pair  $\Sigma(N, N_b) = (N \& N, f_{\Sigma(N, N_b)})$ , where  $f_{\Sigma(N, N_b)} : \langle \underline{k}, \underline{n} \rangle \mapsto N \& N_b(\underline{k})$ .

$$\begin{array}{ccc}
 \frac{\Sigma(N \quad , \quad N_b)}{q_{\Sigma(N, N_b)} \langle \underline{1}, \underline{0} \rangle} & & \frac{\Sigma(N \quad , \quad N_b)}{q_{\Sigma(N, N_b)} \langle \underline{1}, \underline{0} \rangle} \\
 q & & q \\
 1 & & 
 \end{array}$$



## My solution: strategy filtering and game changing (2/3)

For instance, we can interpret the Sigma-type  $\Sigma_{x:N} N_b(x)$  by the pair  $\Sigma(N, N_b) = (N \& N, f_{\Sigma(N, N_b)})$ , where  $f_{\Sigma(N, N_b)} : \langle \underline{k}, \underline{n} \rangle \mapsto N \& N_b(\underline{k})$ .

$$\begin{array}{ccc}
 \frac{\Sigma(N \quad , \quad N_b)}{q_{\Sigma(N, N_b)} \langle \underline{1}, \underline{0} \rangle} & & \frac{\Sigma(N \quad , \quad N_b)}{q_{\Sigma(N, N_b)} \langle \underline{1}, \underline{0} \rangle} \\
 q & & q \\
 1 & & 0
 \end{array}$$

## My solution: strategy filtering and game changing (2/3)

For instance, we can interpret the Sigma-type  $\Sigma_{x:N} N_b(x)$  by the pair  $\Sigma(N, N_b) = (N \& N, f_{\Sigma(N, N_b)})$ , where  $f_{\Sigma(N, N_b)} : \langle \underline{k}, \underline{n} \rangle \mapsto N \& N_b(\underline{k})$ .

$$\frac{\Sigma(N \quad , \quad N_b)}{q_{\Sigma(N, N_b)} \langle \underline{1}, \underline{0} \rangle} \qquad \frac{\Sigma(N \quad , \quad N_b)}{q_{\Sigma(N, N_b)} \langle \underline{1}, \underline{0} \rangle}$$

$$q \qquad \qquad \qquad q$$

$$1 \qquad \qquad \qquad 0$$

*In contrast, the pairing  $\langle \underline{0}, \underline{1} \rangle$  is prohibited by  $f_{\Sigma(N, N_b)}(\langle \underline{0}, \underline{1} \rangle) = N \& N_b(\underline{0})$  since  $\underline{1}$  violates the rules of  $N_b(\underline{0}) = \text{Pref}(\{q0\})$ .*

## My solution: strategy filtering and game changing (2/3)

For instance, we can interpret the Sigma-type  $\Sigma_{x:N} N_b(x)$  by the pair  $\Sigma(N, N_b) = (N \& N, f_{\Sigma(N, N_b)})$ , where  $f_{\Sigma(N, N_b)} : \langle \underline{k}, \underline{n} \rangle \mapsto N \& N_b(\underline{k})$ .

$$\frac{\Sigma(N \quad , \quad N_b)}{q_{\Sigma(N, N_b)} \langle \underline{1}, \underline{0} \rangle} \qquad \frac{\Sigma(N \quad , \quad N_b)}{q_{\Sigma(N, N_b)} \langle \underline{1}, \underline{0} \rangle}$$

$$q \qquad \qquad \qquad q$$

$$1 \qquad \qquad \qquad 0$$

*In contrast, the pairing  $\langle \underline{0}, \underline{1} \rangle$  is prohibited by  $f_{\Sigma(N, N_b)}(\langle \underline{0}, \underline{1} \rangle) = N \& N_b(\underline{0})$  since  $\underline{1}$  violates the rules of  $N_b(\underline{0}) = \text{Pref}(\{q0\})$ .*

**Intuition:** One may think of the map  $f_G$  as giving an *additional specification* of the rules of the game  $G$  that filters strategies  $\sigma : G$ .

# My solution: strategy filtering and game changing (3/3)

## My solution: strategy filtering and game changing (3/3)

For another example, we can interpret the Sigma-type  $\Sigma_{x:N} \text{List}_N(x)$  by the pair  $\Sigma(N, \text{List}_N) = (N \& \bigcup_{k \in \mathbb{N}} \text{List}_N(\underline{k}), f_{\Sigma(N, \text{List}_N)})$ , where  $f_{\Sigma(N, \text{List}_N)} : \langle \underline{k}, \underline{n_1} \otimes \underline{n_2} \otimes \dots \rangle \mapsto N \& \text{List}_N(\underline{k})$ .

## My solution: strategy filtering and game changing (3/3)

For another example, we can interpret the Sigma-type  $\Sigma_{x:N} \text{List}_N(x)$  by the pair  $\Sigma(N, \text{List}_N) = (N \& \bigcup_{k \in \mathbb{N}} \text{List}_N(\underline{k}), f_{\Sigma(N, \text{List}_N)})$ , where  $f_{\Sigma(N, \text{List}_N)} : \langle \underline{k}, \underline{n_1} \otimes \underline{n_2} \otimes \dots \rangle \mapsto N \& \text{List}_N(\underline{k})$ .

$$\frac{\Sigma(N \quad , \quad \text{List}_N)}{\quad}$$

## My solution: strategy filtering and game changing (3/3)

For another example, we can interpret the Sigma-type  $\Sigma_{x:N} \text{List}_N(x)$  by the pair  $\Sigma(N, \text{List}_N) = (N \& \bigcup_{k \in \mathbb{N}} \text{List}_N(\underline{k}), f_{\Sigma(N, \text{List}_N)})$ , where  $f_{\Sigma(N, \text{List}_N)} : \langle \underline{k}, \underline{n_1} \otimes \underline{n_2} \otimes \dots \rangle \mapsto N \& \text{List}_N(\underline{k})$ .

$$\frac{\Sigma(N \quad , \quad \text{List}_N)}{q_{\Sigma(N, \text{List}_N)}}$$

## My solution: strategy filtering and game changing (3/3)

For another example, we can interpret the Sigma-type  $\Sigma_{x:N} \text{List}_N(x)$  by the pair  $\Sigma(N, \text{List}_N) = (N \& \bigcup_{k \in \mathbb{N}} \text{List}_N(\underline{k}), f_{\Sigma(N, \text{List}_N)})$ , where  $f_{\Sigma(N, \text{List}_N)} : \langle \underline{k}, \underline{n_1} \otimes \underline{n_2} \otimes \dots \rangle \mapsto N \& \text{List}_N(\underline{k})$ .

$$\frac{\Sigma(N \quad , \quad \text{List}_N)}{q_{\Sigma(N, \text{List}_N)} \langle \underline{2}, \underline{0} \otimes \underline{1} \rangle}$$



## My solution: strategy filtering and game changing (3/3)

For another example, we can interpret the Sigma-type  $\Sigma_{x:N} \text{List}_N(x)$  by the pair  $\Sigma(N, \text{List}_N) = (N \ \& \ \bigcup_{k \in \mathbb{N}} \text{List}_N(\underline{k}), f_{\Sigma(N, \text{List}_N)})$ , where  $f_{\Sigma(N, \text{List}_N)} : \langle \underline{k}, \underline{n_1} \otimes \underline{n_2} \otimes \dots \rangle \mapsto N \ \& \ \text{List}_N(\underline{k})$ .

$$\frac{\Sigma(N \quad , \quad \text{List}_N)}{q_{\Sigma(N, \text{List}_N)} \langle \underline{2}, \underline{0} \otimes \underline{1} \rangle} q$$

## My solution: strategy filtering and game changing (3/3)

For another example, we can interpret the Sigma-type  $\Sigma_{x:N} \text{List}_N(x)$  by the pair  $\Sigma(N, \text{List}_N) = (N \ \& \ \bigcup_{k \in \mathbb{N}} \text{List}_N(\underline{k}), f_{\Sigma(N, \text{List}_N)})$ , where  $f_{\Sigma(N, \text{List}_N)} : \langle \underline{k}, \underline{n_1} \otimes \underline{n_2} \otimes \dots \rangle \mapsto N \ \& \ \text{List}_N(\underline{k})$ .

$$\frac{\Sigma(N \quad , \quad \text{List}_N)}{q_{\Sigma(N, \text{List}_N)} \langle \underline{2}, \underline{0} \otimes \underline{1} \rangle}$$

$q$   
 $2$

## My solution: strategy filtering and game changing (3/3)

For another example, we can interpret the Sigma-type  $\Sigma_{x:N} \text{List}_N(x)$  by the pair  $\Sigma(N, \text{List}_N) = (N \& \bigcup_{k \in \mathbb{N}} \text{List}_N(\underline{k}), f_{\Sigma(N, \text{List}_N)})$ , where  $f_{\Sigma(N, \text{List}_N)} : \langle \underline{k}, \underline{n_1} \otimes \underline{n_2} \otimes \dots \rangle \mapsto N \& \text{List}_N(\underline{k})$ .

$$\frac{\Sigma(N \quad , \quad \text{List}_N)}{q_{\Sigma(N, \text{List}_N)} \langle \underline{2}, \underline{0} \otimes \underline{1} \rangle}$$

$$\frac{\Sigma(N \quad , \quad \text{List}_N)}{\quad}$$

$q$   
 $2$

## My solution: strategy filtering and game changing (3/3)

For another example, we can interpret the Sigma-type  $\Sigma_{x:N} \text{List}_N(x)$  by the pair  $\Sigma(N, \text{List}_N) = (N \& \bigcup_{k \in \mathbb{N}} \text{List}_N(\underline{k}), f_{\Sigma(N, \text{List}_N)})$ , where  $f_{\Sigma(N, \text{List}_N)} : \langle \underline{k}, \underline{n}_1 \otimes \underline{n}_2 \otimes \dots \rangle \mapsto N \& \text{List}_N(\underline{k})$ .

$$\frac{\Sigma(N \quad , \quad \text{List}_N)}{q_{\Sigma(N, \text{List}_N)} \langle \underline{2}, \underline{0} \otimes \underline{1} \rangle} \quad \frac{\Sigma(N \quad , \quad \text{List}_N)}{q_{\Sigma(N, \text{List}_N)}}$$

$q$

$2$

## My solution: strategy filtering and game changing (3/3)

For another example, we can interpret the Sigma-type  $\Sigma_{x:N} \text{List}_N(x)$  by the pair  $\Sigma(N, \text{List}_N) = (N \ \& \ \bigcup_{k \in \mathbb{N}} \text{List}_N(\underline{k}), f_{\Sigma(N, \text{List}_N)})$ , where  $f_{\Sigma(N, \text{List}_N)} : \langle \underline{k}, \underline{n_1} \otimes \underline{n_2} \otimes \dots \rangle \mapsto N \ \& \ \text{List}_N(\underline{k})$ .

$$\frac{\Sigma(N \quad , \quad \text{List}_N)}{q_{\Sigma(N, \text{List}_N)} \langle \underline{2}, \underline{0} \otimes \underline{1} \rangle} \qquad \frac{\Sigma(N \quad , \quad \text{List}_N)}{q_{\Sigma(N, \text{List}_N)} \langle \underline{2}, \underline{0} \otimes \underline{1} \rangle}$$

$q$   
 $2$

## My solution: strategy filtering and game changing (3/3)

For another example, we can interpret the Sigma-type  $\Sigma_{x:N} \text{List}_N(x)$  by the pair  $\Sigma(N, \text{List}_N) = (N \& \bigcup_{k \in \mathbb{N}} \text{List}_N(\underline{k}), f_{\Sigma(N, \text{List}_N)})$ , where  $f_{\Sigma(N, \text{List}_N)} : \langle \underline{k}, \underline{n}_1 \otimes \underline{n}_2 \otimes \dots \rangle \mapsto N \& \text{List}_N(\underline{k})$ .

$$\begin{array}{c}
 \frac{\Sigma(N \quad , \quad \text{List}_N)}{q_{\Sigma(N, \text{List}_N)} \langle \underline{2}, \underline{0} \otimes \underline{1} \rangle} \\
 q \\
 2
 \end{array}
 \qquad
 \frac{\Sigma(N \quad , \quad \text{List}_N)}{q_{\Sigma(N, \text{List}_N)} \langle \underline{2}, \underline{0} \otimes \underline{1} \rangle}
 \qquad
 q$$

## My solution: strategy filtering and game changing (3/3)

For another example, we can interpret the Sigma-type  $\Sigma_{x:N} \text{List}_N(x)$  by the pair  $\Sigma(N, \text{List}_N) = (N \& \bigcup_{k \in \mathbb{N}} \text{List}_N(\underline{k}), f_{\Sigma(N, \text{List}_N)})$ , where  $f_{\Sigma(N, \text{List}_N)} : \langle \underline{k}, \underline{n}_1 \otimes \underline{n}_2 \otimes \dots \rangle \mapsto N \& \text{List}_N(\underline{k})$ .

$$\begin{array}{c}
 \frac{\Sigma(N \quad , \quad \text{List}_N)}{q_{\Sigma(N, \text{List}_N)} \langle \underline{2}, \underline{0} \otimes \underline{1} \rangle} \\
 q \\
 2
 \end{array}
 \qquad
 \frac{\Sigma(N \quad , \quad \text{List}_N)}{q_{\Sigma(N, \text{List}_N)} \langle \underline{2}, \underline{0} \otimes \underline{1} \rangle}
 \qquad
 \begin{array}{c}
 q \\
 1
 \end{array}$$

## My solution: strategy filtering and game changing (3/3)

For another example, we can interpret the Sigma-type  $\Sigma_{x:N} \text{List}_N(x)$  by the pair  $\Sigma(N, \text{List}_N) = (N \& \bigcup_{k \in \mathbb{N}} \text{List}_N(\underline{k}), f_{\Sigma(N, \text{List}_N)})$ , where  $f_{\Sigma(N, \text{List}_N)} : \langle \underline{k}, \underline{n}_1 \otimes \underline{n}_2 \otimes \dots \rangle \mapsto N \& \text{List}_N(\underline{k})$ .

$$\begin{array}{c}
 \frac{\Sigma(N \quad , \quad \text{List}_N)}{q_{\Sigma(N, \text{List}_N)} \langle \underline{2}, \underline{0} \otimes \underline{1} \rangle} \\
 q \\
 2
 \end{array}
 \qquad
 \frac{\Sigma(N \quad , \quad \text{List}_N)}{q_{\Sigma(N, \text{List}_N)} \langle \underline{2}, \underline{0} \otimes \underline{1} \rangle}
 \qquad
 \begin{array}{c}
 q \\
 1 \\
 q
 \end{array}$$



## My solution: strategy filtering and game changing (3/3)

For another example, we can interpret the Sigma-type  $\Sigma_{x:N} \text{List}_N(x)$  by the pair  $\Sigma(N, \text{List}_N) = (N \& \bigcup_{k \in \mathbb{N}} \text{List}_N(\underline{k}), f_{\Sigma(N, \text{List}_N)})$ , where  $f_{\Sigma(N, \text{List}_N)} : \langle \underline{k}, \underline{n_1} \otimes \underline{n_2} \otimes \dots \rangle \mapsto N \& \text{List}_N(\underline{k})$ .

$$\begin{array}{c}
 \frac{\Sigma(N \quad , \quad \text{List}_N)}{q_{\Sigma(N, \text{List}_N)} \langle \underline{2}, \underline{0} \otimes \underline{1} \rangle} \\
 q \\
 2
 \end{array}
 \qquad
 \frac{\Sigma(N \quad , \quad \text{List}_N)}{q_{\Sigma(N, \text{List}_N)} \langle \underline{2}, \underline{0} \otimes \underline{1} \rangle}
 \qquad
 \begin{array}{c}
 q \\
 1 \\
 q \\
 0
 \end{array}$$

## My solution: strategy filtering and game changing (3/3)

For another example, we can interpret the Sigma-type  $\Sigma_{x:\mathbb{N}}\text{List}_N(x)$  by the pair  $\Sigma(N, \text{List}_N) = (N \& \bigcup_{k \in \mathbb{N}} \text{List}_N(k), f_{\Sigma(N, \text{List}_N)})$ , where  $f_{\Sigma(N, \text{List}_N)} : \langle \underline{k}, \underline{n}_1 \otimes \underline{n}_2 \otimes \dots \rangle \mapsto N \& \text{List}_N(\underline{k})$ .

$$\begin{array}{c}
 \frac{\Sigma(N \quad , \quad \text{List}_N)}{q_{\Sigma(N, \text{List}_N)} \langle \underline{2}, \underline{0} \otimes \underline{1} \rangle} \\
 q \\
 2
 \end{array}
 \qquad
 \frac{\Sigma(N \quad , \quad \text{List}_N)}{q_{\Sigma(N, \text{List}_N)} \langle \underline{2}, \underline{0} \otimes \underline{1} \rangle}
 \begin{array}{c}
 q \\
 1 \\
 q \\
 0
 \end{array}$$

The declaration of the strategy  $\langle \underline{2}, \underline{0} \otimes \underline{1} \rangle : N \& \bigcup_{k \in \mathbb{N}} \text{List}_N(k)$  fixes the underlying game  $N \& (N \otimes N) \subseteq N \& \bigcup_{k \in \mathbb{N}} \text{List}_N(k)$ , so that  $\langle \underline{2}, \underline{0} \otimes \underline{1} \rangle$  is total *without redundant computation*.

## My solution: strategy filtering and game changing (3/3)

For another example, we can interpret the Sigma-type  $\Sigma_{x:\mathbb{N}}\text{List}_N(x)$  by the pair  $\Sigma(N, \text{List}_N) = (N \& \bigcup_{k \in \mathbb{N}} \text{List}_N(k), f_{\Sigma(N, \text{List}_N)})$ , where  $f_{\Sigma(N, \text{List}_N)} : \langle \underline{k}, \underline{n_1} \otimes \underline{n_2} \otimes \dots \rangle \mapsto N \& \text{List}_N(\underline{k})$ .

$$\begin{array}{c}
 \Sigma(N, \text{List}_N) \\
 \hline
 q_{\Sigma(N, \text{List}_N)} \\
 \langle \underline{2}, \underline{0} \otimes \underline{1} \rangle \\
 q \\
 2
 \end{array}
 \qquad
 \frac{\Sigma(N, \text{List}_N)}{q_{\Sigma(N, \text{List}_N)} \langle \underline{2}, \underline{0} \otimes \underline{1} \rangle}
 \qquad
 \begin{array}{c}
 q \\
 1 \\
 q \\
 0
 \end{array}$$

The declaration of the strategy  $\langle \underline{2}, \underline{0} \otimes \underline{1} \rangle : N \& \bigcup_{k \in \mathbb{N}} \text{List}_N(k)$  fixes the underlying game  $N \& (N \otimes N) \subseteq N \& \bigcup_{k \in \mathbb{N}} \text{List}_N(k)$ , so that  $\langle \underline{2}, \underline{0} \otimes \underline{1} \rangle$  is total *without redundant computation*.

**Intuition:** One may think of the map  $f_G$  as *giving an additional power for Player to change the rules* of the game  $G$ .

Caution: not so easy!

## Caution: not so easy!

This bold idea requires extremely careful attention to technical details.

## Caution: not so easy!

This bold idea requires extremely careful attention to technical details. For instance, *the initial two moves  $q_G\sigma$  should not be part of ordinary plays* between Player and Opponent

## Caution: not so easy!

This bold idea requires extremely careful attention to technical details. For instance, *the initial two moves  $q_G\sigma$  should not be part of ordinary plays* between Player and Opponent since otherwise by duality

## Caution: not so easy!

This bold idea requires extremely careful attention to technical details. For instance, *the initial two moves  $q_G\sigma$  should not be part of ordinary plays* between Player and Opponent since otherwise by duality

$$\frac{N \multimap N \quad \multimap \quad N \multimap N}{\quad}$$



## Caution: not so easy!

This bold idea requires extremely careful attention to technical details. For instance, *the initial two moves  $q_G\sigma$  should not be part of ordinary plays* between Player and Opponent since otherwise by duality

$$\frac{N \multimap N \quad \quad \quad \multimap \quad \quad \quad N \multimap N}{q(N \multimap N) \multimap (N \multimap N)}$$

## Caution: not so easy!

This bold idea requires extremely careful attention to technical details. For instance, *the initial two moves  $q_G\sigma$  should not be part of ordinary plays* between Player and Opponent since otherwise by duality

$$\frac{N \multimap N \quad \multimap \quad N \multimap N}{q(N \multimap N) \multimap (N \multimap N) \text{ succ } \circ (-)}$$

## Caution: not so easy!

This bold idea requires extremely careful attention to technical details. For instance, *the initial two moves  $q_G\sigma$  should not be part of ordinary plays* between Player and Opponent since otherwise by duality

$$\frac{N \multimap N \quad \multimap \quad N \multimap N}{q(N \multimap N) \multimap (N \multimap N) \text{ succ } \circ (-)} q_{N \multimap N}$$

## Caution: not so easy!

This bold idea requires extremely careful attention to technical details. For instance, *the initial two moves  $q_G\sigma$  should not be part of ordinary plays* between Player and Opponent since otherwise by duality

$$\begin{array}{c}
 N \multimap N \qquad \multimap \qquad N \multimap N \\
 \hline
 q_{(N \multimap N) \multimap (N \multimap N)} \\
 \text{succ} \circ (-) \\
 \\
 q_{N \multimap N} \qquad \qquad \qquad q_{N \multimap N}
 \end{array}$$

## Caution: not so easy!

This bold idea requires extremely careful attention to technical details. For instance, *the initial two moves  $q_G\sigma$  should not be part of ordinary plays* between Player and Opponent since otherwise by duality

$$\begin{array}{c}
 N \multimap N \qquad \qquad \multimap \qquad \qquad N \multimap N \\
 \hline
 q(N \multimap N) \multimap (N \multimap N) \\
 \text{succ} \circ (-) \\
 \\
 q_{N \multimap N} \qquad \qquad \qquad q_{N \multimap N} \\
 \sigma
 \end{array}$$

## Caution: not so easy!

This bold idea requires extremely careful attention to technical details. For instance, *the initial two moves  $q_G\sigma$  should not be part of ordinary plays* between Player and Opponent since otherwise by duality

$$\begin{array}{c}
 \frac{N \multimap N \quad \multimap \quad N \multimap N}{q(N \multimap N) \multimap (N \multimap N)} \\
 \text{succ} \circ (-) \\
 \\
 \begin{array}{ccc}
 q_{N \multimap N} & & q_{N \multimap N} \\
 \sigma & & \\
 & & \text{succ} \circ \sigma
 \end{array}
 \end{array}$$

## Caution: not so easy!

This bold idea requires extremely careful attention to technical details. For instance, *the initial two moves  $q_G\sigma$  should not be part of ordinary plays* between Player and Opponent since otherwise by duality

$$\begin{array}{c}
 \frac{N \multimap N \quad \multimap \quad N \multimap N}{q(N \multimap N) \multimap (N \multimap N)} \\
 \text{succ} \circ (-) \\
 \\
 \begin{array}{ccc}
 q_{N \multimap N} & & q_{N \multimap N} \\
 \sigma & & \\
 & & \text{succ} \circ \sigma \\
 & & \\
 & \vdots & 
 \end{array}
 \end{array}$$

## Caution: not so easy!

This bold idea requires extremely careful attention to technical details. For instance, *the initial two moves  $q_G\sigma$  should not be part of ordinary plays* between Player and Opponent since otherwise by duality

$$\begin{array}{c}
 \frac{N \multimap N \quad \quad \quad \multimap \quad \quad \quad N \multimap N}{q(N \multimap N) \multimap (N \multimap N)} \\
 \text{succ} \circ (-) \\
 \\
 \begin{array}{ccc}
 q_{N \multimap N} & & q_{N \multimap N} \\
 \sigma & & \\
 & & \text{succ} \circ \sigma \\
 & & \\
 & \vdots & 
 \end{array}
 \end{array}$$

the intensionality of game semantics would *collapse*.



## Caution: not so easy!

This bold idea requires extremely careful attention to technical details. For instance, *the initial two moves  $q_G\sigma$  should not be part of ordinary plays* between Player and Opponent since otherwise by duality

$$\begin{array}{c}
 \frac{N \multimap N \quad \multimap \quad N \multimap N}{q(N \multimap N) \multimap (N \multimap N)} \\
 \text{succ} \circ (-) \\
 \\
 \begin{array}{ccc}
 q_{N \multimap N} & & q_{N \multimap N} \\
 \sigma & & \\
 & & \text{succ} \circ \sigma \\
 & & \\
 & & \vdots
 \end{array}
 \end{array}$$

the intensionality of game semantics would *collapse*.

However, without a declaration of  $\sigma : N \multimap N$  on the domain by Opponent, *what is the underlying game  $f_{N \multimap N}(\sigma)$  on the domain?*

# Pi-types in predicate games (1/3)

# Pi-types in predicate games (1/3)

Let me call my generalised games *predicate (p-) games*.

# Pi-types in predicate games (1/3)

Let me call my generalised games *predicate (p-) games*.

Notation:

## Pi-types in predicate games (1/3)

Let me call my generalised games *predicate (p-) games*.

**Notation:** Write  $\Gamma = (|\Gamma|, f_\Gamma)$  for p-games, and  $\gamma : \Gamma$  if  $\gamma : |\Gamma|$  passes the test by  $f_\Gamma$ .

## Pi-types in predicate games (1/3)

Let me call my generalised games *predicate (p-) games*.

**Notation:** Write  $\Gamma = (|\Gamma|, f_\Gamma)$  for p-games, and  $\gamma : \Gamma$  if  $\gamma : |\Gamma|$  passes the test by  $f_\Gamma$ . Let  $A = (A(\gamma))_{\gamma:\Gamma}$  be a family of p-games  $A(\gamma)$  equipped with a game  $|A|$  such that  $|A(\gamma)| = |A|$  for all  $\gamma : \Gamma$ .

## Pi-types in predicate games (1/3)

Let me call my generalised games *predicate (p-) games*.

**Notation:** Write  $\Gamma = (|\Gamma|, f_\Gamma)$  for p-games, and  $\gamma : \Gamma$  if  $\gamma : |\Gamma|$  passes the test by  $f_\Gamma$ .  
Let  $A = (A(\gamma))_{\gamma:\Gamma}$  be a family of p-games  $A(\gamma)$  equipped with a game  $|A|$  such that  $|A(\gamma)| = |A|$  for all  $\gamma : \Gamma$ .

**Idea:**

## Pi-types in predicate games (1/3)

Let me call my generalised games *predicate (p-) games*.

**Notation:** Write  $\Gamma = (|\Gamma|, f_\Gamma)$  for p-games, and  $\gamma : \Gamma$  if  $\gamma : |\Gamma|$  passes the test by  $f_\Gamma$ .  
 Let  $A = (A(\gamma))_{\gamma:\Gamma}$  be a family of p-games  $A(\gamma)$  equipped with a game  $|A|$  such that  $|A(\gamma)| = |A|$  for all  $\gamma : \Gamma$ .

**Idea:** Define the *linear implication*  $\Gamma \multimap \Delta$  between p-games by



## Pi-types in predicate games (1/3)

Let me call my generalised games *predicate (p-) games*.

**Notation:** Write  $\Gamma = (|\Gamma|, f_\Gamma)$  for p-games, and  $\gamma : \Gamma$  if  $\gamma : |\Gamma|$  passes the test by  $f_\Gamma$ .  
 Let  $A = (A(\gamma))_{\gamma:\Gamma}$  be a family of p-games  $A(\gamma)$  equipped with a game  $|A|$  such that  $|A(\gamma)| = |A|$  for all  $\gamma : \Gamma$ .

**Idea:** Define the *linear implication*  $\Gamma \multimap \Delta$  between p-games by

- $|\Gamma \multimap \Delta| := |\Gamma| \multimap |\Delta|$ ;

# Pi-types in predicate games (1/3)

Let me call my generalised games *predicate (p-) games*.

**Notation:** Write  $\Gamma = (|\Gamma|, f_\Gamma)$  for p-games, and  $\gamma : \Gamma$  if  $\gamma : |\Gamma|$  passes the test by  $f_\Gamma$ . Let  $A = (A(\gamma))_{\gamma:\Gamma}$  be a family of p-games  $A(\gamma)$  equipped with a game  $|A|$  such that  $|A(\gamma)| = |A|$  for all  $\gamma : \Gamma$ .

**Idea:** Define the *linear implication*  $\Gamma \multimap \Delta$  between p-games by

- $|\Gamma \multimap \Delta| := |\Gamma| \multimap |\Delta|$ ;
- $\phi : |\Gamma \multimap \Delta|$  passes the test *if it follows the rules of the subgame*  
 $\bar{\gamma}_\Gamma \multimap f_\Delta(\phi \circ \gamma) \subseteq |\Gamma \multimap \Delta|$  for *all  $\gamma : \Gamma$  that is not yet excluded, i.e., the current position  $\mathbf{s}$  is compatible with  $\gamma$  (or  $\mathbf{s} \upharpoonright |\Gamma| \in \bar{\gamma}_\Gamma$ )*, where  $\bar{\gamma}_\Gamma$  is the subgame of  $f_\Gamma(\gamma)$  played by Player and  $\gamma$  (Opponent), i.e.,

$$\bar{\gamma}_\Gamma := \{\epsilon\} \cup \{sm \in f_\Gamma(\gamma) \mid \mathbf{s} \in \bar{\gamma}_\Gamma\} \cup \{tlr \in \gamma \mid \mathbf{tl} \in \bar{\gamma}_\Gamma\}.$$

# Pi-types in predicate games (1/3)

Let me call my generalised games *predicate (p-) games*.

**Notation:** Write  $\Gamma = (|\Gamma|, f_\Gamma)$  for p-games, and  $\gamma : \Gamma$  if  $\gamma : |\Gamma|$  passes the test by  $f_\Gamma$ . Let  $A = (A(\gamma))_{\gamma:\Gamma}$  be a family of p-games  $A(\gamma)$  equipped with a game  $|A|$  such that  $|A(\gamma)| = |A|$  for all  $\gamma : \Gamma$ .

**Idea:** Define the *linear implication*  $\Gamma \multimap \Delta$  between p-games by

- $|\Gamma \multimap \Delta| := |\Gamma| \multimap |\Delta|$ ;
- $\phi : |\Gamma \multimap \Delta|$  passes the test *if it follows the rules of the subgame*  
 $\bar{\gamma}_\Gamma \multimap f_\Delta(\phi \circ \gamma) \subseteq |\Gamma \multimap \Delta|$  for *all*  $\gamma : \Gamma$  that is not yet excluded, i.e., the current position  $\mathbf{s}$  is compatible with  $\gamma$  (or  $\mathbf{s} \upharpoonright |\Gamma| \in \bar{\gamma}_\Gamma$ ), where  $\bar{\gamma}_\Gamma$  is the subgame of  $f_\Gamma(\gamma)$  played by Player and  $\gamma$  (Opponent), i.e.,

$$\bar{\gamma}_\Gamma := \{\epsilon\} \cup \{ \mathbf{sm} \in f_\Gamma(\gamma) \mid \mathbf{s} \in \bar{\gamma}_\Gamma \} \cup \{ \mathbf{tlr} \in \gamma \mid \mathbf{tl} \in \bar{\gamma}_\Gamma \}.$$

$$\frac{\Sigma(N, \text{List}_N) \quad \multimap \quad N}{\quad}$$

# Pi-types in predicate games (1/3)

Let me call my generalised games *predicate (p-) games*.

**Notation:** Write  $\Gamma = (|\Gamma|, f_\Gamma)$  for p-games, and  $\gamma : \Gamma$  if  $\gamma : |\Gamma|$  passes the test by  $f_\Gamma$ . Let  $A = (A(\gamma))_{\gamma:\Gamma}$  be a family of p-games  $A(\gamma)$  equipped with a game  $|A|$  such that  $|A(\gamma)| = |A|$  for all  $\gamma : \Gamma$ .

**Idea:** Define the *linear implication*  $\Gamma \multimap \Delta$  between p-games by

- $|\Gamma \multimap \Delta| := |\Gamma| \multimap |\Delta|$ ;
- $\phi : |\Gamma \multimap \Delta|$  passes the test *if it follows the rules of the subgame*  
 $\bar{\gamma}_\Gamma \multimap f_\Delta(\phi \circ \gamma) \subseteq |\Gamma \multimap \Delta|$  for *all*  $\gamma : \Gamma$  that is not yet excluded, i.e., the current position  $\mathbf{s}$  is compatible with  $\gamma$  (or  $\mathbf{s} \upharpoonright |\Gamma| \in \bar{\gamma}_\Gamma$ ), where  $\bar{\gamma}_\Gamma$  is the subgame of  $f_\Gamma(\gamma)$  played by Player and  $\gamma$  (Opponent), i.e.,

$$\bar{\gamma}_\Gamma := \{\epsilon\} \cup \{\mathbf{sm} \in f_\Gamma(\gamma) \mid \mathbf{s} \in \bar{\gamma}_\Gamma\} \cup \{\mathbf{tlr} \in \gamma \mid \mathbf{tl} \in \bar{\gamma}_\Gamma\}.$$

$$\frac{\Sigma(N, \text{List}_N) \quad \multimap \quad N}{q_{\Delta \multimap \Gamma}}$$

# Pi-types in predicate games (1/3)

Let me call my generalised games *predicate (p-) games*.

**Notation:** Write  $\Gamma = (|\Gamma|, f_\Gamma)$  for p-games, and  $\gamma : \Gamma$  if  $\gamma : |\Gamma|$  passes the test by  $f_\Gamma$ . Let  $A = (A(\gamma))_{\gamma:\Gamma}$  be a family of p-games  $A(\gamma)$  equipped with a game  $|A|$  such that  $|A(\gamma)| = |A|$  for all  $\gamma : \Gamma$ .

**Idea:** Define the *linear implication*  $\Gamma \multimap \Delta$  between p-games by

- $|\Gamma \multimap \Delta| := |\Gamma| \multimap |\Delta|$ ;
- $\phi : |\Gamma \multimap \Delta|$  passes the test *if it follows the rules of the subgame*  $\bar{\gamma}_\Gamma \multimap f_\Delta(\phi \circ \gamma) \subseteq |\Gamma \multimap \Delta|$  for *all  $\gamma : \Gamma$  that is not yet excluded, i.e., the current position  $\mathbf{s}$  is compatible with  $\gamma$  (or  $\mathbf{s} \upharpoonright |\Gamma| \in \bar{\gamma}_\Gamma$ )*, where  $\bar{\gamma}_\Gamma$  is the subgame of  $f_\Gamma(\gamma)$  played by Player and  $\gamma$  (Opponent), i.e.,

$$\bar{\gamma}_\Gamma := \{\epsilon\} \cup \{ \mathbf{sm} \in f_\Gamma(\gamma) \mid \mathbf{s} \in \bar{\gamma}_\Gamma \} \cup \{ \mathbf{tlr} \in \gamma \mid \mathbf{tl} \in \bar{\gamma}_\Gamma \}.$$

$$\frac{\Sigma(N, \text{List}_N) \quad \multimap \quad N}{q_{\Delta \multimap \Gamma} \quad \pi_1}$$

# Pi-types in predicate games (1/3)

Let me call my generalised games *predicate (p-) games*.

**Notation:** Write  $\Gamma = (|\Gamma|, f_\Gamma)$  for p-games, and  $\gamma : \Gamma$  if  $\gamma : |\Gamma|$  passes the test by  $f_\Gamma$ . Let  $A = (A(\gamma))_{\gamma:\Gamma}$  be a family of p-games  $A(\gamma)$  equipped with a game  $|A|$  such that  $|A(\gamma)| = |A|$  for all  $\gamma : \Gamma$ .

**Idea:** Define the *linear implication*  $\Gamma \multimap \Delta$  between p-games by

- $|\Gamma \multimap \Delta| := |\Gamma| \multimap |\Delta|$ ;
- $\phi : |\Gamma \multimap \Delta|$  passes the test *if it follows the rules of the subgame*  
 $\bar{\gamma}_\Gamma \multimap f_\Delta(\phi \circ \gamma) \subseteq |\Gamma \multimap \Delta|$  for *all*  $\gamma : \Gamma$  that is not yet excluded, i.e., the current position  $\mathbf{s}$  is compatible with  $\gamma$  (or  $\mathbf{s} \upharpoonright |\Gamma| \in \bar{\gamma}_\Gamma$ ), where  $\bar{\gamma}_\Gamma$  is the subgame of  $f_\Gamma(\gamma)$  played by Player and  $\gamma$  (Opponent), i.e.,

$$\bar{\gamma}_\Gamma := \{\epsilon\} \cup \{ \mathbf{sm} \in f_\Gamma(\gamma) \mid \mathbf{s} \in \bar{\gamma}_\Gamma \} \cup \{ \mathbf{tlr} \in \gamma \mid \mathbf{tl} \in \bar{\gamma}_\Gamma \}.$$

$$\frac{\Sigma(N, \text{List}_N) \quad \multimap \quad N}{q_{\Delta \multimap \Gamma}} \quad \pi_1 \quad q$$

# Pi-types in predicate games (1/3)

Let me call my generalised games *predicate (p-) games*.

**Notation:** Write  $\Gamma = (|\Gamma|, f_\Gamma)$  for p-games, and  $\gamma : \Gamma$  if  $\gamma : |\Gamma|$  passes the test by  $f_\Gamma$ . Let  $A = (A(\gamma))_{\gamma:\Gamma}$  be a family of p-games  $A(\gamma)$  equipped with a game  $|A|$  such that  $|A(\gamma)| = |A|$  for all  $\gamma : \Gamma$ .

**Idea:** Define the *linear implication*  $\Gamma \multimap \Delta$  between p-games by

- $|\Gamma \multimap \Delta| := |\Gamma| \multimap |\Delta|$ ;
- $\phi : |\Gamma \multimap \Delta|$  passes the test *if it follows the rules of the subgame*  
 $\bar{\gamma}_\Gamma \multimap f_\Delta(\phi \circ \gamma) \subseteq |\Gamma \multimap \Delta|$  for *all*  $\gamma : \Gamma$  that is not yet excluded, i.e., the current position  $\mathbf{s}$  is compatible with  $\gamma$  (or  $\mathbf{s} \upharpoonright |\Gamma| \in \bar{\gamma}_\Gamma$ ), where  $\bar{\gamma}_\Gamma$  is the subgame of  $f_\Gamma(\gamma)$  played by Player and  $\gamma$  (Opponent), i.e.,

$$\bar{\gamma}_\Gamma := \{\epsilon\} \cup \{sm \in f_\Gamma(\gamma) \mid \mathbf{s} \in \bar{\gamma}_\Gamma\} \cup \{tlr \in \gamma \mid \mathbf{tl} \in \bar{\gamma}_\Gamma\}.$$

$$\frac{\Sigma(N, \text{List}_N) \quad \multimap \quad N}{q_{\Delta \multimap \Gamma} \quad \pi_1} \quad q$$

$q$

# Pi-types in predicate games (1/3)

Let me call my generalised games *predicate (p-) games*.

**Notation:** Write  $\Gamma = (|\Gamma|, f_\Gamma)$  for p-games, and  $\gamma : \Gamma$  if  $\gamma : |\Gamma|$  passes the test by  $f_\Gamma$ . Let  $A = (A(\gamma))_{\gamma:\Gamma}$  be a family of p-games  $A(\gamma)$  equipped with a game  $|A|$  such that  $|A(\gamma)| = |A|$  for all  $\gamma : \Gamma$ .

**Idea:** Define the *linear implication*  $\Gamma \multimap \Delta$  between p-games by

- $|\Gamma \multimap \Delta| := |\Gamma| \multimap |\Delta|$ ;
- $\phi : |\Gamma \multimap \Delta|$  passes the test *if it follows the rules of the subgame*  
 $\bar{\gamma}_\Gamma \multimap f_\Delta(\phi \circ \gamma) \subseteq |\Gamma \multimap \Delta|$  for *all*  $\gamma : \Gamma$  that is not yet excluded, i.e., the current position  $\mathbf{s}$  is compatible with  $\gamma$  (or  $\mathbf{s} \upharpoonright |\Gamma| \in \bar{\gamma}_\Gamma$ ), where  $\bar{\gamma}_\Gamma$  is the subgame of  $f_\Gamma(\gamma)$  played by Player and  $\gamma$  (Opponent), i.e.,

$$\bar{\gamma}_\Gamma := \{\epsilon\} \cup \{ \mathbf{sm} \in f_\Gamma(\gamma) \mid \mathbf{s} \in \bar{\gamma}_\Gamma \} \cup \{ \mathbf{t}l\mathbf{r} \in \gamma \mid \mathbf{t}l \in \bar{\gamma}_\Gamma \}.$$

$$\frac{\Sigma(N, \text{List}_N) \quad \multimap \quad N}{q_{\Delta \multimap \Gamma} \quad \pi_1} \quad q$$

$$q$$

$$k$$



# Pi-types in predicate games (1/3)

Let me call my generalised games *predicate (p-) games*.

**Notation:** Write  $\Gamma = (|\Gamma|, f_\Gamma)$  for p-games, and  $\gamma : \Gamma$  if  $\gamma : |\Gamma|$  passes the test by  $f_\Gamma$ . Let  $A = (A(\gamma))_{\gamma:\Gamma}$  be a family of p-games  $A(\gamma)$  equipped with a game  $|A|$  such that  $|A(\gamma)| = |A|$  for all  $\gamma : \Gamma$ .

**Idea:** Define the *linear implication*  $\Gamma \multimap \Delta$  between p-games by

- $|\Gamma \multimap \Delta| := |\Gamma| \multimap |\Delta|$ ;
- $\phi : |\Gamma \multimap \Delta|$  passes the test *if it follows the rules of the subgame*  
 $\bar{\gamma}_\Gamma \multimap f_\Delta(\phi \circ \gamma) \subseteq |\Gamma \multimap \Delta|$  for *all*  $\gamma : \Gamma$  that is not yet excluded, i.e., the current position  $\mathbf{s}$  is compatible with  $\gamma$  (or  $\mathbf{s} \upharpoonright |\Gamma| \in \bar{\gamma}_\Gamma$ ), where  $\bar{\gamma}_\Gamma$  is the subgame of  $f_\Gamma(\gamma)$  played by Player and  $\gamma$  (Opponent), i.e.,

$$\bar{\gamma}_\Gamma := \{\epsilon\} \cup \{ \mathbf{sm} \in f_\Gamma(\gamma) \mid \mathbf{s} \in \bar{\gamma}_\Gamma \} \cup \{ \mathbf{t}l\mathbf{r} \in \gamma \mid \mathbf{t}l \in \bar{\gamma}_\Gamma \}.$$

$$\frac{\Sigma(N, \text{List}_N) \quad \multimap \quad N}{q_{\Delta \multimap \Gamma} \quad \pi_1} \quad q$$

$$q$$

$$k$$

$$k$$

# Pi-types in predicate games (1/3)

Let me call my generalised games *predicate (p-) games*.

**Notation:** Write  $\Gamma = (|\Gamma|, f_\Gamma)$  for p-games, and  $\gamma : \Gamma$  if  $\gamma : |\Gamma|$  passes the test by  $f_\Gamma$ . Let  $A = (A(\gamma))_{\gamma:\Gamma}$  be a family of p-games  $A(\gamma)$  equipped with a game  $|A|$  such that  $|A(\gamma)| = |A|$  for all  $\gamma : \Gamma$ .

**Idea:** Define the *linear implication*  $\Gamma \multimap \Delta$  between p-games by

- $|\Gamma \multimap \Delta| := |\Gamma| \multimap |\Delta|$ ;
- $\phi : |\Gamma \multimap \Delta|$  passes the test *if it follows the rules of the subgame*  
 $\bar{\gamma}_\Gamma \multimap f_\Delta(\phi \circ \gamma) \subseteq |\Gamma \multimap \Delta|$  for *all*  $\gamma : \Gamma$  that is not yet excluded, i.e., the current position  $\mathbf{s}$  is compatible with  $\gamma$  (or  $\mathbf{s} \upharpoonright |\Gamma| \in \bar{\gamma}_\Gamma$ ), where  $\bar{\gamma}_\Gamma$  is the subgame of  $f_\Gamma(\gamma)$  played by Player and  $\gamma$  (Opponent), i.e.,

$$\bar{\gamma}_\Gamma := \{\epsilon\} \cup \{sm \in f_\Gamma(\gamma) \mid \mathbf{s} \in \bar{\gamma}_\Gamma\} \cup \{tlr \in \gamma \mid tl \in \bar{\gamma}_\Gamma\}.$$

$$\frac{\Sigma(N, \text{List}_N) \multimap N}{\frac{q_{\Delta \multimap \Gamma}}{\pi_1} \quad q} \quad \frac{\Sigma(N, \text{List}_N) \multimap N}{q} \quad k$$

# Pi-types in predicate games (1/3)

Let me call my generalised games *predicate (p-) games*.

**Notation:** Write  $\Gamma = (|\Gamma|, f_\Gamma)$  for p-games, and  $\gamma : \Gamma$  if  $\gamma : |\Gamma|$  passes the test by  $f_\Gamma$ . Let  $A = (A(\gamma))_{\gamma:\Gamma}$  be a family of p-games  $A(\gamma)$  equipped with a game  $|A|$  such that  $|A(\gamma)| = |A|$  for all  $\gamma : \Gamma$ .

**Idea:** Define the *linear implication*  $\Gamma \multimap \Delta$  between p-games by

- $|\Gamma \multimap \Delta| := |\Gamma| \multimap |\Delta|$ ;
- $\phi : |\Gamma \multimap \Delta|$  passes the test *if it follows the rules of the subgame*  
 $\bar{\gamma}_\Gamma \multimap f_\Delta(\phi \circ \gamma) \subseteq |\Gamma \multimap \Delta|$  for *all*  $\gamma : \Gamma$  that is not yet excluded, i.e., the current position  $\mathbf{s}$  is compatible with  $\gamma$  (or  $\mathbf{s} \upharpoonright |\Gamma| \in \bar{\gamma}_\Gamma$ ), where  $\bar{\gamma}_\Gamma$  is the subgame of  $f_\Gamma(\gamma)$  played by Player and  $\gamma$  (Opponent), i.e.,

$$\bar{\gamma}_\Gamma := \{\epsilon\} \cup \{sm \in f_\Gamma(\gamma) \mid \mathbf{s} \in \bar{\gamma}_\Gamma\} \cup \{tlr \in \gamma \mid tl \in \bar{\gamma}_\Gamma\}.$$

$$\frac{\Sigma(N, \text{List}_N) \quad \multimap \quad N}{q_{\Delta \multimap \Gamma} \quad \pi_1} \quad \frac{\Sigma(N, \text{List}_N) \quad \multimap \quad N}{q_{\Delta \multimap \Gamma}}$$

$$q$$

$$q$$

$$k$$

$$k$$

# Pi-types in predicate games (1/3)

Let me call my generalised games *predicate (p-) games*.

**Notation:** Write  $\Gamma = (|\Gamma|, f_\Gamma)$  for p-games, and  $\gamma : \Gamma$  if  $\gamma : |\Gamma|$  passes the test by  $f_\Gamma$ . Let  $A = (A(\gamma))_{\gamma:\Gamma}$  be a family of p-games  $A(\gamma)$  equipped with a game  $|A|$  such that  $|A(\gamma)| = |A|$  for all  $\gamma : \Gamma$ .

**Idea:** Define the *linear implication*  $\Gamma \multimap \Delta$  between p-games by

- $|\Gamma \multimap \Delta| := |\Gamma| \multimap |\Delta|$ ;
- $\phi : |\Gamma \multimap \Delta|$  passes the test *if it follows the rules of the subgame*  
 $\bar{\gamma}_\Gamma \multimap f_\Delta(\phi \circ \gamma) \subseteq |\Gamma \multimap \Delta|$  for *all*  $\gamma : \Gamma$  that is not yet excluded, i.e., the current position  $\mathbf{s}$  is compatible with  $\gamma$  (or  $\mathbf{s} \upharpoonright |\Gamma| \in \bar{\gamma}_\Gamma$ ), where  $\bar{\gamma}_\Gamma$  is the subgame of  $f_\Gamma(\gamma)$  played by Player and  $\gamma$  (Opponent), i.e.,

$$\bar{\gamma}_\Gamma := \{\epsilon\} \cup \{sm \in f_\Gamma(\gamma) \mid \mathbf{s} \in \bar{\gamma}_\Gamma\} \cup \{tlr \in \gamma \mid tl \in \bar{\gamma}_\Gamma\}.$$

$$\frac{\Sigma(N, \text{List}_N) \quad \multimap \quad N}{q_{\Delta \multimap \Gamma} \quad \pi_1} \quad \frac{\Sigma(N, \text{List}_N) \quad \multimap \quad N}{q_{\Delta \multimap \Gamma} \quad \phi}$$

$$q$$

$$k$$

$$k$$

# Pi-types in predicate games (1/3)

Let me call my generalised games *predicate (p-) games*.

**Notation:** Write  $\Gamma = (|\Gamma|, f_\Gamma)$  for p-games, and  $\gamma : \Gamma$  if  $\gamma : |\Gamma|$  passes the test by  $f_\Gamma$ . Let  $A = (A(\gamma))_{\gamma:\Gamma}$  be a family of p-games  $A(\gamma)$  equipped with a game  $|A|$  such that  $|A(\gamma)| = |A|$  for all  $\gamma : \Gamma$ .

**Idea:** Define the *linear implication*  $\Gamma \multimap \Delta$  between p-games by

- $|\Gamma \multimap \Delta| := |\Gamma| \multimap |\Delta|$ ;
- $\phi : |\Gamma \multimap \Delta|$  passes the test *if it follows the rules of the subgame*  
 $\bar{\gamma}_\Gamma \multimap f_\Delta(\phi \circ \gamma) \subseteq |\Gamma \multimap \Delta|$  for *all  $\gamma : \Gamma$  that is not yet excluded, i.e., the current position  $\mathbf{s}$  is compatible with  $\gamma$  (or  $\mathbf{s} \upharpoonright |\Gamma| \in \bar{\gamma}_\Gamma$ )*, where  $\bar{\gamma}_\Gamma$  is the subgame of  $f_\Gamma(\gamma)$  played by Player and  $\gamma$  (Opponent), i.e.,

$$\bar{\gamma}_\Gamma := \{\epsilon\} \cup \{\mathbf{sm} \in f_\Gamma(\gamma) \mid \mathbf{s} \in \bar{\gamma}_\Gamma\} \cup \{\mathbf{tlr} \in \gamma \mid \mathbf{tl} \in \bar{\gamma}_\Gamma\}.$$

$$\frac{\Sigma(N, \text{List}_N) \quad \multimap \quad N}{q_{\Delta \multimap \Gamma} \quad \pi_1} \quad \frac{\Sigma(N, \text{List}_N) \quad \multimap \quad N}{q_{\Delta \multimap \Gamma} \quad \phi} \quad q$$

$$q$$

$$k$$

# Pi-types in predicate games (1/3)

Let me call my generalised games *predicate (p-) games*.

**Notation:** Write  $\Gamma = (|\Gamma|, f_\Gamma)$  for p-games, and  $\gamma : \Gamma$  if  $\gamma : |\Gamma|$  passes the test by  $f_\Gamma$ . Let  $A = (A(\gamma))_{\gamma:\Gamma}$  be a family of p-games  $A(\gamma)$  equipped with a game  $|A|$  such that  $|A(\gamma)| = |A|$  for all  $\gamma : \Gamma$ .

**Idea:** Define the *linear implication*  $\Gamma \multimap \Delta$  between p-games by

- $|\Gamma \multimap \Delta| := |\Gamma| \multimap |\Delta|$ ;
- $\phi : |\Gamma \multimap \Delta|$  passes the test *if it follows the rules of the subgame*  
 $\bar{\gamma}_\Gamma \multimap f_\Delta(\phi \circ \gamma) \subseteq |\Gamma \multimap \Delta|$  for *all*  $\gamma : \Gamma$  that is not yet excluded, i.e., the current position  $\mathbf{s}$  is compatible with  $\gamma$  (or  $\mathbf{s} \upharpoonright |\Gamma| \in \bar{\gamma}_\Gamma$ ), where  $\bar{\gamma}_\Gamma$  is the subgame of  $f_\Gamma(\gamma)$  played by Player and  $\gamma$  (Opponent), i.e.,

$$\bar{\gamma}_\Gamma := \{\epsilon\} \cup \{sm \in f_\Gamma(\gamma) \mid \mathbf{s} \in \bar{\gamma}_\Gamma\} \cup \{tlr \in \gamma \mid tl \in \bar{\gamma}_\Gamma\}.$$

$\Sigma(N, \text{List}_N)$	$\multimap$	$N$		$\Sigma(N, \text{List}_N)$	$\multimap$	$N$
	$q_{\Delta \multimap \Gamma}$				$q_{\Delta \multimap \Gamma}$	
	$\pi_1$				$\phi$	
		$q$				$q$
$q$				$q$		
$k$						
		$k$				

# Pi-types in predicate games (1/3)

Let me call my generalised games *predicate (p-) games*.

**Notation:** Write  $\Gamma = (|\Gamma|, f_\Gamma)$  for p-games, and  $\gamma : \Gamma$  if  $\gamma : |\Gamma|$  passes the test by  $f_\Gamma$ . Let  $A = (A(\gamma))_{\gamma:\Gamma}$  be a family of p-games  $A(\gamma)$  equipped with a game  $|A|$  such that  $|A(\gamma)| = |A|$  for all  $\gamma : \Gamma$ .

**Idea:** Define the *linear implication*  $\Gamma \multimap \Delta$  between p-games by

- $|\Gamma \multimap \Delta| := |\Gamma| \multimap |\Delta|$ ;
- $\phi : |\Gamma \multimap \Delta|$  passes the test *if it follows the rules of the subgame*  $\bar{\gamma}_\Gamma \multimap f_\Delta(\phi \circ \gamma) \subseteq |\Gamma \multimap \Delta|$  for *all  $\gamma : \Gamma$  that is not yet excluded, i.e., the current position  $\mathbf{s}$  is compatible with  $\gamma$  (or  $\mathbf{s} \upharpoonright |\Gamma| \in \bar{\gamma}_\Gamma$ )*, where  $\bar{\gamma}_\Gamma$  is the subgame of  $f_\Gamma(\gamma)$  played by Player and  $\gamma$  (Opponent), i.e.,

$$\bar{\gamma}_\Gamma := \{\epsilon\} \cup \{sm \in f_\Gamma(\gamma) \mid \mathbf{s} \in \bar{\gamma}_\Gamma\} \cup \{tlr \in \gamma \mid tl \in \bar{\gamma}_\Gamma\}.$$

$\Sigma(N, \text{List}_N)$	$\multimap$	$N$		$\Sigma(N, \text{List}_N)$	$\multimap$	$N$
	$q_{\Delta \multimap \Gamma}$				$q_{\Delta \multimap \Gamma}$	
	$\pi_1$				$\phi$	
		$q$				$q$
$q$				$q$		
$k$				$\phi$ fails the test!		
		$k$				

# Pi-types in predicate games (2/3)



## Pi-types in predicate games (2/3)

We then generalise the linear implication to the *linear-pi*  $\Pi_\ell(\Gamma, A)$  by

## Pi-types in predicate games (2/3)

We then generalise the linear implication to the *linear-pi*  $\Pi_\ell(\Gamma, A)$  by

- $|\Pi_\ell(\Gamma, A)| := |\Gamma| \multimap |A|$ ;

## Pi-types in predicate games (2/3)

We then generalise the linear implication to the *linear-pi*  $\Pi_\ell(\Gamma, A)$  by

- $|\Pi_\ell(\Gamma, A)| := |\Gamma| \multimap |A|$ ;
- $\phi : |\Pi_\ell(\Gamma, A)|$  passes the test if it follows the rules of the subgame  $\bar{\gamma}_\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma) \subseteq |\Pi_\ell(\Gamma, A)|$  for all  $\gamma : \Gamma$  not yet excluded,

## Pi-types in predicate games (2/3)

We then generalise the linear implication to the *linear-pi*  $\Pi_\ell(\Gamma, A)$  by

- $|\Pi_\ell(\Gamma, A)| := |\Gamma| \multimap |A|$ ;
- $\phi : |\Pi_\ell(\Gamma, A)|$  passes the test if it follows the rules of the subgame  $\bar{\gamma}_\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma) \subseteq |\Pi_\ell(\Gamma, A)|$  for all  $\gamma : \Gamma$  not yet excluded,

where the only difference from linear implication is the additional type dependency on the codomain  $|A|$ .

## Pi-types in predicate games (2/3)

We then generalise the linear implication to the *linear-pi*  $\Pi_\ell(\Gamma, A)$  by

- $|\Pi_\ell(\Gamma, A)| := |\Gamma| \multimap |A|$ ;
- $\phi : |\Pi_\ell(\Gamma, A)|$  passes the test if it follows the rules of the subgame  $\bar{\gamma}_\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma) \subseteq |\Pi_\ell(\Gamma, A)|$  for all  $\gamma : \Gamma$  not yet excluded,

where the only difference from linear implication is the additional type dependency on the codomain  $|A|$ .

$$\frac{\Pi_\ell(N, \quad \text{List}_N)}{\quad}$$

## Pi-types in predicate games (2/3)

We then generalise the linear implication to the *linear-pi*  $\Pi_\ell(\Gamma, A)$  by

- $|\Pi_\ell(\Gamma, A)| := |\Gamma| \multimap |A|$ ;
- $\phi : |\Pi_\ell(\Gamma, A)|$  passes the test if it follows the rules of the subgame  $\bar{\gamma}_\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma) \subseteq |\Pi_\ell(\Gamma, A)|$  for all  $\gamma : \Gamma$  not yet excluded,

where the only difference from linear implication is the additional type dependency on the codomain  $|A|$ .

$$\frac{\Pi_\ell(N, \quad \text{List}_N)}{q_{\Pi_\ell}}$$

## Pi-types in predicate games (2/3)

We then generalise the linear implication to the *linear-pi*  $\Pi_\ell(\Gamma, A)$  by

- $|\Pi_\ell(\Gamma, A)| := |\Gamma| \multimap |A|$ ;
- $\phi : |\Pi_\ell(\Gamma, A)|$  passes the test if it follows the rules of the subgame  $\bar{\gamma}_\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma) \subseteq |\Pi_\ell(\Gamma, A)|$  for all  $\gamma : \Gamma$  not yet excluded,

where the only difference from linear implication is the additional type dependency on the codomain  $|A|$ .

$$\frac{\Pi_\ell(N, \text{List}_N)}{k \mapsto 0^k} \quad q_{\Pi_\ell}$$

## Pi-types in predicate games (2/3)

We then generalise the linear implication to the *linear-pi*  $\Pi_\ell(\Gamma, A)$  by

- $|\Pi_\ell(\Gamma, A)| := |\Gamma| \multimap |A|$ ;
- $\phi : |\Pi_\ell(\Gamma, A)|$  passes the test if it follows the rules of the subgame  $\bar{\gamma}_\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma) \subseteq |\Pi_\ell(\Gamma, A)|$  for all  $\gamma : \Gamma$  not yet excluded,

where the only difference from linear implication is the additional type dependency on the codomain  $|A|$ .

$$\frac{\Pi_\ell(N, \text{List}_N)}{k \mapsto 0^k} \quad q$$



## Pi-types in predicate games (2/3)

We then generalise the linear implication to the *linear-pi*  $\Pi_\ell(\Gamma, A)$  by

- $|\Pi_\ell(\Gamma, A)| := |\Gamma| \multimap |A|$ ;
- $\phi : |\Pi_\ell(\Gamma, A)|$  passes the test if it follows the rules of the subgame  $\bar{\gamma}_\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma) \subseteq |\Pi_\ell(\Gamma, A)|$  for all  $\gamma : \Gamma$  not yet excluded,

where the only difference from linear implication is the additional type dependency on the codomain  $|A|$ .

$$\frac{\Pi_\ell(N, \text{List}_N)}{k \mapsto 0^k} \quad q$$

$q$

## Pi-types in predicate games (2/3)

We then generalise the linear implication to the *linear-pi*  $\Pi_\ell(\Gamma, A)$  by

- $|\Pi_\ell(\Gamma, A)| := |\Gamma| \multimap |A|$ ;
- $\phi : |\Pi_\ell(\Gamma, A)|$  passes the test if it follows the rules of the subgame  $\bar{\gamma}_\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma) \subseteq |\Pi_\ell(\Gamma, A)|$  for all  $\gamma : \Gamma$  not yet excluded,

where the only difference from linear implication is the additional type dependency on the codomain  $|A|$ .

$$\frac{\Pi_\ell(N, \text{List}_N)}{k \mapsto 0^k} \quad q$$

$q$

$1$

## Pi-types in predicate games (2/3)

We then generalise the linear implication to the *linear-pi*  $\Pi_\ell(\Gamma, A)$  by

- $|\Pi_\ell(\Gamma, A)| := |\Gamma| \multimap |A|$ ;
- $\phi : |\Pi_\ell(\Gamma, A)|$  passes the test if it follows the rules of the subgame  $\bar{\gamma}_\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma) \subseteq |\Pi_\ell(\Gamma, A)|$  for all  $\gamma : \Gamma$  not yet excluded,

where the only difference from linear implication is the additional type dependency on the codomain  $|A|$ .

$$\frac{\Pi_\ell(N, \quad \text{List}_N)}{q \Pi_\ell} \\ k \mapsto 0^k$$

*q*

*q*  
**1**

0

## Pi-types in predicate games (2/3)

We then generalise the linear implication to the *linear-pi*  $\Pi_\ell(\Gamma, A)$  by

- $|\Pi_\ell(\Gamma, A)| := |\Gamma| \multimap |A|$ ;
- $\phi : |\Pi_\ell(\Gamma, A)|$  passes the test if it follows the rules of the subgame  $\bar{\gamma}_\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma) \subseteq |\Pi_\ell(\Gamma, A)|$  for all  $\gamma : \Gamma$  not yet excluded,

where the only difference from linear implication is the additional type dependency on the codomain  $|A|$ .

$$\frac{\frac{\Pi_\ell(N, \text{List}_N)}{q \mapsto 0^k} \quad \frac{\Pi_\ell(\Sigma(N, N_b), N_b), N_b\{\pi_1\}}{q}}{1} \quad q$$

0

## Pi-types in predicate games (2/3)

We then generalise the linear implication to the *linear-pi*  $\Pi_\ell(\Gamma, A)$  by

- $|\Pi_\ell(\Gamma, A)| := |\Gamma| \multimap |A|$ ;
- $\phi : |\Pi_\ell(\Gamma, A)|$  passes the test if it follows the rules of the subgame  $\bar{\gamma}_\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma) \subseteq |\Pi_\ell(\Gamma, A)|$  for all  $\gamma : \Gamma$  not yet excluded,

where the only difference from linear implication is the additional type dependency on the codomain  $|A|$ .

$$\frac{\frac{\Pi_\ell(N, \text{List}_N)}{q_{\Pi_\ell} \quad k \mapsto 0^k} \quad q}{q} \quad \frac{\Pi_\ell(\Sigma(N, N_b), N_b), N_b\{\pi_1\}}{q_{\Pi_\ell}}$$

## Pi-types in predicate games (2/3)

We then generalise the linear implication to the *linear-pi*  $\Pi_\ell(\Gamma, A)$  by

- $|\Pi_\ell(\Gamma, A)| := |\Gamma| \multimap |A|$ ;
- $\phi : |\Pi_\ell(\Gamma, A)|$  passes the test if it follows the rules of the subgame  $\bar{\gamma}_\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma) \subseteq |\Pi_\ell(\Gamma, A)|$  for all  $\gamma : \Gamma$  not yet excluded,

where the only difference from linear implication is the additional type dependency on the codomain  $|A|$ .

$$\frac{\frac{\Pi_\ell(N, \text{List}_N)}{q \Pi_\ell} \quad k \mapsto 0^k}{q} \quad q \quad \frac{\Pi_\ell(\Sigma(N, N_b), N_b), N_b\{\pi_1\}}{q \Pi_\ell} \quad \underline{1}$$

$0$

## Pi-types in predicate games (2/3)

We then generalise the linear implication to the *linear-pi*  $\Pi_\ell(\Gamma, A)$  by

- $|\Pi_\ell(\Gamma, A)| := |\Gamma| \multimap |A|$ ;
- $\phi : |\Pi_\ell(\Gamma, A)|$  passes the test if it follows the rules of the subgame  $\bar{\gamma}_\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma) \subseteq |\Pi_\ell(\Gamma, A)|$  for all  $\gamma : \Gamma$  not yet excluded,

where the only difference from linear implication is the additional type dependency on the codomain  $|A|$ .

$$\frac{\frac{\Pi_\ell(N, \text{List}_N)}{q \mapsto 0^k} \quad q}{q} \quad \frac{\frac{\Pi_\ell(\Sigma(N, N_b), N_b), N_b\{\pi_1\}}{q \mapsto \underline{1}} \quad q}{q}$$

## Pi-types in predicate games (2/3)

We then generalise the linear implication to the *linear-pi*  $\Pi_\ell(\Gamma, A)$  by

- $|\Pi_\ell(\Gamma, A)| := |\Gamma| \multimap |A|$ ;
- $\phi : |\Pi_\ell(\Gamma, A)|$  passes the test if it follows the rules of the subgame  $\bar{\gamma}_\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma) \subseteq |\Pi_\ell(\Gamma, A)|$  for all  $\gamma : \Gamma$  not yet excluded,

where the only difference from linear implication is the additional type dependency on the codomain  $|A|$ .

$$\frac{\frac{\Pi_\ell(N, \text{List}_N)}{q \quad \mathbf{1}} \quad \frac{q_{\Pi_\ell} \quad k \mapsto 0^k}{q}}{q \quad \mathbf{1}} \quad q \quad 0$$

$$\frac{\frac{\Pi_\ell(\Sigma(N, N_b), N_b)}{q_{\Pi_\ell} \quad \mathbf{1}} \quad , \quad N_b\{\pi_1\}}{q \quad \mathbf{1}}$$



## Pi-types in predicate games (2/3)

We then generalise the linear implication to the *linear-pi*  $\Pi_\ell(\Gamma, A)$  by

- $|\Pi_\ell(\Gamma, A)| := |\Gamma| \multimap |A|$ ;
- $\phi : |\Pi_\ell(\Gamma, A)|$  passes the test if it follows the rules of the subgame  $\bar{\gamma}_\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma) \subseteq |\Pi_\ell(\Gamma, A)|$  for all  $\gamma : \Gamma$  not yet excluded,

where the only difference from linear implication is the additional type dependency on the codomain  $|A|$ .

$$\begin{array}{c}
 \frac{\Pi_\ell(N, \quad \text{List}_N)}{q \mapsto 0^k} \quad \frac{\Pi_\ell(\Sigma(N, \quad N_b), \quad N_b \{\pi_1\})}{\underline{1}} \\
 \begin{array}{ccc}
 q & q & q \\
 \underline{1} & & \underline{1} \\
 & 0 & \underline{1} \text{ fails the test!}
 \end{array}
 \end{array}$$

## Pi-types in predicate games (2/3)

We then generalise the linear implication to the *linear-pi*  $\Pi_\ell(\Gamma, A)$  by

- $|\Pi_\ell(\Gamma, A)| := |\Gamma| \multimap |A|$ ;
- $\phi : |\Pi_\ell(\Gamma, A)|$  passes the test if it follows the rules of the subgame  $\bar{\gamma}_\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma) \subseteq |\Pi_\ell(\Gamma, A)|$  for all  $\gamma : \Gamma$  not yet excluded,

where the only difference from linear implication is the additional type dependency on the codomain  $|A|$ .

$$\begin{array}{c}
 \frac{\Pi_\ell(N, \quad \text{List}_N)}{q \Pi_\ell} \\
 \frac{\quad}{k \mapsto 0^k} \\
 q \\
 \underline{1}
 \end{array}
 \qquad
 \frac{\Pi_\ell(\Sigma(N, \quad N_b) \quad , \quad N_b\{\pi_1\})}{q \Pi_\ell} \\
 \frac{\quad}{\underline{1}} \\
 q \\
 \underline{1}$$

0 1 fails the test!

*The intensionality of game semantics is preserved.*



# Pi-types in predicate games (3/3)

## Pi-types in predicate games (3/3)

Definition (Pi-types in predicate games)

## Pi-types in predicate games (3/3)

Definition (Pi-types in predicate games)

The *linear-pi*  $\Pi_\ell(\Gamma, A)$  is defined by  $|\Pi_\ell(\Gamma, A)| := |\Gamma| \multimap |A|$  and

## Pi-types in predicate games (3/3)

### Definition (Pi-types in predicate games)

The *linear-pi*  $\Pi_\ell(\Gamma, A)$  is defined by  $|\Pi_\ell(\Gamma, A)| := |\Gamma| \multimap |A|$  and

$$\begin{aligned} \Pi_\ell(\Gamma, A)(\phi) := & \{ \epsilon \} \cup \{ \mathbf{sm} \in |\Pi_\ell(\Gamma, A)|^{\text{Odd}} \mid \mathbf{s} \in \Pi_\ell(\Gamma, A)(\phi), \exists \gamma : \Gamma. \mathbf{sm} \in A(\gamma)(\phi \circ \gamma)^{\overline{\gamma}\Gamma} \} \\ & \cup \{ \mathbf{tlr} \in |\Pi_\ell(\Gamma, A)|^{\text{Even}} \mid \mathbf{tl} \in \Pi_\ell(\Gamma, A)(\phi), \forall \gamma : \Gamma. \mathbf{tl} \in A(\gamma)(\phi \circ \gamma)^{\overline{\gamma}\Gamma} \Rightarrow \mathbf{tlr} \in A(\gamma)(\phi \circ \gamma)^{\overline{\gamma}\Gamma} \} \end{aligned}$$

for all  $\phi : |\Pi_\ell(\Gamma, A)|$ .

## Pi-types in predicate games (3/3)

### Definition (Pi-types in predicate games)

The *linear-pi*  $\Pi_\ell(\Gamma, A)$  is defined by  $|\Pi_\ell(\Gamma, A)| := |\Gamma| \multimap |A|$  and

$$\begin{aligned} \Pi_\ell(\Gamma, A)(\phi) := & \{ \epsilon \} \cup \{ sm \in |\Pi_\ell(\Gamma, A)|^{\text{Odd}} \mid s \in \Pi_\ell(\Gamma, A)(\phi), \exists \gamma : \Gamma. sm \in A(\gamma)(\phi \circ \gamma)^{\overline{\gamma}\Gamma} \} \\ & \cup \{ tlr \in |\Pi_\ell(\Gamma, A)|^{\text{Even}} \mid tl \in \Pi_\ell(\Gamma, A)(\phi), \forall \gamma : \Gamma. tl \in A(\gamma)(\phi \circ \gamma)^{\overline{\gamma}\Gamma} \Rightarrow tlr \in A(\gamma)(\phi \circ \gamma)^{\overline{\gamma}\Gamma} \} \end{aligned}$$

for all  $\phi : |\Pi_\ell(\Gamma, A)|$ . Based on [Abramsky and Jagadeesan, 2005].



# Pi-types in predicate games (3/3)

## Definition (Pi-types in predicate games)

The *linear-pi*  $\Pi_\ell(\Gamma, A)$  is defined by  $|\Pi_\ell(\Gamma, A)| := |\Gamma| \multimap |A|$  and

$$\begin{aligned} \Pi_\ell(\Gamma, A)(\phi) := & \{ \epsilon \} \cup \{ sm \in |\Pi_\ell(\Gamma, A)|^{\text{Odd}} \mid s \in \Pi_\ell(\Gamma, A)(\phi), \exists \gamma : \Gamma. sm \in A(\gamma)(\phi \circ \gamma)^{\overline{\gamma}\Gamma} \} \\ & \cup \{ tlr \in |\Pi_\ell(\Gamma, A)|^{\text{Even}} \mid tl \in \Pi_\ell(\Gamma, A)(\phi), \forall \gamma : \Gamma. tl \in A(\gamma)(\phi \circ \gamma)^{\overline{\gamma}\Gamma} \Rightarrow tlr \in A(\gamma)(\phi \circ \gamma)^{\overline{\gamma}\Gamma} \} \end{aligned}$$

for all  $\phi : |\Pi_\ell(\Gamma, A)|$ . Based on [Abramsky and Jagadeesan, 2005].

- ① The base case;

# Pi-types in predicate games (3/3)

## Definition (Pi-types in predicate games)

The *linear-pi*  $\Pi_\ell(\Gamma, A)$  is defined by  $|\Pi_\ell(\Gamma, A)| := |\Gamma| \multimap |A|$  and

$$\begin{aligned} \Pi_\ell(\Gamma, A)(\phi) := & \{ \epsilon \} \cup \{ sm \in |\Pi_\ell(\Gamma, A)|^{\text{Odd}} \mid s \in \Pi_\ell(\Gamma, A)(\phi), \exists \gamma : \Gamma. sm \in A(\gamma)(\phi \circ \gamma)^{\bar{\gamma}\Gamma} \} \\ & \cup \{ tlr \in |\Pi_\ell(\Gamma, A)|^{\text{Even}} \mid tl \in \Pi_\ell(\Gamma, A)(\phi), \forall \gamma : \Gamma. tl \in A(\gamma)(\phi \circ \gamma)^{\bar{\gamma}\Gamma} \Rightarrow tlr \in A(\gamma)(\phi \circ \gamma)^{\bar{\gamma}\Gamma} \} \end{aligned}$$

for all  $\phi : |\Pi_\ell(\Gamma, A)|$ . Based on [Abramsky and Jagadeesan, 2005].

- ① The base case;
- ② Given  $s \in \Pi_\ell(\Gamma, A)(\phi)^{\text{Even}}$ , O *can* perform the next move  $m$  as in  $\bar{\gamma}\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma) \subseteq |\Pi_\ell(\Gamma, A)|$ , i.e.,  $sm \in \bar{\gamma}\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma)$ , for *any*  $\gamma : \Gamma$  not yet excluded, i.e.,  $s \in \bar{\gamma}\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma)$ ;

# Pi-types in predicate games (3/3)

## Definition (Pi-types in predicate games)

The *linear-pi*  $\Pi_\ell(\Gamma, A)$  is defined by  $|\Pi_\ell(\Gamma, A)| := |\Gamma| \multimap |A|$  and

$$\begin{aligned} \Pi_\ell(\Gamma, A)(\phi) := & \{ \epsilon \} \cup \{ sm \in |\Pi_\ell(\Gamma, A)|^{\text{Odd}} \mid s \in \Pi_\ell(\Gamma, A)(\phi), \exists \gamma : \Gamma. sm \in A(\gamma)(\phi \circ \gamma)^{\bar{\gamma}\Gamma} \} \\ & \cup \{ tlr \in |\Pi_\ell(\Gamma, A)|^{\text{Even}} \mid tl \in \Pi_\ell(\Gamma, A)(\phi), \forall \gamma : \Gamma. tl \in A(\gamma)(\phi \circ \gamma)^{\bar{\gamma}\Gamma} \Rightarrow tlr \in A(\gamma)(\phi \circ \gamma)^{\bar{\gamma}\Gamma} \} \end{aligned}$$

for all  $\phi : |\Pi_\ell(\Gamma, A)|$ . Based on [Abramsky and Jagadeesan, 2005].

- ① The base case;
- ② Given  $s \in \Pi_\ell(\Gamma, A)(\phi)^{\text{Even}}$ , O *can* perform the next move  $m$  as in  $\bar{\gamma}\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma) \subseteq |\Pi_\ell(\Gamma, A)|$ , i.e.,  $sm \in \bar{\gamma}\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma)$ , for *any*  $\gamma : \Gamma$  not yet excluded, i.e.,  $s \in \bar{\gamma}\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma)$ ;
- ③ Given  $tl \in \Pi_\ell(\Gamma, A)(\phi)^{\text{Odd}}$ , the next move  $r$  by  $\phi$  *must* be as in  $\bar{\gamma}\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma) \subseteq |\Pi_\ell(\Gamma, A)|$ , i.e.,  $tlr \in \bar{\gamma}\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma)$ , for *any*  $\gamma : \Gamma$  not yet excluded, i.e.,  $tl \in \bar{\gamma}\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma)$ .

# Sigma-types and finite limits in predicate games

# Sigma-types and finite limits in predicate games

For completeness, we define the *sigma*  $\Sigma(\Gamma, B)$  by

# Sigma-types and finite limits in predicate games

For completeness, we define the *sigma*  $\Sigma(\Gamma, B)$  by

- $|\Sigma(\Gamma, B)| := |\Gamma| \& |B|$ ;

# Sigma-types and finite limits in predicate games

For completeness, we define the *sigma*  $\Sigma(\Gamma, B)$  by

- $|\Sigma(\Gamma, B)| := |\Gamma| \ \& \ |B|$ ;
- $f_{\Sigma(\Gamma, B)}(\langle \gamma, \beta \rangle) := f_{\Gamma}(\gamma) \ \& \ f_{B(\gamma^{\dagger})}(\beta)$  for all  $\langle \gamma, \beta \rangle : |\Sigma(\Gamma, B)|$ .

## Sigma-types and finite limits in predicate games

For completeness, we define the *sigma*  $\Sigma(\Gamma, B)$  by

- $|\Sigma(\Gamma, B)| := |\Gamma| \& |B|$ ;
- $f_{\Sigma(\Gamma, B)}(\langle \gamma, \beta \rangle) := f_{\Gamma}(\gamma) \& f_{B(\gamma^\dagger)}(\beta)$  for all  $\langle \gamma, \beta \rangle : |\Sigma(\Gamma, B)|$ .

Theorem (finite limits in predicate games)

*The categories of p-games and strict strategies are finitely complete.*



## Sigma-types and finite limits in predicate games

For completeness, we define the *sigma*  $\Sigma(\Gamma, B)$  by

- $|\Sigma(\Gamma, B)| := |\Gamma| \& |B|$ ;
- $f_{\Sigma(\Gamma, B)}(\langle \gamma, \beta \rangle) := f_{\Gamma}(\gamma) \& f_{B(\gamma^{\dagger})}(\beta)$  for all  $\langle \gamma, \beta \rangle : |\Sigma(\Gamma, B)|$ .

**Theorem (finite limits in predicate games)**

*The categories of p-games and **strict** strategies are finitely complete.*

**Proof (sketch).**

The equaliser of given morphisms  $\phi_1, \phi_2 : \Gamma \rightrightarrows \Delta$  is the p-game  $\Theta$  defined by  $|\Theta| := |\Gamma|$  and  $f_{\Theta}(\theta) := \begin{cases} f_{\Gamma}(\theta) & \text{if } \phi_1 \bullet \theta = \phi_2 \bullet \theta \\ T & \text{otherwise} \end{cases}$  for all  $\theta : |\Theta|$ , together with the identity  $\text{id}_{|\Theta|} : \Theta \hookrightarrow \Gamma$ . □

# Ongoing and future research

# Ongoing and future research

Some of the ongoing and future research:

# Ongoing and future research

Some of the ongoing and future research:

- To combine dependent types and linear logic;

# Ongoing and future research

Some of the ongoing and future research:

- To combine dependent types and linear logic;
- To combine dependent types and effects (esp. classical logic);

# Ongoing and future research

Some of the ongoing and future research:

- To combine dependent types and linear logic;
- To combine dependent types and effects (esp. classical logic);
- To interpret W-types;

# Ongoing and future research

Some of the ongoing and future research:

- To combine dependent types and linear logic;
- To combine dependent types and effects (esp. classical logic);
- To interpret W-types;
- To study predicative topos;

## Ongoing and future research

Some of the ongoing and future research:

- To combine dependent types and linear logic;
- To combine dependent types and effects (esp. classical logic);
- To interpret W-types;
- To study predicative topos;
- To interpret higher inductive types.



## Ongoing and future research

Some of the ongoing and future research:

- To combine dependent types and linear logic;
- To combine dependent types and effects (esp. classical logic);
- To interpret W-types;
- To study predicative topos;
- To interpret higher inductive types.

Game semantics plays the roles of:

## Ongoing and future research

Some of the ongoing and future research:

- To combine dependent types and linear logic;
- To combine dependent types and effects (esp. classical logic);
- To interpret W-types;
- To study predicative topos;
- To interpret higher inductive types.

Game semantics plays the roles of:

- An analytic, semantic example of categories and types;

## Ongoing and future research

Some of the ongoing and future research:

- To combine dependent types and linear logic;
- To combine dependent types and effects (esp. classical logic);
- To interpret W-types;
- To study predicative topos;
- To interpret higher inductive types.

Game semantics plays the roles of:

- An analytic, semantic example of categories and types;
- A mathematical tool for the study of type theories;

## Ongoing and future research

Some of the ongoing and future research:

- To combine dependent types and linear logic;
- To combine dependent types and effects (esp. classical logic);
- To interpret W-types;
- To study predicative topos;
- To interpret higher inductive types.

Game semantics plays the roles of:

- An analytic, semantic example of categories and types;
- A mathematical tool for the study of type theories;
- A foundation of generalised computability/constructivity.

# Not an elementary topos

## Not an elementary topos

The finitely complete category of predicate games and strategies is *not* an elementary topos.

## Not an elementary topos

The finitely complete category of predicate games and strategies is *not* an elementary topos. This indicates its *predicativity*.

## Not an elementary topos

The finitely complete category of predicate games and strategies is *not* an elementary topos. This indicates its *predicativity*.

To see this point, consider the subobject  $(N \multimap N)^+$  of the linear implication  $N \multimap N$  defined by:



## Not an elementary topos

The finitely complete category of predicate games and strategies is *not* an elementary topos. This indicates its *predicativity*.

To see this point, consider the subobject  $(N \multimap N)^+$  of the linear implication  $N \multimap N$  defined by:

- $|(N \multimap N)^+| := N \multimap N;$

## Not an elementary topos

The finitely complete category of predicate games and strategies is *not* an elementary topos. This indicates its *predicativity*.

To see this point, consider the subobject  $(N \multimap N)^+$  of the linear implication  $N \multimap N$  defined by:

- $|(N \multimap N)^+| := N \multimap N$ ;
- $f_{(N \multimap N)^+}(\phi) := \begin{cases} N \multimap N & \text{if } \phi \circ \underline{n} \neq \underline{0} \text{ for all } n \in \mathbb{N} \\ \mathbf{0} & \text{otherwise} \end{cases}$  for all  $\phi : |(N \multimap N)^+|$ .

## Not an elementary topos

The finitely complete category of predicate games and strategies is *not* an elementary topos. This indicates its *predicativity*.

To see this point, consider the subobject  $(N \multimap N)^+$  of the linear implication  $N \multimap N$  defined by:

- $|(N \multimap N)^+| := N \multimap N$ ;
- $f_{(N \multimap N)^+}(\phi) := \begin{cases} N \multimap N & \text{if } \phi \circ \underline{n} \neq \underline{0} \text{ for all } n \in \mathbb{N} \\ \mathbf{0} & \text{otherwise} \end{cases}$  for all  $\phi : |(N \multimap N)^+|$ .

I.e.,  $\phi : N \multimap N$  satisfies  $\phi : (N \multimap N)^+$  if and only if  $\phi$  never outputs  $\underline{0}$ .

## Not an elementary topos

The finitely complete category of predicate games and strategies is *not* an elementary topos. This indicates its *predicativity*.

To see this point, consider the subobject  $(N \multimap N)^+$  of the linear implication  $N \multimap N$  defined by:

- $|(N \multimap N)^+| := N \multimap N$ ;
- $f_{(N \multimap N)^+}(\phi) := \begin{cases} N \multimap N & \text{if } \phi \circ \underline{n} \neq \underline{0} \text{ for all } n \in \mathbb{N} \\ \mathbf{0} & \text{otherwise} \end{cases}$  for all  $\phi : |(N \multimap N)^+|$ .

I.e.,  $\phi : N \multimap N$  satisfies  $\phi : (N \multimap N)^+$  if and only if  $\phi$  never outputs  $\underline{0}$ . Then observe that there is no *winning* morphism  $(N \multimap N) \rightarrow \Omega$  that characterises the subobject  $(N \multimap N)^+$  since *there is no finite interaction with a given  $\phi : N \multimap N$  that decides if  $\phi$  never outputs  $\underline{0}$ .*

 Abramsky, S. and Jagadeesan, R. (2005).

A game semantics for generic polymorphism.

*Annals of Pure and Applied Logic*, 133(1):3–37.

 Abramsky, S., Jagadeesan, R., and Vákár, M. (2015).

Games for dependent types.

In *Automata, Languages, and Programming*, pages 31–43. Springer, Berlin, Heidelberg.