CTCS 2004 Preliminary Version

Concrete Data Structures as Games

Andrea Schalk [1] and José Juan Palacios-Perez [2]

Department of Computer Science
University of Manchester
United Kingdom

Abstract

A result by Curien establishes that filiform concrete data structures can be viewed as games. We extend the idea to cover all stable concrete data structures. This necessitates a theory of games with an equivalence relation on positions. We present a faithful functor from the category of concrete data structures to this new category of games, allowing a game-like reading of the former. It is possible to restrict to a cartesian closed subcategory of these games, where the function space does not decompose and the product is given by the usual tensor product construction. There is a close connection between these games and graph games.

Key words: Linear logic, games, concrete data structures, sequential computation

Concrete data structures were introduced by Kahn and Plotkin [9] as a domain for computation. There is a close connection between certain kinds of concrete data structures, namely those with a linear dependency between cells, and a simple category of games (with an intuitionistic rather than linear function space)—these categories are equivalent. This result is established in [4]. The main of this paper is to extend the result to non-linear data structures. However, rather than extending it to all concrete data structures we have to limit ourselves to the *stable* case. Our result requires the introduction of a new category of games. Here positions within the game can be identified, and that limits the number of possible strategies similar to the *conflict-freeness* condition for graph games. We define a functor from the category of stable concrete data structures to this new category of games. By characterizing its image we exhibit a subcategory of the latter which is equivalent to the former.

[1] Email:A.Schalk@cs.man.ac.uk
[2] Email:palacioj@cs.man.ac.uk

This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

1 Background

Notation. If A is a subset of B, and r is a word over the alphabet B then we write $r|_A$ for the word that arises by removing all those letters from r which are not in A. Given words r, r' over the same alphabet we write rr' for their concatenation, $r \wedge r'$ for their greatest joint prefix and if r is a prefix of r' we write $r \trianglelefteq r'$. We extend the concatenation notation to sets of symbols and use $(\,)^*$ for the Kleene star. Given a set S of words over some alphabet we use $\mathsf{pre}(S)$ for the prefix closure of S.

We allow ourselves to view both A and B as subsets of their sum $A + B$. Where this could cause confusion (for example in the sum $A + A$), we use the injections inl and inr.

1.1 Games

We describe a simple category of games.

Definition 1.1 A **game** A consists of

- two disjoint sets A_P (of *Player or P-moves*) and A_O (of *Opponent or O-moves*);
- a prefix-closed subset $\mathsf{pos}(A)$ of the language $\mathsf{pre}((A_O A_P)^*)$ (the *game tree*).

Hence a *position* in a game A is a word in $\mathsf{pre}((A_O A_P)^*)$ which belongs to the game tree, that is an element of $\mathsf{pos}(A)$. A position is a *P-position* if its last move is a P-move, and otherwise it is an *O-position*. In other words, P-positions are the even length words in the language $\mathsf{pos}(A)$ while O-positions are the odd length ones. We use $\mathsf{pos}_P(A)$ to refer to the set of all P-positions and similarly for $\mathsf{pos}_O(A)$.

The set $\mathsf{pos}(A)$ can easily be seen as a tree whose edges are labelled by moves and whose nodes correspond to positions, that is the elements of $\mathsf{pos}(A)$. The 'initial position' or root of the tree is given by the empty word. For $r, r' \in \mathsf{pos}(A)$ we sometimes write $r \longrightarrow r'$ if there is a letter a such that $r' = ra$. A *play* of the game A is a word in $\mathsf{pos}(A)$ (or a path from the root in the game tree).

Definition 1.2 A **strategy** (for Player) for a game A is a prefix-closed sublanguage α of $\mathsf{pos}(A)$ with the following properties:

- If $r \in \mathsf{pos}(A)$ is an O-position and ra, ra' are two positions in α then $a = a'$ (this is known as *determinacy*);
- if $r \in \mathsf{pos}(A)$ is a P-position in α then all $ra \in \mathsf{pos}(A)$ are also in α.

We sometimes write $\alpha\colon A$ if α is a strategy for A. If we think of $\mathsf{pos}(A)$ as a (game) tree then a strategy α must be a subtree. A *play in accord with a strategy α on A* is a word in α.

Definition 1.3 The **linear function space** $A \multimap B$ of two games has

- sets of moves $(A \multimap B)_O = A_P + B_O$, $(A \multimap B)_P = A_O + B_P$, and
- game tree $\mathsf{pos}(A \multimap B) = \{t \in \mathsf{pre}(((A \multimap B)_O (A \multimap B)_P)^*) \mid t|_A \in \mathsf{pos}(A),$ $t|_B \in \mathsf{pos}(B)\}$.

In the usual way a (linear) morphism from A to B in our category of games is a strategy on $A \multimap B$. We solve the problem of composition. Let $\xi\colon A \longrightarrow B$ and $\zeta\colon B \longrightarrow C$. We define a *joint sequence in accord with* ξ *and* ζ to be a word w over the alphabet $A_P + A_O + B_P + B_O + C_P + C_O$ such that

- $w|_{A+B}$ is a play in accord with ξ and
- $w|_{B+C}$ is a play in accord with ζ.

We define the composite to be

$$\zeta \circ \xi = \{w|_{A+C} \mid w \text{ is a joint sequence in accord with } \xi \text{ and } \zeta\}.$$

It is easy to show that if t is a position in $\zeta \circ \xi$ then there is a unique minimal joint sequence in accord with ξ and ζ, say w, with $w|_{A+C} = t$. Further, if $t \trianglelefteq t'$ and w and w' are the respective minimal joint sequences then $w \trianglelefteq w'$. The identity on a game A is given by the 'copycat' strategy, where Player copies Opponent's moves from one copy of A to the other. This composition can also be described as a relational composition by using a faithful functor to the category of sets and relations, see [6].

Definition 1.4 The **tensor product** $A \otimes B$ of two games has

- sets of moves $(A \otimes B)_O = A_O + B_O$, $(A \otimes B)_P = A_P + B_P$, and
- game tree $\mathsf{pos}(A \otimes B) = \{t \in \mathsf{pre}(((A \otimes B)_O (A \otimes B)_P)^*) \mid t|_A \in \mathsf{pos}(A),$ $t|_B \in \mathsf{pos}(B)\}$.

Definition 1.5 The **product** $A \times B$ of two games A and B has

- sets of moves $(A \times B)_O = A_O + B_O$ and $(A \times B)_P = A_P + B_P$;
- positions $\mathsf{pos}(A + B) = \mathsf{pos}(A) \cup \mathsf{pos}(B)$.

The terminal game $\mathbf{1}$ is the game which has no moves.

This defines a well-known symmetric monoidal closed category $\mathbf{Gam}_\ell$. We can define a linear exponential comonad in the sense of [8] so that the resulting Kleisli category **Gam** is cartesian closed. The linear exponential is due to Lamarche and Curien (see for example [4]). Further $\mathbf{Gam}_\ell$ embeds faithfully into **Gam**.

1.2 Concrete data structures

Concrete data structures were introduced by Kahn and Plotkin [9] as a particular kind of computation domain. Their intended use was as a semantic universe for programming languages. Since then, they have been used to understand sequentiality in a number of ways. Accounts can be found in [2,3].

Definition 1.6 A **concrete data structure (cds)** $R = (C, V, \mathsf{ev}(R), \vdash)$ has

- a set C (of *cells*);
- a set V (of *values*) disjoint from C;
- a relation $\mathsf{ev}(R) \subseteq C \times V$ (of *events*);
- a relation $\vdash$ between finite sets of events and cells (the *enabling relation*).

We assume without further comment that the enabling relation is *well-founded* in the sense of [2].

Definition 1.7 A **state** ρ of a concrete data structure R is a set of events with the following properties:

- If (c, v) and (c, v') are both in ρ then $v = v'$ (this is known as *consistency*).
- For all (c, v) in ρ there are $(c_1, v_1,), \ldots, (c_n, v_n) = (c, v)$ in ρ such that for all $1 \leq i \leq n$, $\{(c_1, v_1), \ldots, (c_{i-1}, v_{i-1})\}$ contains an enabling of c_i (this is known as *safety*).

We typically use $\mathsf{st}(R)$ to refer to the set of all states of R, and $\mathsf{st_f}(R)$ for the set of all finite states of R. We sometimes write $\rho\colon R$ to indicate that ρ is a state of R. Given a set of events ρ we say that a cell c is *enabled but not filled in* ρ provided that ρ contains an enabling of c and c does not occur in ρ.

When we translate cdss into games we want to view the cells as Opponent and the values as Player moves. In particular we want to establish a reading of a sequence of events satisfying the safety and consistency conditions as a *play* which is a string in $\mathsf{pre}((CV)^*)$.

On the other hand, given a string $r \in \mathsf{pre}((CV)^*)$ we can obtain a subset of $C \times V$ which we refer to as $\lfloor r \rfloor$ by setting

$$\lfloor r \rfloor = \{(c, v) \in C \times V \mid \exists r' \in (CV)^*.\, r'cv \trianglelefteq r\}.$$

Under the right circumstances, $\lfloor r \rfloor$ will be a set of events or even a state.

A **run** of a cds R is a word r in $\mathsf{pre}((CV)^*)$ such that for each prefix r' of r

- $\lfloor r' \rfloor$ is a state;
- if $r' = r''c$ has odd length then c is enabled but not filled in $\lfloor r'' \rfloor$.

A *run of a state* ρ of R is a run r of R such that $\lfloor r \rfloor \subseteq \rho$. We can recover a state from the set of all its runs. We can think of the enabling relation as specifying that some events must have taken place before others can occur—we are given some information about possible *sequentializations* of states. In general there will be many ways of sequentializing a state, that is, for a given state ρ there will be a number of runs r such that $\lfloor r \rfloor = \rho$. Some of the substrings of such an r will be freely interchangeable, because nothing that occurs in the one substring depends on anything occurring in the other. When we turn cdss into games and states into strategies we do this in terms of such sequentializations.

Definition 1.8 We say that a cds is **stable** if for all enablings $x \vdash c$, $x' \vdash c$ such that there is a state containing both, x and x', there is $x'' \subseteq x \cap x'$ with $x'' \vdash c$.

We find it convenient to assume that *all* cdss are stable and will do so from now on without further notice.

Definition 1.9 The **function space** $R \Rightarrow S$ of two cdss $R = (C, V, \mathsf{ev}(R), \vdash)$ and $S = (D, W, \mathsf{ev}(S), \vdash)$ has

- cells: $\mathsf{st_f}(R) \times D$;
- values: $C + W$;
- events: all pairs
  · $((\rho, d), c)$ such that c is enabled but not filled in ρ,
  · $((\rho, d), w)$ such that (d, w) is an event in S;
- enablings: all instances of

$$((\rho, d), c) \vdash (\rho', d)$$

  such that there exists $v \in V$ with $\rho' = \rho \cup \{(c, v)\}$, and

$$((\rho_1, d_1), w_1), \ldots, ((\rho_n, d_n), w_n) \vdash (\rho, d)$$

  such that $\rho = \rho_1 \cup \cdots \cup \rho_n$ and $(d_1, w_1), \ldots, (d_n, w_n) \vdash d$ in S.

A **morphism of cdss** from $R = (C, V, E, \vdash)$ to $S = (D, W, F, \vdash)$ is a state of the function space. We can view such a morphism as a set of instructions which allows us to use information (values in cells from R) in the left hand game in order to fill cells in the right hand game.

It is non-trivial to define composition. We give a brief account here which is taken from [2]. Given a state ϕ in $R \Rightarrow S$ one can define a partial function

$$g_\phi \colon \mathsf{st_f} R \times D \rightharpoonup C + W,$$

that is a partial function from cells of $R \Rightarrow S$ to values of the same cds. This works by setting

$$g_\phi(\rho, d) = z \text{ iff } \exists \rho' \subseteq \rho \text{ with } ((\rho', d), z) \in \phi \text{ and } ((\rho, d), z) \text{ an event in } R \Rightarrow S.$$

Such a g_ϕ is known as an '*abstract algorithm*'. Given $g \colon \mathsf{st_f} R \times D \rightharpoonup C + W$ there exists $\phi \colon R \Rightarrow S$ with $g = g_\phi$ if and only if the following hold:

- If $g(\rho, d) = z$ then $((\rho, d), z)$ is an event in $R \Rightarrow S$.
- If $g(\rho, d) = z$ and if there is $\rho' \supseteq \rho$ such that $((\rho', d), z)$ is an event in $R \Rightarrow S$ then $g(\rho', d) = z$.
- If $g(\rho, d)$ is defined then d is enabled in $\{(d', w) \mid g(\rho, d') = w\}$, and if there is $\rho' \subseteq \rho$ such that d is also enabled in $\{(d', w) \mid g(\rho', d') = w\}$ then $g(\rho', d)$ is also defined.

We can recover ϕ from g_ϕ

$$\phi = \{((\rho, d), x) \mid g_\phi(\rho, d) = x \text{ and } \rho \text{ is minimal with that property}\}.$$

In order to compose abstract algorithms we have to note that a state $\phi\colon R \Rightarrow S$ defines a function $\mathsf{st}(\phi)\colon \mathsf{st}(R) \longrightarrow \mathsf{st}(S)$ by setting

$$\mathsf{st}(\phi)(\rho) = \{(d, w) \in \mathsf{ev}(S) \mid \exists \rho' \subseteq \rho.\, ((\rho', d), w) \in \phi\}.$$

Given two states $\phi\colon R \Rightarrow S$ and $\psi\colon S \Rightarrow T$ the standard definition of their composite is to give the corresponding abstract algorithm, which is

$$(g_\psi \circ g_\phi)(\rho, e) = \begin{cases} x & g_\psi(\mathsf{st}(\phi)(\rho), e) = x \\ c & g_\psi(\mathsf{st}(\phi)(\rho), e) = d \text{ and } g_\phi(\rho, d) = c. \end{cases}$$

The identity of a cds R is described by

$$g_{\mathsf{id}_R}(\rho, c) = \begin{cases} v & (c, v) \in \rho \\ c & (c, v) \notin \rho. \end{cases}$$

There is an alternative definition of composition with a more game-like flavour. Assume we have a morphism $\phi\colon R \longrightarrow S$, that is a state ϕ of $R \Rightarrow S$. The following lemma is a consequence of material contained in [2].

Lemma 1.10 *The following hold for a cds of the form $R \Rightarrow S$. Call a run of some ϕ of $R \Rightarrow S$* **normal** *if it is of the form*

$$(\rho^1_1, d^1)c^1_1 \cdots (\rho^1_{n_1-1}, d^1)c^1_{n_1-1}(\rho^1_{n_1}, d^1)w^1(\rho^2_1, d^2)c^2_1 \cdots$$
$$(\rho^m_1, d^m)c^m_1 \cdots (\rho^m_{n_m-1}, d^m)c^m_{n_m-1}(\rho^m_{n_m}, d^m)w^m$$

where the last element is optional, such that $\bigcup_{i,j} \rho^i_j$ is a state of R. Then

$$\phi = \{((\rho, d), x) \in \phi \mid \exists \textit{normal run of } \phi \textit{ ending in } (\rho, d)x\}.$$

In other words, in a normal run as soon as a cell d in D has been played we cannot mention other cells from D until d has been filled. Further, we only allow runs which are consistent in their requirement of information from R.

A normal run of ϕ contains a lot of information, some of which we can discard. Given a normal run we want to build a string in $\mathsf{pre}((D(CV)^*W)^*)$ which we require below to interpret ϕ as a strategy. An **economic run** of $\phi\colon R \Rightarrow S$ is a word in $\mathsf{pre}((D(CV)^*W)^*)$ which is generated recursively by a normal run of ϕ as follows:

- **Run of length** 0**.** For an empty run take the empty string.
- **Run of length** $n+1$**.** Let $(\rho_1, d_1)x_1 \cdots (\rho_n, d_n)x_n(\rho_{n+1}, d_{n+1})x_{n+1}$ be a normal run of ϕ and r be the economic run corresponding to the given run

up to $(\rho_n, d_n)x_n$. We make a case distinction based on whether x_n is an element of C or one of W

· $x_n \in C$. In this case $d_n = d_{n+1}$ and $\rho_{n+1} = \rho_n \cup \{(x_n, v)\}$ for some $v \in V$, and we take rvx_{n+1}.

· $x_n \in W$. In this case, d_n and d_{n+1} are distinct, and we take $rd_{n+1}x_{n+1}$.

Clearly no information is really lost by moving from a normal run to the corresponding economic one, and we can easily reconstruct the former from the latter. However, we require stripping away still further information.

Note that by the definition of normal run, if an economic run contains a cell c more than once then its successor, a value from V, has to be the same for all occurrences. Given an economic run r we obtain the corresponding **basic run** by removing pairs $cv \in \mathsf{ev}(R)$ which are repetitions. It is in general not possible to reconstruct an economic run from its corresponding basic version. However, taken together the basic runs of a morphism are sufficient to characterize it.

Example 1.11 Consider the cds R which has one initially enabled cell c and one value for it, v, and the cds S with two initially enabled cells d and d', both of which can be filled with value w. Then

$$\phi = \{((\emptyset, d), c),\ ((\{(c, v)\}, d), w),\ ((\emptyset, d'), c),\ ((\{(c, v)\}, d'), w)\}$$

is a state of $R \Rightarrow S$ with maximal normal runs

$$\begin{aligned} &(\emptyset, d)c(\{(c, v)\}, d)w(\emptyset, d')c(\{(c, v)\}, d')w \\ \text{and } &(\emptyset, d')c(\{(c, v)\}, d')w(\emptyset, d)c(\{(c, v)\}, d)w, \end{aligned}$$

maximal economic runs $dcvwd'cvw$ and $d'cvwdcvw$ and maximal basic runs $dcvwd'w$ and $d'cvwdw$. Both these runs together allow us to deduce that both, d and d' require the cell c to be filled in order be assigned a value.

Note that basic runs of the identity on R are very reminiscent of plays of a copycat strategy.

Lemma 1.12 *A state ϕ of $R \Rightarrow S$ is uniquely determined by its set of basic runs.*

Proof (Sketch) Given an event $((\rho, d), x)$ in $R \Rightarrow S$ we have to decide whether it is in ϕ. Consider all basic runs t which contain d as their last symbol from D and x as their last letter, and which are such that $\lfloor t|_{C+V} \rfloor$ is contained in ρ. If there is such a run which is minimal among all these in the prefix order with the property that $\lfloor t|_{C+V} \rfloor$ is equal to ρ then our event belongs to ϕ, else it does not. □

A **joint basic run of ϕ and ψ** is a word z in $(C + V + D + W + E + X)$ such that

- $z|_{C+V+D+W}$ is a basic run of ϕ and
- $z|_{D+W+E+X}$ is a basic run of ψ.

Proposition 1.13 *The set of basic runs of $\psi \circ \phi$ is the set*

$$\{z|_{C+V+E+X} \mid z \text{ joint basic run of } \phi \text{ and } \psi\}.$$

Proof (Sketch) It is not too difficult to show that the above set of basic runs is contained in the set of basic runs for $\psi \circ \phi$—the proof is an induction by the way joint basic runs are formed. The other inclusion is reminiscent of the proof that composition in **Gam** can be described as a relational composition. $\square$

Definition 1.14 The **product** $R \times S$ of two cdss $R = (C, V, E, \vdash)$ and $S = (D, W, F, \vdash)$ has

- cells: $C + D$;
- values: $V + W$;
- events: $\mathsf{ev}R + \mathsf{ev}(S) \subseteq (C \times V) + (D \times W) \subseteq (C + D) \times (V + W)$;
- enablings: all enablings from R and all enablings from S.

The **terminal cds 1** has empty sets of cells and values.

Theorem 1.15 *The category* **CDS** *is cartesian closed with the structure given above.*

1.3 Games and filiform cdss

There are very special concrete data structures which are closely related to games. They have the property that the enabling relation has a very restricted form; as a result of which the minimal states containing a given event have a unique sequentialization.

Definition 1.16 We say that a cds is **filiform** if all instances of the enabling relation refer to sets of events with at most one element.

We call such a concrete data structure an **fcds**. Let **FCDS** be the full subcategory of **CDS** whose objects are the stable filiform concrete data structures.

Since function space and product in **CDS** preserve the filiform property, **FCDS** is cartesian closed.

For a cds we define a game whose Opponent moves are the cells of R and whose Player moves are the values. A play should then correspond to a certain kind of run, namely one where at each stage the next cell has become enabled for the first time. For a filiform cds R a state ρ is precisely given by its runs of this kind. For a cds R let TR be the game that has

- sets of moves $(TR)_O = C$ and $(TR)_P = V$;
- · The empty word is in $\mathsf{pos}(TR)$.
  · Given $r \in \mathsf{pos}_P(TR)$, $rc \in \mathsf{pos}_O(TR)$ for c in C if and only if
    c is enabled but not filled in $\lfloor r \rfloor$;
    if $r' \trianglelefteq r$ then c enabled in $\lfloor r' \rfloor$ implies $r = r'$.

  - Given $r \in \mathsf{pos}_O(TR)$, $rv \in \mathsf{pos}(TR)$ for v in V if and only if $r = r'c$ for some $c \in C$ and (c, v) is an event;

Then states on R correspond precisely to strategies on TR: Given such a ρ, let the corresponding strategy be given by

$$\{r \in \mathsf{pos}(TR) \mid \lfloor r \rfloor \in \rho\}.$$

This helps in establishing the following result.

Proposition 1.17 *The categories* **FCDS** *and* **Gam** *are equivalent.*

In [4] the author describes a category of sequential data structures which are something of a hybrid between games and fcdss, but it is clear that his category is equivalent to our **Gam** and **FCDS**. It is the main purpose of this paper to extend this idea to a result for all stable cdss. It is clear that the match between games and general cdss cannot be expected to be as close. It can be improved by considering games where some positions are identified.

2 Games with identified positions

It has been noted variously that there may be reasons why one might want to consider positions in a game only up to some equivalence relation. Graph games [8] are a somewhat extreme version of this approach, and we know of at least two others, by Melliès [10] and Houston [5]. Here we try to keep the theory as simple as possible while still making it general enough to be applicable to other situations.

Our motivation comes from trying to view all cdss as games, that is without restricting to the filiform case. Clearly for that we have to come up with another way of constructing a game from a cds.

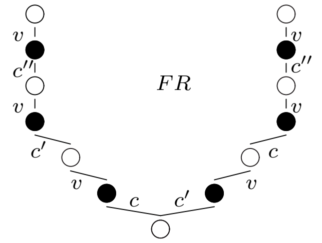
For a cds we want to define a game that contains all the possible runs of the cds. We again turn the cells into Opponent-moves and the values into Player-moves. For filiform cdss we can capture all the structure if we have a cell defining a valid move from a given position if and only if the cell is enabled but not filled in that position, and if the cell is not enabled by any prefix of that position. For general cdss however we have to pick a different way of forming valid plays—we have to drop the condition that the cell not be enabled in an earlier position.

Given a cds R let FR be the game that has:

- sets of moves $(FR)_O = C$ and $(FR)_P = V$;
- · The empty word is in $\mathsf{pos}(FR)$.
  - Given $r \in \mathsf{pos}_P(FR)$, $rc \in \mathsf{pos}_O(FR)$ for c in C if and only if c is enabled but not filled in $\lfloor r \rfloor$.
  - Given $r \in \mathsf{pos}_O(FR)$, $rv \in \mathsf{pos}(FR)$ for v in V if and only if $r = r'c$ for some $c \in C$ and (c, v) is an event;

Remark. There is a one-to-one correspondence between plays of FR and runs of R, whereas in the filiform case for TR it is between plays and restricted runs. Clearly in general, there are more strategies on FR than there are states on R.

Example 2.1 Consider the cds R where $C = \{c, c', c''\}$, $V = \{v\}$, all possible combinations of cells and values are events, and where c and c' are enabled by the empty set whereas $(c, v), (c', v) \vdash c''$. This is the smallest non-filiform cds.



There are four states for the cds, but nine strategies for FR. We aim in what follows to remove at least some undesirable strategies.

One reason for the mismatch between states of R and strategies on FR is that a strategy contains sequentializations of states, and in general a state may have many of those. Moreover, a strategy may be willing to carry out some of these sequentializations, but not others. This problem does not arise for fcdss since there every minimal state containing some event has precisely one sequentialization.

One way of cutting down the number of strategies is by identifying those positions in FR which correspond to the same underlying state, and then demanding that all strategies are well-behaved with respect to that equivalence relation. We want to identify positions $r, r' \in \mathsf{pos}_P(FR)$ if and only if

$$\lfloor r \rfloor = \lfloor r' \rfloor,$$

which is equivalent to saying that the pairs consisting of an Opponent move followed by a Player move performed in r and r' are the same, they merely occur in a different order. We identify $rc, r'c' \in \mathsf{pos}_O(FR)$ if and only if $r \sim r'$ and $c = c'$.

Definition 2.2 A **$\sim$-game** $(A, \sim)$ is a game A with an equivalence relation $\sim$ on positions of A satisfying the following conditions.

- P-positions can be related only to other P-positions, and so the analogue is true for O-positions.
- If r and r' are two $\sim$-related positions then if $ra \in \mathsf{pos}(A)$ then $r'a \in \mathsf{pos}(A)$ and $ra \sim r'a$.
- No position is $\sim$-related to one of its proper prefixes.
- If ra and ra' are $\sim$-related then $a = a'$.

Note that if r and r' are two $\sim$-related positions then the full subtree of the game rooted at r is the same as the subtree of the game rooted at r', including the $\sim$ relation.

Definition 2.3 The **linear function space** of $(A, \sim)$ and $(B, \sim)$ is $(A \multimap B, \sim)$ where for positions t and t' of $A \multimap B$ we set

$$t \sim t' \qquad \text{if and only if} \qquad (t|_A \sim t'|_A \text{ and } t|_B \sim t'_B).$$

An analogous definition is used to define a tensor product for $\sim$-games.

More interestingly, what notion of strategy should we consider for $\sim$-games? Clearly if our motivation for identifying positions is to make sense, then at a minimum one would only want to allow those strategies α on A for which if $r \sim r'$ are O-positions in α then α has an answer to r if and only if it has an answer to r', and the two resulting positions are once more equivalent. A stronger version would be to demand, in this situation, that $ra \in \alpha$ if and only if $r'a \in \alpha$. But it turns out that neither of those is strong enough on its own to ensure that the resulting strategies compose. See [5] for a discussion of this issue in a slightly different setting.

Before we give the definition we introduce the following notation. For two positions r, r' in a $\sim$-game we write

$$r \preceq r' \qquad \text{if and only if} \qquad \text{there exists } \hat{r} \text{ with } r \trianglelefteq \hat{r} \sim r'.$$

In other words r is a play which can become equivalent to r' once further moves have been added. Put differently again, in the acyclic directed graph obtained by identifying those positions which are $\sim$-related there is a path from the node corresponding to (the equivalence class of) r to (that of) r'. We write $r \prec r'$ if $r \preceq r'$ but not $r \sim r'$—in other words, to get from r to a play which is $\sim$-related to r' we have to add at least one move.

If A and B are two games then we can determine the relation $\preceq$ on $A \multimap B$ from those on A and B, although there is a side-condition: If, for example, for positions t and t' in $A \multimap B$ we have $t|_A \sim t'|_A$, $t|_B \prec t'|_B$ and all these are P-positions, then the only way of reaching something related to t' from t would be by making further moves in B. However, if $t|_A$ and $t|_B$ are both P-positions then the next move has to occur in A, so we are 'stuck' on our way to something related to t'.

Lemma 2.4 *Let t and t' be two positions in $A \multimap B$. Then*

$$\begin{array}{lll} t \preceq t' & \textit{if and only if} & (t|_A \preceq t'|_A) \textit{ and } (t|_B \preceq t'|_B) \textit{ and} \\ & & \textit{not } ((t|_A \sim t'|_A,\ t|_B \prec t'_B,\ \ \textit{all } P\textit{-positions}) \\ & & \ \textit{or } (t|_A \prec t'|_A,\ t|_B \sim t'_B,\ \ \textit{all } O\textit{-positions})). \end{array}$$

Definition 2.5 A **$\sim$-strategy** α on $(A, \sim)$ is a strategy α on A such that

- if $r, r' \in \alpha$ are O-positions and $ra \in \alpha$ for some P-move a then $r'a \in \alpha$;
- if $r \in \mathsf{pos}_O(A)$ and $r' \in \mathsf{pos}_P(A)$ are two positions in α with $r \prec r'$ then there exists $ra \in \alpha$ such that $ra \preceq r'$.

Identities are obviously $\sim$-strategies. To show that $\sim$-strategies compose we need an auxiliary result. Let $\xi\colon A \multimap B$ and $\zeta\colon B \multimap C$ be two $\sim$-strategies.

Lemma 2.6 *Let t and t' be two positions in $\zeta \circ \xi$, and let w and w' be the unique minimal joint sequences in accord with ξ and ζ for t and t' respectively, so $w|_{A+C} = t$, $w'|_{A+C} = t'$. If $t \sim t'$ then $w|_B \sim w'|_B$.*

Proof (Sketch) The proof proceeds by induction over the construction of w. So assume we have built $v \trianglelefteq w$ up to length n such that $v|_A \preceq w'|_A$, $v|_B \preceq w'|_B$ and $v|_C \preceq w'|_C$—clearly the empty sequence has these properties. We stop when we have reached $v = w$, so assume that at least one of $v|_A \prec w'|_A$ and $v|_C \prec w'|_C$. There are four cases which can arise when lengthening v by one move, namely $(v|_A, v|_B, v|_C)$ in

- $\mathsf{pos}_P(A) \times \mathsf{pos}_P(B) \times \mathsf{pos}_P(C)$: The next move is for Opponent in C, call it c. But $vc|_C \trianglelefteq w|_C \preceq w'$, and the restrictions to moves from A or B haven't changed.
- $\mathsf{pos}_P(A) \times \mathsf{pos}_P(B) \times \mathsf{pos}_O(C)$: The next move is ζ's answer to $w|_{B+C}$, call it d (from either B or C). But whichever it is, $vd|_{B+C} \preceq w'|_{B+C}$ since ζ is a $\sim$-strategy, and so $vd|_B \preceq w'|_B$ as well as $vd|_C \preceq w'|_C$.
- $\mathsf{pos}_P(A) \times \mathsf{pos}_O(B) \times \mathsf{pos}_O(C)$: This case is dual to the previous one.
- $\mathsf{pos}_O(A) \times \mathsf{pos}_O(B) \times \mathsf{pos}_O(C)$: This case is dual to the first one.

$\square$

Corollary 2.7 *Let t and t' be two positions in $\zeta \circ \xi$, and w and w' be the respective unique joint sequences. Then*

$$t \preceq t' \qquad \text{if and only if} \qquad w|_A \prec w'|_A \text{ and } w|_B \prec w'|_B \text{ and } w|_C \prec w'_C.$$

Proposition 2.8 *The composite of two $\sim$-strategies is a $\sim$-strategy.*

Proof (Sketch) The first condition is shown by making use of the way joint sequences are built; it is a simple case distinction (a new move added to an existing position in a composite can be from either the source or the target) and uses that the composed strategies satisfy that condition. The second condition can be shown by an application of the preceding two Lemmas. $\square$

It is obvious that $(A \otimes B) \multimap C$ is isomorphic to $A \multimap (B \multimap C)$ and so we obtain a symmetric monoidal closed category $\mathbf{Gam}_\ell^\sim$. It is also obvious how to define the product of two $\sim$-games, the equivalence relation is the union of the two equivalence relations. The terminal object has only one position and therefore there is only one equivalence relation turning it into a $\sim$-game. We can give a linear exponential comonad for $\sim$-games, but since we make no use of it here we do not include a definition.

There is an obvious forgetful functor from $\mathbf{Gam}_\ell^\sim$ to $\mathbf{Gam}_\ell$ which preserves the categorical structure and which is faithful. On the other hand if we equip a game with the trivial equivalence relation, where every position is equivalent only to itself, then we obtain a monoidal functor which is full and faithful.

However, these two functors do not form an adjunction. The reason for this is that the only candidates for the unit and co-units are copycat strategies, but the one which lives in $\mathbf{Gam}^{\sim}_{\ell}$ is not a $\sim$-strategy.

Proposition 2.9 *The category* $\mathbf{Gam}^{\sim}_{\ell}$ *is equivalent to the category of graph games* $\mathbf{GGam}$ *in [7].*

Proof (Sketch) We map a $\sim$-game A to the graph game whose positions are equivalence classes of positions A, where there is a move from $[r]$ to $[r']$ if and only if there exists a with $ra \sim r'$. It is obvious what to do for $\sim$-strategies, the main challenge is to prove that the results satisfies the *conflict-freeness* condition, but that's obvious from the condition for $\sim$-strategies. In the opposite direction, map a graph game A to the game whose moves are of the form (rr') whenever $r \longrightarrow r'$ in A. Two positions are equivalent if and only if their last moves end in the same position from A. This is the 'treeification' functor from [7] equipped with the obvious equivalence relation. □

For this application we prefer to work in $\mathbf{Gam}^{\sim}_{\ell}$ because in $\mathbf{GGam}$ we do not have the notion of named moves. Our category bears a vague resemblance to Melliès category of *extensional data structures (edss)* [10]: Given a $\sim$-game, read it as an eds and take the equivalence classes of P-positions as extensions, where such a class realizes all its members. Our $\sim$-strategies are then strategies of edss which implement all extensions. However, Melliès has a more general notion of morphism, so while we get a functor the connection is not close. Further, given an eds there is no obvious way of turning it into a $\sim$-game.

3 Cdss as $\sim$-games

We want to turn the assignment F into a functor $\mathbf{CDS} \longrightarrow \mathbf{Gam}^{\sim}$. Note that

$$F(R \times S) \cong FR \otimes FS \qquad \text{and} \qquad F\mathbf{1} \cong \mathbf{I},$$

so F maps products in $\mathbf{CDS}$ to tensor products in $\mathbf{Gam}^{\sim}_{\ell}$.

For fcdss we had a nice relation between states on R and strategies on TR. If we look at the situation for general cdss and F then we note that if ρ is a state of R then we can derive a $\sim$-strategy $F\rho$ on FR from it. The idea is that $F\rho$ is trying to 'mimic' ρ by attempting to fill all the cells that ρ fills with the appropriate value. Formally, for a play r of FR,

$$r \in F\rho \qquad \text{if and only if} \qquad \lfloor r \rfloor \in \rho.$$

However, in general there are more $\sim$-strategies for FR than there are states of R. The reason for this is that while a $\sim$-strategy might be prepared to fill some cell c with value v under some circumstances there is nothing that compels it to do so whenever it can.

In particular in Example 2.1 consider the strategy $\mathsf{pre}\{cvc', c'vc\}$. The only state this could possibly correspond to is $\rho = \{(c, v), (c', v)\}$, but that is much better interpreted by $F\rho = \mathsf{pre}\{cvc'v,\ c'vcv\}$.

Slightly more generally, consider $F(R \times S) \cong FR \otimes FS$. The only strategies on $F(R \times S)$ which arise from $F\tau$ for some state τ on $R \times S$ are of the form $F\rho \otimes F\sigma$ with $\rho\colon R$ and $\sigma\colon S$.

We have defined an action on objects of **CDS** and it remains to extend it to morphisms to obtain a functor $F\colon \mathbf{CDS} \longrightarrow \mathbf{Gam}^{\sim}_{\ell}$. Assume we have a morphism $\phi\colon R \longrightarrow S$, that is a state ϕ of $R \Rightarrow S$. Then we define $F\phi\colon FR \longrightarrow FS$ to be the set of all runs of ϕ. Note that we can read $\rho\colon R$ as a morphism $\rho\colon \mathbf{1} \longrightarrow R$ which is mapped to $F\rho\colon \mathbf{I} = F\mathbf{1} \longrightarrow FR$, and this corresponds to $F\rho\colon FR$ defined above.

Proposition 3.1 *This defines a faithful functor $F\colon \mathbf{CDS} \longrightarrow \mathbf{Gam}^{\sim}_{\ell}$.*

Proof (Sketch) It is easy to show that every run of ϕ is indeed a play of $FR \multimap FS$, and not much harder to establish that this set defines a $\sim$-strategy. For functoriality all the hard work is done by Proposition 1.13, and we know the assignment is faithful by Lemma 1.12. □

Note that we can describe $F\phi$ inductively by making use of the abstract algorithm corresponding to ϕ, g_ϕ.

Proposition 3.2 *The following are equivalent given $\xi\colon FR \longrightarrow FS$:*

- *there exists $\phi\colon R \Rightarrow S$ with $\xi = F\phi$;*
- *there exists an abstract algorithm $g_\xi\colon \mathsf{st_f}R \times D \rightharpoonup C + W$ defining ξ in the following sense: Let t be an odd-length word in $F\phi$ and let d be the last symbol from D occurring in t. Then ξ's unique answer to t is given by $g_\xi(\lfloor t|_{C+V} \rfloor, d)$ (if g_ξ is undefined at that point then so is ξ);*
- *ϕ maps $\sim$-strategies of the form $F\rho$ for FR by post-composition to $\sim$-strategies of the form $F\sigma$ for FS.*

This condition is a generalization of history-freeness [1] in the sense that if $\alpha\colon FR$ then if α is equal to $F\rho$ for some $\rho\colon R$ if and only if it is history-free.

It is instructive to see a definition of the functor $G\colon \mathbf{CDS} \longrightarrow \mathbf{GGam}$ which is F composed with the equivalence of Proposition 2.9. For a cds R let GR be the graph game that has P-positions the finite states of R, and O-positions such a state ρ together with a cell c enabled but not filled in ρ. Moves are obvious. For a state $\phi\colon R \Rightarrow S$, let $G\phi$ be the strategy defined by

$$G\phi(\rho, (\sigma, d)) = \begin{cases} (\rho, \sigma \cup \{(d, w)\}) & g_\phi(\rho, d) = w \\ ((\rho, c), (\sigma, d)) & g_\phi(\rho, d) = c \\ \text{undefined} & \text{otherwise.} \end{cases}$$

The image of morphisms under G has nice characterizations, but we do not give them here since that would require introducing more material about graph games.

We can identify a subcategory of $\mathbf{Gam}_\ell^\sim$ which is the image of **CDS** under F. For this we generalize the definition of $\lfloor\ \rfloor$ to a play r of an arbitrary game A by setting

$$\lfloor r \rfloor = \{(a, a') \in A_O \times A_P \mid \exists r' \in (A_O A_P)^*.\, r'aa' \trianglelefteq r\}.$$

Proposition 3.3 *Given a $\sim$-game A, there is a cds R with $FR = A$ if and only if for all plays r, r' of A*

- *$rar'a \in \mathsf{pos}_O(A)$ implies r is the empty string (every O-move may occur at most once in every play);*
- *· for $r, r' \in \mathsf{pos}_P(A)$, $r \sim r'$ if and only if $\lfloor r \rfloor = \lfloor r' \rfloor$,*
  · for $ra, r'a' \in \mathsf{pos}_O(A)$, $ra \sim r'a'$ if and only if $r \sim r'$ and $a = a'$
  (two plays are equivalent if they consist of the same (O-move, P-move) pairs possibly in a different order, with the appropriate handling of trailing O-moves);
- *$r \sim r'$ implies $r \wedge r' \in \mathsf{pos}_P(A)$ (equivalent positions diverge at an O-move);*
- *$raa' \in \mathsf{pos}_P(A)$, $r'a \in \mathsf{pos}_O(A)$ imply $r'aa' \in \mathsf{pos}_P(A)$ (if the P-move a' is once a valid answer to the O-move a then this is true everywhere);*
- *$ra \in \mathsf{pos}_O(A)$, $r' \in \mathsf{pos}_P(A)$, $\lfloor r \rfloor \subseteq \lfloor r' \rfloor$, a does not appear in r' imply $r'a \in \mathsf{pos}_O(A)$ (a being enabled remains valid until it has been answered);*
- *$ra, r'a \in \mathsf{pos}_O(A)$ and $r \wedge r' \in \mathsf{pos}_P(A)$ imply $(r \wedge r')a \in \mathsf{pos}_O(A)$ (stability).*

Proof (Sketch) The cds R has cells A_O, values A_P, events all $(a, a') \in A_O \times A_P$ such that there exists $r \in \mathsf{pos}(A)$ with $raa' \in \mathsf{pos}(A)$ and enablings all instances of $\lfloor r \rfloor \vdash a$ such that $ra \in \mathsf{pos}(A)$ and for all $r' \trianglelefteq r$, $r'a \notin \mathsf{pos}(A)$.$\Box$

Let $\mathbf{CDSGam}_\ell^\sim$ be the category given by the image of F.

Theorem 3.4 *The category $\mathbf{CDSGam}_\ell^\sim$ is cartesian closed and equivalent to* **CDS**.

This is somewhat surprising in that this is a cartesian closed category of games which is not based on a decomposition of the function space—instead, the function space is the linear function space of $\mathbf{Gam}_\ell^\sim$ and the product is the tensor product from that category. It is instructive to see why the tensor product for the identified subcategory becomes a product. There are projections for $\otimes$ in $\mathbf{Gam}_\ell$ already, for example $\pi_{\mathsf{l}} \colon A \otimes B \longrightarrow A$ is given by the copy-cat strategy which copies the moves of the right hand copy of A to the left hand copy and *vice versa*, the game B never gets started. In $\mathbf{CDSGam}_\ell^\sim$ these are the images of projections under F. The problem lies in defining the pairing operation.

But in the image of F we have a diagonal $\delta_R \colon FR \longrightarrow FR \otimes FR$ (the image of the diagonal $R \longrightarrow R \times R$ in **CDS**): Given an O-position t in δ_R we calculate its restriction to all three components to obtain three plays r, r' and r'' of FR, where r is an interleaving of r' and r'', plus or minus one extra move at the end. If there is an extra move in r we copy it as our answer to t, else there is an extra move to one of r' or r'', and we copy that. Note that this is only possible because we know that the appropriate move is available in FR, so we are making considerable use of the special nature of games of this form.

For $F\phi\colon FR \multimap FS$, $F\psi\colon FR \multimap FT$ we set $\langle F\phi, F\psi\rangle\colon FR \longrightarrow FS \otimes FT$ to

$$FR \xrightarrow{\delta_R} FR \otimes FR \xrightarrow{F\phi \otimes F\psi} FS \otimes FT,$$

and this is equal to $F\langle \phi, \psi\rangle\colon FR \longrightarrow F(S \times T) \cong FS \otimes FT$.

Note that in order to compose strategies in our cartesian closed category we can just use the composition of linear morphisms and do not have to go *via* some Kleisli category. In some ways, objects of the form FR behave a bit like an object of type $!A$, but to our knowledge there is no linear exponential which would allow us to write FR as such.

Conclusions and future work. We have established a category of games which is equivalent to the category of concrete data structures. This yields a description of a cartesian closed category of games which is not the Kleisli-category for some linear exponential. For this work it was vital to define a category of games with the additional structure of an equivalence relation on positions. This category is equivalent to the category of graph games, but it has named moves without which a cartesian closed subcategory cannot be defined. For this category we have to restrict to games of a very special nature and to morphisms which can be described using a partial function.

Concrete data structures can be viewed as special event structures, and we argue above that some events in a state are independent from others, yielding a primitive notion of *concurrency*. An interesting question is whether these games can be used to model more elaborate versions. But the category $\mathbf{Gam}_\ell^\sim$ throws up other questions: What is so special about this choice of game and strategy?

References

[1] Abramsky, S. and R. Jagadeesan, *Games and full completeness for multiplicative linear logic*, J. Symbolic Logic **59** (1994), pp. 543–574.

[2] Amadio, R. and P.-L. Curien, “Domains and Lambda-Calculi,” Number 46 in Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, 1998.

[3] Curien, P.-L., “Categorical Combinators, Sequential Algorithms and Functional Programming,” Birkhäuser, 1993, 2nd edn. edition.

[4] Curien, P.-L., *On the symmetry of sequentiality*, in: *Proceedings of MFPS93*, Lecture Notes in Computer Science **802** (1994), pp. 29–71.

[5] Houston, R., "Categories of Games," Master's thesis, University of Manchester (2003).

[6] Hyland, M. and A. Schalk, *Abstract games for linear logic. Extended abstract*, in: *Proceedings of CTCS '99*, Electronic Notes on Theoretical Computer Science **29**, 1999.

[7] Hyland, M. and A. Schalk, *Games on graphs and sequentially realizable functionals. Extended abstract*, in: *Proc. Logic in Computer Science 2002* (2002), pp. 257–264.

[8] Hyland, M. and A. Schalk, *Glueing and orthogonality for models of linear logic*, Theoretical Computer Science **294** (2003), pp. 183–231.

[9] Kahn, G. and G. Plotkin, *Concrete domains*, Theoretical Computer Science **121** (1993), pp. 187–277.

[10] Melliés, P.-A., *Sequential algorithms and strongly stable functions*, to appear in the special issue of Theoretical Computer Science: Game Theory Meets Theoretical Computer Science.