Types, Event Structures and the $\pi$-Calculus

Daniele Varacca, Nobuko Yoshida

Imperial College London

Sussex, November 30th 2005
Historical perspective

An unfair and myopic view of the last 40 years
Historical perspective

An unfair and myopic view of the last 40 years

Petri [’60]

Petri nets
An unfair and myopic view of the last 40 years

Petri ['60]  Scott and Strachey ['70]

Denotational semantics - Domain theory
An unfair and myopic view of the last 40 years

Petri ['60] → Scott and Strachey ['70] → Nielsen, Plotkin and Winskel ['80]

Event structures
Historical perspective

An unfair and myopic view of the last 40 years

- Petri ['60]
- Scott and Strachey ['70]
- Park and Milner ['80]
- Nielsen, Plotkin and Winskel ['80]

Transition systems and bisimulation
An unfair and myopic view of the last 40 years

- Petri ['60]
- Scott and Strachey ['70]
- Park and Milner ['80]
- Nielsen, Plotkin and Winskel ['80]
- Berry and Boudol ['90]

Reduction semantics
Historical perspective

An unfair and myopic view of the last 40 years

- Petri ['60]
- Scott and Strachey ['70]
- Park and Milner ['80]
- Nielsen, Plotkin, and Winskel ['80]
- Berry and Boudol ['90]
- Girard ['80]

Linear logic
An unfair and myopic view of the last 40 years

Petri ['60]

Park and Milner ['80]

Scott and Strachey ['70]

Nielsen, Plotkin and Winskel ['80]

Girard ['80]

Blass et al. ['90]

Game semantics
Historical perspective

An unfair and myopic view of the last 40 years

Petri [’60]

Park and Milner [’80]

Berry and Boudol [’90]

Honda, Berger and Yoshida [’00]

Scott and Strachey [’70]

Nielsen, Plotkin and Winskel [’80]

Girard [’80]

Blass et al. [’90]

Linearly typed $\pi$ calculus
Historical perspective

An unfair and myopic view of the last 40 years

Petri ['60]

Park and Milner ['80]

Berry and Boudol ['90]

Honda, Berger and Yoshida ['00]

Scott and Strachey ['70]

Nielsen, Plotkin and Winskel ['80]

Girard ['80]

Blass et al. ['90]

Melliès ['00]

True concurrent games

Varacca, Yoshida

Types, Event Structures and the $\pi$-Calculus
Historical perspective

An unfair and myopic view of the last 40 years

Petri ['60]

Park and Milner ['80]

Berry and Boudol ['90]

Honda, Berger and Yoshida ['00]

Scott and Strachey ['70]

Nielsen, Plotkin and Winskel ['80]

Girard ['80]

Blass et al. ['90]

Melliès ['00]

This talk

Types, Event Structures and the $\pi$-Calculus
We all know what the $\pi$-calculus is

$$x(\tilde{y}).P \mid \bar{x}\langle\tilde{z}\rangle.Q \rightarrow P\{\tilde{z}/\tilde{y}\} \mid Q$$
We all know what the $\pi$-calculus is

$$x(\tilde{y}).P \mid \overline{x}(\tilde{z}).Q \longrightarrow P\{\tilde{z}/\tilde{y}\} \mid Q$$

We consider a restricted version:
bound output only ("internal" mobility)
We all know what the $\pi$-calculus is

$$x(\tilde{y}).P \parallel \overline{x}(\tilde{y}).Q \longrightarrow (\nu \tilde{y})(P \parallel Q)$$

We consider a restricted version: bound output only ("internal" mobility)
We all know what the $\pi$-calculus is

$$x(\tilde{y}).P \mid \overline{x}(\tilde{y}).Q \longrightarrow (\nu \tilde{y})(P \mid Q)$$

We consider a restricted version:
bound output only ("internal" mobility)

A linear type discipline:

(A) for each linear name there are a unique input and a unique output

(B) for each replicated name there is a unique stateless replicated input with zero or more dual outputs
Linearly typed $\pi$ is confluent
Road Map

1. Event Structures
   - Confusion Freeness
   - Conflict Freeness

2. Types
   - Syntax and Semantics
   - Typed Event Structures

3. Semantics of $\pi$
   - Syntax
   - Event Structure Semantics
   - Correspondence
Road Map

1. Event Structures
   - Confusion Freeness
   - Conflict Freeness

2. Types
   - Syntax and Semantics
   - Typed Event Structures

3. Semantics of $\pi$
   - Syntax
   - Event Structure Semantics
   - Correspondence
True concurrency

Standard “interleaving” semantics

- reduces parallelism to nondeterministic interleaving ("expansion law")
- Labelled transition systems, reduction semantics
True concurrency

Standard “interleaving” semantics
- reduces parallelism to nondeterministic interleaving (“expansion law”)
- Labelled transition systems, reduction semantics

“True concurrent” models
- Represent explicitly causality, conflict, independence
- Petri nets, Mazurkiewicz traces, event structures
Event structures

An event structure is a partial order \( \langle E, \leq \rangle \) together with a conflict relation \( \perp \)

- order represents causal dependency
- conflict is irreflexive and symmetric
- conflict is “hereditary”:

\[
e_1 \perp e \text{ and } e_1 \leq e_2 \text{ implies } e_2 \perp e
\]

A conflict is immediate if it is not inherited from another conflict
Event structures

Example

\[ \begin{align*}
  &d \\
  &\downarrow \quad \downarrow \\
  &b \sim \sim \sim \sim \sim c \\
  &\quad \downarrow \quad \downarrow \\
  &a
\end{align*} \]
Event structures

Example

Events can also be labelled: \( \lambda : E \rightarrow L \)
Event structures

Example

\[ \gamma_1 \quad \beta_1 \sim \beta_2 \quad \gamma_2 \]

Events can also be labelled: \( \lambda : E \to L \)
Operators on event structures

Prefixing $\alpha . \mathcal{E}$

\[
\begin{array}{c}
\gamma_1 \\
\beta_1 \\
\end{array}
\quad \sim 
\quad 
\begin{array}{c}
\gamma_2 \\
\beta_2 \\
\end{array}
\]
Operators on event structures

Prefixing $\alpha.\mathcal{E}$
Operators on event structures

Prefix sum $\sum_{i \in I} \alpha_i \cdot \mathcal{E}_i$

$\gamma_1$

$\beta_1$

$\gamma_2$

$\beta_2$
Operators on event structures

Prefix sum $\sum_{i \in I} \alpha_i \cdot \mathcal{E}_i$
Parallel composition $\mathcal{E}_1 \parallel \mathcal{E}_2$

\[ \gamma_1 \quad \gamma_2 \]

\[ \beta \quad \overline{\beta} \]
Operators on event structures

Parallel composition $\mathcal{E}_1 \parallel \mathcal{E}_2$

A complex construction involving synchronisation
Consider

- $\mathcal{E} = \langle E, \leq, \vdash, \lambda \rangle$, a labelled event structure
- $e$, one of its minimal events

We define $\mathcal{E} \mid e$ as $\mathcal{E}$ minus event $e$, and minus all events that are in conflict with $e$

We can then generate a labelled transition system as follows: if $\lambda(e) = \beta$, then

$$
\mathcal{E} \xrightarrow{\beta} \mathcal{E} \mid e
$$
Event structures and transition systems

Example

An event structure $E$
Example

Eliminate a minimal event $e$ (labelled by $\beta_2$)
Event structures and transition systems

Example

\[
\begin{array}{c}
\gamma_1 \\
\gamma_2 \sim \gamma_3 \\
\beta_1 \sim \sim \sim \sim \sim \\
\end{array}
\]

Eliminate a minimal event \(e\) (labelled by \(\beta_2\))
Event structures and transition systems

Example

\[
\begin{align*}
\gamma_1 & \quad \gamma_2 \sim \sim \gamma_3 \\
\beta_1 & \sim \sim \sim
\end{align*}
\]

And every event in conflict with it
Example

And every event in conflict with it
Event structures and transition systems

Example

\[ \begin{array}{c}
\gamma_1 \\
\beta_1 \\
\gamma_2 \\
\beta_2 \\
\gamma_3 \\
\end{array} \xrightarrow[\beta_2]{\gamma_2 \sim \gamma_3} \begin{array}{c}
\gamma_2 \\
\gamma_3 \\
\end{array} \]

\[ E \xrightarrow[\beta_2]{e} E \]
An event structure is confusion-free when
- “reflexive” immediate conflict is an equivalence
- any two events in immediate conflict have the same predecessors

The equivalence classes are the cells
Cells represent local choices
Examples

Confusion Free
Examples

Confusion!

Varacca, Yoshida
Types, Event Structures and the $\pi$-Calculus
Examples

Confusion!
Confusion arises from synchronisation
Consider $(\bar{a} \mid a)$
The event structure for this is

\[
\begin{array}{c}
\bar{a} \\
\tau \\
a
\end{array}
\]

Confusion - the choice is not local
Where confusion arises

Confusion arises from synchronisation
Consider \((\overline{\alpha} \mid \alpha)\)
The event structure for this is

\[
\overline{\alpha} \sim \tau \sim \alpha
\]

Confusion - the choice is not local

Issue: how to perform synchronisation without introducing confusion
Conflict freeness

When the conflict relation is empty, the corresponding transition system is confluent
A special case of confusion freeness
Conflict freeness

When the conflict relation is empty, the corresponding transition system is confluent
A special case of confusion freeness

Idea: give a conflict free event structure semantics to the linear $\pi$-calculus
Issues:
  - difficult to handle name generation
  - hidden conflicts appear
The post office

Example:

- Stateless replicated resource: post office $!a.P$
- Clients: customers $\overline{a}.C$

Every customer wants to send a letter $a$
The post office

The process $\bar{a}.D \parallel \bar{a}.N \parallel !a.P$ is confluent

\[
\begin{array}{c}
\bar{a}.D \parallel \bar{a}.N \parallel !a.P \\
D \parallel P \parallel \bar{a}.N \parallel !a.P \\
N \parallel P \parallel \bar{a}.D \parallel !a.P \\
N \parallel P \parallel D \parallel P \parallel !a.P
\end{array}
\]
Situation 1: two customers, one till
A conflict to resolve: who goes first?
Eventually, it does not matter, but the two events are not independent
The post office

Situation 1: two customers, one till
A conflict to resolve: who goes first?
Eventually, it does not matter, but the two events are not independent

Situation 2: two customers, infinitely many identical tills
if the two customers want to go to the same till, there is a conflict
The post office

Situation 1: two customers, one till
A conflict to resolve: who goes first?
Eventually, it does not matter, but the two events are not independent

Situation 2: two customers, infinitely many identical tills
if the two customers want to go to the same till, there is a conflict

Situation 3: one customer, infinitely many identical tills
the customer has to choose which till to go to
The post office

Solution: no conflict arises if every possible customer is assigned a specific till \textit{in advance}.
Road Map

1. Event Structures
   - Confusion Freeness
   - Conflict Freeness

2. Types
   - Syntax and Semantics
   - Typed Event Structures

3. Semantics of $\pi$
   - Syntax
   - Event Structure Semantics
   - Correspondence
Types for event structures

\[ \Gamma, \Delta ::= y_1 : \sigma_1, \ldots, y_n : \sigma_n \quad \text{type environment} \]

\[ \tau, \sigma ::= \land_{i \in I} \Gamma_i \quad \text{branching} \]

\[ \text{subject to } \bigg| \begin{array}{l}
\lor_{i \in I} \Gamma_i \\
\bigoplus_{i \in I} \Gamma_i \\
\bigotimes_{i \in I} \Gamma_i \\
\biguplus_{i \in I} \Gamma_i \\
\end{array} \quad \text{selection, server, client, closed type} \]

Linearity condition: no name appears more than once
Composing environments

Notion of **matching** of types

- A branching type $\&$ matches the dual selection types $\oplus$, and the residual type is $\uparrow$
Notion of matching of types

- A branching type $\&$ matches the dual selection types $\oplus$, and the residual type is $\uparrow$

- A server type $\otimes$ matches a client type $\odot$ if all requests correspond to an available resource. The residual is again a server type $\otimes$ that records which resources are still available.
Composing environments

Notion of matching of types

- A branching type $\&$ matches the dual selection types $\oplus$, and the residual type is $\uparrow$

- A server type $\otimes$ matches a client type $\bowtie$ if all requests correspond to an available resource. The residual is again a server type $\otimes$ that records which resources are still available

- Two environments $\Gamma_1, \Gamma_2$ be composed if the types of the common names match
Composing environments

Notion of matching of types

- A branching type \& matches the dual selection types \( \oplus \), and the residual type is \( \uparrow \)

- A server type \( \otimes \) matches a client type \( \bowtie \) if all requests correspond to an available resource. The residual is again a server type \( \otimes \) that records which resources are still available.

- Two environments \( \Gamma_1, \Gamma_2 \) be composed if the types of the common names match

- Such names are given the residual type by the resulting environment \( \Gamma_1 \odot \Gamma_2 \)
Type environments

Example

\[ \tau_1 = \&_{i \in \{1,2\}} (x_i : \bigvee_{j \in J}) \]
\[ \tau_2 = \bigvee_{i \in \{1,2\}} (x_i : \&_{j \in J}) \]
\[ \sigma_1 = \forall_{i \in \{1\}} (y_i : \uparrow) \]
\[ \sigma_2 = \bigotimes_{i \in \{1,2\}} (y_i : \uparrow) \]
Type environments

Example

- $\tau_1 = \&_{i \in \{1,2\}} (x_i : \bigoplus_{j \in J})$
- $\tau_2 = \bigoplus_{i \in \{1,2\}} (x_i : \&_{j \in J})$
- $\sigma_1 = \&_{i \in \{1\}} (y_i : \uparrow)$
- $\sigma_2 = \bigotimes_{i \in \{1,2\}} (y_i : \uparrow)$

- $\tau_1$ matches $\tau_2$
- the residual type is $\uparrow$
Type environments

Example

- $\tau_1 = \&_{i \in \{1,2\}} (x_i : \bigoplus_{j \in J})$
- $\tau_2 = \bigoplus_{i \in \{1,2\}} (x_i : \&_{j \in J})$
- $\sigma_1 = \forall_{i \in \{1\}} (y_i : \uparrow)$
- $\sigma_2 = \bigotimes_{i \in \{1,2\}} (y_i : \uparrow)$

- $\sigma_1$ matches $\sigma_2$
- the residual type is $\bigotimes_{i \in \{2\}} (y_i : \uparrow)$
Type environments

Example

- $\tau_1 = \land_{i \in \{1,2\}} (x_i : \bigoplus_{j \in J})$
- $\tau_2 = \bigoplus_{i \in \{1,2\}} (x_i : \land_{j \in J})$
- $\sigma_1 = \bigwedge_{i \in \{1\}} (y_i : \uparrow)$
- $\sigma_2 = \bigotimes_{i \in \{1,2\}} (y_i : \uparrow)$
- $\Gamma_1 = a : \tau_1, b : \sigma_1,$
- $\Gamma_2 = a : \tau_2, b : \sigma_2$
Type environments

Example

\[ \tau_1 = \&_{i \in \{1, 2\}}(x_i : \bigoplus_{j \in J}) \]
\[ \tau_2 = \bigoplus_{i \in \{1, 2\}}(x_i : \&_{j \in J}) \]
\[ \sigma_1 = \bigwedge_{i \in \{1\}}(y_i : \downarrow) \]
\[ \sigma_2 = \bigotimes_{i \in \{1, 2\}}(y_i : \downarrow) \]
\[ \Gamma_1 = a : \tau_1, b : \sigma_1, \]
\[ \Gamma_2 = a : \tau_2, b : \sigma_2 \]
\[ \Gamma_1 \odot \Gamma_2 = a : \downarrow, b : \bigotimes_{i \in \{2\}}(y_i : \downarrow) \]
Labelled event structures

Labels:

\[ \alpha, \beta ::= x \mathbin{\text{in}}_i \langle \check{y} \rangle \quad \text{branching} \quad \tau ::= (x, \check{x}) \mathbin{\text{in}}_i \langle \check{y} \rangle \]

\[ \quad \mid \bar{x} \mathbin{\text{in}}_i \langle \check{y} \rangle \quad \text{selection} \quad \mid (x, \check{x}) \langle \check{y} \rangle \]

\[ \quad \mid x \langle \check{y} \rangle \quad \text{server} \quad \mid (x, \bar{x}) \langle \check{y} \rangle \]

\[ \quad \mid \bar{x} \langle \check{y} \rangle \quad \text{client} \quad \mid x \langle \check{y} \rangle \langle \check{y} \rangle \]

\[ \quad \mid \tau \quad \text{synchronisation} \]
An event structure $\mathcal{E}$ is well typed in $\Gamma$ if

- $\mathcal{E}$ is confusion free
- cells are partitioned in branching, selection, client, server and synchronisation cells
- all the non-synchronisation events of are represented in $\Gamma$
- causality in $\mathcal{E}$ refines name causality of $\Gamma$

Technically: via morphisms of the category of event structures
A typed event structure

\[
\begin{align*}
\varepsilon \triangleright a &: \: \&_{i \in \{1,2\}} (x_i : \&_{k \in \{1\}}) \\
& \quad b &: \: \otimes_{j \in \{1\}} (z_j : \not\exists_{l \in \{1\}})
\end{align*}
\]
Typed event structures

Properties

- Typed event structures are confusion free (by definition)
- Prefixing, prefixed sum and parallel composition preserve typing

In particular parallel composition of typed event structures is confusion free
Typed event structures

Properties
- Typed event structures are confusion free (by definition)
- Prefixing, prefixed sum and parallel composition preserve typing

In particular parallel composition of typed event structures is confusion free

Theorem [Parallel composition]
If $E_1 \triangleright \Gamma_1$ and $E_2 \triangleright \Gamma_2$ and $\Gamma_1 \odot \Gamma_2$ is defined, then

$$(E_1 \parallel E_2) \setminus S \triangleright \Gamma_1 \odot \Gamma_2$$
Typed event structures

Properties

- Typed event structures are confusion free (by definition)
- Prefixing, prefixed sum and parallel composition preserve typing

In particular parallel composition of typed event structures is confusion free

**Theorem [Parallel composition]**

If $E_1 \triangleright \Gamma_1$ and $E_2 \triangleright \Gamma_2$ and $\Gamma_1 \circ \Gamma_2$ is defined, then

$$(E_1 \parallel E_2) \setminus S \triangleright \Gamma_1 \circ \Gamma_2$$

(S is the set of names not allowed by the new environment)
Typed event structures

When branching and selection types are trivial:

- Typed event structures are conflict free
- Prefixing, and parallel composition preserve typing

In particular parallel composition of typed event structures is conflict free
Road Map

1. Event Structures
   - Confusion Freeness
   - Conflict Freeness

2. Types
   - Syntax and Semantics
   - Typed Event Structures

3. Semantics of $\pi$
   - Syntax
   - Event Structure Semantics
   - Correspondence
The syntax

$\pi$ processes

$$P ::= x \& \prod_{i \in I} \text{in}_i(\tilde{y}_i).P_i \quad \text{branching}$$
$$\quad \mid \bar{x}\text{in}_j(\tilde{y}).P \quad \text{selection}$$
$$\mid !x(\tilde{y}).P \quad \text{server}$$
$$\mid \bar{x}(\tilde{y}).P \quad \text{client}$$
$$\mid P \mid Q \quad \text{parallel}$$
$$\mid (\nu x)P \quad \text{restriction}$$
$$\mid 0 \quad \text{inaction}$$
The syntax

\( \pi \) processes

\[
P ::= \begin{align*}
& x \land_{i \in I} i_n_i(\tilde{y}_i).P_i & \text{branching} \\
| & \bar{x} \bigoplus_{i \in I} i_n_i(\tilde{y}_i).P_i & \text{selection} \\
| & !x(\tilde{y}).P & \text{server} \\
| & \bar{x}(\tilde{y}).P & \text{client} \\
| & P | Q & \text{parallel} \\
| & (\nu x)P & \text{restriction} \\
| & 0 & \text{inaction}
\end{align*}
\]
The types

$\pi$ types

$\sigma ::= \&_{i \in l} (\tilde{\sigma}_i) \downarrow$ branching

| $\bigoplus_{i \in l} (\tilde{\sigma}_i) \uparrow$ selection

| $(\tilde{\sigma})!$ server

| $(\tilde{\sigma})?$ client

$\tau ::= \sigma \mid \uparrow$

Environments compose in a similar way as event structure environments
<table>
<thead>
<tr>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;bar;a.b</td>
</tr>
</tbody>
</table>

This is not typable as $a$ appears twice as output
Examples

\[ b \bar{a} \mid c \bar{b} \mid a.(\bar{c} \mid \bar{e}) \]

This is typable since each channel appears at most once as input and output
Examples

\[ \overline{a}(b \oplus c) | a(\overline{d} \& \overline{e}) \]

This process is typable, and contains nondeterminism:

\[ Q_3 \rightarrow (b | \overline{d}) \]

\[ Q_3 \rightarrow (c | \overline{e}) \]

Varacca, Yoshida
Types, Event Structures and the \( \pi \)-Calculus
Examples

\[ !b.\overline{a} | !b.\overline{c} \]

This is **not** typable as there are two different servers associated with \( b \).
Examples

\[ !b.\overline{a} \mid \overline{b} \mid !c.\overline{b} \]

This is typable: the two clients on \( b \) are associated to a unique server
Operational semantics

- As usual $P \triangleright \Gamma \xrightarrow{\beta} P' \triangleright \Gamma'$
- The transition must be allowed by the environment
- (The environment performs implicit restrictions)
The semantics has the form $\llbracket P \triangleright \Gamma \rrbracket^\Delta$, where $\Delta$ is an event structure environment.
The semantics has the form $[[P \triangleright \Gamma]]^\Delta$, where $\Delta$ is an event structure environment.

$\Delta$ fixes a choice of the newly generated names.
The semantics has the form $\sem{\mathcal{P} \triangleright \Gamma}^\Delta$, where $\Delta$ is an event structure environment.

$\Delta$ fixes a choice of the newly generated names.

$\Delta$ assigns each client a specific instance of its server.
Event structure semantics of $\pi$

$$\sem{\sum_{i \in I} \text{in}_i(y_i) \cdot P_i \triangleright \Gamma, a : \bigoplus_{i \in I}(\tau_i)}^{\Delta, a : \bigoplus_{i \in I} z_i : \hat{\tau}_i} = \sum_{i \in I} \sem{\text{in}_i(z_i) \cdot [P_i[z_i/y_i] \triangleright \Gamma, z_i : \tau_i]}^{\Delta, z_i : \hat{\tau}_i}$$
Event structure semantics of $\pi$

$$
\left[!a(y).P \triangleright \Gamma, a : (\tau)!\right]^{\Delta,a: \bigotimes_{k \in K} (y^k : \hat{\tau}^k)}
= \\
\|_{k \in K} a \langle y^k \rangle . [P[y^k / y] \triangleright \Gamma [y^k / y]]^{\Delta, y^k : \hat{\tau}^k}
$$
Event structure semantics of $\pi$

$$
\begin{align*}
\sem{P_1 | P_2 \triangleright \Gamma_1 \odot \Gamma_2}^{\Delta_1 \odot \Delta_2} &= \\
&= (\sem{P_1 \triangleright \Gamma_1}^{\Delta_1} \parallel \sem{P_2 \triangleright \Gamma_2}^{\Delta_2}) \setminus S
\end{align*}
$$
The interpretation functions are partial functions: for the wrong choice of $\Delta_1, \Delta_2$, the interpretation of the parallel composition could be undefined, because $\Delta_1 \circ \Delta_2$ may be undefined.
The interpretation functions are partial functions: for the wrong choice of $\Delta_1, \Delta_2$, the interpretation of the parallel composition could be undefined, because $\Delta_1 \odot \Delta_2$ may be undefined.

It is always possible to find suitable $\Delta_1, \Delta_2$. We perform $\alpha$-conversion “at compile time”.
The interpretation functions are partial functions: for the wrong choice of $\Delta_1, \Delta_2$, the interpretation of the parallel composition could be undefined, because $\Delta_1 \circ \Delta_2$ may be undefined.

Theorem: [Event structure semantics]
For every judgement $P \triangleright \Gamma$ in the $\pi$-calculus, there exists an environment $\Delta$ such that $\llbracket P \triangleright \Gamma \rrbracket^\Delta$ is defined
Also: $\llbracket P \triangleright \Gamma \rrbracket^\Delta \triangleright \Delta$
Correspondence between transition system and event structure:

**Theorem: [Operational correspondence]**

If $P \triangleright \Gamma \xrightarrow{\beta} P' \triangleright \Gamma'$, then $\llbracket P \triangleright \Gamma \rrbracket^\Delta \xrightarrow{\beta} \cong \llbracket P' \triangleright \Gamma' \rrbracket^{\Delta'}$
Correspondence between transition system and event structure:

**Theorem: [Operational correspondence]**

If $P \triangleright \Gamma \xrightarrow{\beta} P' \triangleright \Gamma'$, then $\llbracket P \triangleright \Gamma \rrbracket^\Delta \xrightarrow{\beta} \simeq \llbracket P' \triangleright \Gamma' \rrbracket^\Delta'$

If $\llbracket P \triangleright \Gamma \rrbracket^\Delta \xrightarrow{\beta} \mathcal{E}'$, then there exists $P'$ such that $P \triangleright \Gamma \xrightarrow{\beta} P' \triangleright \Gamma'$ and $\llbracket P' \triangleright \Gamma' \rrbracket^\Delta' \simeq \mathcal{E}'$
What we have done

- First typing system for event structures
- Typing system for true concurrent behavioural properties
- First \textit{explicit} event structure semantics of $\pi$
Summary

What we have done
- First typing system for event structures
- Typing system for true concurrent behavioural properties
- First *explicit* event structure semantics of $\pi$

What we will do
- Probabilistic event structures
- Connections with true concurrent games
- Connections with Beffara’s thesis