Game Semantics for Dependent Types

Samson Abramsky, Radha Jagadeesan and Matthijs Vákár

Kyoto, 8 July, 2015
Game theoretic model of dependent type theory (DTT):

- refines model in domains and (total) continuous functions;
- faithful model of (total) DTT with \( \Sigma \)-, \( \Pi \)-, \( \text{Id} \)-types and finite inductive type families;
- fully complete if \( \text{Id} \)-types limited;
Game theoretic model of dependent type theory (DTT):

- refines model in domains and (total) continuous functions;
- faithful model of (total) DTT with $\Sigma$-, $\Pi$-, Id-types and finite inductive type families;
- fully complete if Id-types limited;
- Id-types more intensional than domain model: function extensionality fails;
- intensional in orthogonal way to HoTT (time vs space): UIP holds.
Game Semantics?

- Interpolates between operational and denotational semantics: very intensional with structural clarity of categorical model;
- Unified framework for intensional, computational semantics:
  - PCF (HO, N, \textbf{AJM});
- Various evaluation strategies;
- Recursive types, polymorphism;
- Propositional logic, impredicative second order quantification, external first order quantification; internal first order quantification / dependent types surprisingly absent (and surprisingly hard!);
- Tight correspondence with syntax (full abstraction, full faithful completeness): often unique semantics in this respect.

Samson Abramsky, Radha Jagadeesan and Matthijs Vákár

Game Semantics for Dependent Types
Interpolates between operational and denotational semantics: very intensional with structural clarity of categorical model;

Unified framework for intensional, computational semantics:
- PCF (HO, N, AJM);
- references, non-local control, dynamically generated local names, probability, non-determinism, concurrency…;
- various evaluation strategies;
- recursive types, polymorphism;
- propositional logic, impredicative $2^{nd}$ order quantification, external $1^{st}$ order quantification;
Game Semantics?

- Interpolates between operational and denotational semantics: very intensional with structural clarity of categorical model;
- Unified framework for intensional, computational semantics:
  - PCF (HO, N, AJM);
  - references, non-local control, dynamically generated local names, probability, non-determinism, concurrency...;
  - various evaluation strategies;
  - recursive types, polymorphism;
  - propositional logic, impredicative $2^{nd}$ order quantification, external $1^{st}$ order quantification;
  - **internal $1^{st}$ order quantification / dependent types** surprisingly absent (and surprisingly hard!);
Interpolates between operational and denotational semantics: very intensional with structural clarity of categorical model;

Unified framework for intensional, computational semantics:
- PCF (HO, N, AJM);
- references, non-local control, dynamically generated local names, probability, non-determinism, concurrency...;
- various evaluation strategies;
- recursive types, polymorphism;
- propositional logic, impredicative $2^{nd}$ order quantification, external $1^{st}$ order quantification;
- **internal $1^{st}$ order quantification / dependent types surprisingly absent (and surprisingly hard!);**
- Tight correspondence with syntax (full abstraction, full faithful completeness): often unique semantics in this respect.
<table>
<thead>
<tr>
<th><strong>Computation / Logic</strong></th>
<th><strong>Games</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>data type / proposition</td>
<td>2-player game (duality!)</td>
</tr>
<tr>
<td>computational process / argument</td>
<td>play: alternating sequence of moves</td>
</tr>
<tr>
<td>program / proof</td>
<td>Player ($P$) strategy</td>
</tr>
<tr>
<td>environment / refutation</td>
<td>Opponent ($O$) strategy</td>
</tr>
<tr>
<td>variable declarations / axiom links</td>
<td>copycat strategies</td>
</tr>
<tr>
<td>substitution / cut</td>
<td>interaction</td>
</tr>
<tr>
<td>termination / correctness</td>
<td>winning</td>
</tr>
<tr>
<td>pure functional behaviour (stateless + local control) / ...</td>
<td>history-freeness + well-bracketing of strategies</td>
</tr>
</tbody>
</table>
An example:

Player: \( x : \mathbb{B}, y : (\mathbb{B} \Rightarrow \mathbb{B}) \vdash y(x) : \mathbb{B} \)

Opponent: \(- [\text{tt}/x, (\lambda z: \mathbb{B} \neg z)/y]\)

\[
\begin{array}{cccc}
\mathbb{B} \Rightarrow (\mathbb{B} \Rightarrow \mathbb{B}) \Rightarrow \mathbb{B} & \Rightarrow & \mathbb{B} & \\
\hline
\text{tt} & \Rightarrow & \text{tt} & \Rightarrow & \text{tt} & \Rightarrow & \text{ff} & \Rightarrow & \text{ff} & \Rightarrow & \mathbb{B} & \\
\end{array}
\]
An example:

Player: \( x : \mathcal{B}, y : (\mathcal{B} \Rightarrow \mathcal{B}) \vdash y(x) : \mathcal{B} \)

Opponent: \(- [ (\lambda z : \mathcal{B} \text{ff})/y] \)

\[
\begin{array}{c|ccc}
\mathcal{B} & (\mathcal{B} & \Rightarrow & \mathcal{B}) & \Rightarrow & \mathcal{B} \\
\hline
* & * & O & P \\
* & ff & O & P \\
ff & ff & P & P \\
\end{array}
\]

history-free!
Games and winning history-free strategies form a smcc \textbf{Game}: 

- \textbf{I}: the game with one play of length 0;
- \textbf{A \otimes B}: playing \( A \) and \( B \) simultaneously, where only Opponent can switch games;
- \textbf{A \rightarrow B}: \( swap_{O,P}(A) \) and \( B \) simultaneously, Player switches.
Games and winning history-free strategies form a smcc \textbf{Game}:

- \textit{I}: the game with one play of length 0;
- \textit{A \otimes B}: playing \( A \) and \( B \) simultaneously, where only Opponent can switch games;
- \textit{A \rightarrow B}: \textit{swap}_{O,P}(A) \text{ and } B \text{ simultaneously, Player switches.}

Also model simple type theory (STT): have a ccc \textbf{Game}!:

- \textit{A \Rightarrow B} := !A \rightarrow B;
- \textit{!A}: playing \( \omega \equiv \) equivalent copies of \( A \) simultaneously, where only Opponent can switch games;
- Product \textit{A\&B}: Opponent chooses to play \( A \) or \( B \) (unit: \textit{I}).
Games and winning history-free strategies form a smcc **Game**:

- \( I \): the game with one play of length 0;
- \( A \otimes B \): playing \( A \) and \( B \) simultaneously, where only Opponent can switch games;
- \( A \rightarrow B \): \( \text{swap}_{O,P}(A) \) and \( B \) simultaneously, Player switches.

Also model simple type theory (STT): have a ccc **Game**!:

- \( A \Rightarrow B \) := \( !A \rightarrow B \);
- \( !A \): playing \( \omega \) equivalent copies of \( A \) simultaneously, where only Opponent can switch games;
- Product \( A \& B \): Opponent chooses to play \( A \) or \( B \) (unit: \( I \)).

Ground types (finite inductive types): for a set \( X \), game \( \tilde{X}_* \) with one Opponent move \( * \), followed by any of the Player moves \( x \in X \).
Use the term simple type theory (STT) to refer to a simple \(\lambda\)-calculus with binary products \(\times\), function types \(\Rightarrow\) and finite inductive types \(\{a_i \mid 1 \leq i \leq n\}\), or a total finitary PCF with binary products, with \(\beta\eta\)-rules and PCF commutative conversions for case-constructs.

Straightforward consequence of AJM:

**Theorem**

The interpretation of STT in \(\text{Game}_1\) is **fully and faithfully complete**.
Dependent type theory (DTT)?

What is it?

- Curry-Howard for predicate logic: types with free (term) variables, constructions $\Sigma$, $\Pi$, $\text{Id}$ on types.
- Judgements: $\vdash \Gamma \text{ ctxt}, \Gamma \vdash A$ type, $\Gamma \vdash a : A$, equations.
- Order in context matters!
- No clean separation syntax types and terms.

Why care?

- Move towards richer type systems: e.g. GADTs in Haskell.
- Types allowed to refer to data: e.g. $n : N \vdash \text{List}(n)$ type.
- Specification by typing: certification by type checking.
- Proof assistants.
- Logical Frameworks.
Dependent type theory (DTT)?

What is it?

- Curry-Howard for predicate logic: types with free (term) variables, constructions $\Sigma$, $\Pi$, $\text{Id}$ on types.
- Judgements: $\vdash \Gamma \text{ ctxt}, \Gamma \vdash A \text{ type}, \Gamma \vdash a : A$, equations.
- Order in context matters!
- No clean separation syntax types and terms.

Why care?

- Move towards richer type systems: e.g. GADTs in Haskell.
- Types allowed to refer to data: e.g. $n : \mathbb{N} \vdash \text{List}(n) \text{ type}$.
- Specification by typing: certification by type checking.
- Proof assistants.
- Logical Frameworks.
A Faithful Translation to Simple Type Theory (STT)

Idea: DTT talks about same algorithms as STT but can assign them a more precise type/specification.

Formally: have translation of DTT into STT. Let DTT inherit the equational theory of STT to make translation faithful.
Idea: DTT talks about same algorithms as STT but can assign them a more precise type/specification.

Formally: have translation of DTT into STT. Let DTT inherit the equational theory of STT to make translation faithful.

\[
\begin{align*}
x : A \vdash (a_i \mapsto_i \{b_{i,j} | j\}) \text{ type} & \quad \mapsto \quad \vdash \{b_{i,j} | i,j\} \text{ type} \\
x : A \vdash \Sigma_{y:B} C \text{ type} & \quad \mapsto \quad \vdash B^T \times C^T \text{ type} \\
x : A \vdash \Pi_{y:B} C \text{ type} & \quad \mapsto \quad \vdash B^T \Rightarrow C^T \text{ type} \\
x : A, y : B, y' : B \vdash \text{Id}_B(y, y') \text{ type} & \quad \mapsto \quad \vdash B^T \text{ type} \\
x' : A' \vdash B[a/x] \text{ type} & \quad \mapsto \quad \vdash B^T \text{ type}
\end{align*}
\]

+ translation on terms.
The idea of our interpretation $\llbracket \cdot \rrbracket$ will be to construct a category $\text{CtxtGame}_!$ of dependent context games with a functor $\otimes (-)$ to $\text{Game}_!$, such that

$$\begin{array}{ccc}
\text{Syntax}_{\text{DTT}} & \xrightarrow{\llbracket - \rrbracket} & \text{CtxtGame}_!\\
\downarrow (-)^T & & \downarrow (\otimes (-))\\n\text{Syntax}_{\text{STT}} & \xrightarrow{\llbracket - \rrbracket} & \text{Game}_!.
\end{array}$$

Games model of DTT will therefore automatically be faithful.
Game $B \in \text{ob}(\text{DGame}_!(A))$ with dependency on $A$:

- game $\ominus(B) \in \text{ob}(\text{Game}_!)$ (without dependency);
- continuous function $\text{str}(A) \xrightarrow{B} \text{Sub}(\ominus(B))$ from strategies on $A$ to subgames of $\ominus(B)$.
Dependent Games and Strategies

Game $B \in \text{ob}(\text{DGame}_!(A))$ with dependency on $A$:

- game $\ominus(B) \in \text{ob}(\text{Game}_!)$ (without dependency);
- continuous function $\text{str}(A) \xrightarrow{B} \text{Sub}(\ominus(B))$ from strategies on $A$ to subgames of $\ominus(B)$.

Note: $\text{ob}(\text{Game}_!) \nsubseteq \text{ob}(\text{DGame}_!(I))$. Indeed, a dependent game $A$ in empty context is a pair $A(\bot) \subseteq \ominus(A)$, for empty strategy $\bot$.

Define $\text{ob}(\text{DGame}_!(A)) := \text{ob}(\text{DGame}_!(\ominus(A)))$.

Can define $\&$, $\Rightarrow$ on games with dependency and make into ccc with homset $\text{DGame}_!(A)(B, C) := \text{wstr}(\Pi_A(B \Rightarrow C))$.
We define a game $\Pi_A B \subseteq \mathcal{O}(A) \Rightarrow \mathcal{O}(B)$ of dependent functions.

- Idea: the choice of a fibre $B[a/x]$ for the output of a dependent function $f : \Pi_A B$ is entirely the responsibility of the context that provides the argument $a$. 

Opponent can determine fibre $B[\tau]$ of $B$:

- explicitly, revealing winning history-free strategy $\tau$ on $A(\bot)$, by playing in $! A$;
- implicitly, by playing in $! B$;

Player has to stay within $B[\tau]$ for all $\tau$ consistent with Opponent's behaviour. That is, as long as there is such a $\tau$; otherwise, anything goes.

Indeed, Opponent is totally free and might not play along a winning strategy, as $\mathcal{O}(-)$ should be faithful (to match $(\mathcal{O}(-))_T$).
We define a game $\Pi_A B \subseteq \mathcal{G}(A) \Rightarrow \mathcal{G}(B)$ of dependent functions.

- Idea: the choice of a fibre $B[a/x]$ for the output of a dependent function $f : \Pi_A B$ is entirely the responsibility of the context that provides the argument $a$.
- Opponent can determine fibre $B(\tau)$ of $B$:
  - explicitly, revealing winning history-free strategy $\tau$ on $A(\bot)$, by playing in $!\mathcal{G}(A)$;
  - implicitly, by playing in $\mathcal{G}(B)$;
- Player has to stay within $B(\tau)$ for all $\tau$ consistent with Opponent’s behaviour.
We define a game $\Pi_A B \triangleleft \& (A) \Rightarrow \& (B)$ of dependent functions.

- Idea: the choice of a fibre $B[a/x]$ for the output of a dependent function $f : \Pi_A B$ is entirely the responsibility of the context that provides the argument $a$.

- Opponent can determine fibre $B(\tau)$ of $B$:
  - explicitly, revealing winning history-free strategy $\tau$ on $A(\bot)$, by playing in $!\& (A)$;
  - implicitly, by playing in $\& (B)$;

- Player has to stay within $B(\tau)$ for all $\tau$ consistent with Opponent’s behaviour.

- That is, as long as there is such a $\tau$; otherwise, anything goes.

- Indeed, Opponent is totally free and might not play along a winning strategy, as $\& (\bot)$ should be faithful (to match $(\bot)^T$).
Example of dependently typed algorithm (let the year be 1984):

Player:  \( x : \text{months} \vdash \text{case}_{\text{days, months}}(x, \{1, 1, 1, 1, 1, 31, 1, 1, 1, 1\}) : \text{days}(x) \)

Opponent:  \( - \text{[July/}x] \)

<table>
<thead>
<tr>
<th>( \prod(\text{months}<em>*, \text{days}</em>*) )</th>
<th>( \ast )</th>
<th>( O )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \ast )</td>
<td>( P )</td>
<td></td>
</tr>
<tr>
<td>\text{July}</td>
<td>( O )</td>
<td></td>
</tr>
<tr>
<td>( 31 )</td>
<td>( P )</td>
<td></td>
</tr>
</tbody>
</table>
Non-example:

Player: \( x : \text{months} \vdash 31 : \text{days}(x) \)

Opponent:

\[
\Pi(\sim\text{months}_*, \sim\text{days}_*) \quad \text{*} \\
\quad 31 \quad O \\
\quad P
\]

Player chooses fibre: e.g. February doesn’t have a 31\(^{st}\) day.
Example (fibre-wise identities):

Player: \[ x : A, y : B(x) \vdash y : B(x) \]

Opponent: choose your favourite

\[
\Pi( [A], [B] \Rightarrow [B] )
\]

\[
\begin{array}{c|c|c}
  b & O \\
  b & P \\
  b' & O \\
  b' & P \\
  \vdots \\
\end{array}
\]
Theorem

Have strict indexed ccc \( \text{DGame}_! (I)^{op} \xrightarrow{\text{DGame}} \text{CCCat} \) of dependent games:

- \( \text{ob}(\text{DGame}_! (A)) := \{ \text{continuous str}(\circ(A)) \xrightarrow{B} \text{Sub}(\circ(B)) \} \)
- \( \text{hom-sets} \ \text{DGame}_! (A)(B, C) := \text{wstr}(\Pi_A (B \Rightarrow C)) \)
- identities as in example;
- composition: usual AJM-composition;
- change of base: usual AJM-composition (works out!).

As we don’t have (additive) \( \Sigma \)-types, this is not a model of DTT!
Theorem

Have strict indexed ccc $\mathbf{DGame}_!(I)^{op} \to \mathbf{CCCat}$ of dependent games:

- $\text{ob}(\mathbf{DGame}_!(A)) := \{\text{continuous str}(\odot(A)) \to B \to \text{Sub}(\odot(B))\}$
- $\text{hom-sets } \mathbf{DGame}_!(A)(B, C) := \text{wstr}(\Pi_A(B \Rightarrow C))$
- identities as in example;
- composition: usual AJM-composition;
- change of base: usual AJM-composition (works out!).

As we don’t have (additive) $\Sigma$-types, this is not a model of DTT!

Theorem

Formally add them: get model $\mathbf{CtxtGame}_!$ of DTT with $\Sigma\Pi\Id$ (the last through intersection) and fin. inductive type families!
Place in intensionality spectrum Id-types:

<table>
<thead>
<tr>
<th></th>
<th>Domains</th>
<th>HoTT</th>
<th>Games</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failure of Equality Reflection</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Streicher Intensionality Criteria (I1) and (I2)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Streicher Intensionality Criterion (I3)</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Failure of Function Extensionality (FunExt)</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Failure of Uniqueness of Identity Proofs (UIP)</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
</tbody>
</table>

Discrete ground types (0-types), but function hierarchy generates (open) propositional identities: observational equivalences.
Summarising, we have

Theorem (Soundness and Faithfulness)

We have a faithful model of DTT with $\Sigma$-, $\Pi$-, $\text{Id}$-types and finite inductive type families: faithful functor

$$
\text{Syntax}_{DTT} \xrightarrow{\llbracket - \rrbracket} \text{CtxtGame}!.
$$
Summarising, we have

**Theorem (Soundness and Faithfulness)**

*We have a faithful model of DTT with $\Sigma$-, $\Pi$-, Id-types and finite inductive type families: faithful functor*

\[
\text{Syntax}_{DTT} \xrightarrow{\llbracket-\rrbracket} \text{CtxtGame}!.
\]

Actually, the model has strong completeness properties.

**Theorem (Full Completeness)**

*This interpretation is full when restricted to the types of the form $A$ or $\Pi_A \text{Id}_B(f, g)$ with $A$ and $B$ built without Id-types:*

\[
\text{Syntax}_{DTT}^{\text{Restricted Types}} \xrightarrow{\llbracket-\rrbracket} \text{CtxtGame}! \quad \text{full.}
\]
Future Work

Ultimate goal: intensional, computational analysis of HoTT.

- game semantics of higher inductive types / quotient types;
- examining function extensionality and univalence;
- universes and a more intensional notion of type family;
- infinite inductive type families and their definability results;
- examining completeness properties of the model for the complete type hierarchy, including Id-types;
- constructing models of DTT with side effects.
In particular, for the first item:

\[
\begin{align*}
\text{HtpyGame} & \quad \infty - \text{Gpd} \times \text{Set} \quad \text{Game} \\
& \quad \text{collapsing time-like identity,} \\
& \quad \text{a.k.a. extensional collapse} \\
& \quad \infty - \text{Gpd} \quad \text{collapsing space-like identity,} \\
& \quad \text{a.k.a. 0-truncation} \\
& \quad X \quad ||X||_0.
\end{align*}
\]