Università degli Studi di Pisa
Corso di Laurea Magistrale in Informatica

Anno Accademico 2021-2022

Insegnamento di

Foundation of Computing

Pagina del corso: `http:/pages.di.unipi.it/montanari/FOC.html`

Note di

Teoria della Concorrenza

1. Specifiche algebriche, categorie di modelli e aggiunzioni
2. Le reti di Petri e i loro modelli monoidali (strettamente) simmetrici

a cura di
Ugo Montanari
Dipartimento di Informatica
Università degli Studi di Pisa
`ugo@di.unipi.it`

Theories

Theory = Specification closed under entailment $\frac{\mathcal{E} \vdash E}{E \in \mathcal{E}}$

Lawvere theories (1967) basis of algebraic semantics

$\Gamma = (\Sigma, \mathcal{E})$ sentences (equation, membership, conditional = Horn)

Σ: signature (many sorted)

Γ-algebra A $\quad \frac{s \in S}{A_s \in A} \quad \frac{f \in \Sigma_{s_1 \ldots s_k, s}}{f^A : A_{s_1} \times \ldots \times A_{s_k} \to A_s \in A}$

Σ-algebra which satisfies $\mathcal{E}$ $\quad \frac{M : s \in E}{\forall \eta.\ [\![M]\!]\eta \text{ is defined}} \quad \frac{M = N : s \in E}{\forall \eta.\ [\![M]\!]\eta = [\![N]\!]\eta}$ defined and equal

Σ-homomorphism $h = \{h_s\}_{s \in S} : A \to B$ total functions with

$f \in \Sigma_{s_1 \ldots s_k, s} \quad a_i \in A_{s_i} \quad f^A(a_1, \ldots, a_k)$ defined

implies

$f^B(h_{s_1}(a_1) \ldots h_{s_k}(a_k))$ defined and

$f^B(h_{s_1}(a_1) \ldots h_{s_k}(a_k)) = h_s(f^A(a_1 \ldots a_k))$

Category of models $\underline{Alg}_\Gamma$: Γ-algebras and Γ-homomorphisms (loose semantics)

Initial model: only sentences $\mathcal{E}$ are satisfied

In general: general logics, institutions

liberal institutions correspond to what we will discuss in the following

Algebras as Categories of Functors

(30 hrs)

(Only in the simplest case: Lawvere Theories)

A specification $\Gamma = (\Sigma, \mathcal{E})$ closed theory corresponds exactly to an A-Cartesian category

- build the category $\underline{Subst}(\Sigma)$ of terms & substitutions on the signature Σ;
- for every equation $e: L = R$ equate the two coinitial, 1-final arrows $[\![L]\!] = [\![R]\!]$

Since $\mathcal{E}$ is closed, no additional arrows will be equated. Let $L(\Gamma)$ be the resulting Cartesian category.

Conversely, any skeletal (isomorphic objects are identified) Cartesian category is $L(\Gamma)$ for some (possibly infinite) Γ.

The category $\underline{Alg}_\Gamma$ of models (algebras) satisfying specification Γ is the category of A-Cartesian functors: $\underline{Alg}_\Gamma = L(\Gamma) \to \underline{Set}$

$\underline{Set}$ can be considered as an A-Cartesian category

Final object 1 is a chosen singleton

Tupling is associative, etc.

A-Cartesian functors respect all the operations

Functor Categories $\mathcal{F} : \underline{C} \to \underline{D}$

$|\mathcal{F}|$ are functors $F : \underline{C} \to \underline{D}$

$\mathcal{F}[F, G]$ are natural transformations $\varphi : F \Rightarrow G$

$F, G : \underline{C} \to \underline{D}$ $\quad \varphi : |\underline{C}| \to \|\underline{D}\|$ $\quad \varphi(c) \in \underline{D}[F(c), G(c)]$

$$\begin{tikzcd} c_1 \arrow[d, "f"'] & F(c_1) \arrow[r, "\varphi(c_1)"] \arrow[d, "F(f)"'] \arrow[dr, phantom, "="] & G(c_1) \arrow[d, "G(f)"] \\ c_2 & F(c_2) \arrow[r, "\varphi(c_2)"] & G(c_2) \end{tikzcd}$$

In the case of algebras:

An algebra A is a functor $A : \underline{L(\Gamma)} \to \underline{Set}$ which:

- associates to every sort s the set $A(s) = A_s$ of all the values of sort s.
- associates to every arrow $\sigma : s_1 \otimes s_2 \otimes \cdots s_n \to s$ an n-adic function (arrow in $\underline{Set}$)
  $A(\sigma) = A^\sigma : A_{s_1} \times \ldots \times A_{s_n} \to A_s$

An arrow $h : A \to B$ is a natural transformation

$$\begin{tikzcd} s_1 \otimes \cdots \otimes s_n \arrow[d, "\sigma"'] & A_{s_1} \times \cdots \times A_{s_n} \arrow[r, "h_{s_1} \times \cdots h_{s_n}"] \arrow[d, "A^\sigma"'] & B_{s_1} \times \cdots \times B_{s_n} \arrow[d, "B^\sigma"] \\ s & A_s \arrow[r, "h_s"] & B_s \end{tikzcd}$$

i.e. a family of functions $\{h_s\}_{s \in S}$

arrows of $\underline{Alg}_\Gamma = \underline{L(\Gamma)} \to \underline{Set}$ are exactly Σ-homomorphisms!

30, quater

Tensor product of theories

$$\mathrm{Alg}_{\Gamma} = L_{\Gamma} \to \mathrm{Set}$$

$$\mathrm{Alg}(\Gamma, C) = L_{\Gamma} \to C \quad \text{generalisation}$$

$$\mathrm{Alg}(\Gamma_1 \otimes \Gamma_2, \mathrm{Set}) = L_{\Gamma_1 \otimes \Gamma_2} \to \mathrm{Set}$$
$$= L_{\Gamma_1} \to (L_{\Gamma_2} \to \mathrm{Set})$$
$$= L_{\Gamma_2} \to (L_{\Gamma_1} \to \mathrm{Set})$$

$$a^{b \times c} \cong (a^b)^c \cong (a^c)^b$$

$$b \times c \to a \cong c \to b \to a$$
$$\cong b \to c \to a$$

Example: $\Gamma_1 = \Gamma_{Cat}$ $\Gamma_2 = \Gamma_{Mon}$

$$\mathrm{Alg}(\Gamma_{Cat} \otimes \Gamma_{Mon}) = L_{MonCat} \to \mathrm{Set}$$
$$= L_{Mon} \to (L_{Cat} \to \mathrm{Set}) = L_{Mon} \to \mathrm{Cat}$$
$$= L_{Cat} \to (L_{Mon} \to \mathrm{Set})$$

$$A^{B \times C} = (A^B)^C = (A^C)^B$$

30.5

Symmetries in monoidal categories as a natural isomorphism

Monoidal specification interpreted in CAT

$$(x \otimes y) \otimes z = x \otimes (y \otimes z)$$

becomes

$$(C_1 \otimes C_2) \otimes C_3 = C_1 \otimes (C_2 \otimes C_3)$$

where $|C_1 \otimes C_2| = \{(x,y) \mid x \in |C_1|, y \in |C_2|\}$

$$C_1 \otimes C_2 [u \otimes v, u' \otimes v'] = \{(h,k) \mid h \in C_1[u,u'],\ k \in C_2[v,v']\}$$

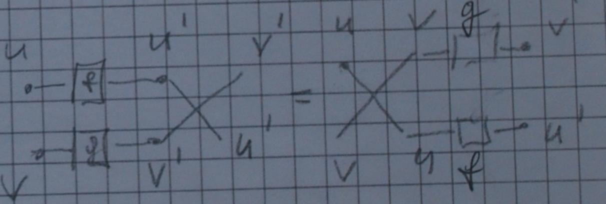
$(-,-)$ associative

What about commutative monoids?

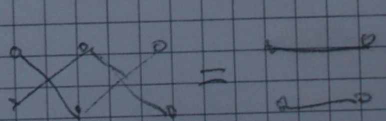
$C_1 \otimes C_2 = C_2 \otimes C_1$ "on the nose" too strict

$C_1 \otimes C_2 \cong C_2 \otimes C_1$ up to a natural isomorphism

$$\begin{tikzcd}
(u,v) \arrow[r, "\rho_{uv}"] \arrow[d, "{(f,g)}"'] \arrow[dr, phantom, "="] & (v,u) \arrow[d, "{(g,f)}"] \\
(u',v') \arrow[r, "\rho_{u'v'}"'] & (v',u')
\end{tikzcd}$$



naturality axioms



$\rho : C_1 \otimes C_2 \to C_2 \otimes C_1$

$\rho' : C_2 \otimes C_1 \to C_1 \otimes C_2$

$\rho_{uv} ; \rho_{vu} = id_{uv}$ $\rho_{vu} ; \rho_{uv} = id_{vu}$

natural isomorphism

Theory Morphisms

$$\underbrace{(\Sigma^1, \mathcal{E}^1)}_{\Gamma_1} \xrightarrow{V} \underbrace{(\Sigma^2, \mathcal{E}^2)}_{\Gamma_2}$$

V for view

$\mathcal{E}_1 \mathcal{E}_2$ closed theories

$V_{sort} : S_1 \to S_2$

$V_{operation} : \Sigma_{s_1 \ldots s_k, s} \to \Sigma_{V_{sort}(s_1) \ldots V_{sort}(s_k), V_{sort}(s)}$

Just V.

V extended to terms N and sentences E.

We require
$$\frac{E \in \mathcal{E}_1}{V(E) \in \mathcal{E}_2}$$

Every theory morphism V defines a functor

$U_V : \underline{Alg}_{\Gamma_2} \to \underline{Alg}_{\Gamma_1}$ <u>forgetful</u> functor

$$\frac{V(s_1) = s_2}{(U_V(A))_{s_1} = A_{s_2}} \qquad \frac{V(f_1) = f_2}{f_1^{U_V(A)} = f_2^A}$$

$$\frac{V(s_1) = s_2 \quad h_{s_2} : A_{s_2} \to B_{s_2}}{(U_V(h))_{s_1} = h_{s_2}}$$

31 δ's

Theory Morphisms as Functors

$V : L(\Gamma_1) \to L(\Gamma_2)$ in $\underline{Cart}$ — $V(s)$ on sorts, $V(\sigma)$ on operations

$U_V : \underline{Alg}_{\Gamma_2} \to \underline{Alg}_{\Gamma_1}$

On the objects:

$A_2 : L(\Gamma_2) \to Set$

$U_V(A_2) = V ; A_2 : L(\Gamma_1) \to Set$

$$\frac{V(s_1) = s_2}{(U_V(A))_{s_1} = A_{s_2}} \qquad \frac{V(\sigma_1) = \sigma_2}{\sigma_1^{U_V(A)} = \sigma_2^{A}}$$

On the arrows

$$\frac{V(s_1) = s_2 \quad h_{s_2} : A^{s_2} \to B^{s_2}}{(U_V(h))_{s_1} = h_{s_2}}$$

$h : A \to B$ trasformazione naturale $h(s) \in Set[A(s) \to B(s)]$

$$\begin{tikzcd}
s_1 \otimes \cdots \otimes s_n \arrow[d, "\sigma"'] & A_{s_1} \times \cdots \times A_{s_n} \arrow[r, "h_{s_1} \times \cdots \times h_{s_n}"] \arrow[d, "A^{\sigma}"'] & B_{s_1} \times \cdots \times B_{s_n} \arrow[d, "B^{\sigma}"] \\
s & A_s \arrow[r, "h_s"] & B_s
\end{tikzcd}$$

$$\begin{tikzcd}
V(s_1') \otimes \cdots V(s_n') \arrow[d, "V(\sigma')"'] & A_{V(s_1')} \times \cdots \times A_{V(s_n')} \arrow[r, "h_{V(s_1')} \times \cdots \times h_{V(s_n')}"] \arrow[d, "A^{V(\sigma')}"'] & B_{V(s_1')} \times \cdots \times B_{V(s_n')} \arrow[d, "B^{V(\sigma')}"] \\
V(s') & A_{V(s')} \arrow[r, "h_{V(s')}"] & B_{V(s')}
\end{tikzcd}$$

$U_V(h) = V ; h : U_V(A) \to U_V(B)$

31.2

$$
\begin{tikzcd}
\mathbf{A} & & \mathcal{U}(b) & b & \mathbf{B} \\
a \arrow[r, "\eta_a"'] \arrow[urr, "f"] & \mathcal{U}(F_a) \arrow[ur, "\mathcal{U}(g)"'] & & F_a \arrow[u, dotted, "g"] &
\end{tikzcd}
$$

Figura 3: Diagramma per l'oggetto libero F_a

La seconda nozione che ci serve è quella di oggetto libero.

DEFINIZIONE 1.4 (OGGETTO LIBERO)
Dato un funtore $\mathcal{U} : \mathbf{B} \to \mathbf{A}$ *e un oggetto* $a \in \mathbf{A}$ *si dice che un certo oggetto* $F_a \in \mathbf{B}$ è libero *su* $\mathbf{A}$ *se esiste una freccia* $\eta_a : a \to \mathcal{U}(F_a)$ *tale che per ogni altro oggetto* $b \in \mathbf{B}$ *e freccia* $f : a \to \mathcal{U}(b) \in \mathbf{A}$ *esiste un'unica freccia* $g : F_a \to b \in \mathbf{B}$ *tale che* $\eta_a;\mathcal{U}(g) = f$ *(si veda la Figura 3, dove la punteggiatura della freccia* g *ne simboleggia l'unicità).*

La proprietà che definisce un oggetto libero è detta *proprietà universale* e la freccia η_a è detta *freccia universale*. Sostanzialmente la proprietà universale ci garantisce che l'oggetto F_a è uno dei migliori rappresentanti possibili che possiamo trovare per a all'interno di $\mathbf{B}$ rispetto al funtore $\mathcal{U}$. Per capire meglio, immaginiamo che un morfismo $f : a \to a'$ di una categoria definisca un modo di rappresentare l'oggetto a attraverso a', ovvero di vedere a' come un'approssimazione di a. L'idea è di cercare la migliore approssimazione possibile dell'oggetto a tra gli oggetti che appartengono all'immagine di $\mathcal{U}(\mathbf{B})$. Quindi ogni oggetto $b \in \mathbf{B}$ tale che esiste una freccia $f : a \to \mathcal{U}(b) \in \mathbf{A}$ offre implicitamente un possibile candidato (cioè $\mathcal{U}(b)$). La proprietà universale garantisce che il candidato F_a (mediante η_a) fornisce l'approssimazione più vicina ad a, perché consente di decomporre in modo univoco ogni altra approssimazione possibile $f : a \to \mathcal{U}(b) \in \mathbf{A}$ di a attraverso $g : F_a \to b$ in $\mathbf{B}$. Si noti che, come avevamo notato prima a proposito delle caratteristiche della teoria delle categorie, un oggetto libero quando esiste non è necessariamente unico, ma se ve ne sono più di uno questi sono tutti isomorfi tra loro.

Nel caso di interesse, la categoria delle computazioni gioca il ruolo di $\mathbf{B}$ e la categoria dei modelli computazionali quello di $\mathbf{A}$. Il funtore $\mathcal{U}$ associa un modello ad ogni computazione e dato un modello a, l'oggetto libero F_a è lo spazio delle computazioni che meglio descrive a tra quelli che potremmo scegliere in $\mathbf{B}$.

La terza nozione è quella di *aggiunzione* tra due categorie. Lo scenario comprende due categorie $\mathbf{A}$ e $\mathbf{B}$ con due funtori $\mathcal{F} : \mathbf{A} \to \mathbf{B}$ e $\mathcal{U} : \mathbf{B} \to \mathbf{A}$ tali che il funtore $\mathcal{F}$ manda ogni oggetto a in un oggetto libero $\mathcal{F}(a)$ rispetto a $\mathcal{U}$.

DEFINIZIONE 1.5 (AGGIUNTO SINISTRO)
Sia dato un funtore $\mathcal{U} : \mathbf{B} \to \mathbf{A}$. *Se per ogni oggetto* $a \in \mathbf{A}$ *esiste un oggetto libero* $F_a \in \mathbf{B}$ *con freccia universale* $\eta_a : a \to \mathcal{U}(F_a)$, *allora possiamo definire un funtore* $\mathcal{F} : \mathbf{A} \to \mathbf{B}$, *detto* aggiunto sinistro *di* $\mathcal{U}$, *come segue:*

$$\begin{array}{lcc}
\mathbf{A} & \begin{tikzcd} a' \arrow[r, "\eta_{a'}"] & \mathcal{U}(\mathcal{F}(a')) \\ a \arrow[u, "f"] \arrow[r, "\eta_a"'] \arrow[ur, "f;\eta_{a'}"] & \mathcal{U}(\mathcal{F}(a)) \arrow[u, "\mathcal{U}(\mathcal{F}(f))"'] \end{tikzcd} & \begin{tikzcd} \mathcal{F}(a') \\ \mathcal{F}(a) \arrow[u, dotted, "\mathcal{F}(f)"] \end{tikzcd} \quad \mathbf{B}
\end{array}$$

Figura 4: Diagramma per l'aggiunzione $\mathcal{F} \dashv \mathcal{U}$

$$\begin{array}{lcc}
\mathbf{A} & \begin{tikzcd} a \arrow[d, "f"'] \\ b \end{tikzcd} & \begin{tikzcd} \mathcal{F}(a) \arrow[r, "\varphi_a"] \arrow[d, "\mathcal{F}(f)"'] & \mathcal{G}(a) \arrow[d, "\mathcal{G}(f)"] \\ \mathcal{F}(b) \arrow[r, "\varphi_b"'] & \mathcal{G}(b) \end{tikzcd} \quad \mathbf{B}
\end{array}$$

Figura 5: Diagramma che esprime la naturalità di $\varphi : \mathcal{F} \Rightarrow \mathcal{G} : \mathbf{A} \to \mathbf{B}$.

- *per ogni oggetto* $a \in \mathbf{A}$, *poniamo* $\mathcal{F}(a) = F_a$;
- *per ogni freccia* $f : a \to a' \in \mathbf{A}$, *definiamo come* $\mathcal{F}(f) : F_a \to F_{a'}$ *l'unica freccia* g *tale che* $\eta_a;\mathcal{U}(g) = f;\eta_{a'}$, *la cui esistenza e unicità sono garantite dalla proprietà universale di* F_a *rispetto alla freccia* $f;\eta_{a'}$ *(si veda la Figura 4).*

Si noti che $\mathcal{F}$ è un funtore perché rispetta identità e composizione grazie alla unicità della proprietà universale. Il funtore $\mathcal{U} : \mathbf{B} \to \mathbf{A}$ è detto *aggiunto destro*, la coppia $(\mathcal{F}, \mathcal{U})$ è detta *aggiunzione* e si usa la notazione $\mathcal{F} \dashv \mathcal{U}$ per evidenziarlo. La famiglia di frecce $\{\eta_a\}_{a \in \mathbf{A}}$ è detta *unit* dell'aggiunzione e costituisce una cosiddetta *trasformazione naturale.*

DEFINIZIONE 1.6 (TRASFORMAZIONE NATURALE)
Dati due funtori $\mathcal{F}, \mathcal{G} : \mathbf{A} \to \mathbf{B}$, *una* trasformazione naturale $\varphi : \mathcal{F} \Rightarrow \mathcal{G}$ *è una famiglia* $\{\varphi_a : \mathcal{F}(a) \to \mathcal{G}(a)\}_{a \in \mathbf{A}}$ *di frecce in* $\mathbf{B}$ *tali che per ogni* $f : a \to b \in \mathbf{A}$ *vale* $\varphi_a; \mathcal{G}(f) = \mathcal{F}(f); \varphi_b$ *(si veda la Figura 5). Una trasformazione naturale* φ *tale che per ogni* $a \in \mathbf{A}$ *la freccia* φ_a *è un isomorfismo è detta* isomorfismo naturale.

È immediato verificare che la unit di un'aggiunzione $\mathcal{F} \dashv \mathcal{U}$ definisce una trasformazione naturale $\eta : 1_{\mathbf{A}} \Rightarrow \mathcal{F};\mathcal{U}$ tra il funtore identità e quello ottenuto componendo i due aggiunti (infatti, per definizione $\mathcal{F}(f)$ è tale che $\eta_a;\mathcal{U}(\mathcal{F}(f)) = f;\eta_{a'}$). Sfruttando il concetto di dualità si può facilmente dare una definizione alternativa ma del tutto equivalente di aggiunzione, basata sull'esistenza di una cosiddetta *counit* $\epsilon : \mathcal{U};\mathcal{F} \Rightarrow 1_{\mathbf{B}}$, a partire dal funtore $\mathcal{F}$ e dal diagramma di oggetto libero (duale) in Figura 6: in questo caso $\mathcal{U}(b) = U_b$ e data $f : b' \to b \in \mathbf{B}$ si ha che $\mathcal{U}(f)$ è l'unica freccia g tale

Adjunctions

Very general and useful notion. Given two functors

$F: A \to B$

$U: B \to A$

$$A \underset{U}{\overset{F}{\rightleftarrows}} B$$

Natural Transformation

$S, T: C \to B$

$\tau: S \to T$

$$\begin{tikzcd} c \arrow[d, "f"] & Sc \arrow[r, "\tau c"] \arrow[d, "Sf"'] \arrow[dr, phantom, "="] & Tc \arrow[d, "Tf"] \\ c' & Sc' \arrow[r, "\tau c'"] & Tc' \end{tikzcd}$$

an adjunction is a collection of isomorphisms $\varphi = \{\varphi_{a,b}\}$

$$\varphi_{a,b} : B[F(a), b] \to A[a, U(b)]$$

Hom functors : COVARIANT one for every a

$C[a,-] : C \to Set$

$g: b \to b'$ $\quad C[a,g] : C[a,b] \to C[a,b']$

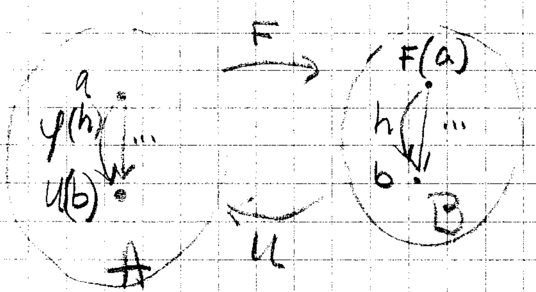
$C[a,g](h) = h;g$

CONTRAVARIANT

$C[-,b] : C^{op} \to Set$

$f: a \to a'$ $\quad C[f,b] : C[a',b] \to C[a,b]$

$C[f,b](h) = f;h$



Such that the following naturality conditions are satisfied

$$\begin{tikzcd} b \arrow[d, "g"] & B[F(a), b] \arrow[r, "\varphi_{ab}"] \arrow[d, "-;g"'] & A[a, U(b)] \arrow[d, "-;U(g)"] \\ b' & B[F(a), b'] \arrow[r, "\varphi_{ab'}"] & A[a, U(b')] \end{tikzcd}$$

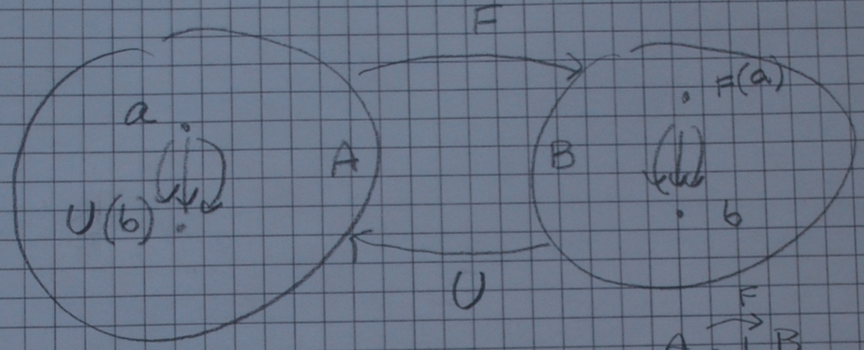
$$\begin{tikzcd} a & B[F(a), b] \arrow[r, "\varphi_{ab}"] \arrow[d, "F(f);-"'] & A[a, U(b)] \arrow[d, "f;-"] \\ a' \arrow[u, "f"] & B[F(a'), b] \arrow[r, "\varphi_{a'b}"] & A[a', U(b)] \end{tikzcd}$$

Here the naturality is the natural isomorphism concerning

$$\underbrace{A^{op} \times B \xrightarrow{F \times Id} B^{op} \times B \xrightarrow{hom} Set}_{\text{left functor}} \xrightarrow{\varphi} \underbrace{A^{op} \times B \xrightarrow{id \times U} A^{op} \times A \xrightarrow{hom} Set}_{\text{right functor}}$$

(31.5)

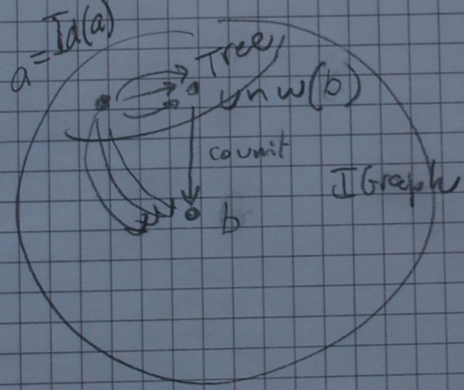
An example of right adjoint (coreflection)



$$\begin{tikzcd} A \arrow[r, bend left, "F"] \arrow[r, phantom, "\perp"] & B \arrow[l, bend left, "U"] \end{tikzcd}$$

F: inclusion

U: unwinding



$$\begin{tikzcd} \mathrm{Tree} \arrow[r, bend left, "\mathrm{Id}"] \arrow[r, phantom, "\perp"] & \mathrm{IGraph} \arrow[l, bend left, "\mathrm{Unw}"] \end{tikzcd}$$

IGraph = con vertice iniziale

$$\frac{}{([\,], \mathrm{init}(G)) \in \mathrm{unw}(G)}$$

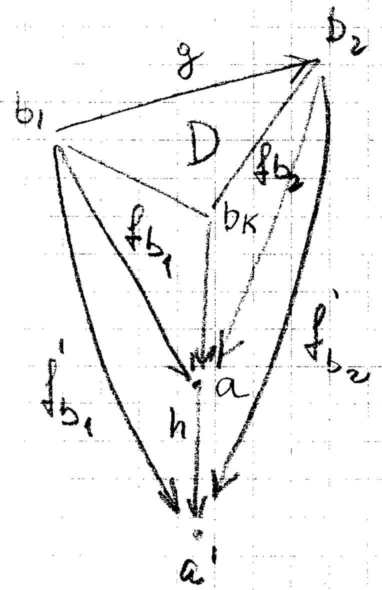
$$\frac{(x,a) \in \mathrm{unw}(G) \quad a \to b \in G}{((x,a), b) \in \mathrm{unw}(G)}$$

Colimits

Given a diagram $D = (\{b_i\}, \{g_j\})$, a <u>colimit</u> is a cocone (i.e. an object a and an arrow $f_b : b \to a$ for every object b in D) such that

i) for every arrow $g : b_1 \to b_2$ we have $f_{b_1} = g ; f_{b_2}$

ii) for every cocone $(a', \{f'_b\})$ satisfying the above condition there is one and only one arrow $h : a \to a'$ with $f_b ; h = f'_b$ for every b in D.



in <u>Set</u> $a = \bigcup b_i$
with all the identifications entailed by

$$\frac{g : b_1 \to b_2 \quad x \in b_1}{x = g(x)}$$

In a generic category, the usual intended meaning of colimits is to provide composition of objects

Special Colimits

Initial object : empty diagram

$$\begin{tikzcd} a \arrow[d] \\ a' \end{tikzcd}$$

for every a' there is a unique $¡_a$ (co-discharger)

Coproduct : two objects a, b, no arrow

$$\begin{tikzcd} a \arrow[r, "inj_a"'] \arrow[dr, "f"'] & a+b \arrow[d, "{[f,g]}"] & b \arrow[l, "inj_b"] \arrow[dl, "g"] \\ & \cdot & \end{tikzcd}$$

an object $a+b$ (union)
and two injection arrows inj_a, inj_b
such that for every f, g
there is a unique copairing $[f,g]$
making the two triangle commute.

Pushout : a span $\langle g_1, g_2 \rangle$, namely
two arrows $g_1 : b \to b_1$ $g_2 : b \to b_2$

$$\begin{tikzcd} & b \arrow[dl, "g_1"'] \arrow[dr, "g_2"] & \\ b_1 \arrow[dr, "f_1"] \arrow[ddr, bend right, "f_1'"'] & & b_2 \arrow[dl, "f_2"'] \arrow[ddl, bend left, "f_2'"] \\ & a \arrow[d, "h"] & \\ & a' & \end{tikzcd}$$

the pushout is a cospan $\langle f_1, f_2 \rangle$
such that

$$g_1 ; f_1 = g_2 ; f_2$$

and for every other cospan $\langle f_1', f_2' \rangle$
there is a unique h which
makes the two triangles commute.

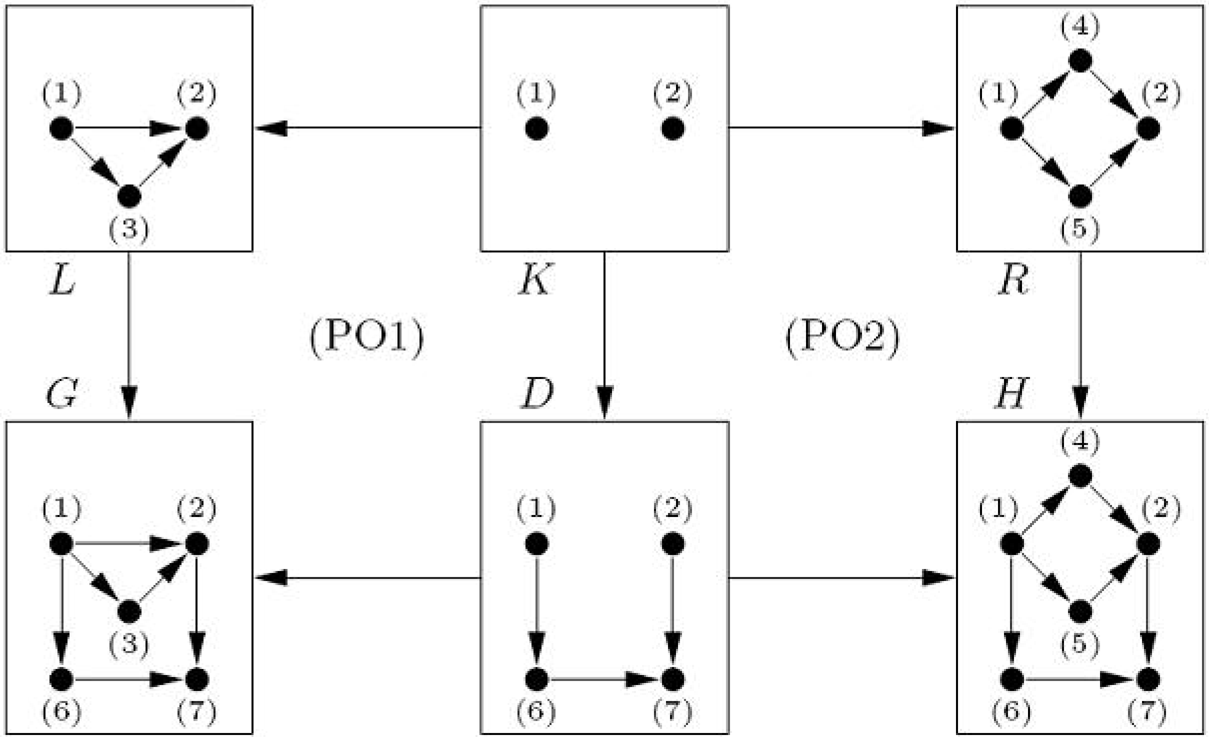
Double-pushout step by step process (continued)

$$
\begin{tikzcd}
L \arrow[d, "m"'] \arrow[dr, phantom, "(1)"] & K \arrow[l] \arrow[r] \arrow[d] \arrow[dr, phantom, "(2)"] & R \arrow[d] \\
G & D \arrow[l] \arrow[r] & H
\end{tikzcd}
$$

(1) construct D such that the gluing of L and D via K is equal to G

(2) construct the gluing of R and D via K leading to the graph H

Simple Example



Example: Category Graph

$\mathbf{Graph} = \mathbf{Alg}_{\Gamma_{Graph}}$ for Γ_{Graph} as follows.

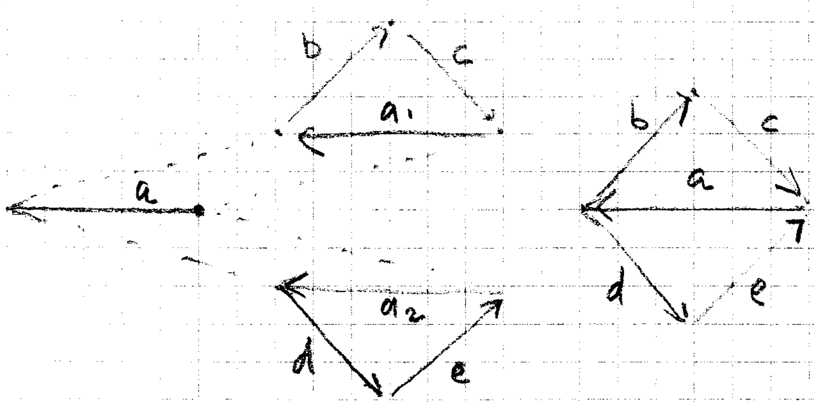
$S = \{node, arc\}$ $\quad h = \{h_{node}, h_{arc}\}$ preserving source & target

$s, t : arc \to node$

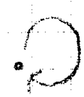
Initial object: the empty graph

Coproduct: disjoint union

Pushout: "composing" graphs.



Also:

Final object: 

Product: $G \times H$

$$\frac{n \in G_{node} \quad m \in H_{node}}{(n,m) \in (G \times H)_{node}}$$

like product of automata

$$\frac{a : n \to m \in G_{arc} \quad b : i \to j \in H_{arc}}{(a,b) : (n,i) \to (m,j) \in (G \times H)_{arc}}$$

Spiegare

Unification as a Pullback

(34 bis)

Logic Programming

$$\frac{H :- F \in \mathbb{P} \quad \sigma = mgu(A, \rho(H))}{P \models A \Rightarrow_\sigma \sigma(\rho(F))}$$

$$\frac{P \models G \Rightarrow_\sigma F}{P \models G' \wedge G \Rightarrow_\sigma \sigma(G') \wedge F}$$

$sum(0, y, y)$

$sum(s(x), y, s(z')) :- sum(x, y, z')$

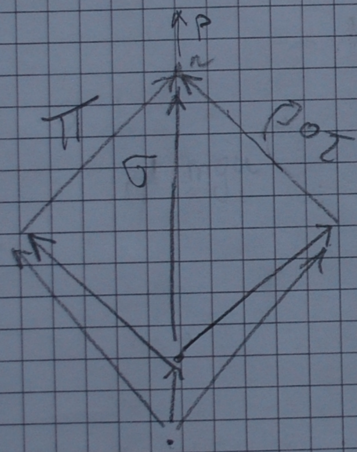
$$sum(s(0), s(0), z) \xleftarrow[y = s(0),\ z = s(z')]{x = 0} sum(0, s(0), z') \xleftarrow[z = s(s(0))]{s(0) = z'}$$

Unification

$$mgu(\underbrace{\pi\, \underline{P}}_{A}, \rho\, \underbrace{\tau\, \underline{P}}_{\mathbb{H}}) = \sigma\, \underline{P}$$

pullback in $L(\Sigma)$

double pushout



σ: minimal instantiation of variables

most general unifier

ritornare a unfolding

Some Fundamental Properties

Free constructions

Given a theory morphism $V : (\Sigma^1, \mathcal{E}^1) \to (\Sigma^2, \mathcal{E}^2)$ its forgetful functor

$$U_V : \underline{Alg}_{\Gamma_2} \to \underline{Alg}_{\Gamma_1}$$

conservative = injective unit
complete = surjective "
persistent = bijective "

has a left adjoint (See pag. 21)

$$F_V : \underline{Alg}_{\Gamma_1} \to \underline{Alg}_{\Gamma_2}.$$

F_V freely adds to every algebra $A \in |\underline{Alg}_{\Gamma_1}|$ all the additional carriers and operations which are in Σ_2 but not in Σ_1, and imposes the satisfaction of all the additional sentences which are in $\mathcal{E}^2$ and not in $\mathcal{E}^1$.

Uniqueness of the adjoint

Given categories A and B and a functor $U : B \to A$, its left adjoint $F : A \to B$, if it exists, is unique up to (natural) isomorphism.

Preservation of the colimits

Given an adjunction $A \underset{U}{\overset{F}{\rightleftarrows}} B$ ($\perp$), the left adjoint F preserves colimits. Namely, if cocone C is the colimit of diagram D in A, then $F(C)$ is the colimit of diagram $F(D)$ in B.

Example of operational semantics as a free construction

$\Gamma_{Graph} =$ {node, arc} $\qquad$ no axioms

$\qquad s, t :$ arc $\to$ node

$\Gamma_{Cat} =$ {object, arrow} $\qquad$ plus all the axioms of categories

$\qquad s, t :$ arrow $\to$ object

$\qquad id :$ object $\to$ arrow

$\qquad ; :$ arrow arrow $\to$ arrow

We define a theory morphism

$$V : \Gamma_{Graph} \to \Gamma_{Cat}$$

$$V(\text{node}) = \text{object} \qquad V(\text{arc}) = \text{arrow}$$

$$V(s) = s \qquad V(t) = t$$

Thus we have an adjunction

$$\underline{Graph} \underset{U_V}{\overset{F_V}{\rightleftarrows}} \underline{Cat}$$

F_V on objects (graphs) is defined by the inference rules

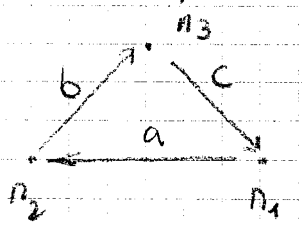
$$\frac{n \in G^{node}}{n \in (F_V(G))^{object}} \qquad \frac{a \in G^{arc}}{a \in (F_V(G))^{arrow}}$$

$$\frac{a \in (F_V(G))^{object}}{id_a \in (F_V(G))_{arrow}} \qquad \frac{f : a \to b,\ g : b \to c \in (F_V(G))^{arrow}}{f;g : a \to c \ \in (F_V(G))^{arrow}}$$

plus associativity and identity on $(F_V(G))^{arrow}$.

Example (Cont.)

Given a graph G

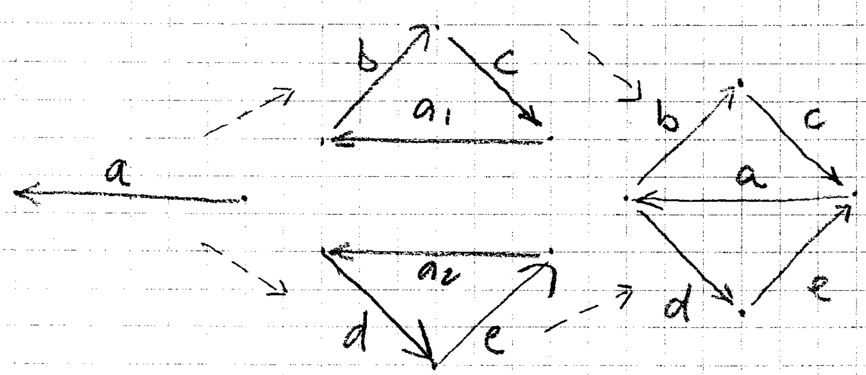


the category $F_V(G)$ has three objects and an infinite number of arcs: e.g.

$$F_V(G)[n_1 \to n_1] = \{ id_{n_1}, abc, (abc)^2, \dots \}$$

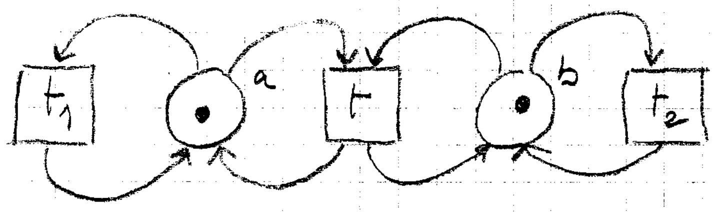
it is the path category freely generated by G.

Also in the pushout

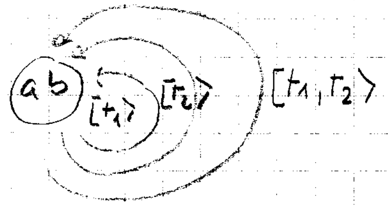


We get the same cocone (cospan) in Cat by taking the colimit in Graphs (program composition) of the given span and then mapping to Cat via F_V the resulting cospan (operational semantics of the result) or alternatively by mapping the given span to Cat via F_V (operational semantics of the components) and then taking the colimit in Cat (composition of the models of computation).

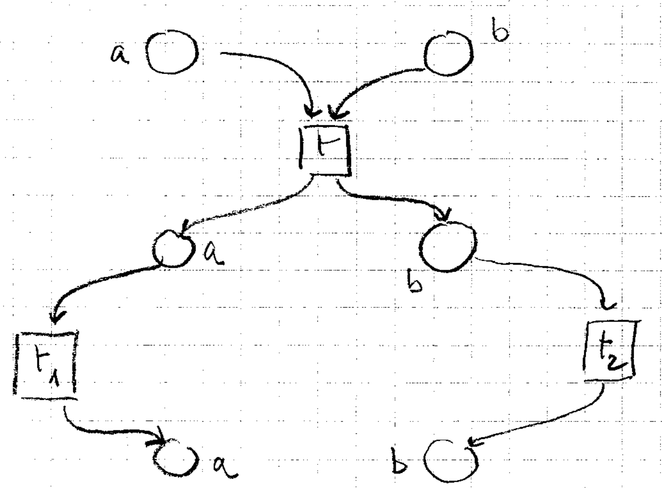
The case study of Petri Place-Transition (P/T) nets



token game



case graph



non sequential process:
a deterministic
concurrent trace

defined as an acyclic net,
at most one input and
output from every place,
plus a mapping to the given net

Set of places S
Markings $u: S \to N$ natural numbers
Set of transitions T
Markings of preconditions ${}^\circ t$ and postconditions t°, $t \in T$.

A step $u\,[X\rangle\, v$ is possible, u, v markings, X multiset of transitions, if

$\forall s.\ \sum_{t \in X} {}^\circ t(s) \leq u(s)$. Then

$$v(s) = \left(u(s) - \sum_{t \in X} {}^\circ t(s)\right) + \sum_{t \in X} t^\circ(s)$$

Petri Nets are Monoids (commutative)

i) Commutative monoids: Γ_{CMon}

$S = \{s\}$

$\otimes : s\ s \to s$

$1 : \quad \to s$

$\otimes$ assoc, commutative, 1 identity (ACI)

$\underline{\text{CMon}} = \underline{\text{Alg}}_{\Gamma_{CMon}}$

$\underline{\text{Set}} \underset{U}{\overset{-^{\otimes}}{\rightleftarrows}} \underline{\text{CMon}}$ $\quad\quad$ $\underline{\text{Set}}$ can be seen as a trivial algebra

Given the set S of tokens, $S^{\otimes}$ is the free commutative monoid freely generated by S. Its elements are markings.

ii) Semi c-monoidal graphs: $\Gamma_{SCMonGraph}$

$S = \{\text{node}, \text{arc}\}$

$\otimes : \text{node} \times \text{node} \to \text{node}$ $\quad$ (ACI)

$1 : \quad \to \text{node}$

$s, t : \text{arc} \to \text{node}$

Petri nets are those semi c-monoidal graphs whose commutative monoid of nodes is free.

Their full subcategory of $\underline{\text{SCMonGraph}}$ is $\underline{\text{Petri}}$

iii) C-monoidal graphs: $\Gamma_{CMonRGraph}$

$S = \{node, arc\}$ independent structures

$\otimes$: node node $\to$ node of commutative monoids and

1 : $\to$ node reflexive graphs (= with identity)

$\otimes$: arc arc $\to$ arc

1 : $\to$ arc ACI on nodes and arcs

s, t : arc $\to$ node

id : node $\to$ arcs $id_n : n \to n$

Distributive axioms

$$\frac{a_1 : u_1 \to v_1 \quad a_2 : u_2 \to v_2}{a_1 \otimes a_2 : u_1 \otimes u_2 \to v_1 \otimes v_2}$$

$id_1 = 1$

Tensorproduct $\Gamma_{CMonRGraph} = \Gamma_{CMon} \otimes \Gamma_{RGraph}$

$$\underline{SCMonGraph} \underset{U}{\overset{C}{\rightleftarrows}} \underline{CMonRGraph} \quad \perp$$

C freely generates the monoid of arcs

If N is a P/T net, $C[N]$ is the case graph of N.
Logically C is defined on nets as:

$$\frac{u \in N^{node}}{u \in C[N]^{node}}$$ markings are left unchanged

$$\frac{a : u \to v \in N^{arc}}{a : u \to v \in C[N]^{arc}}$$ transitions are inserted as steps

$$\frac{u \in C[N]^{node}}{id_u : u \to u \in C[N]^{arc}}$$ identity steps are created

$$\frac{a_1 : u_1 \to v_1 \;,\; a_2 : u_2 \to v_2 \in C[N]^{arc}}{a_1 \otimes a_2 : u_1 \otimes u_2 \to v_1 \otimes v_2 \in C[N]^{arc}}$$ parallel steps are created

plus all the monoidal and distributive axioms

iv) Strictly symmetric monoidal categories : $\Gamma_{SSMonCat}$

All the operations and axioms of monoidal categories plus commutativity of $-\otimes-$ on objects and arrows

$$\underline{SCMonGraph} \rightleftarrows \underline{CMonRGraph} \rightleftarrows \underline{SSMonCat}$$

or directly $\qquad \Gamma_{SSMonCat} = \Gamma_{CMon} \otimes \Gamma_{Cat}$

$$\underline{SCMonGraph} \underset{U}{\overset{T}{\rightleftarrows}} \underline{SSMonCat}$$

If N is a net, $T[N][u \to v]$ consists exactly of all (Best & Devillers) nonsequential processes from marking u to marking v.

Logically T is defined by:

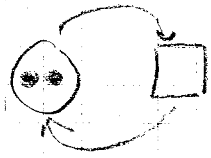
$$\frac{u \in N^{node}}{u \in T[N]^{object}} \qquad \frac{a : u \to v \in N^{arc}}{a : u \to v \in T[N]^{arrow}}$$

$$\frac{u \in T[N]^{object}}{id_u \in T[N]^{arrow}} \qquad \frac{a_1 : u_1 \to v_1 ,\ a_2 : u_2 \to v_2 \in T[N]^{arrow}}{a_1 \otimes a_2 : u_1 \otimes u_2 \to v_1 \otimes v_2 \in T[N]^{arrow}}$$

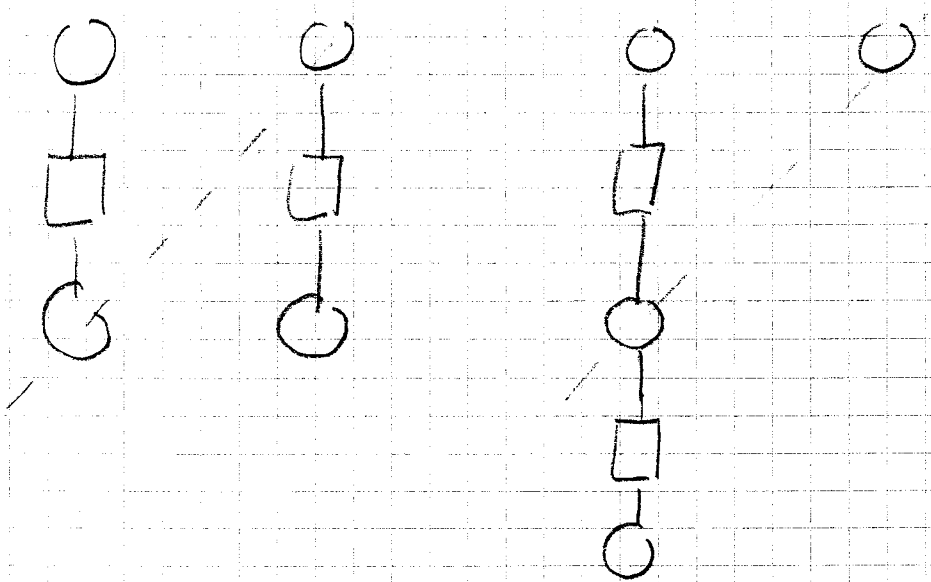
$$\frac{a_1 : u \to v \quad a_2 : v \to w \ \in T[N]^{arrow}}{a_1 ; a_2 : u \to w \in T[N]^{arrow}}$$ plus all the axioms.

The exchange axiom of monoidal categories (see p. 12) yields the partial ordering structure of nonsequential processes. Strictly symmetric monoidal categories are just defined as having independently the structure of commutative monoids (= commutative parallel composition) and categories (= sequential composition). Semantics of concurrency is automatic.

Best-Devillers vs. Goltz-Reisig Processes



Two transitions in sequence: the second transition uses the token returned by the first transition or the other one?



The two traces are different Goltz-Reisig processes but they are the same Best-Deviller process. The difference is a "swapping" operation when the two token are on the same place.

Best-Deviller are the concurrent semantics of nets in the collective token semantics, which is the one obtained using SSMonCat: commutativity of objects causes the swapping identification. The other is individual token.

Collective token semantics is of little use to model causal dependencies, since it mixes up causality with concurrency. Goltz-Reisig processes provide the right semantics: how to get them?

Monoidal Categories

Additional operations and axioms

$- \otimes_o - $: object object $\to$ object
$- \otimes_a - $: arrow arrow $\to$ arrow
$\Rightarrow \otimes$ generic : tensor

1_o : object
1_a : arrow
$\Rightarrow 1$ generic

Monoidal axioms:

$$a \otimes 1 = 1 \otimes a = a$$
$$f \otimes 1 = 1 \otimes f = f$$
$$a \otimes (b \otimes c) = (a \otimes b) \otimes c$$
$$f \otimes (g \otimes h) = (f \otimes g) \otimes h$$

All the categorical operations respect the monoidal structure

$$\frac{f : a \to b \quad g : c \to d}{f \otimes g : a \otimes c \to b \otimes d} \qquad 1 : 1 \to 1$$

$$id(a \otimes b) = id(a) \otimes id(b)$$

$$\frac{f_1 : a_1 \to b_1 \quad g_1 : b_1 \to c_1 \quad f_2 : a_2 \to c_2 \quad g_2 : b_2 \to c_2}{(f_1 ; g_1) \otimes (f_2 ; g_2) = f_1 \otimes f_2 ; g_1 \otimes g_2}$$

; and $\otimes$ represent sequential and parallel composition:

$f_1 : u_1 \to v_1 \quad f_2 : u_2 \to v_2$ — basic interleaving equations can be derived

$$f_1 \otimes id(u_2) ; id(v_1) \otimes f_2 = (f_1 ; id(v_1)) \otimes (id\, u_2 ; f_2) = f_1 \otimes f_2$$

$$f_1 \otimes f_2 = (id(u_1) ; f_1) \otimes (f_2 ; id(v_2)) = id(u_1) \otimes f_2 ; f_1 \otimes id(v_2)$$

Petri nets and Symmetric Monoidal Categories

Symmetric monoidal categories : $\Gamma_{SMonCat}$ see p. 13

Object monoidal operation is not commutative. Arrow operation

$$\frac{f : a \to b \quad g : c \to d}{f \otimes g \,;\, \rho_{b,d} = \rho_{a,c} \,;\, g \otimes f}$$

is commutative up to symmetries (a natural isomorphism).
Symmetries are exactly what is needed, since
they take into account permutations of tokens, but they
still yield equivalence of Goltz-Reisig processes
up to isomorphism.
Unfortunately there is no theory morphism from
$\Gamma_{SCMonGraph}$ (which includes Petri) and $\Gamma_{SMonCat}$
since the commutativity axiom for $\otimes$ on nodes
in $\Gamma_{SCMonGraph}$ has no counterpart in $\Gamma_{SMonCat}$.
It can be proved that there is no functorial
construction (let alone colimit-preserving)
from Petri to SMonCat.

Prenets

Prenets are like nets, but the monoid of places is not commutative.

i) Monoids : T_{Mon}

$$\underline{Set} \underset{\perp}{\overset{-^*}{\rightleftarrows}} \underline{Mon}$$

S^* is the free monoid (strings!) freely generated by S.

ii) Semi-monoidal graphs : $T_{SMonGraph}$

Prenets are those semi-monoidal graphs whose monoid of nodes is free. PreNet is the full sub category

iii) Symmetric monoidal categories : $T_{SMonCat}$

$$\underline{SMonGraph} \underset{\perp}{\overset{H}{\rightleftarrows}} \underline{SMonCat}$$

The adjunction guarantees compositional operational semantics for prenets. But what about nets?

Theorem

Consider the obvious adjunction

$$\underline{SMonGraph} \underset{\perp}{\overset{K}{\rightleftarrows}} \underline{SCMonGraph}$$

Given a Petri net N in $|\underline{SCMonGraph}|$, let P_1 and P_2 any two prenets in $\underline{SMonGraph}$ such that $K[P_1] = K[P_2] = N$. Then $H[P_1]$ and $H[P_2]$ are isomorphic symmetric monoidal categories. Namely, all the prenets corresponding to the same net yield the same model of computation, up to isomorphism

General Construction of Rewriting Systems

i) Specify the basic structure Γ_{Struct}.

ii) Specify the Semi-Struct graphs $\Gamma_{SStruct\text{-}Graph}$ imposing the (free) structure Γ_{Struct} only on nodes (states) and assuming a set (of transitions): rewriting system.

iii) Specify the Struct graphs $\Gamma_{StructRGraph}$ by imposing the independent structures of Struct and reflexive graphs (tensor product of structures). Use the obvious adjunction to build the transition systems.

iv) Specify the Struct categories $\Gamma_{Struct\text{-}Cat}$ by imposing the independent structures of Struct and categories (tensor). Use the obvious adjunction to build the model of computation. The exchange axioms between Struct and categories will equate those arrows corresponding to the same transitions executed in different orders (concurrency).

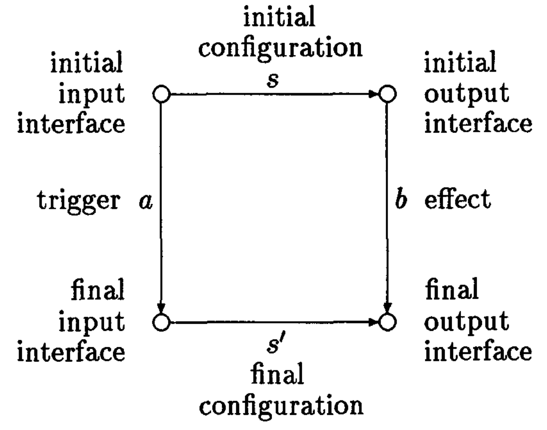
Examples of Struct:

Monoids → Petri nets

GS-monoidal categories → concurrent graph rewriting

Cartesian categories → concurrent term rewriting

Two-sorted cartesian categories → logic programming.


initial
configuration
s
initial
input
interface
initial
output
interface
trigger a
b effect
final
input
interface
final
output
interface
s′
final
configuration

A *double category* has been defined [14] as a category structure on **Cat**, the category of categories, that is, an object of $\mathbf{PAlg}_{CAT}(\mathbf{PAlg}_{CAT}) = \mathbf{DCat}$. The theory $CAT \otimes CAT$ then axiomatizes double categories in partial membership equational logic.

Spelling out the specification of $T \otimes T'$ for the case of $T = T' = CAT$ we get the following poset of sorts, where *Square* is the top:

$$\begin{array}{ll}
(Object, Object) = Object, & (Arrow, Arrow) = Square, \\
(Arrow, Object) = Harrow, & (Object, Arrow) = Varrow, \\
Object \leq Harrow \leq Square, & Object \leq Varrow \leq Square.
\end{array}$$

Rules of the CCS Rewrite System.

$$\mathtt{Pref}_{\mu} : \mu \xrightarrow[\underline{\mu}]{\underline{1}} \underline{1} \qquad \mathtt{Res}_{\mu} : \backslash\alpha \xrightarrow[\underline{\mu}]{\underline{\mu}} \backslash\alpha \qquad \text{for} \quad \mu \notin \{\alpha, \overline{\alpha}\}$$

$$\mathtt{Suml}_{\mu} : + \xrightarrow[\underline{\mu}]{\underline{\mu}\otimes\underline{1}} \underline{1}\otimes! \qquad \mathtt{Sumr}_{\mu} : + \xrightarrow[\underline{\mu}]{\underline{1}\otimes\underline{\mu}} ! \otimes \underline{1}$$

$$\mathtt{Compl}_{\mu} : | \xrightarrow[\underline{\mu}]{\underline{\mu}\otimes\underline{1}} | \qquad \mathtt{Compr}_{\mu} : | \xrightarrow[\underline{\mu}]{\underline{1}\otimes\underline{\mu}} | \qquad \mathtt{Synch}_{\lambda} : | \xrightarrow[\underline{\tau}]{\underline{\lambda}\otimes\underline{\overline{\lambda}}} | .$$

the thesis here:

→ cartesian category

graphs

categories → cartesian categories

↓ arrows

objects

cells

↓ arrow

no good meaning, only identities

They are rewriting rules $L \to R$

$L \Rightarrow R$

$\Downarrow$ cartesian two-category

Rewriting logic by José Meseguer!

The arrow-arrow exchange equation

$$(\alpha ;_h \beta) ;_v (\gamma ;_h \delta) = (\alpha ;_v \gamma) ;_h (\beta ;_v \delta)$$

guarantees concurrency

Rules can be instantiated & contextualised automatically

$$\frac{L \to R}{I;L;C \to I;R;C}$$

Concurrency does NOT give proper algebraic bounds

given $f(x) \to g(x,x)$ $h \to K$ the 2 computations

$f(h) \to g(h,h) \to g(K,h) \to g(K,K)$

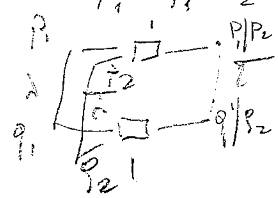
$f(h) \to f(K) \to g(K,K)$

are the same cell (computation)

the "copy" manager destroys concurrency

Truly useful 2nd dimension? Monoidal double cat = tiles

CCS rule: $P_1 \xrightarrow{\lambda} q_1$, $P_2 \xrightarrow{\bar{\lambda}} q_2 \Rightarrow P_1|P_2 \xrightarrow{\tau} q_1|q_2$



synchronization