Dependent Types and Finite Limits in Games

Norihiro Yamada

yamad041@umn.edu
University of Minnesota

Categories and Types Seminar
Institute for Logic, Language and Computation
University of Amsterdam
May 11, 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, $f : A \to \bigcup_{x \in A} B(x)$ 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, $f : A \to \bigcup_{x \in A} B(x)$ 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, $f : A \to \bigcup_{x \in A} B(x)$ 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, $f : A \to \bigcup_{x \in A} B(x)$ 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, $f : A \to \bigcup_{x \in A} B(x)$ 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 $\big(\phi : \Pi(A, B), \psi : \Pi(A, C\{\phi\})\big)$;

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, $f : A \to \bigcup_{x \in A} B(x)$ 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 $\big(\phi : \Pi(A, B), \psi : \Pi(A, C\{\phi\})\big)$;
- 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)

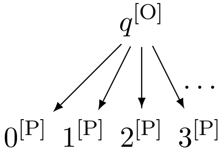
Definition (games)

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)

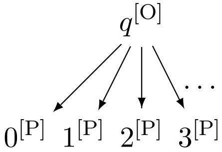
A ***game*** is a rooted DAG whose vertices (or ***moves***) have parity O/P, and paths from a root (or ***positions***) have parity OPOP...

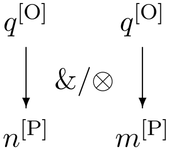


Games and strategies (1/2)

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...

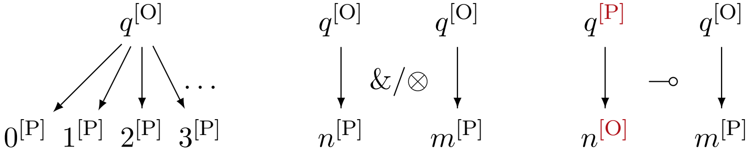




Games and strategies (1/2)

Definition (games)

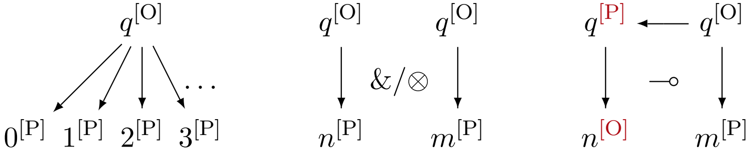
A ***game*** is a rooted DAG whose vertices (or ***moves***) have parity O/P, and paths from a root (or ***positions***) have parity OPOP...



Games and strategies (1/2)

Definition (games)

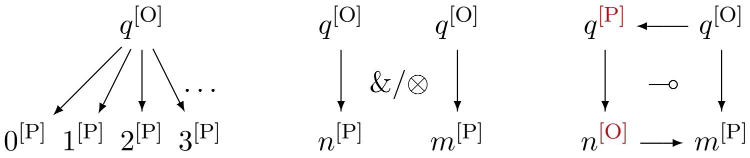
A ***game*** is a rooted DAG whose vertices (or ***moves***) have parity O/P, and paths from a root (or ***positions***) have parity OPOP...



Games and strategies (1/2)

Definition (games)

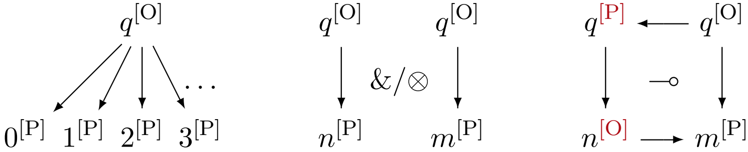
A ***game*** is a rooted DAG whose vertices (or ***moves***) have parity O/P, and paths from a root (or ***positions***) have parity OPOP...



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...

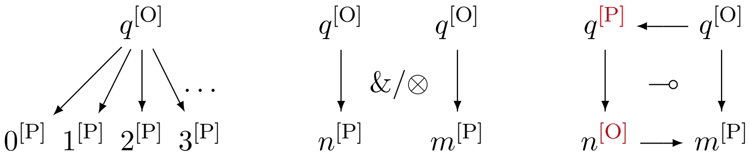


Definition (product, tensor and linear implication on games)

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...



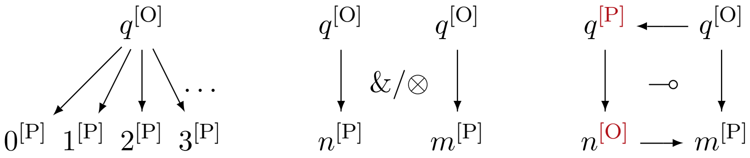
Definition (product, tensor and linear implication on games)

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...



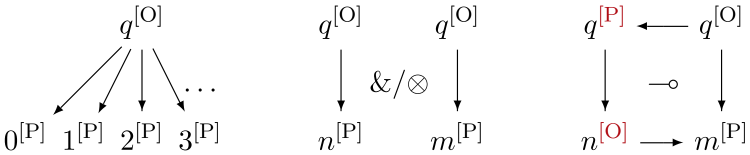
Definition (product, tensor and linear implication on games)

Product: $G \,\&\, H := G \uplus H$;
Tensor: $G \otimes H := \{\, \boldsymbol{s} \in (M_G \uplus M_H)^{\mathrm{alt}} \mid \boldsymbol{s} \upharpoonright G \in G, \boldsymbol{s} \upharpoonright H \in H \,\}$;

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: $G \mathbin{\&} H := G \uplus H$;
Tensor: $G \otimes H := \{ \boldsymbol{s} \in (M_G \uplus M_H)^{\mathrm{alt}} \mid \boldsymbol{s} \upharpoonright G \in G, \boldsymbol{s} \upharpoonright H \in H \}$;
Linear implication:
$G \multimap H := \{ \boldsymbol{s} \in (M_G^{\mathrm{flip}} \uplus M_H)^{\mathrm{alt}} \mid \boldsymbol{s} \upharpoonright G \in G, \boldsymbol{s} \upharpoonright H \in H \}$.

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

Games and strategies (2/2)

Example (games as sets of positions)

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

Games and strategies (2/2)

Example (games as sets of positions)

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

Games and strategies (2/2)

Example (games as sets of positions)

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

Games and strategies (2/2)

Example (games as sets of positions)

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

Definition (strategies)

Games and strategies (2/2)

Example (games as sets of positions)

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

Definition (strategies)

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

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

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

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

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

It is ***total*** if it always responds: $\forall \boldsymbol{s}m \in G^{\mathrm{Odd}}. \boldsymbol{s} \in \sigma \Rightarrow \exists \boldsymbol{s}mn \in \sigma$.

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

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

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

It is ***total*** if it always responds: $\forall \boldsymbol{s}m \in G^{\mathrm{Odd}}.\, \boldsymbol{s} \in \sigma \Rightarrow \exists \boldsymbol{s}mn \in \sigma$.

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

Games and strategies (2/2)

Example (games as sets of positions)

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

Definition (strategies)

A ***strategy*** σ 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: $\forall \boldsymbol{s}m \in G^{\mathrm{Odd}}. \boldsymbol{s} \in \sigma \Rightarrow \exists \boldsymbol{s}mn \in \sigma$.

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

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

Games and strategies (2/2)

Example (games as sets of positions)

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

Definition (strategies)

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

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

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

It is ***total*** if it always responds: $\forall \boldsymbol{s}m \in G^{\mathrm{Odd}}. \boldsymbol{s} \in \sigma \Rightarrow \exists \boldsymbol{s}mn \in \sigma$.

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

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

Games and strategies (2/2)

Example (games as sets of positions)

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

Definition (strategies)

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

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

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

It is ***total*** if it always responds: $\forall \boldsymbol{s}m \in G^{\mathrm{Odd}}.\, \boldsymbol{s} \in \sigma \Rightarrow \exists \boldsymbol{s}mn \in \sigma$.

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

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

Games and strategies (2/2)

Example (games as sets of positions)

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

Definition (strategies)

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

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

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

It is ***total*** if it always responds: $\forall \boldsymbol{s}m \in G^{\mathrm{Odd}}.\, \boldsymbol{s} \in \sigma \Rightarrow \exists \boldsymbol{s}mn \in \sigma$.

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

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

Games and strategies (2/2)

Example (games as sets of positions)

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

Definition (strategies)

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

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

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

It is ***total*** if it always responds: $\forall \boldsymbol{s}m \in G^{\mathrm{Odd}}.\, \boldsymbol{s} \in \sigma \Rightarrow \exists \boldsymbol{s}mn \in \sigma$.

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

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

Games and strategies (2/2)

Example (games as sets of positions)

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

Definition (strategies)

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

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

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

It is ***total*** if it always responds: $\forall \boldsymbol{s}m \in G^{\mathrm{Odd}}.\, \boldsymbol{s} \in \sigma \Rightarrow \exists \boldsymbol{s}mn \in \sigma$.

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

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

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 $\mathsf{x} : \mathsf{C} \vdash \mathsf{D}(\mathsf{x})\ \mathsf{Type}$ 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 $\mathsf{x} : \mathsf{C} \vdash \mathsf{D}(\mathsf{x})\ \mathsf{Type}$ 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 $\Sigma_{\mathsf{x}:\mathsf{C}}\mathsf{D}(\mathsf{x})$ by a subgame $\Sigma(C, D) \subseteq C \mathbin{\&} \bigcup_{x:C} D(x)$ 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 $\mathsf{x} : \mathsf{C} \vdash \mathsf{D}(\mathsf{x})\ \mathsf{Type}$ 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 $\Sigma_{\mathsf{x}:\mathsf{C}}\mathsf{D}(\mathsf{x})$ 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 $\mathsf{x} : \mathsf{C} \vdash \mathsf{D}(\mathsf{x})\ \mathsf{Type}$ 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 $\Sigma_{\mathsf{x}:\mathsf{C}}\mathsf{D}(\mathsf{x})$ 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 $\mathsf{x : C \vdash D(x)\ Type}$ 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 $\mathsf{\Sigma_{x:C} D(x)}$ 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 $\mathsf{x} : \mathsf{N} \vdash \mathsf{N_b}(\mathsf{x})\ \mathsf{Type}$ such that the canonical terms of the simple type $\mathsf{N_b}(\underline{\mathsf{k}})$ for each $k \in \mathbb{N}$ are the numerals $\underline{\mathsf{n}}$ that satisfy $n \leqslant k$.

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

First example: Consider the dependent type $\mathsf{x} : \mathsf{N} \vdash \mathsf{N_b}(\mathsf{x})$ Type such that the canonical terms of the simple type $\mathsf{N_b}(\underline{\mathsf{k}})$ for each $k \in \mathbb{N}$ are the numerals $\underline{\mathsf{n}}$ that satisfy $n \leqslant k$.
We model $\mathsf{N_b}$ by the family N_b of games $N_b(\underline{k}) := \mathrm{Pref}(\{ qn \mid n \leqslant k \})$.

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

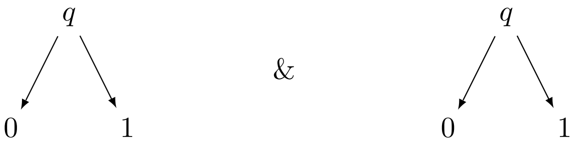
First example: Consider the dependent type $\mathsf{x} : \mathsf{N} \vdash \mathsf{N_b}(\mathsf{x})$ Type such that the canonical terms of the simple type $\mathsf{N_b}(\underline{\mathsf{k}})$ for each $k \in \mathbb{N}$ are the numerals $\underline{\mathsf{n}}$ that satisfy $n \leqslant k$.
We model $\mathsf{N_b}$ by the family N_b of games $N_b(\underline{k}) := \mathrm{Pref}(\{ qn \mid n \leqslant k \})$.
Then, a subgame $\Sigma(N, N_b) \subseteq N \,\&\, N$ that models the Sigma-type $\Sigma_{\mathsf{x:N}}\mathsf{N_b}(\mathsf{x})$ must satisfy $\langle \underline{k}, \underline{n} \rangle : \Sigma(N, N_b) \Leftrightarrow k \geqslant n$ for all $k, n \in \mathbb{N}$.

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

First example: Consider the dependent type $\mathsf{x} : \mathsf{N} \vdash \mathsf{N_b}(\mathsf{x})\ \mathsf{Type}$ such that the canonical terms of the simple type $\mathsf{N_b}(\underline{\mathsf{k}})$ for each $k \in \mathbb{N}$ are the numerals $\underline{\mathsf{n}}$ that satisfy $n \leqslant k$.
We model $\mathsf{N_b}$ by the family N_b of games $N_b(\underline{k}) := \mathrm{Pref}(\{ qn \mid n \leqslant k \})$.
Then, a subgame $\Sigma(N, N_b) \subseteq N \mathbin{\&} N$ that models the Sigma-type $\Sigma_{\mathsf{x}:\mathsf{N}} \mathsf{N_b}(\mathsf{x})$ must satisfy $\langle \underline{k}, \underline{n} \rangle : \Sigma(N, N_b) \Leftrightarrow k \geqslant n$ 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 $\langle \underline{0}, \underline{1} \rangle : \Sigma(N, N_b)$ by the definition of strategies on a game.

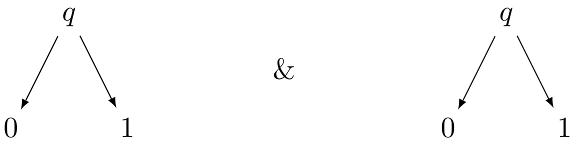
Existing games cannot interpret Sigma-types (2/4)

First example: Consider the dependent type $\mathsf{x} : \mathsf{N} \vdash \mathsf{N_b}(\mathsf{x})$ Type such that the canonical terms of the simple type $\mathsf{N_b}(\underline{\mathsf{k}})$ for each $k \in \mathbb{N}$ are the numerals $\underline{\mathsf{n}}$ that satisfy $n \leqslant k$.
We model $\mathsf{N_b}$ by the family N_b of games $N_b(\underline{k}) := \mathrm{Pref}(\{ qn \mid n \leqslant k \})$.
Then, a subgame $\Sigma(N, N_b) \subseteq N \,\&\, N$ that models the Sigma-type $\Sigma_{\mathsf{x}:\mathsf{N}}\mathsf{N_b}(\mathsf{x})$ must satisfy $\langle \underline{k}, \underline{n} \rangle : \Sigma(N, N_b) \Leftrightarrow k \geqslant n$ 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 $\langle \underline{0}, \underline{1} \rangle : \Sigma(N, N_b)$ by the definition of strategies on a game.



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

First example: Consider the dependent type $\mathsf{x} : \mathsf{N} \vdash \mathsf{N_b}(\mathsf{x})$ Type such that the canonical terms of the simple type $\mathsf{N_b}(\underline{\mathsf{k}})$ for each $k \in \mathbb{N}$ are the numerals $\underline{\mathsf{n}}$ that satisfy $n \leqslant k$.
We model $\mathsf{N_b}$ by the family N_b of games $N_b(\underline{k}) := \mathrm{Pref}(\{ qn \mid n \leqslant k \})$.
Then, a subgame $\Sigma(N, N_b) \subseteq N \& N$ that models the Sigma-type $\Sigma_{\mathsf{x:N}}\mathsf{N_b}(\mathsf{x})$ must satisfy $\langle \underline{k}, \underline{n} \rangle : \Sigma(N, N_b) \Leftrightarrow k \geqslant n$ 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 $\langle \underline{0}, \underline{1} \rangle : \Sigma(N, N_b)$ 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 $\mathsf{x} : \mathsf{N} \vdash \mathsf{List}_{\mathsf{N}}(\mathsf{x})\ \mathsf{Type}$ such that the canonical terms of the simple type $\mathsf{List}_{\mathsf{N}}(\underline{\mathsf{k}})$ for each $k \in \mathbb{N}$ are k-lists $(\underline{\mathsf{n_1}}, \underline{\mathsf{n_2}}, \dots, \underline{\mathsf{n_k}})$ of numerals.

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

Second example: Consider the dependent type $\mathsf{x} : \mathsf{N} \vdash \mathsf{List_N(x)}$ Type such that the canonical terms of the simple type $\mathsf{List_N}(\underline{\mathsf{k}})$ for each $k \in \mathbb{N}$ are k-lists $(\underline{\mathsf{n_1}}, \underline{\mathsf{n_2}}, \dots, \underline{\mathsf{n_k}})$ of numerals.
We interpret $\mathsf{List_N}$ by the family List_N of games $\mathrm{List}_N(\underline{k})$ $(k \in \mathbb{N})$, which are the k-times iteration of tensor $\otimes$ on N (n.b., $\mathrm{List}_N(\underline{0}) = T$).

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

Second example: Consider the dependent type $\mathsf{x} : \mathsf{N} \vdash \mathsf{List_N}(\mathsf{x})$ Type such that the canonical terms of the simple type $\mathsf{List_N}(\underline{\mathsf{k}})$ for each $k \in \mathbb{N}$ are k-lists $(\underline{\mathsf{n_1}}, \underline{\mathsf{n_2}}, \dots, \underline{\mathsf{n_k}})$ of numerals.
We interpret $\mathsf{List_N}$ by the family List_N of games $\mathrm{List}_N(\underline{k})$ $(k \in \mathbb{N})$, which are the k-times iteration of tensor $\otimes$ on N (n.b., $\mathrm{List}_N(\underline{0}) = T$).
If a subgame $\Sigma(N, \mathrm{List}_N) \subseteq N \,\&\, \bigcup_{k \in \mathbb{N}} \mathrm{List}_N(\underline{k})$ models the Sigma-type $\Sigma_{\mathsf{x}:\mathsf{N}} \mathsf{List_N}(\mathsf{x})$, then $\langle \underline{0}, \top \rangle$ and $\langle \underline{1}, \underline{0} \rangle$ are *total* on $\Sigma(N, \mathrm{List}_N)$.

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

Second example: Consider the dependent type $\mathsf{x} : \mathsf{N} \vdash \mathsf{List_N}(\mathsf{x})$ Type such that the canonical terms of the simple type $\mathsf{List_N}(\underline{\mathsf{k}})$ for each $k \in \mathbb{N}$ are k-lists $(\underline{\mathsf{n_1}}, \underline{\mathsf{n_2}}, \dots, \underline{\mathsf{n_k}})$ of numerals.
We interpret $\mathsf{List_N}$ by the family List_N of games $\mathrm{List}_N(\underline{k})$ ($k \in \mathbb{N}$), which are the k-times iteration of tensor $\otimes$ on N (n.b., $\mathrm{List}_N(\underline{0}) = T$).
If a subgame $\Sigma(N, \mathrm{List}_N) \subseteq N \,\&\, \bigcup_{k \in \mathbb{N}} \mathrm{List}_N(\underline{k})$ models the Sigma-type $\Sigma_{\mathsf{x:N}} \mathsf{List_N}(\mathsf{x})$, then $\langle \underline{0}, \top \rangle$ and $\langle \underline{1}, \underline{0} \rangle$ are *total* on $\Sigma(N, \mathrm{List}_N)$.
However, the totality of $\langle \underline{0}, \top \rangle$ contradicts $\langle \underline{1}, \underline{0} \rangle : \Sigma(N, \mathrm{List}_N)$.

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

Second example: Consider the dependent type $\mathsf{x} : \mathsf{N} \vdash \mathsf{List_N}(\mathsf{x})\ \mathsf{Type}$ such that the canonical terms of the simple type $\mathsf{List_N}(\underline{\mathsf{k}})$ for each $k \in \mathbb{N}$ are k-lists $(\underline{\mathsf{n_1}}, \underline{\mathsf{n_2}}, \dots, \underline{\mathsf{n_k}})$ of numerals.
We interpret $\mathsf{List_N}$ by the family List_N of games $\mathrm{List}_N(\underline{k})$ $(k \in \mathbb{N})$, which are the k-times iteration of tensor $\otimes$ on N (n.b., $\mathrm{List}_N(\underline{0}) = T$).
If a subgame $\Sigma(N, \mathrm{List}_N) \subseteq N \mathbin{\&} \bigcup_{k \in \mathbb{N}} \mathrm{List}_N(\underline{k})$ models the Sigma-type $\Sigma_{\mathsf{x}:\mathsf{N}} \mathsf{List_N}(\mathsf{x})$, then $\langle \underline{0}, \top \rangle$ and $\langle \underline{1}, \underline{0} \rangle$ are *total* on $\Sigma(N, \mathrm{List}_N)$.
However, the totality of $\langle \underline{0}, \top \rangle$ contradicts $\langle \underline{1}, \underline{0} \rangle : \Sigma(N, \mathrm{List}_N)$.

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

Second example: Consider the dependent type $\mathsf{x} : \mathsf{N} \vdash \mathsf{List_N}(\mathsf{x})\ \mathsf{Type}$ such that the canonical terms of the simple type $\mathsf{List_N}(\underline{\mathsf{k}})$ for each $k \in \mathbb{N}$ are k-lists $(\underline{\mathsf{n_1}}, \underline{\mathsf{n_2}}, \dots, \underline{\mathsf{n_k}})$ of numerals.
We interpret $\mathsf{List_N}$ by the family List_N of games $\mathrm{List}_N(\underline{k})$ $(k \in \mathbb{N})$, which are the k-times iteration of tensor $\otimes$ on N (n.b., $\mathrm{List}_N(\underline{0}) = T$).
If a subgame $\Sigma(N, \mathrm{List}_N) \subseteq N \,\&\, \bigcup_{k \in \mathbb{N}} \mathrm{List}_N(\underline{k})$ models the Sigma-type $\Sigma_{\mathsf{x:N}}\mathsf{List_N}(\mathsf{x})$, then $\langle \underline{0}, \top \rangle$ and $\langle \underline{1}, \underline{0} \rangle$ are *total* on $\Sigma(N, \mathrm{List}_N)$.
However, the totality of $\langle \underline{0}, \top \rangle$ contradicts $\langle \underline{1}, \underline{0} \rangle : \Sigma(N, \mathrm{List}_N)$.

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 $\Sigma_{\mathsf{x}:\mathsf{N}}\mathsf{N_b}(\mathsf{x})$ 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*

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

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* $\Gamma = A_1.A_2 \ldots A_n$ and *context morphisms*

$$(\Delta \stackrel{\phi_1}{\to} A_1, \Delta \stackrel{\phi_2}{\to} A_2\{\phi_1\}, \ldots, \Delta \stackrel{\phi_n}{\to} A_n\{(\phi_1, \phi_2, \ldots, \phi_{n-1})\}) : \Delta \to \Gamma.$$

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*

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

This list construction is nothing about game semantics.

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

$$(\underline{k}, \underline{n_1} \otimes \underline{n_2} \otimes \cdots) \quad \text{(whatever } k \in \mathbb{N} \text{ is)}$$

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*

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

This list construction is nothing about game semantics.

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

$$(\underline{k}, \underline{n_1} \otimes \underline{n_2} \otimes \cdots) \quad (\text{whatever } k \in \mathbb{N} \text{ is})$$

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 $\Sigma_{\mathsf{x:N}}\mathsf{N_b}(\mathsf{x})$ 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* $\Gamma = A_1.A_2 \ldots A_n$ and *context morphisms*

$$(\Delta \stackrel{\phi_1}{\to} A_1, \Delta \stackrel{\phi_2}{\to} A_2\{\phi_1\}, \ldots, \Delta \stackrel{\phi_n}{\to} A_n\{(\phi_1, \phi_2, \ldots, \phi_{n-1})\}) : \Delta \to \Gamma.$$

This list construction is nothing about game semantics.

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

$$(\underline{k}, \underline{n_1} \otimes \underline{n_2} \otimes \cdots) \quad (\text{whatever } k \in \mathbb{N} \text{ is})$$

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., $\mathsf{List_N}$*).*

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

$$N \mathbin{\&} N \underset{\geqslant}{\overset{\underline{\mathrm{tt}}}{\rightrightarrows}} \Omega$$

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

$$N \,\&\, N \underset{\geqslant}{\overset{\underline{\mathrm{tt}}}{\rightrightarrows}} \Omega$$

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

$$N \,\&\, N \overset{\underline{\mathrm{tt}}}{\underset{\geqslant}{\rightrightarrows}} \Omega$$

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

Why does it matter?

- To internalise ∞-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

$$N \,\&\, N \underset{\geqslant}{\overset{\underline{\mathrm{tt}}}{\rightrightarrows}} \Omega$$

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

Why does it matter?

- To internalise ∞-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 $\boldsymbol{s} \in G$ is contained in some strategy $\sigma : G$.

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

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

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

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

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 $\boldsymbol{s} \in G$ is contained in some strategy $\sigma : G$. Specifically, define $\sigma := \mathrm{Pref}(\{\boldsymbol{s}\})^{\mathrm{Even}}$.

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 $\boldsymbol{s} \in G$ is contained in some strategy $\sigma : G$. Specifically, define $\sigma := \mathrm{Pref}(\{\boldsymbol{s}\})^{\mathrm{Even}}$.

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

Idea: To equip each game G with a map $f_G : \mathrm{st}(G) \to \mathrm{sub}(G)$, 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 $\boldsymbol{s} \in G$ is contained in some strategy $\sigma : G$. Specifically, define $\sigma := \mathrm{Pref}(\{\boldsymbol{s}\})^{\mathrm{Even}}$.

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

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

$$\forall \boldsymbol{s}mn \in \sigma.\, \boldsymbol{s}m \in f_G(\sigma) \Rightarrow \boldsymbol{s}mn \in f_G(\sigma). \tag{1}$$

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

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

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

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

$$\forall \boldsymbol{s}mn \in \sigma.\, \boldsymbol{s}m \in f_G(\sigma) \Rightarrow \boldsymbol{s}mn \in f_G(\sigma). \tag{1}$$

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 $\boldsymbol{s} \in G$ is contained in some strategy $\sigma : G$. Specifically, define $\sigma := \mathrm{Pref}(\{\boldsymbol{s}\})^{\mathrm{Even}}$.

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

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

$$\forall \boldsymbol{s}mn \in \sigma.\, \boldsymbol{s}m \in f_G(\sigma) \Rightarrow \boldsymbol{s}mn \in f_G(\sigma). \tag{1}$$

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

$$\frac{G}{\quad}$$

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

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

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

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

$$\forall \boldsymbol{s}mn \in \sigma.\, \boldsymbol{s}m \in f_G(\sigma) \Rightarrow \boldsymbol{s}mn \in f_G(\sigma). \tag{1}$$

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 $\boldsymbol{s} \in G$ is contained in some strategy $\sigma : G$. Specifically, define $\sigma := \mathrm{Pref}(\{\boldsymbol{s}\})^{\mathrm{Even}}$.

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

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

$$\forall \boldsymbol{s}mn \in \sigma . \, \boldsymbol{s}m \in f_G(\sigma) \Rightarrow \boldsymbol{s}mn \in f_G(\sigma). \tag{1}$$

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

G	
q_G	What is your strategy?
σ	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 $\boldsymbol{s} \in G$ is contained in some strategy $\sigma : G$. Specifically, define $\sigma := \mathrm{Pref}(\{\boldsymbol{s}\})^{\mathrm{Even}}$.

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

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

$$\forall \boldsymbol{s}mn \in \sigma.\, \boldsymbol{s}m \in f_G(\sigma) \Rightarrow \boldsymbol{s}mn \in f_G(\sigma). \tag{1}$$

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

G	
q_G	What is your strategy?
σ	It is $\sigma : G$, which satisfies the axiom (1)!
m_1	

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

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

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

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

$$\forall \boldsymbol{s}mn \in \sigma.\, \boldsymbol{s}m \in f_G(\sigma) \Rightarrow \boldsymbol{s}mn \in f_G(\sigma). \quad (1)$$

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

G	
q_G	What is your strategy?
σ	It is $\sigma : G$, which satisfies the axiom (1)!
m_1	
m_2	

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

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

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

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

$$\forall \boldsymbol{s}mn \in \sigma.\, \boldsymbol{s}m \in f_G(\sigma) \Rightarrow \boldsymbol{s}mn \in f_G(\sigma). \tag{1}$$

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

$$\begin{array}{c} G \\ \hline q_G \\ \sigma \\ m_1 \\ m_2 \\ \vdots \end{array}$$

What is your strategy?

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 $\boldsymbol{s} \in G$ is contained in some strategy $\sigma : G$. Specifically, define $\sigma := \mathrm{Pref}(\{\boldsymbol{s}\})^{\mathrm{Even}}$.

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

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

$$\forall \boldsymbol{s}mn \in \sigma.\, \boldsymbol{s}m \in f_G(\sigma) \Rightarrow \boldsymbol{s}mn \in f_G(\sigma). \qquad (1)$$

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

G	
q_G	What is your strategy?
σ	It is $\sigma : G$, which satisfies the axiom (1)!
m_1	
m_2	
$\vdots$	($m_1 m_2 \dots$ is played by σ in $f_G(\sigma) \subseteq G$)

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 $\Sigma_{\mathsf{x}:\mathsf{N}}\mathsf{N_b}(\mathsf{x})$ by the pair $\Sigma(N, N_b) = (N \mathbin{\&} N, f_{\Sigma(N,N_b)})$, where $f_{\Sigma(N,N_b)} : \langle \underline{k}, \underline{n} \rangle \mapsto N \mathbin{\&} N_b(\underline{k})$.

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

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

$$\frac{\Sigma(N \qquad , \qquad N_b)}{\qquad}$$

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

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

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

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

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

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

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

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

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

q

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

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

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

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

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

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

q

1

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

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

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

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

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

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

q

1

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

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

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

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

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

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

$$\begin{array}{lr} q & \\ 1 & \end{array} \qquad\qquad\qquad\qquad \begin{array}{lr} & q \\ & 0 \end{array}$$

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

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

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

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 $N_b(\underline{0}) = \mathrm{Pref}(\{q0\})$.

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

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

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

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 $N_b(\underline{0}) = \mathrm{Pref}(\{q0\})$.

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 $\Sigma_{\mathsf{x:N}}\mathsf{List_N}(\mathsf{x})$ by the pair $\Sigma(N, \mathrm{List}_N) = (N \,\&\, \bigcup_{k \in \mathbb{N}} \mathrm{List}_N(\underline{k}), f_{\Sigma(N,\mathrm{List}_N)})$, where $f_{\Sigma(N,\mathrm{List}_N)} : \langle \underline{k}, \underline{n_1} \otimes \underline{n_2} \otimes \dots \rangle \mapsto N \,\&\, \mathrm{List}_N(\underline{k})$.

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

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

$$\frac{\Sigma(N \qquad , \qquad \mathrm{List}_N)}{}$$

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

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

$$\frac{\Sigma(N \qquad\qquad , \qquad\qquad \mathrm{List}_N)}{q_{\Sigma(N, \mathrm{List}_N)}}$$

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

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

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

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

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

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

q

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

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

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

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

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

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

q

2

$$\frac{\Sigma(N \qquad , \qquad \mathrm{List}_N)}{}$$

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

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

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

$$\frac{\Sigma(N \qquad , \qquad \mathrm{List}_N)}{q_{\Sigma(N,\mathrm{List}_N)}}$$

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

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

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

q

2

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

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

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

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

q (right diagram)

q

2

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

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

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

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

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

$$\begin{array}{ccc} \Sigma(N & , & \mathrm{List}_N) \\ \hline & q_{\Sigma(N,\mathrm{List}_N)} & \\ & \langle \underline{2}, \underline{0} \otimes \underline{1} \rangle & \\ q & & \\ 2 & & \end{array}$$

$$\begin{array}{cccc} \Sigma(N & , & \mathrm{List}_N) & \\ \hline & q_{\Sigma(N,\mathrm{List}_N)} & & \\ & \langle \underline{2}, \underline{0} \otimes \underline{1} \rangle & & \\ & & & q \\ & & & 1 \\ & & q & \end{array}$$

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

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

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

$q_{\Sigma(N,\mathrm{List}_N)}$

$\langle \underline{2}, \underline{0} \otimes \underline{1} \rangle$

q

2

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

$q_{\Sigma(N,\mathrm{List}_N)}$

$\langle \underline{2}, \underline{0} \otimes \underline{1} \rangle$

q

1

q

0

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

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

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

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

q (in N)
2

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

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

q
1

q
0

The declaration of the strategy $\langle \underline{2}, \underline{0} \otimes \underline{1} \rangle : N \mathbin{\&} \bigcup_{k \in \mathbb{N}} \mathrm{List}_N(\underline{k})$ *fixes the underlying game* $N \mathbin{\&} (N \otimes N) \subseteq N \mathbin{\&} \bigcup_{k \in \mathbb{N}} \mathrm{List}_N(\underline{k})$, *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 $\Sigma_{x:N}\mathsf{List}_N(x)$ by the pair $\Sigma(N, \mathrm{List}_N) = (N \,\&\, \bigcup_{k \in \mathbb{N}} \mathrm{List}_N(\underline{k}), f_{\Sigma(N,\mathrm{List}_N)})$, where $f_{\Sigma(N,\mathrm{List}_N)} : \langle \underline{k}, \underline{n_1} \otimes \underline{n_2} \otimes \dots \rangle \mapsto N \,\&\, \mathrm{List}_N(\underline{k})$.

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

The declaration of the strategy $\langle \underline{2}, \underline{0} \otimes \underline{1} \rangle : N \,\&\, \bigcup_{k \in \mathbb{N}} \mathrm{List}_N(\underline{k})$ *fixes the underlying game* $N \,\&\, (N \otimes N) \subseteq N \,\&\, \bigcup_{k \in \mathbb{N}} \mathrm{List}_N(\underline{k})$, *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

$$\frac{N \multimap N \qquad\qquad \multimap \qquad\qquad N \multimap N}{}$$

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

$$\frac{N \multimap N \qquad\qquad \multimap \qquad\qquad N \multimap N}{q_{(N \multimap N) \multimap (N \multimap N)}}$$

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

$$\frac{N \multimap N \qquad\qquad \multimap \qquad\qquad N \multimap N}{\begin{array}{c} q_{(N \multimap N) \multimap (N \multimap N)} \\ \mathrm{succ} \circ (\_) \end{array}}$$

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

$$\begin{array}{ccc} N \multimap N & \multimap & N \multimap N \\ \hline & q_{(N \multimap N) \multimap (N \multimap N)} & \\ & \mathrm{succ} \circ (\_) & \\ & & q_{N \multimap N} \end{array}$$

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

$$\frac{N \multimap N \qquad \multimap \qquad N \multimap N}{\begin{array}{c} q_{(N \multimap N) \multimap (N \multimap N)} \\ \mathrm{succ} \circ (\_) \end{array}}$$

$$q_{N \multimap N}$$

$$q_{N \multimap N}$$

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

$$\begin{array}{ccc} N \multimap N & \multimap & N \multimap N \\ \hline & q_{(N \multimap N) \multimap (N \multimap N)} & \\ & \mathrm{succ} \circ (\_) & \\ & & q_{N \multimap N} \\ q_{N \multimap N} & & \\ \sigma & & \end{array}$$

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

$$N \multimap N \qquad \multimap \qquad N \multimap N$$

$$q_{(N \multimap N) \multimap (N \multimap N)}$$

$$\mathrm{succ} \circ (\_)$$

$$q_{N \multimap N}$$

$$q_{N \multimap N}$$

$$\sigma$$

$$\mathrm{succ} \circ \sigma$$

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

$$
\begin{array}{ccc}
N \multimap N & \multimap & N \multimap N \\
\hline
 & q_{(N \multimap N) \multimap (N \multimap N)} & \\
 & \mathrm{succ} \circ (\_) & \\
 & & q_{N \multimap N} \\
q_{N \multimap N} & & \\
\sigma & & \\
 & & \mathrm{succ} \circ \sigma \\
 & \vdots &
\end{array}
$$

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

$$
\begin{array}{ccc}
N \multimap N & \multimap & N \multimap N \\
\hline
& q_{(N \multimap N) \multimap (N \multimap N)} & \\
& \mathrm{succ} \circ (\_) & \\
& & q_{N \multimap N} \\
q_{N \multimap N} & & \\
\sigma & & \\
& & \mathrm{succ} \circ \sigma \\
& \vdots &
\end{array}
$$

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

$N \multimap N$	$\multimap$	$N \multimap N$
	$q_{(N \multimap N) \multimap (N \multimap N)}$	
	$\mathrm{succ} \circ (\_)$	
		$q_{N \multimap N}$
$q_{N \multimap N}$		
σ		
		$\mathrm{succ} \circ \sigma$
	$\vdots$	

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_Γ.

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_Γ.
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_Γ.
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_Γ.

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_Γ.

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_Γ.

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* $\overline{\gamma}_\Gamma \multimap f_\Delta(\phi \circ \gamma) \subseteq |\Gamma \multimap \Delta|$ for *all* $\gamma : \Gamma$ that is not yet excluded, i.e., the current position $\boldsymbol{s}$ is compatible with γ (or $\boldsymbol{s} \upharpoonright |\Gamma| \in \overline{\gamma}_\Gamma$), where $\overline{\gamma}_\Gamma$ is the subgame of $f_\Gamma(\gamma)$ played by Player and γ (Opponent), i.e.,

$$\overline{\gamma}_\Gamma := \{\boldsymbol{\epsilon}\} \cup \{\, \boldsymbol{s}m \in f_\Gamma(\gamma) \mid \boldsymbol{s} \in \overline{\gamma}_\Gamma \,\} \cup \{\, \boldsymbol{t}lr \in \gamma \mid \boldsymbol{t}l \in \overline{\gamma}_\Gamma \,\}.$$

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_Γ.

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* $\overline{\gamma}_\Gamma \multimap f_\Delta(\phi \circ \gamma) \subseteq |\Gamma \multimap \Delta|$ for *all* $\gamma : \Gamma$ that is not yet excluded, i.e., the current position $\boldsymbol{s}$ is compatible with γ (or $\boldsymbol{s} \upharpoonright |\Gamma| \in \overline{\gamma}_\Gamma$), where $\overline{\gamma}_\Gamma$ is the subgame of $f_\Gamma(\gamma)$ played by Player and γ (Opponent), i.e.,

$$\overline{\gamma}_\Gamma := \{\boldsymbol{\epsilon}\} \cup \{\, \boldsymbol{s}m \in f_\Gamma(\gamma) \mid \boldsymbol{s} \in \overline{\gamma}_\Gamma \,\} \cup \{\, \boldsymbol{t}lr \in \gamma \mid \boldsymbol{t}l \in \overline{\gamma}_\Gamma \,\}.$$

$$\underline{\Sigma(N, \quad \mathrm{List}_N) \qquad \multimap \qquad N}$$

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_Γ.

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* $\overline{\gamma}_\Gamma \multimap f_\Delta(\phi \circ \gamma) \subseteq |\Gamma \multimap \Delta|$ for *all* $\gamma : \Gamma$ that is not yet excluded, i.e., the current position $\boldsymbol{s}$ is compatible with γ (or $\boldsymbol{s} \upharpoonright |\Gamma| \in \overline{\gamma}_\Gamma$), where $\overline{\gamma}_\Gamma$ is the subgame of $f_\Gamma(\gamma)$ played by Player and γ (Opponent), i.e.,

$$\overline{\gamma}_\Gamma := \{\boldsymbol{\epsilon}\} \cup \{\, \boldsymbol{s}m \in f_\Gamma(\gamma) \mid \boldsymbol{s} \in \overline{\gamma}_\Gamma \,\} \cup \{\, \boldsymbol{t}lr \in \gamma \mid \boldsymbol{t}l \in \overline{\gamma}_\Gamma \,\}.$$

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

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_Γ.

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* $\overline{\gamma}_\Gamma \multimap f_\Delta(\phi \circ \gamma) \subseteq |\Gamma \multimap \Delta|$ for *all* $\gamma : \Gamma$ that is not yet excluded, i.e., the current position $\boldsymbol{s}$ is compatible with γ (or $\boldsymbol{s} \upharpoonright |\Gamma| \in \overline{\gamma}_\Gamma$), where $\overline{\gamma}_\Gamma$ is the subgame of $f_\Gamma(\gamma)$ played by Player and γ (Opponent), i.e.,

$$\overline{\gamma}_\Gamma := \{\boldsymbol{\epsilon}\} \cup \{\, \boldsymbol{s}m \in f_\Gamma(\gamma) \mid \boldsymbol{s} \in \overline{\gamma}_\Gamma \,\} \cup \{\, \boldsymbol{t}lr \in \gamma \mid \boldsymbol{t}l \in \overline{\gamma}_\Gamma \,\}.$$

$$\begin{array}{cccc} \Sigma(N, & \mathrm{List}_N) & \multimap & N \\ \hline & & q_{\Delta \multimap \Gamma} & \\ & & \pi_1 & \end{array}$$

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_Γ.

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* $\overline{\gamma}_\Gamma \multimap f_\Delta(\phi \circ \gamma) \subseteq |\Gamma \multimap \Delta|$ for *all* $\gamma : \Gamma$ that is not yet excluded, i.e., the current position $\boldsymbol{s}$ is compatible with γ (or $\boldsymbol{s} \upharpoonright |\Gamma| \in \overline{\gamma}_\Gamma$), where $\overline{\gamma}_\Gamma$ is the subgame of $f_\Gamma(\gamma)$ played by Player and γ (Opponent), i.e.,

$$\overline{\gamma}_\Gamma := \{\boldsymbol{\epsilon}\} \cup \{\, \boldsymbol{s}m \in f_\Gamma(\gamma) \mid \boldsymbol{s} \in \overline{\gamma}_\Gamma \,\} \cup \{\, \boldsymbol{t}lr \in \gamma \mid \boldsymbol{t}l \in \overline{\gamma}_\Gamma \,\}.$$

$\Sigma(N,$	$\mathrm{List}_N)$	$\multimap$	N
		$q_{\Delta \multimap \Gamma}$	
		π_1	
			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_Γ.

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* $\overline{\gamma}_\Gamma \multimap f_\Delta(\phi \circ \gamma) \subseteq |\Gamma \multimap \Delta|$ for *all* $\gamma : \Gamma$ that is not yet excluded, i.e., the current position $\boldsymbol{s}$ is compatible with γ (or $\boldsymbol{s} \upharpoonright |\Gamma| \in \overline{\gamma}_\Gamma$), where $\overline{\gamma}_\Gamma$ is the subgame of $f_\Gamma(\gamma)$ played by Player and γ (Opponent), i.e.,

$$\overline{\gamma}_\Gamma := \{\boldsymbol{\epsilon}\} \cup \{\, \boldsymbol{s}m \in f_\Gamma(\gamma) \mid \boldsymbol{s} \in \overline{\gamma}_\Gamma \,\} \cup \{\, \boldsymbol{t}lr \in \gamma \mid \boldsymbol{t}l \in \overline{\gamma}_\Gamma \,\}.$$

$\Sigma(N,$	$\mathrm{List}_N)$	$\multimap$	N
		$q_{\Delta \multimap \Gamma}$	
		π_1	
			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_Γ.

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* $\overline{\gamma}_\Gamma \multimap f_\Delta(\phi \circ \gamma) \subseteq |\Gamma \multimap \Delta|$ for *all* $\gamma : \Gamma$ that is not yet excluded, i.e., the current position $\boldsymbol{s}$ is compatible with γ (or $\boldsymbol{s} \upharpoonright |\Gamma| \in \overline{\gamma}_\Gamma$), where $\overline{\gamma}_\Gamma$ is the subgame of $f_\Gamma(\gamma)$ played by Player and γ (Opponent), i.e.,

$$\overline{\gamma}_\Gamma := \{\boldsymbol{\epsilon}\} \cup \{\, \boldsymbol{s}m \in f_\Gamma(\gamma) \mid \boldsymbol{s} \in \overline{\gamma}_\Gamma \,\} \cup \{\, \boldsymbol{t}lr \in \gamma \mid \boldsymbol{t}l \in \overline{\gamma}_\Gamma \,\}.$$

$\Sigma(N,$	$\mathrm{List}_N)$	$\multimap$	N
		$q_{\Delta \multimap \Gamma}$	
		π_1	
			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_Γ.

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* $\overline{\gamma}_\Gamma \multimap f_\Delta(\phi \circ \gamma) \subseteq |\Gamma \multimap \Delta|$ for *all* $\gamma : \Gamma$ that is not yet excluded, i.e., the current position $\boldsymbol{s}$ is compatible with γ (or $\boldsymbol{s} \upharpoonright |\Gamma| \in \overline{\gamma}_\Gamma$), where $\overline{\gamma}_\Gamma$ is the subgame of $f_\Gamma(\gamma)$ played by Player and γ (Opponent), i.e.,

$$\overline{\gamma}_\Gamma := \{\boldsymbol{\epsilon}\} \cup \{\, \boldsymbol{s}m \in f_\Gamma(\gamma) \mid \boldsymbol{s} \in \overline{\gamma}_\Gamma \,\} \cup \{\, \boldsymbol{t}lr \in \gamma \mid \boldsymbol{t}l \in \overline{\gamma}_\Gamma \,\}.$$

$\Sigma(N,$	$\mathrm{List}_N)$	$\multimap$	N
		$q_{\Delta \multimap \Gamma}$	
		π_1	
			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_Γ.

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* $\overline{\gamma}_\Gamma \multimap f_\Delta(\phi \circ \gamma) \subseteq |\Gamma \multimap \Delta|$ for *all* $\gamma : \Gamma$ that is not yet excluded, i.e., the current position $\boldsymbol{s}$ is compatible with γ (or $\boldsymbol{s} \upharpoonright |\Gamma| \in \overline{\gamma}_\Gamma$), where $\overline{\gamma}_\Gamma$ is the subgame of $f_\Gamma(\gamma)$ played by Player and γ (Opponent), i.e.,

$$\overline{\gamma}_\Gamma := \{\boldsymbol{\epsilon}\} \cup \{\, \boldsymbol{s}m \in f_\Gamma(\gamma) \mid \boldsymbol{s} \in \overline{\gamma}_\Gamma \,\} \cup \{\, \boldsymbol{t}lr \in \gamma \mid \boldsymbol{t}l \in \overline{\gamma}_\Gamma \,\}.$$

$\Sigma(N,$	$\mathrm{List}_N)$	$\multimap$	N
		$q_{\Delta \multimap \Gamma}$	
		π_1	
			q
q			
k			
			k

$\Sigma(N,$	$\mathrm{List}_N)$	$\multimap$	N

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_Γ.

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* $\overline{\gamma}_\Gamma \multimap f_\Delta(\phi \circ \gamma) \subseteq |\Gamma \multimap \Delta|$ for *all* $\gamma : \Gamma$ that is not yet excluded, i.e., the current position $\boldsymbol{s}$ is compatible with γ (or $\boldsymbol{s} \upharpoonright |\Gamma| \in \overline{\gamma}_\Gamma$), where $\overline{\gamma}_\Gamma$ is the subgame of $f_\Gamma(\gamma)$ played by Player and γ (Opponent), i.e.,

$$\overline{\gamma}_\Gamma := \{\boldsymbol{\epsilon}\} \cup \{\, \boldsymbol{s}m \in f_\Gamma(\gamma) \mid \boldsymbol{s} \in \overline{\gamma}_\Gamma \,\} \cup \{\, \boldsymbol{t}lr \in \gamma \mid \boldsymbol{t}l \in \overline{\gamma}_\Gamma \,\}.$$

$\Sigma(N,$	$\mathrm{List}_N)$	$\multimap$	N
		$q_{\Delta \multimap \Gamma}$	
		π_1	
			q
q			
k			
			k

$\Sigma(N,$	$\mathrm{List}_N)$	$\multimap$	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_Γ.

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* $\overline{\gamma}_\Gamma \multimap f_\Delta(\phi \circ \gamma) \subseteq |\Gamma \multimap \Delta|$ for *all* $\gamma : \Gamma$ that is not yet excluded, i.e., the current position $\boldsymbol{s}$ is compatible with γ (or $\boldsymbol{s} \upharpoonright |\Gamma| \in \overline{\gamma}_\Gamma$), where $\overline{\gamma}_\Gamma$ is the subgame of $f_\Gamma(\gamma)$ played by Player and γ (Opponent), i.e.,

$$\overline{\gamma}_\Gamma := \{\boldsymbol{\epsilon}\} \cup \{\, \boldsymbol{s}m \in f_\Gamma(\gamma) \mid \boldsymbol{s} \in \overline{\gamma}_\Gamma \,\} \cup \{\, \boldsymbol{t}lr \in \gamma \mid \boldsymbol{t}l \in \overline{\gamma}_\Gamma \,\}.$$

$\Sigma(N,$	$\mathrm{List}_N)$	$\multimap$	N
		$q_{\Delta \multimap \Gamma}$	
		π_1	
			q
q			
k			
			k

$\Sigma(N,$	$\mathrm{List}_N)$	$\multimap$	N
		$q_{\Delta \multimap \Gamma}$	
		ϕ	

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_Γ.

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* $\overline{\gamma}_\Gamma \multimap f_\Delta(\phi \circ \gamma) \subseteq |\Gamma \multimap \Delta|$ for *all* $\gamma : \Gamma$ that is not yet excluded, i.e., the current position $\boldsymbol{s}$ is compatible with γ (or $\boldsymbol{s} \upharpoonright |\Gamma| \in \overline{\gamma}_\Gamma$), where $\overline{\gamma}_\Gamma$ is the subgame of $f_\Gamma(\gamma)$ played by Player and γ (Opponent), i.e.,

$$\overline{\gamma}_\Gamma := \{\boldsymbol{\epsilon}\} \cup \{\, \boldsymbol{s}m \in f_\Gamma(\gamma) \mid \boldsymbol{s} \in \overline{\gamma}_\Gamma \,\} \cup \{\, \boldsymbol{t}lr \in \gamma \mid \boldsymbol{t}l \in \overline{\gamma}_\Gamma \,\}.$$

$\Sigma(N,$	$\mathrm{List}_N)$	$\multimap$	N
		$q_{\Delta \multimap \Gamma}$	
		π_1	
			q
q			
k			
			k

$\Sigma(N,$	$\mathrm{List}_N)$	$\multimap$	N
		$q_{\Delta \multimap \Gamma}$	
		ϕ	
			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_Γ.

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* $\overline{\gamma}_\Gamma \multimap f_\Delta(\phi \circ \gamma) \subseteq |\Gamma \multimap \Delta|$ for *all* $\gamma : \Gamma$ that is not yet excluded, i.e., the current position $\boldsymbol{s}$ is compatible with γ (or $\boldsymbol{s} \upharpoonright |\Gamma| \in \overline{\gamma}_\Gamma$), where $\overline{\gamma}_\Gamma$ is the subgame of $f_\Gamma(\gamma)$ played by Player and γ (Opponent), i.e.,

$$\overline{\gamma}_\Gamma := \{\boldsymbol{\epsilon}\} \cup \{\, \boldsymbol{s}m \in f_\Gamma(\gamma) \mid \boldsymbol{s} \in \overline{\gamma}_\Gamma \,\} \cup \{\, \boldsymbol{t}lr \in \gamma \mid \boldsymbol{t}l \in \overline{\gamma}_\Gamma \,\}.$$

$\Sigma(N,$	$\mathrm{List}_N)$	$\multimap$	N
		$q_{\Delta \multimap \Gamma}$	
		π_1	
			q
q			
k			
			k

$\Sigma(N,$	$\mathrm{List}_N)$	$\multimap$	N
		$q_{\Delta \multimap \Gamma}$	
		ϕ	
			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_Γ.

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* $\overline{\gamma}_\Gamma \multimap f_\Delta(\phi \circ \gamma) \subseteq |\Gamma \multimap \Delta|$ for *all* $\gamma : \Gamma$ that is not yet excluded, i.e., the current position $\boldsymbol{s}$ is compatible with γ (or $\boldsymbol{s} \upharpoonright |\Gamma| \in \overline{\gamma}_\Gamma$), where $\overline{\gamma}_\Gamma$ is the subgame of $f_\Gamma(\gamma)$ played by Player and γ (Opponent), i.e.,

$$\overline{\gamma}_\Gamma := \{\boldsymbol{\epsilon}\} \cup \{\, \boldsymbol{s}m \in f_\Gamma(\gamma) \mid \boldsymbol{s} \in \overline{\gamma}_\Gamma \,\} \cup \{\, \boldsymbol{t}lr \in \gamma \mid \boldsymbol{t}l \in \overline{\gamma}_\Gamma \,\}.$$

$\Sigma(N,$	$\mathrm{List}_N)$	$\multimap$	N
		$q_{\Delta \multimap \Gamma}$	
		π_1	
			q
q			
k			
			k

$\Sigma(N,$	$\mathrm{List}_N)$	$\multimap$	N
		$q_{\Delta \multimap \Gamma}$	
		ϕ	
			q
	q		

ϕ fails the test!

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 $\overline{\gamma}_\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma) \subseteq |\Pi_\ell(\Gamma, A)|$ 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 $\overline{\gamma}_\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma) \subseteq |\Pi_\ell(\Gamma, A)|$ 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 $\overline{\gamma}_\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma) \subseteq |\Pi_\ell(\Gamma, A)|$ for all $\gamma : \Gamma$ not yet excluded,

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

$\Pi_\ell(N, \qquad \mathrm{List}_N)$

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 $\overline{\gamma}_\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma) \subseteq |\Pi_\ell(\Gamma, A)|$ for all $\gamma : \Gamma$ not yet excluded,

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

$$\frac{\Pi_\ell(N, \qquad\qquad\qquad\qquad \mathrm{List}_N)}{q_{\Pi_\ell}}$$

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 $\overline{\gamma}_\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma) \subseteq |\Pi_\ell(\Gamma, A)|$ for all $\gamma : \Gamma$ not yet excluded,

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

$$\begin{array}{ccc} \Pi_\ell(N, & & \mathrm{List}_N) \\ \hline & q_{\Pi_\ell} & \\ & k \mapsto 0^k & \end{array}$$

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 $\overline{\gamma}_\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma) \subseteq |\Pi_\ell(\Gamma, A)|$ for all $\gamma : \Gamma$ not yet excluded,

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

$$
\begin{array}{ccc}
\Pi_\ell(N, & & \mathrm{List}_N) \\
\hline
 & q_{\Pi_\ell} & \\
 & k \mapsto 0^k & \\
 & & q
\end{array}
$$

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 $\overline{\gamma}_\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma) \subseteq |\Pi_\ell(\Gamma, A)|$ for all $\gamma : \Gamma$ not yet excluded,

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

$\Pi_\ell(N,$		$\mathrm{List}_N)$
	q_{Π_ℓ}	
	$k \mapsto 0^k$	
		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 $\overline{\gamma}_\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma) \subseteq |\Pi_\ell(\Gamma, A)|$ for all $\gamma : \Gamma$ not yet excluded,

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

$\Pi_\ell(N,$		$\mathrm{List}_N)$
	q_{Π_ℓ}	
	$k \mapsto 0^k$	
		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 $\overline{\gamma}_\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma) \subseteq |\Pi_\ell(\Gamma, A)|$ for all $\gamma : \Gamma$ not yet excluded,

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

$\Pi_\ell(N,$		$\mathrm{List}_N)$
	q_{Π_ℓ}	
	$k \mapsto 0^k$	
		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 $\overline{\gamma}_\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma) \subseteq |\Pi_\ell(\Gamma, A)|$ for all $\gamma : \Gamma$ not yet excluded,

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

$\Pi_\ell(N,$		$\mathrm{List}_N)$
	q_{Π_ℓ} $k \mapsto 0^k$	
		q
q		
1		
		0

$\Pi_\ell(\Sigma(N,$	$N_b)$	,	$N_b\{\pi_1\})$

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 $\overline{\gamma}_\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma) \subseteq |\Pi_\ell(\Gamma, A)|$ for all $\gamma : \Gamma$ not yet excluded,

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

$\Pi_\ell(N,$		$\mathrm{List}_N)$
	q_{Π_ℓ}	
	$k \mapsto 0^k$	
		q
q		
1		
		0

$\Pi_\ell(\Sigma(N,$	$N_b)$	,	$N_b\{\pi_1\})$
		q_{Π_ℓ}	

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 $\overline{\gamma}_\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma) \subseteq |\Pi_\ell(\Gamma, A)|$ for all $\gamma : \Gamma$ not yet excluded,

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

$$\begin{array}{ccc} \Pi_\ell(N, & & \mathrm{List}_N) \\ \hline & q_{\Pi_\ell} & \\ & k \mapsto 0^k & \\ & & q \\ q & & \\ 1 & & \\ & & 0 \end{array} \qquad \begin{array}{ccc} \Pi_\ell(\Sigma(N, \; N_b) & , & N_b\{\pi_1\}) \\ \hline & q_{\Pi_\ell} & \\ & \underline{1} & \end{array}$$

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 $\overline{\gamma}_\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma) \subseteq |\Pi_\ell(\Gamma, A)|$ for all $\gamma : \Gamma$ not yet excluded,

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

$\Pi_\ell(N,$	$\mathrm{List}_N)$
q_{Π_ℓ}	
$k \mapsto 0^k$	
	q
q	
1	
	0

$\Pi_\ell(\Sigma(N,$	$N_b)$	,	$N_b\{\pi_1\})$
		q_{Π_ℓ}	
		$\underline{1}$	
			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 $\overline{\gamma}_\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma) \subseteq |\Pi_\ell(\Gamma, A)|$ for all $\gamma : \Gamma$ not yet excluded,

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

$\Pi_\ell(N,$		$\mathrm{List}_N)$
	q_{Π_ℓ}	
	$k \mapsto 0^k$	
		q
q		
1		
		0

$\Pi_\ell(\Sigma(N,$	$N_b)$	,	$N_b\{\pi_1\})$
		q_{Π_ℓ}	
		$\underline{1}$	
			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 $\overline{\gamma}_\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma) \subseteq |\Pi_\ell(\Gamma, A)|$ for all $\gamma : \Gamma$ not yet excluded,

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

$\Pi_\ell(N,$		$\mathrm{List}_N)$
	q_{Π_ℓ}	
	$k \mapsto 0^k$	
		q
q		
1		
		0

$\Pi_\ell(\Sigma(N,$	$N_b)$	$,$	$N_b\{\pi_1\})$
		q_{Π_ℓ}	
		$\underline{1}$	
			q
			1
		$\underline{1}$ fails the test!	

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 $\overline{\gamma}_\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma) \subseteq |\Pi_\ell(\Gamma, A)|$ for all $\gamma : \Gamma$ not yet excluded,

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

$\Pi_\ell(N,$	$\mathrm{List}_N)$
q_{Π_ℓ} $k \mapsto 0^k$	
	q
q	
1	
	0

$\Pi_\ell(\Sigma(N,$	$N_b)$	$,$	$N_b\{\pi_1\})$
		q_{Π_ℓ} $\underline{1}$	
			q
			1
		$\underline{1}$ fails the test!	

The intensionality of game semantics is preserved.

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 $\overline{\gamma}_\Gamma \multimap f_{A(\gamma)}(\phi \circ \gamma) \subseteq |\Pi_\ell(\Gamma, A)|$ for all $\gamma : \Gamma$ not yet excluded,

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

$\Pi_\ell(N,$	$\mathrm{List}_N)$
q_{Π_ℓ} $k \mapsto 0^k$	
	q
q	
1	
	0

$\Pi_\ell(\Sigma(N,$	$N_b)$	,	$N_b\{\pi_1\})$
		q_{Π_ℓ} $\underline{1}$	
			q
			1
		$\underline{1}$ fails the test!	

The intensionality of game semantics is preserved.

We finally define the ***pi*** Π by $\Pi(\Gamma, B) := \Pi_\ell(!\Gamma, B)$, where B is over $!\Gamma$.

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

$$\Pi_\ell(\Gamma, A)(\phi) := \{\boldsymbol{\epsilon}\} \cup \{\, \boldsymbol{s}m \in |\Pi_\ell(\Gamma, A)|^{\mathrm{Odd}} \mid \boldsymbol{s} \in \Pi_\ell(\Gamma, A)(\phi), \exists \gamma : \Gamma.\, \boldsymbol{s}m \in A(\gamma)(\phi \circ \gamma)^{\overline{\gamma}\Gamma} \,\}$$
$$\cup \{\, \boldsymbol{t}lr \in |\Pi_\ell(\Gamma, A)|^{\mathrm{Even}} \mid \boldsymbol{t}l \in \Pi_\ell(\Gamma, A)(\phi), \forall \gamma : \Gamma.\, \boldsymbol{t}l \in A(\gamma)(\phi \circ \gamma)^{\overline{\gamma}\Gamma} \Rightarrow \boldsymbol{t}lr \in A(\gamma)(\phi \circ \gamma)^{\overline{\gamma}\Gamma} \,\}$$

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

$$\Pi_\ell(\Gamma, A)(\phi) := \{\boldsymbol{\epsilon}\} \cup \{\, \boldsymbol{s}m \in |\Pi_\ell(\Gamma, A)|^{\mathrm{Odd}} \mid \boldsymbol{s} \in \Pi_\ell(\Gamma, A)(\phi), \exists \gamma : \Gamma.\, \boldsymbol{s}m \in A(\gamma)(\phi \circ \gamma)^{\overline{\gamma}\Gamma} \,\}$$
$$\cup \{\, \boldsymbol{t}lr \in |\Pi_\ell(\Gamma, A)|^{\mathrm{Even}} \mid \boldsymbol{t}l \in \Pi_\ell(\Gamma, A)(\phi), \forall \gamma : \Gamma.\, \boldsymbol{t}l \in A(\gamma)(\phi \circ \gamma)^{\overline{\gamma}\Gamma} \Rightarrow \boldsymbol{t}lr \in A(\gamma)(\phi \circ \gamma)^{\overline{\gamma}\Gamma} \,\}$$

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

$$\Pi_\ell(\Gamma, A)(\phi) := \{\boldsymbol{\epsilon}\} \cup \{\, \boldsymbol{s}m \in |\Pi_\ell(\Gamma, A)|^{\mathrm{Odd}} \mid \boldsymbol{s} \in \Pi_\ell(\Gamma, A)(\phi), \exists \gamma : \Gamma.\, \boldsymbol{s}m \in A(\gamma)(\phi \circ \gamma)^{\overline{\gamma}\Gamma} \,\}$$
$$\cup \{\, \boldsymbol{t}lr \in |\Pi_\ell(\Gamma, A)|^{\mathrm{Even}} \mid \boldsymbol{t}l \in \Pi_\ell(\Gamma, A)(\phi), \forall \gamma : \Gamma.\, \boldsymbol{t}l \in A(\gamma)(\phi \circ \gamma)^{\overline{\gamma}\Gamma} \Rightarrow \boldsymbol{t}lr \in A(\gamma)(\phi \circ \gamma)^{\overline{\gamma}\Gamma} \,\}$$

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

$$\Pi_\ell(\Gamma, A)(\phi) := \{\boldsymbol{\epsilon}\} \cup \{\, \boldsymbol{s}m \in |\Pi_\ell(\Gamma, A)|^{\mathrm{Odd}} \mid \boldsymbol{s} \in \Pi_\ell(\Gamma, A)(\phi), \exists \gamma : \Gamma.\, \boldsymbol{s}m \in A(\gamma)(\phi \circ \gamma)^{\overline{\gamma}_\Gamma} \,\}$$
$$\cup \{\, \boldsymbol{t}lr \in |\Pi_\ell(\Gamma, A)|^{\mathrm{Even}} \mid \boldsymbol{t}l \in \Pi_\ell(\Gamma, A)(\phi), \forall \gamma : \Gamma.\, \boldsymbol{t}l \in A(\gamma)(\phi \circ \gamma)^{\overline{\gamma}_\Gamma} \Rightarrow \boldsymbol{t}lr \in A(\gamma)(\phi \circ \gamma)^{\overline{\gamma}_\Gamma} \,\}$$

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

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

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

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

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

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

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

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

- $|\Sigma(\Gamma, B)| := |\Gamma| \mathbin{\&} |B|$;
- $f_{\Sigma(\Gamma,B)}(\langle \gamma, \beta \rangle) := f_{\Gamma}(\gamma) \mathbin{\&} f_{B(\gamma^{\dagger})}(\beta)$ 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 Θ 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 $\mathrm{id}_{|\Theta|} : \Theta \hookrightarrow \Gamma$. □

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