Determinisme, Structures d'événements et le π-Calcul

Daniele Varacca

with Nobuko Yoshida

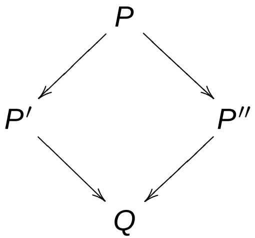
PPS

Sophia, Parsec - 2/2/2007

Determinism

What is determinism

- For functions: only one result
- For reactive systems: confluence

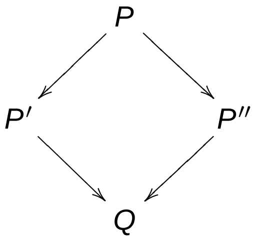


Only one maximal execution, up to order

Determinism

What is determinism

- For functions: only one result
- For reactive systems: confluence



Only one maximal execution, up to order
Some fairness assumptions may be necessary

Probabilistic determinism

What is probabilistic determinism

- For functions: only one probability distribution
- For reactive systems?
  Only one maximal execution, up to order??

Road Map

1. Typed π
   - Syntax
2. Event Structures
   - Conflict Freeness
   - Semantics
   - Correspondence
3. Probabilistic case
   - Syntax
   - Probabilistic event structures

Road Map

Typed π-calculus

We all know what the π-calculus is

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

Typed π-calculus

We all know what the π-calculus is

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

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

Typed π-calculus

We all know what the π-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)

The syntax

π processes

P	$::=$	$x \&_{i \in I} \mathtt{in}_i(\tilde{y}_i).P_i$	branching
	$\mid$	$\overline{x}\mathtt{in}_j(\tilde{y}).P$	selection
	$\mid$	$!x(\tilde{y}).P$	server
	$\mid$	$\overline{x}(\tilde{y}).P$	client
	$\mid$	$P \mid Q$	parallel
	$\mid$	$(\nu x)P$	restriction
	$\mid$	$\mathbf{0}$	inaction

Typed π-calculus

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

This discipline guarantees confluence (determinsim)

Examples

$$\overline{a}.b \mid \overline{a}.c \mid a$$

This is not typable as *a* appears twice as output

Examples

$$b.\overline{a} \mid c.\overline{b} \mid a.(\overline{c} \mid \overline{e})$$

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

Examples

$$!\,b.\overline{a} \mid !\,b.\overline{c}$$

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

Examples

$$P = \overline{a}\mathtt{in}_1.b \mid a[\mathtt{in}_1\overline{d} \,\&\, \mathtt{in}_2\overline{e}]$$

This process is typable, and performs a choice:

$$P \longrightarrow (b \mid \overline{d})$$

Road Map

1. Typed π
   - Syntax
2. Event Structures
   - Conflict Freeness
   - Semantics
   - Correspondence
3. Probabilistic case
   - Syntax
   - Probabilistic event structures

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 $\smile$

- order represents causal dependency
- conflict is irreflexive an symmetric
- conflict is "hereditary":

$$e_1 \smile e \text{ and } e_1 \leq e_2 \text{ implies } e_2 \smile e$$

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

Configurations

A notion of run

A configuration is a set x of events

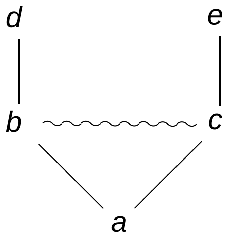
- justified: $e \in x, e' \leq e \implies e' \in x$
- conflict-free: $e, e' \in x \implies \neg e \smile e'$

Example:

$$[e] := \{e' \mid e' \leq e\}$$

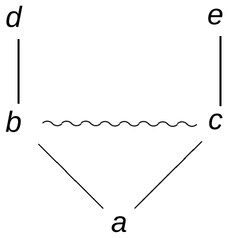
Event structures

Example



Event structures

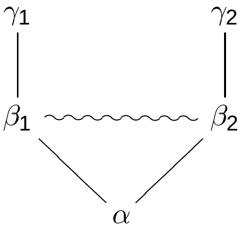
Example



Events can also be labelled: $\lambda : E \longrightarrow L$

Event structures

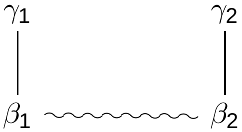
Example



Events can also be labelled: $\lambda : E \longrightarrow L$

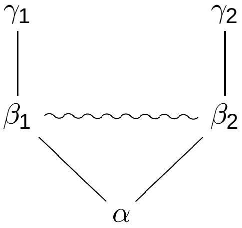
Operators on event structures

Prefixing $\alpha.\mathcal{E}$



Operators on event structures

Prefixing $\alpha.\mathcal{E}$

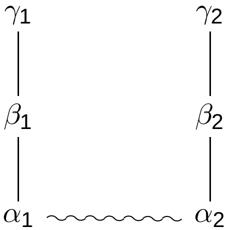


Operators on event structures

Prefixed sum $\sum_{i \in I} \alpha_i.\mathcal{E}_i$

Operators on event structures

Prefixed sum $\sum_{i \in I} \alpha_i.\mathcal{E}_i$

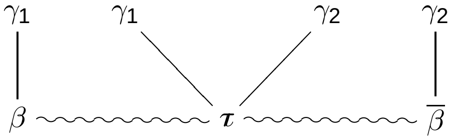


Operators on event structures

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

Operators on event structures

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



A complex construction involving synchronisation

Event structures and transition systems

Consider

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

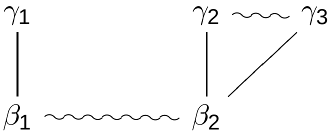
We define $\mathcal{E}\lfloor e$ as $\mathcal{E}$ minus event e, and minus all events that are in conflict with e

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

$$\mathcal{E} \xrightarrow{\beta} \mathcal{E}\lfloor e$$

Event structures and transition systems

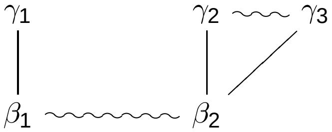
Example



An event structure $\mathcal{E}$

Event structures and transition systems

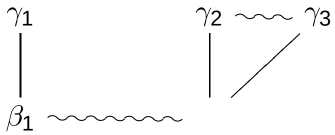
Example



Eliminate a minimal event e (labelled by β_2)

Event structures and transition systems

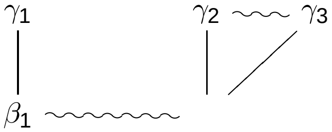
Example



Eliminate a minimal event *e* (labelled by β_2)

Event structures and transition systems

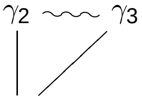
Example



And every event in conflict with it

Event structures and transition systems

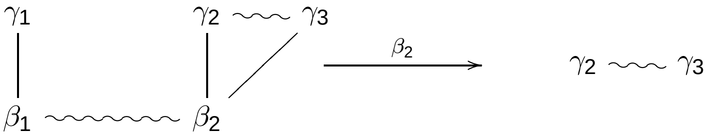
Example



And every event in conflict with it

Event structures and transition systems

Example



$$\mathcal{E} \xrightarrow{\beta_2} \mathcal{E}\lfloor e$$

Conflict freeness

When the conflict relation is empty, the corresponding transition system is confluent

Conflict freeness

When the conflict relation is empty, the corresponding transition system is confluent

Idea: give a conflict free event structure semantics to the linear π-calculus

Conflict freeness

When the conflict relation is empty, the corresponding transition system is confluent

Idea: give a conflict free event structure semantics to the linear π-calculus

Issues:

- perform synchronisation without introducing conflict
- 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 $\overline{a}.D \mid \overline{a}.N \mid !a.P$ is confluent

$$\overline{a}.D \mid \overline{a}.N \mid !a.P$$

$$D \mid P \mid \overline{a}.N \mid !a.P \qquad\qquad N \mid P \mid \overline{a}.D \mid !a.P$$

$$N \mid P \mid D \mid P \mid !a.P$$

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

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 identcal 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 spefic till in advance

Event structure semantics of π

The semantics has the form $[\![P]\!]^{\Delta}$, where Δ assigns each client a specific instance of its server

Event structure semantics of π

The semantics has the form $[\![P]\!]^{\Delta}$, where Δ assigns each client a specific instance of its server

The interpretation functions are partial functions: for the wrong choice of Δ, a post office customer would not find her till.

It is always possible to find suitable Δ: we perform α-conversion "at compile time"

Event structure semantics of π

The semantics has the form $[\![P]\!]^{\Delta}$, where Δ assigns each client a specific instance of its server

The interpretation functions are partial functions: for the wrong choice of Δ, a post office customer would not find her till.

It is always possible to find suitable Δ: we perform α-conversion "at compile time"

Theorem:
For every process P, there exists a choice Δ such that $[\![P]\!]^{\Delta}$ is defined

Correspondence

Correspondence between transition system and event structure:

Theorem: [Operational correspondence]

If $P \xrightarrow{\beta} P'$, then $[\![P]\!]^{\Delta} \xrightarrow{\beta} \cong [\![P']\!]^{\Delta'}$

Correspondence

Correspondence between transition system and event structure:

Theorem: [Operational correspondence]

If $P \xrightarrow{\beta} P'$, then $[\![P]\!]^{\Delta} \xrightarrow{\beta} \cong [\![P']\!]^{\Delta'}$

If $[\![P]\!]^{\Delta} \xrightarrow{\beta} \mathcal{E}'$, then there exists P' such that $P \xrightarrow{\beta} P'$ and $[\![P']\!]^{\Delta'} \cong \mathcal{E}'$

Road Map

The syntax

π processes

P	$::=$	$x \,\&_{i \in I} \, \mathtt{in}_i(\tilde{y}_i).P_i$	branching
	$\mid$	$\overline{x}\mathtt{in}_j(\tilde{y}).P$	selection
	$\mid$	$!x(\tilde{y}).P$	server
	$\mid$	$\overline{x}(\tilde{y}).P$	client
	$\mid$	$P \mid Q$	parallel
	$\mid$	$(\nu x)P$	restriction
	$\mid$	$\mathbf{0}$	inaction

The syntax

π processes

P	$::=$	$x \binampersand_{i \in I} \mathtt{in}_i(\tilde{y}_i).P_i$	branching
	$\mid$	$\overline{x} \bigoplus_{i \in I} p_i \mathtt{in}_i(\tilde{y}_i).P_i$	selection
	$\mid$	$!x(\tilde{y}).P$	server
	$\mid$	$\overline{x}(\tilde{y}).P$	client
	$\mid$	$P \mid Q$	parallel
	$\mid$	$(\nu x)P$	restriction
	$\mid$	$\mathbf{0}$	inaction

Typed π-calculus

The same 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

This discipline guarantees probabilistic confluence?

Example

$$P = \overline{a}[\mathtt{in}_1.b \ \oplus_p \ \mathtt{in}_2.c] \mid a[\mathtt{in}_1.\overline{d} \ \& \ \mathtt{in}_2.\overline{e}]$$

This process is typable, and performs a choice:

$$P \longrightarrow_p (b \mid \overline{d})$$

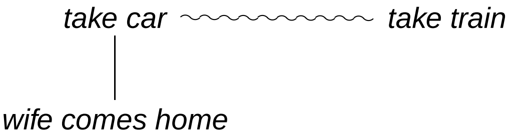
$$P \longrightarrow_{1-p} (c \mid \overline{e})$$

Probabilistic choice?

How to add probabilities to event structures?
Idea: resolve the immediate conflict by flipping a coin

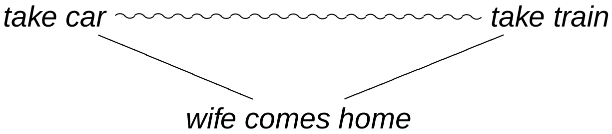
Coins resolve local choices
What does local mean?

Locality: Example



Non local!

Locality: Example



Local!

Confusion freeness

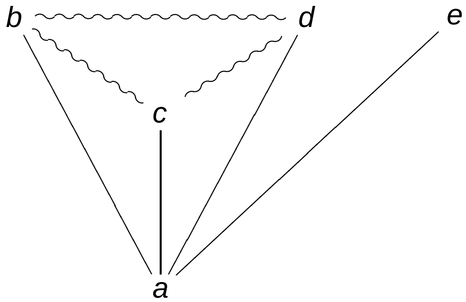
An event structure is confusion-free when

- “reflexive” immediate conflict is an equivalence
- any two events in immediate conflict have the same predecessors

The equivalence classes are the cells

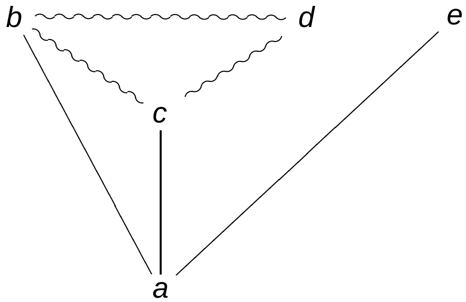
Cells represent local choices

Examples



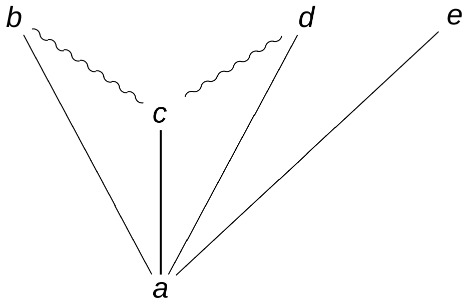
Confusion Free

Examples



Confusion!

Examples



Confusion!

Valuations on event structures

A local valuation on $\mathcal{E}$ associates to every cell a coin/die

It is a function $p : E \to [0,1]$ such that for every cell c

$$\sum_{e \in c} p(e) = 1$$

The weight $v_p(x)$ of a configuration x is the product of the probabilities of the events in x

Probabilistic runs

An conflict free event structure has only one maximal configurations (only one maximal run up to order)

Theorem: [Varacca-Völzer-Winskel]

For every local valuation p there exists a unique probability measure m_p on the set of maximal configurations such that

$$m_p(\uparrow x) = v_p(x)$$

"A probabilistic event structure has only one maximal run up to order"

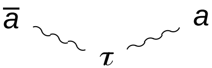
probabilistic determinism

Semantics of π

Confusion arises from synchronisation

Consider $(\overline{a} \mid a)$

The event structure for this is



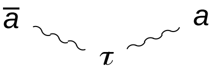
Confusion - the choice is not local

Semantics of π

Confusion arises from synchronisation

Consider $(\overline{a} \mid a)$

The event structure for this is

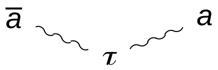


Confusion - the choice is not local

Issue: how to perform synchronisation without introducing confusion

Semantics of π

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



Confusion - the choice is not local

Issue: how to perform synchronisation without introducing confusion
Same machinery as for the conflict free case

Semantics of π

The semantics of π extends to the probabilistic case.

Only one probability distributions over maximal runs: probabilistic determinism.

Relations with interleaving semantics (Segala automata)

Related Work

- Concurrent games (Melliès, Faggian, Curien)
- Untyped π-calculus (with Silvia Crafa)
- Termination
- Encodings

Dessert

Historical perspective

An unfair and myopic view of the last 40 years

An unfair and myopic view of the last 40 years

Petri ['60]

Petri nets

Historical perspective

An unfair and myopic view of the last 40 years

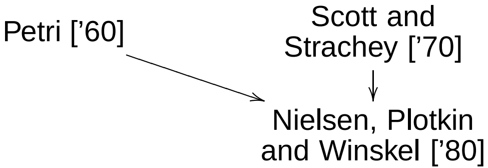
Petri ['60]

Scott and Strachey ['70]

Denotational semantics - Domain theory

Historical perspective

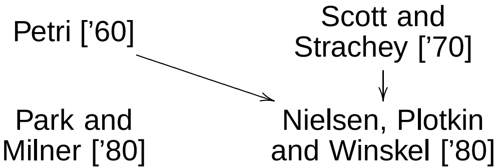
An unfair and myopic view of the last 40 years



Event structures

Historical perspective

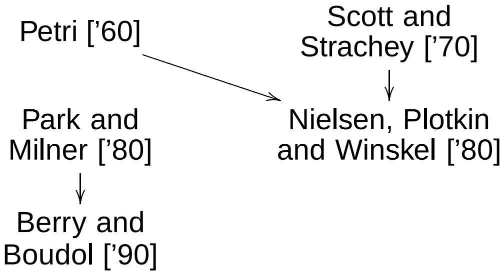
An unfair and myopic view of the last 40 years



Transition systems and bisimulation

Historical perspective

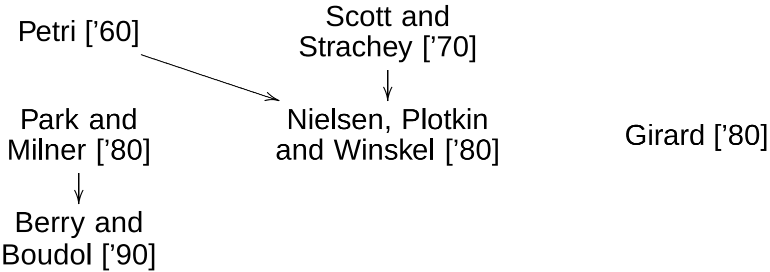
An unfair and myopic view of the last 40 years



Reduction semantics

Historical perspective

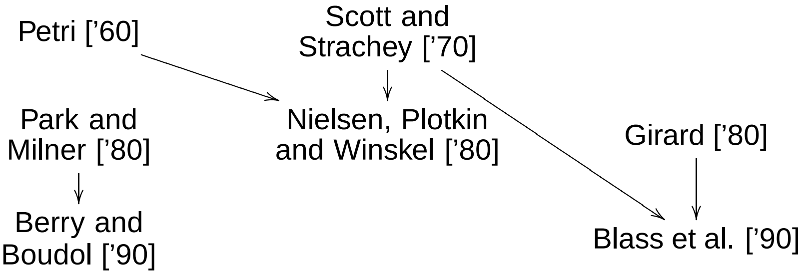
An unfair and myopic view of the last 40 years



Linear logic

Historical perspective

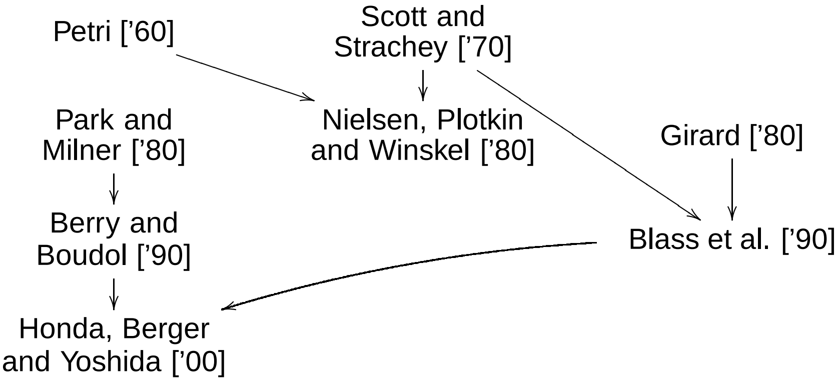
An unfair and myopic view of the last 40 years



Game semantics

Historical perspective

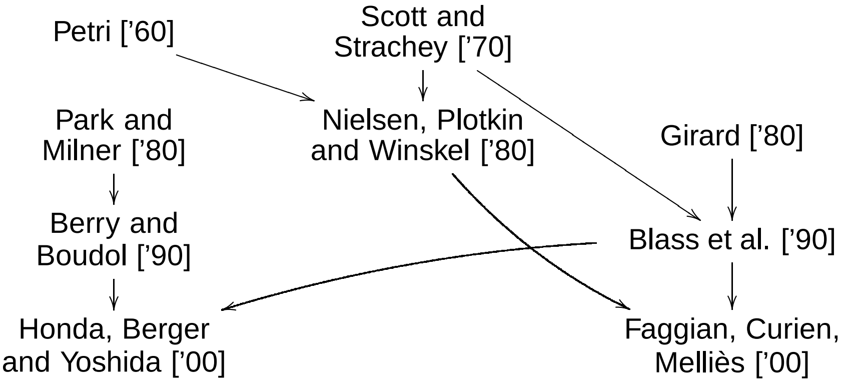
An unfair and myopic view of the last 40 years



Linearly typed π calculus

Historical perspective

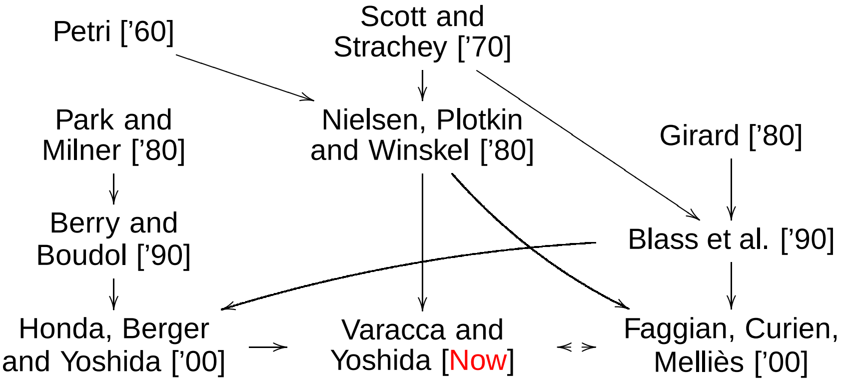
An unfair and myopic view of the last 40 years



True concurrent games

Historical perspective

An unfair and myopic view of the last 40 years



Event structures for π