Allegories of Circuits

Carolyn Brown
carolynb@cogs.susx.ac.uk

Alan Jeffrey
alanje@cogs.susx.ac.uk

School of Cognitive and Computing Sciences
University of Sussex, Falmer, Brighton BN1 9QH, UK

Alan Jeffrey is funded by SERC project GR/H 16537

Abstract

This paper presents three paradigms for circuit design, and investigates the relationships between them. These paradigms are *syntactic* (based on Freyd and Scedrov's *unitary pre-tabular allegories* (*upas*)), *pictorial* (based on the *net list* model of circuit connectivity), and *relational* (based on Sheeran's relational model of circuit design RUBY). We show that net lists over a given signature Σ constitute the free upa on Σ. Our proof demonstrates that nets and upas are equally expressive, and that nets provide a normal form for both upas and pictures. We use Freyd and Scedrov's representation theorem for upas to show that our relational interpretations constitute a sound and complete class of models for the upa axioms. Thus we can reason about circuits using either the upa axioms, pictures or relations. By considering garbage collection, we show that there is no faithful representation of nets in **Rel**: we conjecture that a semantics for nets which takes garbage collection into account is faithfully representable in **Rel**.

1 Introduction

Relational languages such as Sheeran's circuit design language RUBY [6] are used to derive hardware circuits from abstract specifications of their behaviour. Freyd and Scedrov [3] have recently introduced *allegories*, which generalise the notion of sets and relations in the same sense that categories generalise the notion of sets and functions. In [1], Brown and Hutton showed how allegories could be used to reason about a pictorial representation of RUBY programs. In this paper, we develop the mathematical notions used in [1], and give sketch proofs of expressiveness, normalisation and completeness. Our results provide new methods of formal reasoning about relational programs.

We define a syntax for hardware circuit design, based on *unitary pretabular allegories*, which are allegories with a certain product structure. We also give a categorical presentation of the *net list* model of circuit connectivity. We prove these equivalent by showing that, given a signature Σ of basic circuit components, the net lists over

$$\begin{array}{rclrclrcl}
\mathrm{id}_X^\circ & = & \mathrm{id}_X & (f;g)^\circ & = & g^\circ;f^\circ & f \cap (g \cap h) & = & (f \cap g) \cap h \\
f^{\circ\circ} & = & f & f \cap f & = & f & f;(g \cap h) & = & (f;g) \cap f;(g \cap h) \cap (f;h) \\
f \cap g & = & g \cap f & (f \cap g)^\circ & = & g^\circ \cap f^\circ & (f;g) \cap h & = & (f;g) \cap h \cap (f \cap (h;g^\circ));g
\end{array}$$

Table 1: Axioms for an allegory

$$\begin{array}{rclrcl}
f \cap \top_{X,Y} & = & f & (\mathrm{fst}^\circ_{X,Y};\mathrm{fst}_{X,Y}) \cap \mathrm{id}_X & = & \mathrm{id}_X \\
\mathrm{fst}^\circ_{X,Y};\mathrm{snd}_{X,Y} & = & \top_{X,Y} & (\mathrm{snd}^\circ_{X,Y};\mathrm{snd}_{X,Y}) \cap \mathrm{id}_Y & = & \mathrm{id}_Y
\end{array}$$

$$(\mathrm{fst}_{X,Y};\mathrm{fst}^\circ_{X,Y}) \cap (\mathrm{snd}_{X,Y};\mathrm{snd}^\circ_{X,Y}) \;=\; \mathrm{id}_{X \otimes Y}$$

Table 2: Additional axioms for a upa

Σ constitute the free unitary pretabular allegory on Σ. Our proof develops a normal form for both pictures and net lists, and expressiveness results for our allegorical syntax, for pictures and for net lists. Finally, we show that relational interpretations in the style of RUBY constitute a sound and complete class of models for our axioms.

These results enable us to reason about circuits using either the upa axioms, pictures or relations.

2 Allegories as a syntax for circuits

We now recall Freyd and Scedrov's [3] definition of *allegory*, which underlies our equational theory of circuit design.

Definition. An *allegory* is a category equipped with:

- a morphism $f^\circ : Y \to X$ for each morphism $f : X \to Y$.
- a morphism $f \cap g : X \to Y$ for each pair of morphisms $f, g : X \to Y$.

satisfying the equations in Table 1. □

Example. The category **Rel** of sets and relations is an allegory with $(-)^\circ$ given by relational converse and $\cap$ by intersection. □

A morphism $f : X \to Y$ in an allegory is *entire* iff $(f;f^\circ) \cap \mathrm{id}_X = \mathrm{id}_X$, is *simple* iff $(f^\circ;f) \cap \mathrm{id}_X = (f^\circ;f)$, and is a *map* iff it is entire and simple.

Definition. Morphisms f and g with common source X are *jointly monic* iff $(f;f^\circ) \cap (g;g^\circ) = \mathrm{id}_X$. Two maps $g : Z \to X$ and $h : Z \to Y$ *tabulate* a morphism $f : X \to Y$ if g and h are jointly monic and $f = g^\circ;h$. An allegory is *pretabular* iff for every morphism $f : X \to Y$ we can find a tabulated morphism $g : X \to Y$ such that $f \cap g = f$. □

Example. Let $f : X \to Y$ in **Rel**. Then

- f is entire iff $\forall x \,.\, \exists y \,.\, \langle x, y\rangle \in f$
- f is simple iff whenever $\langle x, y\rangle \in f$ and $\langle x, z\rangle \in f$ we have $y = z$.

- f is a map in f is a function.

□

Definition. An allegory is *unitary* iff it has:

- an object U such that $f = f \cap \mathrm{id}_U$ for any $f : U \to U$.
- an entire morphism $u_X : X \to U$ for each object X.

A *unitary pretabular allegory homomorphism* $F : \mathbf{A} \to \mathbf{A}'$ is a functor such that:

$$\begin{array}{rclcrcl} F(f^\circ) & = & (Ff)^{\circ'} & & F(f \cap g) & = & Ff \cap' Fg \\ F(U) & = & U' & & F(u_X) & = & u'_{FX} \end{array}$$

UPA is the category of unitary pretabular allegories (upas) and upa homomorphisms. □

Proposition 1 *Given $f, g : X \to Y$, we define $f \subseteq_{X,Y} g$ iff $f \cap g = f$. Then $(\mathbf{A}(X,Y), \subseteq_{X,Y})$ is a partial order with meet $\cap$ and top $\top_{X,Y} = u_X ; u_Y^\circ$.*

Definition. An allegory has *local products* iff it is unitary and has for each pair of objects X and Y:

- an object $X \otimes Y$
- morphisms $\mathrm{fst}_{X,Y} : X \otimes Y \to X$ and $\mathrm{snd}_{X,Y} : X \otimes Y \to Y$ which tabulate $\top_{X,Y}$. □

Example. **Rel** has local products, given by $X \otimes Y \stackrel{\mathrm{def}}{=} \{\langle x, y\rangle \mid x \in X \text{ and } y \in Y\}$ and the usual projection functions $\mathrm{fst}_{X,Y} : \langle x, y\rangle \mapsto x$ and $\mathrm{snd}_{X,Y} : \langle x, y\rangle \mapsto y$. **Rel** has unit 1, the singleton set, and is pretabular. In fact, any arrow $R : X \to Y$ in **Rel** is tabulated by the projections from $\{\langle x, y\rangle \mid x \; R \; y\}$. □

Proposition 2 *An allegory is unitary pretabular iff it has local products.*

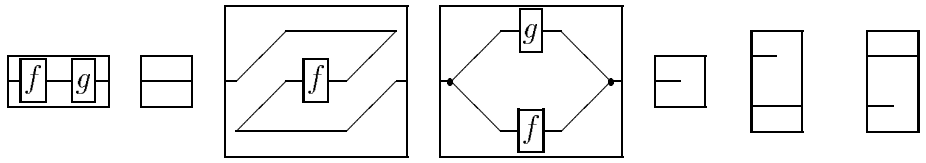
Proposition 2 allows us to extend the axioms of Table 1 to provide an axiomatisation of equality of arrows in a upa. The additional axioms are given in Table 2.

A circuit evidently defines a relation between its input values (the values carried on its source wires) and its output values (the values carried on its target wires). Any two points on the same wire are constrained to carry equal values, while components impose more complex constraints on the values carried by their input and output wires. We view the discrete components of a circuit as arrows in a upa whose objects are types and build complex circuits from the basic components using the operations of a upa. Intuitively, these operations are those of relational algebra augmented with parallel composition and projection. We show in Section 5 that any finite circuit can be built from basic components using the upa operations, and in Section 6 that the upa axioms are sound and complete with respect to the interpretation of circuits as relations.

Arrows in a upa, regarded as terms built up using the upa operations, have a natural representation (made precise in [1]) as *pictures* in which boxes represent generating relations (the circuit components) and lines indicate wiring connections.[1]

[1] We follow the circuit design convention of indicating intersecting wires with blobs '•'.

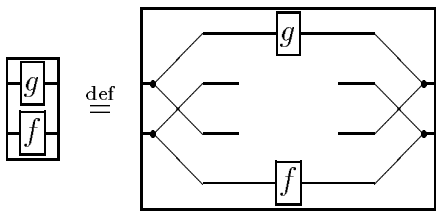
The basic operations of a dpa have the following pictures:



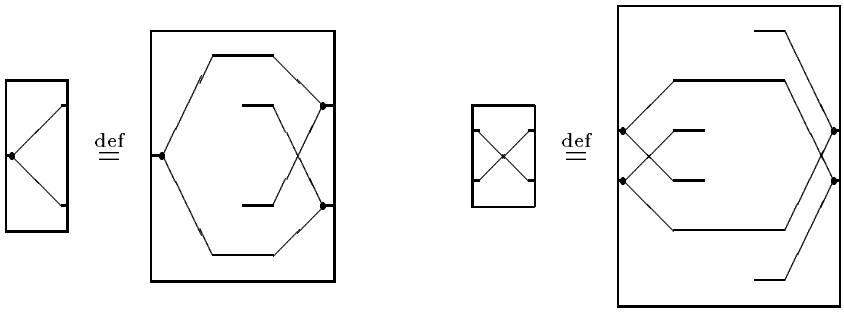
$f;g$ $\quad \mathrm{id}_X$ $\quad f^\circ$ $\quad f \cap g$ $\quad u_X$ $\quad \mathrm{fst}_{X,Y}$ $\quad \mathrm{snd}_{X,Y}$

Note that, whatever value is input to u_X, no value will be output. The input wire is dangling in the sense that it is not connected either to another external connector or to a component.

If $f : W \to X$ and $g : Y \to Z$ then the parallel composition of the circuits f and g is given by the derived operation of local product, $f \otimes g : W \otimes Y \to X \otimes Z \stackrel{\text{def}}{=} (\mathrm{fst}_{W,Y}; f; \mathrm{fst}^\circ_{X,Z}) \cap (\mathrm{snd}_{W,Y}; g; \mathrm{snd}^\circ_{X,Z})$ and has picture:



Two useful derived operations on wires are $\mathrm{fork}_X : X \to (X \otimes X)$, and $\mathrm{swap}_{X,Y} : (X \otimes Y) \to (Y \otimes X)$, which are pictured thus:

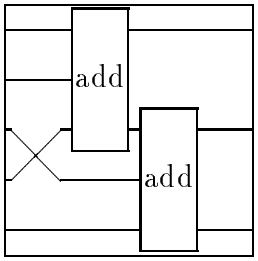


$\mathrm{fork}_X \stackrel{\text{def}}{=} \mathrm{fst}^\circ_{X,X} \cap \mathrm{snd}^\circ_{X,X}$ $\qquad$ $\mathrm{swap}_{X,Y} \stackrel{\text{def}}{=} (\mathrm{fst}_{X,Y}; \mathrm{snd}^\circ_{Y,X}) \cap (\mathrm{snd}_{X,Y}; \mathrm{fst}^\circ_{Y,X})$

The picture below shows how to build a circuit to add two 2-bit binary numbers by combining two circuits which add single bit binary numbers. This picture represents the allegory morphism $\mathrm{2add} : Y \otimes Y \otimes Y \otimes Y \otimes X \to Y \otimes Y \otimes X$, given by

$$\mathrm{2add} \stackrel{\text{def}}{=} (\mathrm{id}_Y \otimes \mathrm{swap}_{Y,Y} \otimes \mathrm{id}_Y \otimes \mathrm{id}_X); (\mathrm{id}_Y \otimes \mathrm{id}_Y \otimes \mathrm{add}); (\mathrm{add} \otimes \mathrm{id}_Y)$$

where Y is the type of booleans and X is the type of the carry bit.



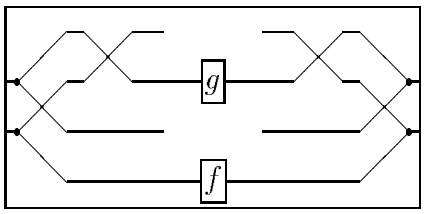
If we take Y to be instead the type of n bit binary numbers, this picture becomes a template for building a $2n$-bit adder from two n bit adders.

The picture of this circuit is much easier to read than the syntactic term. In Section 4 we represent pictures categorically as *nets*, and in Section 5 we show that reasoning with terms is sound and complete for nets.

We remark that pictures provide an elegant proof that **UPA** is isomorphic to **DCB**, the category of *discrete cartesian bicategories* (dcbs) [2] and structure-preserving functors. We define the structure of a dcb in terms of the structure of a upa, and conversely. The proof that these assignments are mutually inverse is facilitated by our pictorial representation: for example, the term obtained by translating $f \otimes g : X \otimes Y \to W \otimes Z$ into a dcb and back is:

$$\begin{aligned}&\mathrm{fork}_{X,Y}; (((\mathrm{id}_X \otimes u_Y); \rho_X; f; \rho_W^\circ; (\mathrm{id}_W \otimes u_Z^\circ))\\ &\quad \otimes (\mathrm{swap}_{X,Y}; (\mathrm{id}_Y \otimes u_X); \rho_Y; g; \rho_Z^\circ; (\mathrm{id}_Z \otimes u_W^\circ); (\mathrm{swap}_{W,Z})^\circ)); \mathrm{fork}_{W,Z}^\circ\end{aligned}$$

The picture of this term given below makes clear why the term equals $f \otimes g$, since the shape of wires is unimportant and the dangling wires can be removed without affecting the relation defined by the circuit:



3 Signatures

We now give a semantics for circuits as allegories generated by typed components from a *signature*.

Definition. A *signature* Σ is a 4-tuple $(\mathrm{Sort}_\Sigma, \mathrm{Comp}_\Sigma, \mathrm{source}_\Sigma, \mathrm{target}_\Sigma)$ where:

- Sort_Σ is a set of *sorts*.
- Comp_Σ is a set of *components*.
- source_Σ and target_Σ are functions from Comp_Σ to finite lists of Sort_Σ.

We shall write $c : \vec{\sigma} \to \vec{\tau}$ in Σ iff $\mathrm{source}_\Sigma(c) = \vec{\sigma}$ and $\mathrm{target}_\Sigma(c) = \vec{\tau}$. □

Each signature Σ provides certain "building blocks" for circuits: Sort_Σ gives the types of the wires in the circuit, Comp_Σ gives the components used in the circuit (each of which may be used many times) and source_Σ and target_Σ give respectively the input and output types of these components.

Example. The signature Σ_n has sorts $1, \ldots, n$ and no components. Σ_n can be regarded as a list of n distinct wires.

Given $c : \sigma_1 \ldots \sigma_m \to \sigma_{m+1} \ldots \sigma_{m+n}$, the signature Σ_c has sorts $1, \ldots, m+n$ and component c. Thus Σ_c lists $n+m$ wires and indicates where each wire connects to c.

The signature Σ_{add} for one-bit adders has one sort Bool, and one component $\mathrm{add} : \mathrm{Bool} \otimes \mathrm{Bool} \otimes \mathrm{Bool} \to \mathrm{Bool} \otimes \mathrm{Bool}$. Thus Σ_{add} indicates that a one-bit adder has three boolean inputs and two boolean outputs.

The template for building a $2n$ bit adder from two n bit adders has signature Σ_{2add} with nine sorts $1 \ldots 9$, and two components $a : 1 \otimes 2 \otimes 4 \to 6 \otimes 9$ and $b : 9 \otimes 3 \otimes 5 \to 7 \otimes 8$. Thus Σ_{2add} describes a circuit containing two components wired together with nine wires as shown on page 2. We use nine sorts because each wire drawn in the picture plays a distinct rôle: the template does not identify any two of the wires depicted. □

Definition. A *signature morphism* $F : \Sigma \to \Sigma'$ assigns:

- a sort $F\sigma$ in Σ' to each sort σ in Σ.
- a component $Fc : F\vec{\sigma} \to F\vec{\tau}$ in Σ' to each component $c : \vec{\sigma} \to \vec{\tau}$ in Σ.

Sig is the category of signatures and signature morphisms. □

Example. Given sorts $\sigma_1 \ldots \sigma_n$ from Σ, the signature morphism $L_{\sigma_1 \ldots \sigma_n} : \Sigma_n \to \Sigma$ which maps i to σ_i is called a *labelling*. A labelling assigns a rôle to each wire in a list. The degree of information given by a labelling may vary: L_{σ_1} may be a type, or a type and a generic position in the circuit (for example, the carry bit input to an adder), or a type and a particular position in the circuit (for example, the carry bit input to the leftmost adder).

Given $c : \sigma_1 \ldots \sigma_m \to \sigma_{m+1} \ldots \sigma_{m+n}$ in Σ:

- the morphism $M_c : \Sigma_c \to \Sigma$ maps i to σ_i, and c to c.
- the labelling $S_c : \Sigma_m \to \Sigma_c$ maps i to σ_i.
- the labelling $T_c : \Sigma_n \to \Sigma_c$ maps i to σ_{m+i}.

M_c assigns an appropriate type to each wire connected to the component c.

For example, the circuit which builds a two-bit adder from two one-bit adders is the signature morphism $M_{\text{2add}} : \Sigma_{\text{2add}} \to \Sigma_{\text{add}}$ which maps each sort to Bool and each component to add. The signature Σ_{2add} expresses the connectivity of the circuit in terms of uninterpreted wire positions and component positions, while M_{2add} assigns a meaning to each wire and each component in terms of sorts and components available in the signature Σ_{add}. □

It is straightforward to prove the following proposition.

Proposition 3 **Sig** *has finite coproducts and coequalisers, and hence pushouts.*

Definition. A Σ-allegory is a upa with:

- an object X_σ for each sort σ in Σ.
- a morphism $f_c : X_{\sigma_1} \otimes \cdots \otimes X_{\sigma_m} \to X_{\tau_1} \otimes \cdots \otimes X_{\tau_n}$ for each component $c : \sigma_1 \ldots \sigma_m \to \tau_1 \ldots \tau_n$ in Σ.

A Σ-allegory morphism $F : \mathbf{A} \to \mathbf{A}'$ is an allegory morphism such that:

$$F(X_\sigma) = X'_\sigma \qquad F(f_c) = f'_c$$

Let Σ**Alleg** be the category of Σ-allegories with Σ-allegory morphisms. □

Example. The free Σ-allegory, Free(Σ) has:

- objects finite lists over Sort_Σ
- morphisms generated by Comp_Σ, together with $\text{fst}_{X,Y}, \text{snd}_{X,Y}$ and u_X for each pair of objects X and Y, subject to the equivalence $\vdash e = f$ if e and f are equal under the equivalence generated by the equivalence of Tables 1 and 2. □

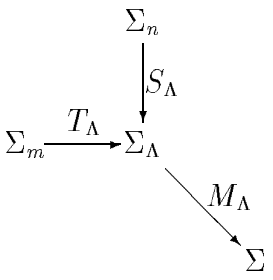
1 Nets

Net lists are a model of circuit connectivity, and are used in circuit extraction [4] and simulation [5]. A net list consists of

- a list of *elements*, a list of *wires*, a list of *input wires* and a list of *output wires*,
- an *assignment* of components to elements, and sorts to wires,
- *connectivity* information saying which wires are connected to which elements, and
- *geometric* information giving the size and position of each element.

We regard elements and wires as respectively the components and sorts of a signature Σ_Λ, and the assignment as a signature morphism from Σ_Λ to Λ. These considerations lead to the following definition of a Σ-*net*, which abstracts away from the geometric information contained in a net list.

Definition. A Σ-*net* $\Lambda : \sigma_1 \ldots \sigma_m \to \tau_1 \ldots \tau_n$ is a 4-tuple $(\Sigma_\Lambda, M_\Lambda, S_\Lambda, T_\Lambda)$ where Σ_Λ is a finite signature, S_Λ and T_Λ are labellings, and:

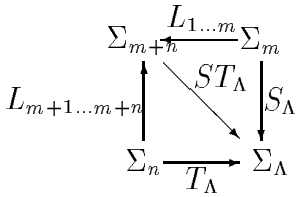


is a diagram in **Sig**. □

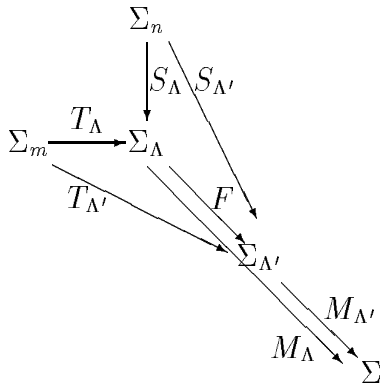
We can evidently view the Σ-net Λ as a multi-graph with an edge for each sort in Σ_Λ and a vertex $M_\Lambda(c)$ for each component c in Σ_Λ. A Σ-net expresses circuit connectivity using names whereas a picture makes connectivity explicit: thus a given Σ-net represents many pictures. Every picture determines a Σ-net uniquely up to the equivalence $\simeq$ defined below.

Example. The Σ_{add}-net which builds a two-bit adder from two one-bit adders is $(\Sigma_{\text{2add}}, M_{\text{2add}}, L_{12345}, L_{678})$. The picture on page 2 is a picture representing the multi-graph Σ_{add}. □

Let ST_Λ be the labelling given by:



A Σ-*net morphism* $F : \Lambda \to \Lambda'$ is a signature morphism $F : \Sigma_\Lambda \to \Sigma_{\Lambda'}$ such that:

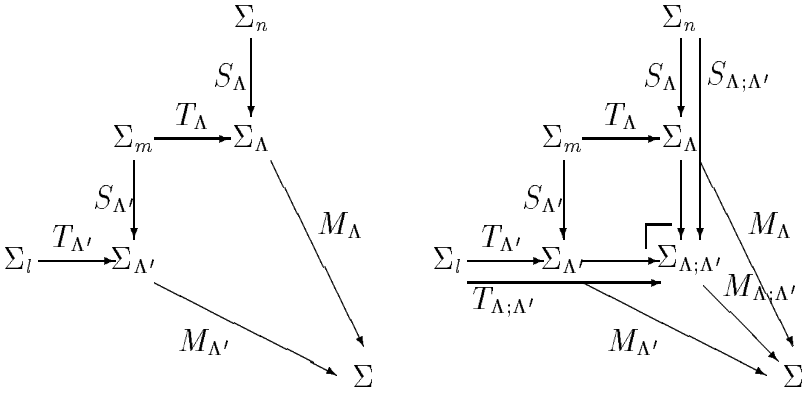


commutes. We write $\Lambda \subseteq \Lambda'$ iff there is a morphism $F : \Lambda \to \Lambda'$ and $\Lambda \simeq \Lambda'$ iff $\Lambda \subseteq \Lambda' \subseteq \Lambda$. If $\Lambda \simeq \Lambda'$ we say that Λ is *bihomomorphic* to Λ'.

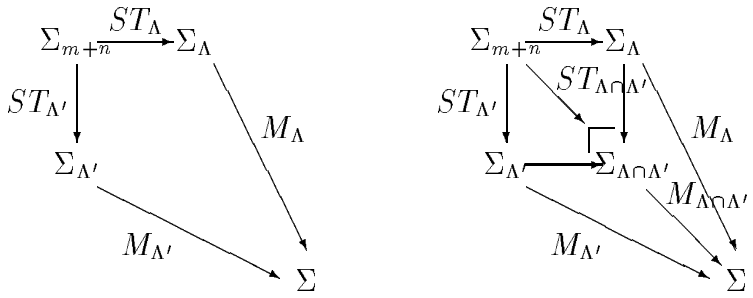
We shall now show that Σ-nets form the arrows of a Σ-allegory. This allows us to combine nets using the upa operations, and to prove a close correspondence between nets and pictures of circuits.

We first construct the composition and meet of two nets using pushouts in **Sig** (which exist by Proposition 3).

Given nets Λ and Λ' such that the diagram on the left commutes, we define the net $\Lambda; \Lambda'$ by the pushout on the right:



Given nets Λ and Λ' such that the diagram on the left commutes, we define the net $\Lambda \cap \Lambda'$ by the pushout on the right:



Given $\vec{\sigma} = \sigma_1 \ldots \sigma_m$, $\vec{\tau} = \tau_1 \ldots \tau_n$, and $c : \vec{\sigma} \to \vec{\tau}$, define:

$$\Lambda^\circ \stackrel{\text{def}}{=} (\Sigma_\Lambda, M_\Lambda, T_\Lambda, S_\Lambda)$$

$$
\begin{aligned}
\mathrm{id}_{\vec{\sigma}} &\stackrel{\mathrm{def}}{=} (\Sigma_m, L_{\vec{\sigma}}, L_{1\ldots m}, L_{1\ldots m}) \\
u_{\vec{\sigma}} &\stackrel{\mathrm{def}}{=} (\Sigma_m, L_{\vec{\sigma}}, L_{1\ldots m}, L_{\epsilon}) \\
\mathrm{fst}_{\vec{\sigma},\vec{\tau}} &\stackrel{\mathrm{def}}{=} (\Sigma_{m+n}, L_{\vec{\sigma}\vec{\tau}}, L_{1\ldots m+n}, L_{1\ldots m}) \\
\mathrm{snd}_{\vec{\sigma},\vec{\tau}} &\stackrel{\mathrm{def}}{=} (\Sigma_{m+n}, L_{\vec{\sigma}\vec{\tau}}, L_{1\ldots m+n}, L_{m+1\ldots m+n}) \\
\Lambda_c &\stackrel{\mathrm{def}}{=} (\Sigma_c, M_c, S_c, T_c)
\end{aligned}
$$

Proposition 4 *Let* $\mathrm{Net}(\Sigma)$ *be the category where*

- *objects are finite vectors of sorts,*
- *morphisms from* $\vec{\sigma}$ *to* $\vec{\tau}$ *are nets* $\Lambda : \vec{\sigma} \to \vec{\tau}$ *(considered up to* $\simeq$*) such that* $S_\Lambda; M_\Lambda = L_{\vec{\sigma}}$ *and* $T_\Lambda; M_\Lambda = L_{\vec{\tau}}$*, and*
- *composition and identities are those defined above.*

$\mathrm{Net}(\Sigma)$ *is a* Σ*-allegory, with the upa structure defined above.*

5 Nets are free

In this section, we sketch the proof that $\mathrm{Net}(\Sigma)$ is isomorphic to $\mathrm{Free}(\Sigma)$. Note that, since $\mathrm{Free}(\Sigma)$ is the initial Σ-allegory, there is a unique Σ-allegory morphism:

$$\nu : \mathrm{Free}(\Sigma) \to \mathrm{Net}(\Sigma)$$

In this section, we define a Σ-allegory morphism:

$$\lambda : \mathrm{Net}(\Sigma) \to \mathrm{Free}(\Sigma)$$

and show that ν and λ form an iso.

Definition. A *pre-net* Π over Σ is a pair (Σ_Π, M_Π) where Σ_Π is a finite signature and $M_\Pi : \Sigma_\Pi \to \Sigma$ a signature morphism. A *pre-net morphism* $F : \Pi \to \Pi'$ is a signature morphism $F : \Sigma_\Pi \to \Sigma_{\Pi'}$ such that $M_\Pi = F; M_{\Pi'}$. □

A pre-net can be thought of as a net in which input and output have not been specified. Each prenet Π with sorts $\{\sigma_1, \ldots \sigma_m\}$ determines an object $\beta(\Pi)$ of $\mathrm{Free}(\Sigma)$, via

$$\beta(\Pi) = M_\Pi \sigma_1 \otimes \cdots \otimes M_\Pi \sigma_m.$$

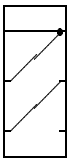
$\beta(\Pi)$ is the bus comprising the wires of Π.

Each prenet morphism $F : \Pi \to \Pi'$ determines a morphism $\pi(F) : \beta(\Pi) \to \beta(\Pi')$ in $\mathrm{Free}(\Sigma)$, via

$$\pi(F) \stackrel{\mathrm{def}}{=} \bigcap_{\substack{i \le m \\ F(\sigma_i) = \tau_j}} i\mathrm{th}_{\beta(\Pi)}; j\mathrm{th}^\circ_{\beta(\Pi')}$$

where Π' has sorts $\tau_1, \ldots \tau_n$ and $i\mathrm{th}_{\sigma_1 \ldots \sigma_n} : \sigma_1 \ldots \sigma_n \to \sigma_i$ is the ith projection function.

Pictures of the terms $\pi(F)$ have no components and connect each input wire to precisely one output wire. For example:

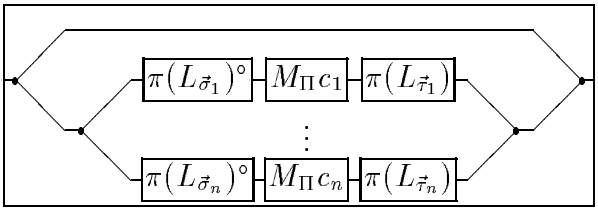


We think of $\pi(\Gamma)$ as providing the interface between the bus of wires of Π and the bus of wires of Π'.

Each pre-net Π determines a morphism $\pi(\Pi) : \beta(\Pi) \to \beta(\Pi)$ in Free(Σ), via

$$\pi(\Pi) \stackrel{\text{def}}{=} \bigcap_{c\,:\,\vec{\sigma} \to \vec{\tau}\ \text{in}\ \Sigma_\Pi} \pi(L_{\vec{\sigma}})^\circ ; M_\Pi c ; \pi(L_{\vec{\tau}}) \cap \mathrm{id}_{\beta(\Pi)}$$

$\pi(\Pi)$ is pictured:



$\pi(\Pi)$ is obtained by composing all components of Π in parallel. For each i, we use the wiring interfaces $\pi(L_{\vec{\sigma}_i})$ and $\pi(L_{\vec{\tau}_i})$ to obtain the appropriate inputs and ouputs to component $M_\Pi c_i$.

Definition. We define λ to be the identity on objects and to map the morphism $\Lambda : \vec{\sigma} \to \vec{\tau}$ in Net(Σ) to

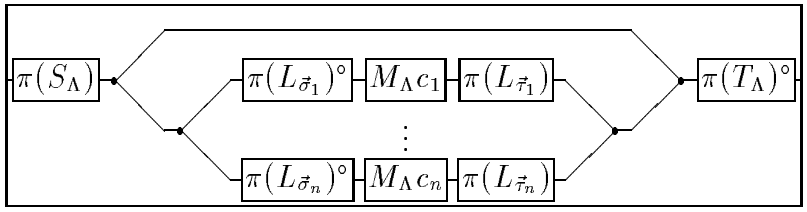
$$\pi(S_\Lambda) ; \pi(\Sigma_\Lambda, M_\Lambda) ; \pi(T_\Lambda)^\circ : \lambda(\vec{\sigma}) \to \lambda(\vec{\tau})$$

where:

$$S_\Lambda : (\Sigma_m, S_\Lambda ; M_\Lambda) \to (\Sigma_\Lambda, M_\Lambda)$$
$$T_\Lambda : (\Sigma_n, T_\Lambda ; M_\Lambda) \to (\Sigma_\Lambda, M_\Lambda)$$

are pre-net morphisms. □

$\lambda(\Lambda)$ is pictured:



Proposition 5 makes precise our intuition that $\pi(\Pi)$ is the normal form of the pre-net Π and $\lambda(\Lambda)$ the normal form of the net Λ. The proof of this proposition includes a proof of the expressiveness of nets (every term is provably equal to a term in the image of λ), the expressiveness of terms (every net is bihomomorphic to a net in the image of ν) and of normalisation (equal nets have equal images under λ).

Proposition 5

1. *If* $\Lambda \subseteq \Lambda'$ *then* $\vdash \lambda(\Lambda) \subseteq \lambda(\Lambda')$.
2. $\vdash e = \lambda(\nu\, e)$ *for any* e.
3. $\Lambda \simeq \nu(\lambda\, \Lambda)$ *for any* Λ.
4. λ *and* ν *are* Σ*-allegory morphisms.*

Thus λ *and* ν *form a* Σ*-allegory isomorphism.*

Proof. 1, 3 and 4 are straightforward. We shall prove 2.

First, define a upa term to be *standard* iff it is given by:

$$e ::= c \mid \mathrm{id}_X \mid e; \mathrm{fst}_{X,Y} \mid e; \mathrm{snd}_{X,Y} \mid \mathrm{fork}_X; e \mid e \otimes e \mid e^\circ$$

We prove by induction on the structure of a upa term t that there is a standard s such that $\vdash t = s$. This proof uses the identities:

$$\begin{aligned} \vdash e; f &= \mathrm{fst}^\circ_{X,Z}; \mathrm{fork}_{X\otimes Z}; (e \otimes f^\circ \otimes \mathrm{snd}_{X,Z}); (\mathrm{fork}^\circ_Y \otimes \mathrm{id}_Z); \mathrm{snd}_{Y,Z} \\ \vdash e \cap f &= \mathrm{fork}_X; (e \otimes f); \mathrm{fork}^\circ_Y \end{aligned}$$

Second, we show by induction on e that if e is standard then $\vdash e = \lambda(\nu\, e)$. The only difficult case is when $e = \mathrm{fork}_X; f$. Let:

$$F : \Sigma_{\nu\, f} \to \Sigma_{\nu\, \mathrm{fork}_X; \nu\, f}$$

be the signature morphism given by the pushout diagram for the composition $\nu\, \mathrm{fork}_X; \nu\, f$. Diagram chasing shows that $F : \nu\, f \to \nu\, \mathrm{fork}_X; \nu\, f$ is a net morphism. Then we show:

$$\begin{aligned} \vdash \mathrm{fork}_{\vec{\sigma}}; \pi(S_{\nu\, f}); &= \pi(S_{\nu\, \mathrm{fork}_{\vec{\sigma}}; \nu\, f}); \pi(F)^\circ \\ \vdash (g \cap h); \pi(F) &= (g; \pi(F)) \cap (h; \pi(F)) \end{aligned}$$

By equational reasoning, we show that:

$$\vdash \mathrm{fork}_{\vec{\sigma}}; f = \lambda(\nu(\mathrm{fork}_{\vec{\sigma}}; f))$$

The other cases are simpler. □

6 Relations

Our model of circuit design is closely related to Sheeran's [6] relational model RUBY. In this section we show that Sheeran's relational interpretation is sound and complete for the axioms of a upa. This result shows that we can reason about equality of RUBY programs using the upa axioms, and that these axioms suffice to prove the equality of any two programs which compute the same relation.

Definition. Let **Rel** be the allegory of sets and relations. A Σ-*interpretation* ρ assigns:

- a set $\rho(\sigma)$ to each sort σ in Σ.
- a relation $\rho(c) : \rho(\sigma_1) \times \cdots \times \rho(\sigma_m) \to \rho(\tau_1) \times \cdots \times \rho(\tau_n)$ to each component $c : \sigma_1 \ldots \sigma_m \to \tau_1 \ldots \tau_n$ in Σ.

We write $[\![-]\!]_\rho$ for the homomorphic extension of ρ to Σ-nets. □

Theorem 6 *There is a faithful representation (that is, a faithful morphism in* **UPA***) from* Free(Σ) *to* $\mathbf{Rel}^J$ *where J indexes the endomorphisms of U in* Free(Σ).

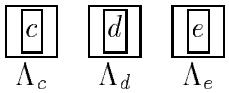
Proof. Follows from the following results of [3]. Free(Σ) is fully and faithfully representable in its tabular reflection Tab(Free(Σ)), which is the category of relations of a regular category **C**. The endomorphisms of U in Free(Σ) are precisely the subobjects of the terminal object 1 in Tab(Free(Σ)), and so by Freyd and Scedrov's proof of the Henkin–Lubkin theorem, there is a collectively faithful family $\{T_j : \mathbf{C} \to \mathbf{Set} \mid j \in J\}$. By regularity, Rel(Map(Tab(Free(Σ)))) $\simeq$ Tab(Free(Σ)), and so we can find a collectively faithful family $\{F_j : \mathrm{Free}(\Sigma) \to \mathbf{Rel} \mid j \in J\}$. □

Furthermore, the Σ allegory axioms are sound and complete for Σ nets in the sense that $\vdash e = f$ iff $[\![e]\!]_\rho = [\![f]\!]_\rho$ for all Σ-interpretations ρ.

Proposition 7 *$\vdash e = f$ iff for all Σ-interpretations ρ we have $[\![e]\!]_\rho = [\![f]\!]_\rho$.*

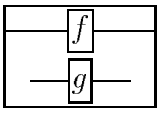
Proof. Soundness ($\Rightarrow$) is immediate as **Rel** is a upa. Completeness ($\Leftarrow$) follows from Theorem 6. We have a collectively faithful family $\{F_j \mid j \in J\}$, and F_j must be the homomorphic extension of a Σ-interpretation ρ_j. Thus if $\forall \rho \,.\, [\![e]\!]_\rho = [\![f]\!]_\rho$ then $\forall j \in J \,.\, F_j e = F_j f$, so $\vdash e = f$ since $\{F_j \mid j \in J\}$ is collectively faithful. □

We now consider why we cannot represent Free(Σ) faithfully in **Rel**. Consider the following net lists of type $U \to U$ in the signature Σ_{cde} with three components $c, d, e : U \to U$:



For any distinct $a, b \in \{c, d, e\}$, there is a Σ_{cde}-interpretation which maps a to $\emptyset$ and b to id, and so a and b must be distinguished in Free(Σ_{cde}). However, any Σ_{cde}-allegory morphism $F : \text{Free}(\Sigma_{cde}) \to \mathbf{Rel}$ identifies at least two of these net lists, since the unit 1 of **Rel** has precisely two endomorphisms $\emptyset$ and id_1. Hence F cannot be faithful.

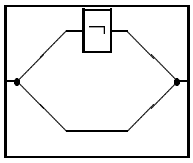
Any net containing an isolated net, such as g in:



has a subnet $u_X^\circ ; g ; u_Y$. The term $u_X^\circ ; g ; u_Y$ is an endomorphism on U.

In a Σ-interpretation ρ where $[\![g]\!]_\rho \neq \emptyset$ the subnet $u_X^\circ ; g ; u_Y$ can be regarded as *garbage* and its removal *garbage collection*. Garbage collection does not alter the relation denoted by a circuit.

In a Σ-interpretation ρ where $[\![g]\!]_\rho = \emptyset$ we regard the subnet g as a *short-circuit*. For example, interpreting $\neg$ as boolean negation, $\text{id} \cap \neg$ is a short circuit, with picture:



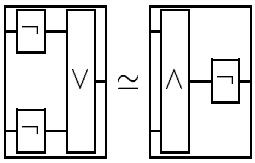
If a circuit f contains a short-circuit then $[\![f]\!]_\rho = \emptyset$, and so short circuits of the form $u_X^\circ ; g ; u_Y$ cannot be garbage collected, even though they are isolated from the rest of the circuit.

The relational semantics of a net depends on which of its isolated subnets can be garbage collected, which is determined precisely by which endomorphisms of U can be garbage collected. This is why there is no faithful representation of Free(Σ) in **Rel**.

7 Future work

This paper has shown how three different approaches to circuit design can be integrated. Some open problems remain.

We have presented signatures for circuits without *equations*. A notion of equation is needed in order to prove equivalences such as:



There is an obvious definition of a set of Σ-equations E, from which we can define the notion of (Σ, E)-allegory, and the free (Σ, E)-allegory Free(Σ, E). However, there is no obvious notion of (Σ, E)-net homomorphism such that $E \vdash \lambda(\Lambda) \subseteq \lambda(\Lambda')$ iff there is a homomorphism $F : \Lambda' \to \Lambda$.

In Section 6, we showed that there was no faithful representation of Free(Σ) in **Rel**, due to the semantics of isolated subnets. We conjecture that a semantics for nets which takes garbage collection into account is faithfully representable in **Rel**.

The issues of simulation, bisimulation, refinement and substitution of nets for components merit further investigation.

Acknowledgements

We thank Graham Hutton for his contribution to the initial development of this work and Edmund Robinson and Paul Taylor for useful comments and criticisms. Peter Freyd developed many of the notions of this paper independently, and we greatly appreciate his comments.

References

[1] Carolyn Brown and Graham Hutton. Categories, allegories and circuit design. To appear in *Proc. LICS*, 1994.

[2] Aurelio Carboni and Bob Walters. Cartesian bicategories I. *J. Pure and Applied Algebra*, 49:11–32, 1987.

[3] Peter J. Freyd and Andre Scedrov. *Categories, Allegories.* North-Holland, 1990.

[4] Randall L. Geiger, Phillip E. Allen, and Noel R. Strader. *VLSI Design Techniques for Analog and Digital Circuits.* McGraw Hill, 1990.

[5] Steven M. Rubin. *Computer Aids for VLSI Design.* Addison Wesley, 1987.

[6] Mary Sheeran. Describing and reasoning about circuits using relations. In J. Tucker et al., editors, *Proc. Workshop in Theoretical Aspects of VLSI*, 1986.