Concurrent games with symmetry

Simon Castellan

May 10, 2014

Internship realized under the supervision of Glynn Winskel and Pierre Clairambault
Computer Laboratory, University of Cambridge.

Abstract

Last year, Glynn Winskel and Sylvain Rideau introduced *concurrent games* [?] as an attempt to provide new foundations for game semantics more suitable to concurrency and non-determinism. These foundations rely on event structures, one of the cornerstones of the theory of concurrency. However these concurrent games were *linear*, meaning that a single event could only appear once. This makes impossible to model functions using their arguments several times. (Thus you cannot build a model of λ-calculus using concurrent games). The goal of the internship was to add duplication through the notion of *symmetry* on event structures to solve this problem and to be able to model non-linear languages.

Contents

Introduction

Thanks I would like to first thank Glynn and Pierre for welcoming me in Cambridge, for their availability and the very rich scientific experience I had in Cambridge. I learned a lot on different subjects and met many interesting people. Thanks to Glynn, Pierre, Julian, Teresa, Grant, Sam, Robert, Marie and many more for the wonderful time in Cambridge, these three months have gone so quickly...

Finally, thanks to Daniel Hirschkoff for making this internship possible.

Game semantics and interaction

Game semantics is a way to model typed programs as *strategies* on specific games. Types are typically seen as two-players games, where the moves of each player are alternating. The interaction between Player and Opponent in a game leads to a different interpretation of programs with an emphasis on the duality between programs and environment. This has been quite successful as it produced models close to the languages being modelled often with completeness results. Through the Curry-Howard correspondence[1], game semantics is also useful to build logic models.

We give a few interpretation of common types into games. Each type A is mapped to a game $[\![A]\!]$ where Opponent starts. The product type $A \times B$ — the type of pairs (a, b) where a has type A and b type B — is mapped to a an arena $[\![A \times B]\!]$. In this arena Opponent starts by asking Player to play on $[\![A]\!]$ or on $[\![B]\!]$. To win on $[\![A \times B]\!]$, Player must be able to win in $[\![A]\!]$ and in $[\![B]\!]$ — otherwise Opponent can defeat him by picking the game in which he cannot win. Dually, $[\![A + B]\!]$ the sum type — the type of elements $i_1(a)$ (where a has type A) and $i_2(b)$ (where b has type B) — is interpreted by a game where Opponent starts by asking Player where he wants to play in, either $[\![A]\!]$ or $[\![B]\!]$ and then the chosen game is played. Basic types (`Boolean`, `Integer`) are interpreted by games where Opponent asks for a value of this type, and Player has a possible answer for each value belonging to the type.

The case of the arrow type $A \to B$ — the type of function expecting an argument of type A and returning a result of type B — is more interesting. In $[\![A \to B]\!]$, Opponent starts playing in $[\![B]\!]$. When Player wants, he can switch and play on $[\![A]\!]^\perp$, $[\![A]\!]$ where the role of Opponent and Player are reversed, and then he can get back to $[\![B]\!]$ (He can still switch back to $[\![A]\!]^\perp$ afterwards, as many times as he wants). Playing on $[\![A]\!]^\perp$ consists in evaluating its argument, and thus Player may need to play several times in $[\![A]\!]^\perp$. For instance if its argument is $A = C \times D$ and he wants to use both components, he will need to play twice on $[\![C \times D]\!]^\perp$ as he can access only one component by play (according to the encoding described above).

[1] The Curry-Howard correspondence relates logic to computation where logical statements corresponds to types and proofs of a given statement, programs of the corresponding type.

The construction $[\![A \to B]\!]$ can be decomposed into two smaller constructions:

- ▷ from a game G, build the duplicated version of G, $!\,G$, where you can open as many parallel copies of G as you like ;
- ▷ from two game G and H, build the linear implication $G \multimap H$ as described above where Player can play in G only once. Once G is consumed, Player has to answer in H.

This decomposition of $A \to B$ into $!\,A \multimap B$ was first discovered by Girard in the coherent semantics of System F, which led to linear logic.

Tools used in games constructions

Categories Categories have become ubiquitous in semantics. They are very useful at defining objects by their abstrat properties — such as models, and will be only used as a language in which models are defined. Only basic category will be used in a major part of the report, altough models definition can use more advanced categorical tools. The interested reader can find the main definitions in any category theory textbook, such as [?].

Linear logic and duplication Linear logic [?] was introduced by Jean-Yves Girard as an attempt to build a logic which would gather the advantages of the classical logic (involutive negation and symmetry of the calculus) and the intuitionistic logic (computational meaning), and is based on this decomposition of the intuitionistic arrow. Our interest in linear logic is mainly to know what is a good notion of duplication. That is why the goal of the internship was to build a model of IMELL with concurrent games, which induces a model of intuitionistic logic, and thus a model of the simply-typed λ-calculus. To do that, we need to build inside concurrent games a *linear exponential comonad* (see A for more details).

The hard part is that defining $!\,A$ as infinitely many copies of A is not enough: copies have to be similar and interchangeable to actually be a valid linear exponential comonad.

Related works

The first game models appeared in the 60s and were models of logic, with for instance [?], where a model for intuitionistic logic using games is given. Then Blass [?] gave the first model for linear logic. In these models, the sought for result is the equivalence between the existence a proof for a formula and a winning strategy for the corresponding game (notion of *complete model*).

These models did not take into account the dynamics of the logics (cut-elimination). Thus these models do not lead to models for programming languages where we want that models are invariant with respect to the reduction (two terms equivalent with respect to the langaguage's reduction are the same in the model). This kind of models organize themselves as categories so they are called *categorical models*. The first categorical model using games to appear is a model for the multiplicative fragment of linear logic [?]. The first model of full propositional linear logic would appear a decade after with the asynchronous strategies of Melliès [?].

In the meanwhile, game semantics has also been used as a denotational semantics tool, to build models of programming languages. One of the first model is for PCF (an extension of the simply typed λ-calculus) [?, ?]. This result has been extended to model different paradigms of computation (control operators, references, ...). The present work is a step towards the generalization of these game-theoretic tools to support truly concurrent interpretations of concurrent and non-deterministic programming languages.

Overview of the report

The report is divided into three parts:

- ▷ in the section 1, the general framework of concurrent games is exposed
- ▷ in the section 2, event structures with symmetry, the main formalism used in this internship is introduced
- ▷ in the section 3, my work is presented, with the main result: concurrent games with symmetry are a model of IMELL (a fragment of linear logic with duplication).

1 Concurrent games

This section introduces the **Games** category, which my work was based on. The fundamental notion of concurrent games is the composition that is really similar to the parallel composition of CCS where dual moves synchronize instead of dual labels.

1.1 Event structures with polarities

1.1.1 Event structures

Event structures are the concurrent analogue of trees. Event structures are a way to speak about causality-related events, with a notion of consistency of events.

Event structures An event structure $(E, \leq, \mathrm{Con}_E)$ is given by a set E of events along with:

- ▷ an order relation $\leq$. $e \leq e'$ means that for e' to happen, e must have happened before.
- ▷ Con_E is a set of subsets of E denoting the *consistent* sets of E.

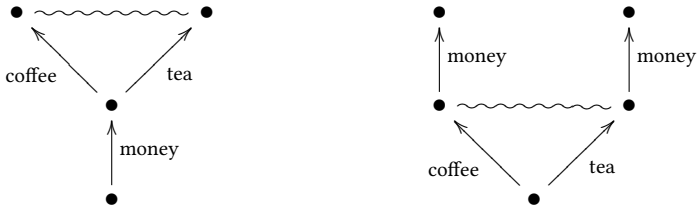
These must satisfy a few axioms :

(i) every event has to depend on a finite number of events, that is the set $[e] = \{e' | e' \leq e\}$ is finite for every e.

(ii) singletons are consistent, i.e. $\{e\} \in \mathrm{Con}_E$

(iii) any consistent set X can be extendend by events that enable events of X, that is if $e' \in X$ and $e \leq e'$ then $X \cup \{e\}$ is consistent as well

Here is two examples of event structures that could represent two kind of coffee machine : one waiting for the coin and then asking the user if wants a tea or a coffee, and the other that let the user choose before requesting money:

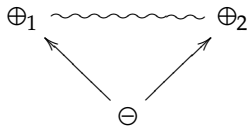


$x \to y$ is the *immediate causality relation* it means that $x < y$ with no events in between, and $\sim\sim\sim$ means conflict (the pair of events is not consistent, and thus their consequences is not either by axiom (iii)). The conflicts in this event structure represents irreversibility of the choice.

To formally speak of "run", we introduce the notion of configuration. A configuration of an event structure is simply a downclosed set of events which is consistent. The down-closed part that if an event e is part of a configuration, then so are all the events on which e causally depends. The set of configurations of an event structure A will be denoted by $\mathscr{C}(A)$.

1.1.2 Games

An event structure with polarities is simply an event structure whose events are either positive (denoted by $\oplus$) or negative ($\ominus$). Here is a simple example of a game



Viewed as a game (and from the positive player's point of view), it means that the player has to wait for an opponent move, and *then* (it is expressed by the causality dependence) can make either one of the two positive moves, and this decision is irreversible: if the player makes one of the two positive move, he cannot do the other one later as $\{\ominus, \oplus_1, \oplus_2\}$ is not a configuration and thus cannot be a valid run of the game.

1.1.3 Operations on games

There are two important operations on games we are going to use:

- ▷ The dual of a game $A^\perp$ is the game A with inverted polarities (but the same underlying event structure). If $a \in A$, we will denote by $\bar{a} \in A^\perp$ the corresponding event.

▷ The parallel composition $A||B$ of two games A and B: it can be seen as the disjoint union of A and B, with no conflicts or causal dependency between them. Formally it is defined as follows: $A||B = (\{0\} \times A \cup \{1\} \times B, \{0\}\times \leq_A \cup \{1\}\times \leq_B, \mathrm{Con}_{A||B})$ where a set of event X is consistent in $A||B$ if $\{a|(0,a) \in X\}$ is consistent in A and $\{b|(1,b) \in X\}$ is consistent in B.

1.2 Strategies

1.2.1 Definition

We have games, we now need strategies. A strategy is a set of runs of a game such that:

▷ it must be compatible with causality: a strategy cannot make a specific move if it has not made all the moves it depends on

▷ it must not prevent an opponent move from happening: after a move, if opponent can make a move then the strategy has to be able to handle it, and it must handle it in a unique way (*receptivity*)

▷ it must preserve causalities $s \to s'$ where s is positive or s' is negative. (It will be clearer in the next section what it means) (*innocence*)

One way to represent this is to define a strategy on a game A as a map of event structures with polarities $\sigma : S \to A$, which is receptive and innocent.

Map of event structures A map $f : A \to B$ of event structures with polarities is a map such that:

▷ the direct image of a configuration in A is a configuration in B

▷ f restricted to a configuration is injective

▷ f preserves polarities

Besides, a map f is receptive when for any configuration x in A, such that the configuration $f\ x$ in B can be extendended by a negative event b (this means that $f\ x \cup \{b\}$ is a configuration of B as well), then x can be extended in a unique way to x' such that $x' = x \cup \{a\}$ and $f(a) = b$.

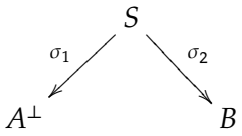
A map f is innocent when, for each causal dependency $s \to s'$ in A where either s is positive or s' negative, then $f(s) \to f(s')$ in B.

We need a notion of isomorphism between strategies since most laws (identity, associativity) will only hold up to isomorphism.

***Definition* 1.** — *Two strategies $\sigma : S \to A$ and $\tau : T \to A$ are isomorphic when there exists an isomorphism $\varphi : S \cong T$ such that $\tau \circ \varphi = \sigma$.*

1.2.2 Spans

We have defined strategies on a game A. Now we define strategies from a game A to a game B as strategies on the game $A^\perp \| B$. A strategy $\sigma : S \to A^\perp \| B$. It can be seen as a span:

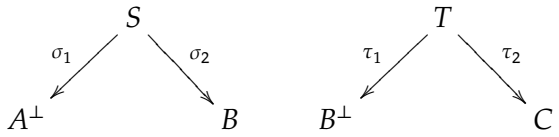


where $\sigma_1 : S \to A^\perp$ and $\sigma_2 : S \to B$ are partial maps of event structures.

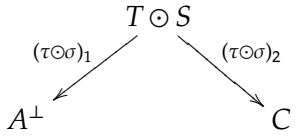
Innocence restricts the way a strategy can switch between A and B. As it has to respect immediate causality $s \to s'$ when s is positive or s' negative it means that a link (in S) $s \to s'$ such that $\sigma_1(s)$ is defined and $\sigma_2(s')$ is defined (or the otherway around) must be a link of the form $\ominus \to \oplus$ since this causality does not exist in $A^\perp \| B$ (there is no causality between an event of $A^\perp$ and an event of B).

1.3 Composition

We now want to compose two spans σ and τ:



in order to obtain a span $\tau \odot \sigma$:



In order to build $T \odot S$ we need to synchronize $\tau \odot \sigma$ on B. Each output of σ in B will be fed as an input to τ and vice-versa. This is very similar to what does the parallel composition in CCS for instance. It consists of two steps:

- ▷ interaction: we consider events in A, synchronized events in B and events in C
- ▷ hiding: we hide synchronized events.

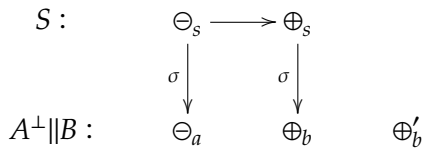
1.3.1 An example

First we give an example of what we mean by synchronization. Consider the following games:

- ▷ $A = \{\oplus_a\}$: a single player move

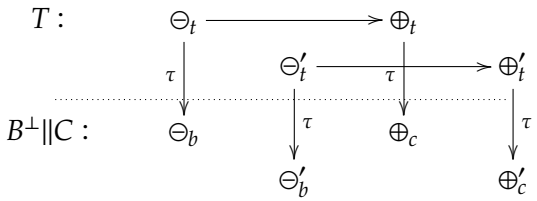
- ▷ $B = \{\oplus_b, \oplus'_b\}$: two concurrent (that is consistent and not causally dependent) player moves
- ▷ $C = \{\oplus_c, \oplus'_c\}$: the same as for B but with different names.

Consider now $\sigma : S \to A^\perp \| B$

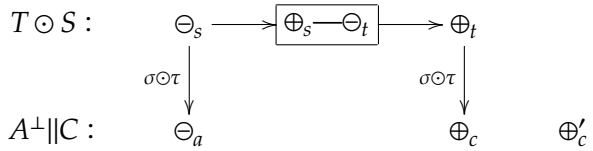


(Labelled arrows denotes the image of the strategy, unlabelled arrows causal dependency in the strategy or the game)

The strategy σ is receptive to the negative event in $A^\perp$ and *then* (because of the link $\ominus_s \to \oplus_s$) plays the first move in B. Consider $\tau : T \to B^\perp \| C$:



The strategy τ is receptive to the two negative events in $B^\perp$ and *then* depending on the first event plays the corresponding event in C. Here we see that only one synchronization can occur: we have $\sigma(\oplus_b) = \overline{\tau(\ominus_b)}$. Here is the composition $\tau \odot \sigma$:



$\oplus_b$—$\ominus_c$ denotes the synchronized event which does not belong directly in $T \odot S$ (because we cannot obtain from such an event, something in A or C). Remark that since $\ominus'_t$ cannot be synchronized, its causal dependency on $\oplus'_t$ disappear in the composition.

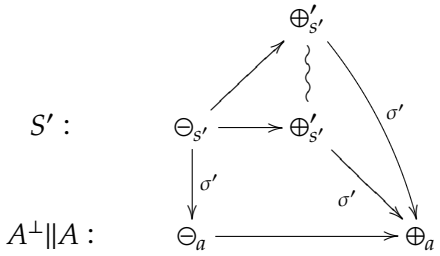
1.3.2 Informal definition

To model this synchronization, we need events of one of the three forms:

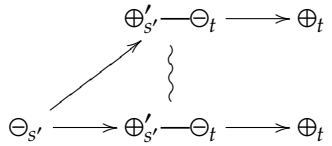
- ▷ $\langle s, * \rangle$ with $s \in S$ and $\sigma_1(s) \in A$ defined
- ▷ $\langle s, t \rangle$ with $(s, t) \in S \times T$ and $\sigma_2(s) = \overline{\tau_1(t)} \in B$ (Recall that $B^\perp$ and B share the same events and have oppose polarities)

▷ $\langle *, t\rangle$ with $t \in T$ and $\tau_2(t) \in B$.

This, however does not define an event structure[2]. To have a proper event structure, we need each event to carry his own history of synchronization, for instance, if we consider $\sigma' : S' \to A^\perp \| B$ defined by



Then in the composition $T \odot S'$ shown below, there is two copies of $\oplus_t$ because the two synchronized events are inconsistent and thus cannot enable the same event.



So events will essentially be pairs $\langle x, y\rangle_h$ labelled by a history h of synchronization.

1.3.3 Formal definition

In this section, we give a formal definition of the event structure $T \odot S$, based on the product of event structures $S \times_* T$ (note that the order is reversed) whose construction is omitted for lack of space. Consider the two projections $\pi_1 : S \times_* T \to S$ and $\pi_2 : S \times_* T \to T$. The projections are *partial* maps of event structures. Define two sets of events in $S \times_* T$:

▷ the set of *visible* events : an event $e \in S \times_* T$ is visible if either $\sigma_1(\pi_1(e))$ is defined and $\pi_2(e)$ is not or $\tau_2(\pi_2(e))$ is defined and $\pi_1(e)$ is not. It corresponds to previous events $\langle s, *\rangle$ and $\langle *, t\rangle$. Only those events will appear directly in $T \odot S$

▷ the set of *interaction* events : an event $e \in S \times_* T$ is an interaction event when $\sigma_2(\pi_1(e)) = \overline{\tau_1(\pi_2(e))}$. Those events will not appear directly on $S \odot T$, but only in the history of events.

As the product of event structures takes care of this notion of history, we can form $T \odot S$ as the following restriction of $S \times_* T$. An event $e \in S \times_* T$ is in $S \odot T$ whenever

[2]To state this more formally, this set with the obvious order ($\langle s, t\rangle \to \langle s', t'\rangle$ iff $s \to s'$ or $t \to t'$) and consistency inherited from S and T does not form an event structure for the following reason: if we have $t \to t'$ in T and two events such that $s, s' \in S$ such that $\sigma_2(s) = \sigma_2(s') = \tau_1(t)$, and $\tau_2(t)$ defined this would imply that the two causalities $\langle s, t\rangle \to \langle *, t'\rangle$ and $\langle s', t\rangle \to \langle *, t\rangle$ but these two events are not consistent since they share the same second projection.

- ▷ it is a visible event
- ▷ every event $e' \in S \times_* T$ such that $e' \leq_{S\times_* T} e$ is either a visible event or an interaction event.

Then one can define $\tau \odot \sigma : T \odot S \to A^\perp \| C$ by letting $\tau \odot \sigma(e)$ to be either $\sigma_1(\pi_1(e))$ or $\tau_2(\pi_2(e))$ depending on which one is defined.

***Lemme* 1.** — *If σ and τ are composable strategies, then $\tau \odot \sigma$ is a strategy. Moreover the composition operation is associative.*

1.4 The copy-cat strategy as an identity for $\odot$

To have a category, we now only need an identity, that is a strategy $\gamma_A : CC_A \to A^\perp \| A$.

Let $\sigma : S \to A^\perp \| A$ and consider what an identity for σ should look like. First, we see that we cannot choose $\mathrm{Id} : A^\perp \| A \to A^\perp \| A$ as a candidate, because $\sigma \odot \mathrm{Id}$ will not have the same causality links as σ. Indeed, in $\sigma \odot \mathrm{Id}$ we do not have any causality links between an event sent by the strategy in $A^\perp$ and an event sent to A because any such causal dependency in $S \odot T$ must "go through a synchronized event" as illustrated before. But as there is no such causal dependency in $A^\perp \| A$ (no causalities between events in $A^\perp$ and A), then there cannot be any in $S \odot \mathrm{Id}$. So we must add to the identity a few causality links between $A^\perp \| A$, and more precisely between negative events c of $A^\perp$ and their counterpart $\bar{c}$ in A (It is the only ones we are allowed to add — because of innocence)

For a given σ, we do not need to include all such causality links for $\sigma \odot \mathrm{Id} \cong \sigma$ to hold, but for the result to hold for every strategy we need to include them all. Adding these dependencies, and the dependency the other way around (going from A to $A^\perp$) to have $\gamma_A \odot \sigma = \sigma$ it gives us the copy-cat strategy.

***Definition* 2.** — *Define the event structure $CC_A = (A^\perp \| A, \leq, Con_{A^\perp \| A})$ where $\leq$ is the reflexive and transitive closure of $\to$ defined by:*

- ▷ *$c \to c'$ when $c, c' \in A$ or $c, c' \in A^\perp$ and $c \to c'$ in A or $A^\perp$*
- ▷ *$\bar{c} \to c$ when $c \in A^\perp \| A$ is positive*

The example τ from the previous section was actually the copycat of B. In term of strategy how can we interpret copycat ? Basically, copycats waits for a move in A or $A^\perp$ and plays the corresponding move in the dual copy.

If you have two chessboards in front of you, and in one you play black, and in the other one you play white, then with this strategy you are sure to win in one of the boards.

***Lemme* 2.** — *For any strategy $\sigma : S \to A^\perp \| B$ we have $\sigma \odot \gamma_A = \gamma_B \odot \sigma = \sigma$.*

1.5 The Games category

The **Games** category[3] is defined as follows:

- ▷ its objects are the event structures with polarities (games)
- ▷ its morphisms from a game A to a game B are strategies $S \to A^{\perp}||B$
- ▷ the composition of strategies is given by $\odot$
- ▷ the identity of A is the copycat strategy γ_A

Rideau, Winskel **Games** is a category. The parallel composition makes **Games** into a compact closed category, with $A \multimap B = A^{\perp}||B$:

(**Games**, $||$) is a compact closed category.

Thus **Games** is a model for IMLL. What about exponentials ?

2 Event structures with symmetry

Concurrent games are based on event structures in which one cannot build an exponential !. This section introduces *event structures with symmetry* which have a good candidate for ?, the dual of !. The next section will be the main result of the internship: we can build a category of games using these event structures with symmetry.

2.1 The problem

If we want to define a monad on the category of event structures, the natural candidate is to define $?A = A||A||... \triangleq \mathbb{N} \times A$, infinitely many copies in parallel of A. Doing this, we have no way to identify the copies that are in a sense equivalent. For instance we want strategies behaving the same but on different copies to be identified. This is needed for the monad law to hold. To give a quickinsight, suppose we define the multiplication of the monad as $\mu_A : ??A \to ?A$ by $(i,(j,a)) \mapsto (\langle i,j\rangle, a)$ where $\langle i,j\rangle : \mathbb{N}^2 \to \mathbb{N}$ is any bijection. Then the associativity of the multiplication amounts to prove that $(i,(j,(k,a))) \mapsto (\langle i,\langle j,k\rangle\rangle, a)$ and $(i,(j,(k,a))) \mapsto (\langle\langle i,j\rangle,k\rangle, a)$ are equal. They are not equal but "equivalent" as defined above: they play the same moves but on different copies.

This is why we need *symmetry* on event structures to be able to have a weaker equivalence than isomorphism on strategies.

[3] It's actually a bicategory since associativity and identities laws hold up to isomorphism. To recover a category one should quotient it by $\cong$

2.2 Isomorphism families

Event structures with symmetry (abbreviated to *ess* in the following) were introduced to support duplication pseudo-monads. They also support unfolding general petri-nets [?].

***Definition* 3.** — *An isomorphism family $\mathbb{S}_A$ on an event structure A is a set of bijections (denoted by $\varphi : x \cong y$ for a bijection φ going from x to y) between configurations of A such that*

- ▷ *(*Identity*) For every configuration, the identity* $\mathrm{Id} : x \cong x$ *is in* $\mathbb{S}_A$
- ▷ *(*Symmetry*) If* $\varphi : x \cong y \in \tilde{A}$ *then* $\varphi^{-1} : y \cong x \in \tilde{A}$
- ▷ *(*Composition*) If* $\varphi : x \cong y \in \tilde{A}$ *and* $\psi : y \cong z \in \tilde{A}$*, then* $\psi \circ \varphi : x \cong z \in \mathbb{S}_A$
- ▷ *(*Restriction*) If* $\varphi : x \cong y \in \tilde{A}$*, then for any subconfigurations* $x_0 \subset x$*, then the restriction* $\varphi|_{x_0} : x_0 \cong \varphi(x_0)$ *is in* $\mathbb{S}_A$
- ▷ *(*Extension*) If* $\varphi : x \cong y \in \mathbb{S}_A$*, then for any superconfiguration* $x_0 \supset x$*, then there exists a way to extend* φ *to* φ' *such that* $\varphi' : x_0 \cong \varphi'(x_0)$ *(then we have* $\varphi'(x_0) \supset y$*).*

The three first conditions ensures that "$x \sim y$ iff there exists $\varphi : x \cong y \in \tilde{A}$" is an equivalence relation, and the last one that two equivalent configurations are bisimilar.

An event structure with symmetry is simply a an event structure A with an associated isomorphism family $\mathbb{S}_A$.

A map of ess from A to B is simply a map $f : A \to B$ such that if $\varphi : x \cong y \in \mathbb{S}_A$ then $f(\varphi) : fx \cong fy \in \mathbb{S}_B$ (where $f(\varphi) = \{(f(a), f(a'))|(a, a') \in \varphi\}$).

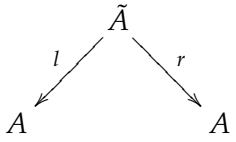
On event structures with symmetry we have an equivalence relation between maps. Two maps $f, g : A \to B$ will be equivalent (notation: $f \sim g$) if for all configuration $x \in \mathscr{C}(A)$ the bijection $\theta_x : fx \cong gx$ is in $\mathbb{S}_B$ where $\theta_x = \{(f(a), g(a))|a \in A\}$.

An *event structure with symmetry and polarities* (essp) is an event structure with polarities A along with an isomorphism family $\mathbb{S}_A$ such that the bijections in $\mathbb{S}_A$ respects polarities.

Note that on essps we have the same operation as for regular esp : parallel composition and duality.

2.3 Alternative definition

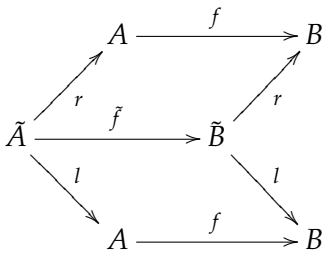
We give an equivalent definition of symmetry, a bit more abstract that will be used in the next section. A symmetry on A is an event structure $\tilde{A}$ equipped with two maps $l : \tilde{A} \to A, r : \tilde{A} \to A$:



We ask that the map $\langle l, r\rangle : \tilde{A} \to A \times A$ is injective and that (l, r) form an equivalence relation (satisfies a category reformulation of reflexivity, symetry, transitivity — details can be found in [?]). Besides we want l and r to preserve the causal dependencies (normal maps only reflects them) and that if we have x a configuration in $\tilde{A}$ and y a configuration in A such that $l(x) \subset y$ then x can be extended to x' such that $l(x') = y$.

To get an isomorphism family out of a symmetry $\tilde{A}$, simply define $\mathbb{S}_A = \{\varphi : lx \cong rx | x \in \mathscr{C}(\tilde{A})\}$ where φ is induced by the injectivity of l and r on x.

A map of event structures with symmetry with this definition is a map $f : A \to B$ along with a map on symmetries $\tilde{f} : \tilde{A} \to \tilde{B}$ making the following commute



2.4 The monad on event structures with symmetry

One can define a monad[4] ? on the category of event structures with symmetry, with maps of ess. As introduced above, defined for any ess A, $?A = \mathbb{N} \times A$. $?A$ represents infinitely many copies of A in parallel. Note $\pi_k : \mathscr{C}(?A) \to \mathscr{C}(A)$ which takes a configuration $x \in \mathscr{C}(?A)$ and returns the moves played on the k-th copy (formally $\pi_k(x) = \{a | (k, a) \in x\}$.

$\mathbb{S}_{?A}$ is defined as follows: a bijection $\varphi : x \cong y$ between two configurations of $?A$ is in $\mathbb{S}_{?A}$ where there exists a bijection $\theta : \mathbb{N} \to \mathbb{N}$ such that x behaves on the k-th copy as y behaves on the $\theta(k)$-th copy in a symmetric way (with respect to the symmetry on A). Formally this means that

- $\varphi(k, a) = \varphi(\theta(k), a')$ for every $(k, a) \in x$
- φ induces a bijection $\varphi_k : \pi_k(x) \cong \pi_{\theta(k)}(x)$ that is in the isomorphism family of A.

It is easy to check that $?A$ is actually an essp. This makes a functor ? from the category of event structures with symmetry and polarities to itself ($?f : ?A \to ?B$ is defined by $?f(i, a) = (i, f(a))$)

Then one can build the unit of ?, $\eta_A : A \to ?A, a \mapsto (0, a)$ and $\mu_A : ??A \to ?A, (i, (j, a)) \mapsto (\langle i, j\rangle, a)$. One can easily check that $(?, \eta_A, \mu_A)$ is a (pseudo-)monad.

[4] It's actually a pseudo-monad since monadic laws hold up to symmetry

3 Games with symmetry

This is where the internship started: trying to create a category **SGames** such that it is suitable to create game models of functional programming language or logic. To do that, we will define a category based on event structures with symmetry and import the monad defined in the previous section to this category.

3.1 The category SGames

We build a category **SGames** of games with symmetry which consists in :

- ▷ Objects: games with symmetry
- ▷ Morphisms: symmetric strategies

We want **SGames** to have a weaker notion of equivalence $\simeq$, inherited from the equivalence on event structures with symmetry.

3.1.1 Games with symmetry

Games with symmetry are essentially event structures with polarities with an extra condition on the isomorphism families:

Games with symmetry A game with symmetry is an event structure with symmetry and polarities whose isomorphism family is such that, for every $\varphi : x \cong y \in \mathbb{S}_A$, if φ can be extended (as a bijection in $\mathbb{S}_A$) by (e, e') and by (e_0, e_0') such that e and e_0 have different polarities and $x \cup \{e, e_0\}$ as well as $y \cup \{e', e_0'\}$ are consistent, then $\varphi \cup \{(e_0, e_0'), (e, e')\} \in \mathbb{S}_A$ (Race-freeness)

This rather technical condition means that symmetry cannot add extra races[5] two symmetric configurations can be extended by two consistent ways of different polarities, then the resulting extensions are symmetric as well. This is needed to ensure that copycat has a symmetry. This condition will also force us to consider exponentials only on polarised games (i.e. games where initial moves have the same polarities). It is needed to have a symmetry on copycat.

3.1.2 Symmetric strategies

A symmetric strategy from A to B is a map of games with symmetry $\sigma : S \to A^{\perp} \| B$ (so S has to have a symmetry with the same technical condition as for games with symmetry) such that

- ▷ σ is innocent and receptive, as in **Games**
- ▷ σ reflects the symmetry in a way: if we have $\varphi : x \cong x' \in \mathbb{S}_S$ and $\sigma\varphi \cup \{e, e'\} : \sigma x \cup \{e\} \cong \sigma x' \cup \{e'\} \in \mathbb{S}_{A^{\perp} \| B}$ and e, e' are negative, *then* φ can be extended to $\varphi' \in \mathbb{S}_S$ in a unique way, and $\varphi' : y \cong y'$ with $\sigma y = \sigma x \cup \{e\}, \sigma y' = \sigma x' \cup \{e'\}$.

[5] A race, in terms of event structure, is when a configuration x can be extended by two incompatible events of different polarities.

3.1.3 Identity and composition

One can check that we can make the previous identity and composition compatible with this new structure. One can construct the symmetry (with the equivalent definition given in sec. 2.3) on copycat of A by considering $\gamma_{\bar{A}}$, the copycat on symmetry, and the symmetry on the composition $\sigma \odot \tau$ by considering $\tilde{\sigma} \odot \tilde{\tau}$ the composition of the symmetries.

SGames is a category and (**SGames**, $\|$) is a symmetric monoidal closed category.

3.1.4 An equivalence on SGames

To have the monadic laws of ? we are going to need a weaker notion of equivalence on strategies than isomorphism.

***Definition* 4.** — *Two strategies $\sigma : S \to A^{\perp}\|B$ and $\tau : T \to A^{\perp}\|B$ are equivalent if there is family of bijection $\mathbb{S}_{S,T}$ between configurations of S and T such that*

- ▷ *(*Restriction*) : If $\varphi : x \cong y \in \mathbb{S}_{S,T}$ and $x' \subset x$ then $\varphi|'_x : x' \cong \varphi(x') \in \mathbb{S}_{S,T}$*
- ▷ *(*Extension*) : If $\varphi : x \cong y \in \mathbb{S}_{S,T}$ and $x \subset x'$ then there exists $y' \in \mathscr{C}(T)$ and $\varphi' \supset \varphi \in \mathbb{S}_{S,T}$ such that $\varphi' : x' \cong y'$.*
- ▷ *(*Compatibility with the symmetries*) : If $\varphi : x \cong y \in \mathbb{S}_{S,T}$ and $\theta : y \cong y' \in \mathbb{S}_T$ then $\theta \circ \varphi : x \cong z \in \mathbb{S}_{S,T}$ and the same thing on the left*
- ▷ *(*Symmetry*) : If $\varphi : x \cong y \in \mathbb{S}_{S,T}$ and $\varphi' : y \cong z \in \mathbb{S}_{T,S}$ then $\varphi \circ \varphi' : x \cong z \in S_S$ and the same thing on the left*

We note $\sigma \simeq \tau$ when σ and τ are equivalent

***Lemme* 3.** — *$\simeq$ is a congruence[6] on **SGames**.*

3.2 A functor $F : \mathscr{E}_{ri} \to$ SGames

What we want is a way to import maps (the monadic components of ?) into **SGames**. To do that we need a functor from $\mathscr{E}_{ri}$ the category of games with symmetry, with rigid and injective maps of essps, into **SGames**. The functor presented here acts on special maps (injective and rigid) but a slightly more general construction exists.

Given a map of event structures $f : A \to B$ (with no symmetry) one can form the event structure S_f which consists of $A^{\perp}\|B$ with extra immediate causality:

- ▷ $\bar{a} \rightarrowtail f(a)$ when $f(a)$ is positive
- ▷ $f(a) \rightarrowtail \bar{a}$ when $f(a)$ is negative

[6] A congruence is an equivalence relation on morphisms of the same type, which is preserved by composition.

This construction is a generalization of copycat's — we have $S_{\mathrm{Id}_A} = CC_A$. When f is a map of essps, one can build an essp S_f, such that $F(f) : S_f \to A^\perp \| B$ which acts like the identity is a symmetric strategy.

Lemme **4.** — *The functor $F : \mathscr{E}_r \to$ **SGames** which maps each game with symmetry A to itself, and each function $f : A \to B$ to $F(f)$ is a functor.*

Besides, if $f, g : A \to B$ are two receptive, injective and rigid maps of event structures with symmetry, then $f \sim g$ implies $F(f) \simeq F(g)$.

So F will lift the monadic laws in $\mathscr{E}_{ri}$ to **SGames** since the morphism we want to lift satisfy all the conditions.

3.3 Polarised games

The problem with the definition of games with symmetry is that if A has two initial moves of different polarities, $?A$ won't be a game with symmetry. If $A = \{\oplus, \ominus\}$, two concurrent events of different polarities, then in $?A$ we have : $\varnothing \cong_{?A} \varnothing$ and $\{1, \oplus\} \cong \{(0, \oplus)\}$ along with $\{1, \ominus\} \cong \{2, \ominus\}$. But we don't have $\{(1, \oplus), (1, \ominus)\} \cong \{(0, \oplus), (2, \ominus)\}$ since $\{1, 1\}$ and $\{0, 2\}$ are not in bijection.

This forces us to consider the two full subcategory of games whose initial move are either positive or negative. This is called *polarisation*: a game whose initial events are negative is called *negatively polarised* (Opponent starts), and a game whose initial events are positive is called *positively polarised* (Player starts). Doing so, we need to restrict the polarities on the strategies as well. To have that the two categories are dual, we need to have the same polarities on strategies. We choose negative to be consistent with usual game semantics models, as it means that Opponent starts.

We define two subcategories of **SGames**:

Definition **5.** — ***SGames***$^+$ *is the category whose objects are games with symmetry positively polarised, and the morphisms from A to B are symmetric strategies $\sigma : S \to A^\perp \| B$ where S is negatively polarised.*

SGames$^-$ *is the category whose objects are games with symmetry negatively polarised, and the morphisms from A to B are symmetric strategies $\sigma : S \to A^\perp \| B$ where S is negatively polarised.*

These two categories are dual via the duality operator $A \mapsto A^\perp$:

We have an isomorphism $(\mathbf{SGames}^+)^{\mathrm{op}} \cong \mathbf{SGames}^-$

We can define ? on the positive category as a functor. By a quite complicated natural transformation lifting lemma, we have that

Lemme **5.** — *? : **SGames***$^+ \to$ ***SGames***$^+$ *is a functor and $F(\eta_A) : A \to ?A$ as well as $F(\mu_A) : ??A \to ?A$ are natural transformations.*

Note that we need to work with positively polarised games so η_A is receptive (This hypothesis is needed to have F preserve equivalences).

This means that $\mathbf{SGames}^+$ is equipped with a monad. With a bit of work, one can show that it is the dual of an exponential comonad. Therefore, by duality we get an exponential comonad on

the category $\mathbf{SGames}^{-}$, thus the following. $\mathbf{SGames}^{-}$ is a linear category, and as such a model of IMELL.

This is the main result of the internship. Concurrent games with symmetry are suitable to model non-linear programming language, and with event structures even concurrent programming languages.

Conclusion

We have seen how to introduce duplication into the framework of concurrent games, making it suitable to model functional programming language. With this result, one can now try to model non-deterministic and concurrent higher-order programming language within this framework, something that does not exist yet.

Another interesting lead that was (and still is) studied at the end of the internship is how to have duplication in a non-polarised setting. This however is much more interesting and could lead to a "natural" model of linear logic. This requires us to change the copycat to a saturated copycat: to make a move, it doesn't wait for the move to be played in the other copy, it waits for a symmetric move to be played. Composition should be changed as well, to take into account symmetry. It is not clear though what strategies (spans composing well with this saturated copycat) should be in this setting.

References

A Categorical models of linear logic: linear categories

For a survey of models of linear logic, see [**?**]. For a more precise account of the models we are interested in, see [**?**]. Basic category theory is assumed (categories, functors, natural transformations). Coherence diagrams are omitted because of the lack of space, but the interested reader will find them in the cited references.

A.1 Symmetric monoidal closed category

A symmetric monoidal closed category SMCC is a model of the intuitionistic multplicative fragment of linear logic (IMLL). It is a model of linear λ-calculus, that is a λ-calculus where each λ-abstraction uses its argument exactly once.

A symmetric monoidal category $(\mathscr{C}, \otimes, e)$ is a category $\mathscr{C}$ along with a functor $— \otimes — : \mathscr{C} \times \mathscr{C} \to \mathscr{C}$ and an object e of $\mathscr{C}$ such that there exists natural isomorphism $A \otimes (B \otimes C) \cong (A \otimes B) \otimes C$, $A \otimes B \cong B \otimes A$ and $A \otimes e \cong e \otimes A \cong A$ natural in A, B and C satisfying some coherence diagrams ommited by lack of space.

A symmetrical monoidal closed category is a category where there exists a functor $— \multimap — : \mathscr{C}^{\mathrm{op}} \times \mathscr{C} \to \mathscr{C}$ such that there exists a natural isomorphism between $A \otimes B \to C$ and $A \to (B \multimap C)$ (natural in A, B, C).

A.2 Linear exponential comonad

A linear exponential comonad represents the duplication in a SMCC. Formally a linear exponential comonad over a SMCC $\mathscr{C}$ is an uple $(!, \varepsilon, \delta, m_A, m_I)$ where $! : \mathscr{C} \to \mathscr{C}$ is a functor, $\varepsilon :! A \to A, \eta :! A \to !!A$ are two natural transformations such that $(!, \varepsilon, \delta)$ is a comonad, and $m_A :! A \to !A \otimes, m_I :! A \to 1$ are natural transformations making each object $!A$ a commutative comonoid, such that $m_I :! A \to 1$ and $m_A :! A \to !A \otimes !A$ are coalgebra morphisms, and finally that each coalgebra morphism $f :! A \to !B$ is also a comonoid morphism.

A linear category $\mathscr{C}$ is a SMCC along with a linear exponential comonad $!$ over $\mathscr{C}$. Such categories are models of IMELL, the multiplicative fragment of linear logic with the exponential. What is interesting and conveys the intuition that $A \to B \equiv !A \multimap B$, is that the category of co-Kleisli over $!$ is a cartesian closed category, and thus a model of λ-calculus: From a linear category $(\mathscr{C}, !)$, we can build a category $K(!)$ whose objects are the same objects as in $\mathscr{C}$ and morphism between A and B are the arrows in $\mathscr{C}$ from $!A \to B$ (i.e. we have $K(!)(A, B) = \mathscr{C}(!A, B)$. This category is cartesian closed.