Some examples in Category Theory

D. Gift Samuel

April 25, 2007

Outline

- SET, $SET_\perp$, Proc
- AUTO, REACH
- THEO, PRES, SPRES
- POSET,Poset,GRAPH, PROOF, LOGI

Definitions

- A category C is a triple $< G, ;, id >$
  - $G = (G_0, G_1, src, trg)$ is a graph
  - ; is a map from G_2 into G_1
  - id is a map from G_0 into G_1
- ; satisfies associative properties.
  - $(f; g); h = f; (g; h)$
- id_x satisfies identity morphism.
  - If $f : x \to y$, $id_x; f = f; id_y = f$

Category: POSET

- Objects are posets $(A, \leq)$
- Morphisms are monotonic functions
- Composition is well defined and it is closed.
  1. Let $(P, \leq_P)$ and $(Q, \leq_Q)$ be posets.
     $f : P \to Q$ and $g : Q \to R$, $(f;g)(x) = g(f(x))$
  2. $x \leq_P y \Rightarrow f(x) \leq_Q f(y)$ by f is monotonic
     $\Rightarrow g(f(x)) \leq_R g(f(y))$ by g is monotonic
     $\Rightarrow (f;g)(x) \leq_R (f;g)(y)$ by definition of composition
  3. $f;(g;h) = (f;g);h$ is true as f,g,h are functions
- for each poset $(P, \leq_P)$, indentity morphism is identity function
  1. $id_P : P \to P$ is monotonic
  2. it satisfies the identity axioms; $f : P \to Q$,
     $id_P; f = f$ and $f; id_Q = f$

Category: SET

- Objects are sets
- Morphisms are total functions
- Composition is functional composition.
  - If $f : A \rightarrow B$ and $g : B \rightarrow c$ are total, then so is.
  - Functional compositional is associative
- Identity morphisms are identity functions
  - Identity function is total
  - for any function $f : A \rightarrow B$, $id_A; f = f; id_B = f$

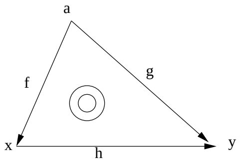
Category: $SET_\perp$

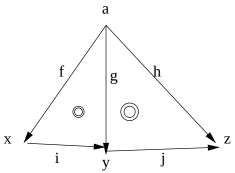
- Objects are pair $< A, \perp_A >$ where $\perp_A \in A$
- morphism between $< A, \perp_A >$ and $< B, \perp_B >$ are total functions s.t. $f(\perp_A) = \perp_B$
- morphism for $SET_\perp$ are all morphism of SET s.t. it satisfies the above condition.

  PROOF
  - composition is defined by functional composition which are inherited from SET
    - composition law is closed for $SET_\perp$
  - identity map assigns to every set the identity function which are also inherited from SET
    - Identities are morphism in $SET_\perp$

Comma Category

- Given a category C and an object $c : C$, we define $a \downarrow C$
- Objects are all the pairs $< f, x >$ where f is a morphism $f : a \longrightarrow x$ in C.
- Morphism between $f : a \longrightarrow x$ and $g : a \longrightarrow y$ s.t. $f; h = g$

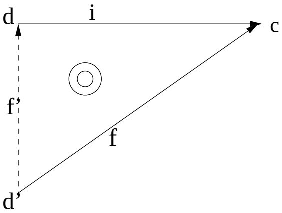




- Category isomorphism between the $1 \downarrow SET$ and $SET_\perp$

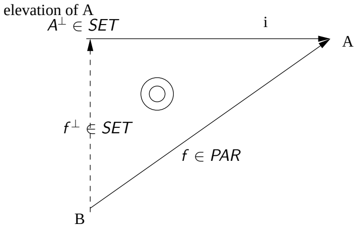
Categories:Co-reflective sub-categories

- D be a co-reflective sub-category of a category C iff every C-object c, co-reflection for c is C-Morphism $i : d \longrightarrow c$ s.t. for any C-morphism $f : d' \longrightarrow c$ where d' is a D-Morphism , there is a unique D-morphism $f' : d' \longrightarrow d$ s.t. f=f';i



SET is co-reflective sub-category of PAR

- PAR is a category where objects are sets and morphisms are partial functions
- SET is subcategory of PAR, but it is not a full subcategory of PAR
- PAR is a Co-reflective sub-categories of SET
- Proof



Category: $FSET_\perp$

- Objects are pair $< A, \perp_A >$ where $\perp_A \in A$ and A is finite
- morphism between $< A, \perp_A >$ and $< B, \perp_B >$ are total functions s.t. $f(\perp_A) = \perp_B$
- composition is functional composition and identity map assigns to every set the identity function
- $FSET_\perp$ is a full subcategory of $SET_\perp$

Category: Proc

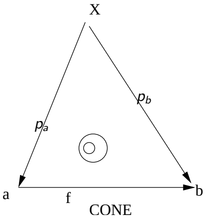
- Objects are process behaviour(or process) $P = < A_\perp, \Lambda >$ where $A_\perp$ is a finite pointed set and $\Lambda \subseteq A_\perp^\omega$.
  - $\lambda \in \Lambda$ is a function from $\lambda : \omega \to A_\perp$. an finite sequence of elements of $A_\perp$
  - Given a process $< A_\perp, \Lambda_A >$, $A_\perp$ is called events of P,and $A_\perp \setminus \perp_A$ is called alphabet of P(denoted as P_α). Λ is called behaviours of P.
  - $\perp$ is called the environment event of P
- process morphism $h : P = < A_\perp, \Lambda_A > \to Q = < B_\perp, \Lambda_B >$ is a morphism $h : A_\perp \to B_\perp$ s.t. $h^\omega(\Lambda_A) \subseteq \Lambda_B$. where $h^\omega(\lambda) = \lambda; h$
- processes and process morphism constitute a category Proc
- Forget functor $U_\perp : Proc \to FSet_\perp$ that sends each process to its alphabets and each morphism $f : (A_1, \Lambda_1) \to (A_2, \Lambda_2)$ to $f : A_1 \to A_2$ is faithful.

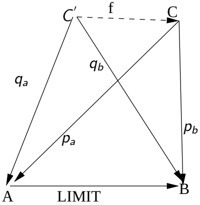
Category: Proc

- Morphism $h : P = < A_\perp, \Lambda_A > \rightarrow Q = < B_\perp, \Lambda_B >$ is a embedding of the process Q within P, making Q a component-of P.
- $h : < A_\perp, \Lambda_A > \rightarrow < B_\perp, \Lambda_B >$ is a morphism $h : A_\perp \rightarrow B_\perp$ s.t. $h^\omega(\Lambda_A) \subseteq \Lambda_B$
  - environment of P are also in the environment of Q
  - any alphabet of P can be mapped onto $\perp_B$: P identifies part of the environment of Q: P doesn't participate on the event.
  - the behaviour of P be compatible with Q: the life cycle of Q is mapped to life cycle of P.

Definition: Cone, Limit

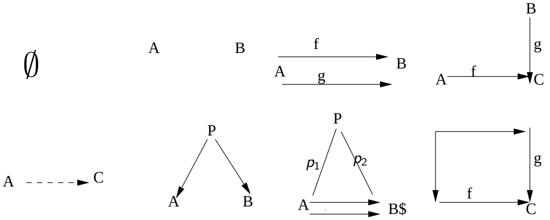
- A communtative cone over diagram δ consists of an object C together with morphism $p : C \rightarrow \delta$ s.t. for every $f : A_i \rightarrow A_j$ we have $f; \delta_i = \delta_j$
- A limit for the diagram D is a commutative cone $p : C \rightarrow \delta$ s.t. for every commutative cone $p' : C' \rightarrow \delta'$ there is a unique morphism $f : C' \rightarrow C$ such that $p; f = p'$ ($p_a; f = p'_a$ for every edge)
- Terminal, products,equalizers and pushback are speciallization of limits





Definition: Universal properties

- Limit of two object without any morphism is a product. The limit of empty diagram is the terminal object
- Limit of two parallel morphism with the same domain and co-domain is the equalizer. Pullback is the limit of two morphism with the same co-domain.



Terminal Object C Product P Equaliser pullback

Proc: Initial and Terminal

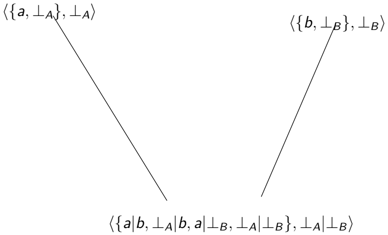
- Terminal process is $< \{\perp_\emptyset\}, \{\perp_\emptyset^\omega\} >$. Its alphabet contains only the witness for action of the environment. It is a model for idle process.
- Terminal process is the innermost component
- Initial process $< \{\perp_\emptyset\}, \emptyset >$. It does nothing. It models a deadlock process. deadlock any process to which it is connected
- Initial process is the outmost component.
- Initial objects and terminals objects are in $SET_\perp$ are singleton sets $< \{a\}, a >$

Universal property: Product

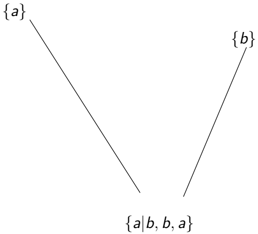
- The object z is the product of x and y with projection $\pi_x : z \to x$ and $\pi_y : z \to y$ iff for any v and pair of morphism $f_x : v \to x$ $f_y : v \to y$, then there is a unique morphism $k : v \to z$
- Every poset$(P, \leq)$ is a category. Objects are elements in P. Morphisms are given by the relation $\leq$.
- Composition is defined by transitivity law. Identity morphism are defined by reflexitivity laws
- Universal Properties
  - The least element is the initial object, the greatest element is the terminal object
  - In a category of poset $(P, \leq_P)$, greatest lower bound z is the product of p and q
  - $glb(p, q) \leq p$, $glb(p, q) \leq q$, are projections
  - if $c \leq p$ and $c \leq q$, then $c \leq glb(p, q)$, which is unique

Universal properties of Proc

- Product of two processes alphabets



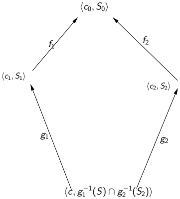
DENOTED AS



- Parallel composition without interaction

Universal properties of Proc

- product represents traditional trace-based semantics of parallel compositions without synchronisation
- Product of two processes $< A_1, \Lambda_1 >$ and $< A_2, \Lambda_2 >$ is obtained
  - by computing the product of the alphabets $(A_1 \times A_2)$
  - consist all lifecycles of the products, once projected into the components alphabets, give life cycles of the component process
- consists λ, $\lambda : \omega \rightarrow (A_1 \times A_2)$ such that $g_1^{\omega}(\lambda) \in \Lambda_1$ and $g_2^{\omega}(\lambda) \in \Lambda_2$



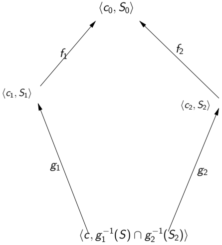
- Parallel composition without interaction

Idle and deadlock process

- When terminal process put in parallel with another process, the result of the parallel composition is the other process
- Initial process(blocking process) absorbs any other process when put in parallel
- product with other process returns an empty behaviour

Universal properties of Proc

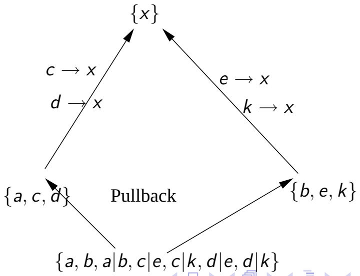
- Pullback represents traditional trace-based semantics of parallel compositions with synchronisation
- Product of two morphism f_1 and f_2 is obtained by computing the product of the alphabets and by taking as set of behaviours the intersection of the inverse images of the set of behaviours of the components and equal
- consists λ, $\lambda : \omega \longrightarrow (A_1 \times_A A_2)$ such that $g_1^\omega(\lambda) \in \Lambda_1$ and $g_2^\omega(\lambda) \in \Lambda_2$



- Parallel composition without interaction

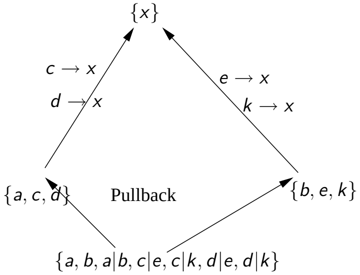
Universal properties of Proc

- $A_1 = \{\bot, a, c, d\}$, $A_2 = \{\bot, b, e, k\}$ and $A = \{\bot, x\}$ with functions $f = \{\bot \to \bot, a \to \bot, c \to x, d \to x\}$ and $g = \{\bot \to \bot, b \to \bot, e \to x, k \to x\}$
- product $A_1 \times A_2 =$ $\{\bot, a, c, d, b, e, k, a|b, a|e, a|k, c|b, c|e, c|k, d|b, d|e, d|k\}$
- pullback is obtained by keeping only the events that after being projected to A_1 and A_2 are mapped through f and g to the same element of A.
- $A_1 \times_A A_2 = \{\bot, a, b, a|b, c|e, c|k, d|e, d|k\}$



An Example

- Λ_{P1} contains all the life cycle of the form $\perp^* a \perp^* c \perp^* a \perp^* c \perp^* \cdots$ and Λ_{P2} contains $\perp^* e \perp^* b \perp^* e \perp^* b \perp^* \cdots$
- Pullback is $\perp^* a \perp^* c|e \perp^* \{a \perp^* b, b \perp^* a, a|b\} \perp^* c|e \perp^* \{a \perp^* b, b \perp^* a, a|b\} \cdots$
- e can synchronised either with c or d, so e has to wait until c or d appears



Reference

- Mirror, Mirror in my Hand: a duality between specifications and models of process behaviour J.L. Fiadeiro and J.F. Costa

Category of Automatas(AUTO)

- Objects in AUTO are automata (X, S, Y, f, s_0, g)
- Morphisms in AUTO are simulations: B simulates A if $A \rightarrow B$
- Morphism from $A = (X, S, Y, f, s_0, g)$ to $B = (X^{'}, S^{'}, Y^{'}, f^{'}, s_0^{'}, g^{'})$ is a tuple $< h : X \rightarrow X^{'}, i : S \rightarrow S^{'}, j : Y \rightarrow Y^{'} >$ such that
  - $i(s_0) = s_0^{'}$
  - $f; i = h \times i; f^{'}$
  - $g; j = i; g^{'}$

Proof for AUTO

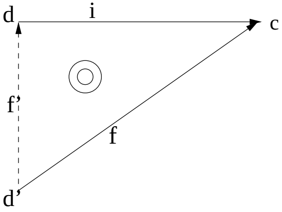
- Composition of two morphisms is a morphism
- (h, i, j) : $A = (X, S, Y, f, s_0, g)$ to $B = (X^{'}, S^{'}, Y^{'}, f^{'}, s_0^{'}, g^{'})$
  $(h^{'}, i^{'}.j^{'})$: $B = (X^{'}, S^{'}, Y^{'}, f^{'}, s_0^{'}, g^{'})$ to $C = (X^{''}, S^{''}, Y^{''}, f^{''}, s_0^{''}, g^{''})$ and
- Composition is $< h; h^{'}, i; i^{'}, j; j^{'} >$
- $(i; i')(s_0)$
  $= i(i'(s_0))$ by composition definition
  $= i(s_0^{'})$ by $< h, i, j >$ is morphism
  $= s_0^{''}$ by $< h^{'}, i^{'}, j^{'} >$ is morphism
- $f; (i; i^{'})$
  $= (f; i); i^{'}$ by associativity
  $= (h \times i; f^{'}); i^{'}$ by morphism
  $= h \times i; (f^{'}; i^{'})$ by associativity
  $= (h \times i; (h^{'} \times i^{'}); f^{''}$ by morphism
  $= (h \times i; h^{'} \times i^{'}); f^{''}$ by associativity
  $= (h; h^{'}) \times (i; i^{'}); f^{''}$ by composition

Initials and Terminals

- $\emptyset$ is the initial object in SET
- $\{a\}$ is a terminal object in SET.
- $(\emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$ is a initial objects in AUTOM
- $(\{i\}, \{s\}, \{o\}, \{i \times s \rightarrow s\}, s, \{s \rightarrow o\})$ is a terminal object in AUTOM

Co-reflective sub-categories of AUTO

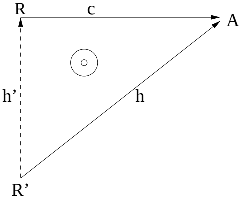
- D be a co-reflective sub-category of a category C iff every C-object c, co-reflection for c is C-Morphism $i : d \rightarrow c$ such that for any C-morphism $f : d' \rightarrow c$ where d' is a D-Morphism , there is a unique D-morphism $f' : d' \rightarrow d$ such that f=f';i



- Reachable Automata: automate that is obtained by removing all non-reachable states.
- In REACH, objects are reachable automata.
- Morphisms are simulations.

REACH is a Co-reflective sub-category of AUTO

- A is related to canonical reachable automata R by $c : R \to A$



- $A = (X, S, Y, s_0, f, g)$ and $R = (X, S_R, Y, s_0, f_R, g_R)$ where $S_R \subseteq S$, X,Y are identities
- Given any reachable automata R' and simulation $h : R' \to A$, there is a unique morphism of reachable automata $h' : R' \to R$ such that $h = h'; c$
- Co-reflector for an object is a morphism through which all communication must go.

Temporal propositions

- A signature of LTL is a set of actions symbols.
- The action symbols provide atomic propositions in the LTL formula.
- The set of temporal propositions $prop(\Sigma)$ for a signature is inductively defined as
  - Every action symbol is a temporal propositions
  - beg is a temporal propositions (denotating the initial state)
  - if ϕ is a temporal proposition so is $\neg\phi$
  - if ϕ_1 and ϕ_2 is a temporal proposition so are $\phi_1 \supset \phi_2$, $\phi_1 U \phi_2$ and $\phi_1 W \phi_2$

Semantics

- An interpretation structure for a signature Σ is a sequence $\lambda \in (2^{\Sigma})^{\omega}$
- $\lambda \in (2^{\Sigma})^{\omega}$ is true at state i which we write $\lambda \models_{\Sigma,i} \phi$
  - if $\phi \in \Sigma$, $\lambda \models_{\Sigma,i} \phi$ iff $\phi \in \lambda(i)$
  - $\phi \in \Sigma$, $\lambda \models_{\Sigma,i} beg$ iff $i = 0$
  - $\phi \in \Sigma$, $\lambda \models_{\Sigma,i} \neg\phi$ iff it is not the case $\lambda \models_{\Sigma,i} \phi$
  - $\phi \in \Sigma$, $\lambda \models_{\Sigma,i} \phi_1 \supset \phi_2$ iff $\lambda \models_{\Sigma,i} \phi_1$ implies $\lambda \models_{\Sigma,i} \phi_2$
  - $\phi \in \Sigma$, $\lambda \models_{\Sigma,i} \phi_1 U \phi_2$ iff for some $j > i$, $\lambda \models_{\Sigma,j} \phi_2$ and $\lambda \models_{\Sigma,k} \phi_1$ for every $i \leq k \leq j$
- (weak until) $(\phi_1 W \phi_2)$ holds $(\phi_1 U \phi_2)$, or ϕ_2 will forever be false and ϕ_1 true.
- ϕ is true in ϕ for λ, written, $\lambda \models_{\Sigma} \phi$ iff $\lambda \models_{\Sigma,i} \phi$ for every state i
- $\Phi \vdash_{\Sigma} \phi$ iff ϕ is true in every sequence that makes all the propositions in Φ true

Examples

- specification vending machine is
  signature coin,cake,cigar
  axioms: $beg \supset$
  $((\neg cake \wedge \neg cigar) \wedge (coin \vee (\neg cake \wedge \neg cigar) W coin))$
  $coin \supset (\neg coin) W (cake \vee cigar)$
  $(cake \vee cigar) \supset (\neg cake \wedge \neg cigar) W coin$
  $cake \supset (\neg cigar)$
- accepts coins, delivers cakes and cigars

Interpretations between theories

- Let Σ be a signature and Ψ a subset of $prop(\Sigma)$ is said to be closed iff for every $\phi \in prop(\Sigma)$, $\Psi \vdash_{\Sigma} \phi$ implies $\phi \in \Psi$.
- $c_{\Sigma}(\Psi)$ denotes the least closed set that contains Ψ.
- Let Σ and$\Sigma^{'}$ be signatures. Every functions $f : \Sigma \rightarrow \Sigma^{'}$ extends to a $prop(f) : prop(\Sigma) \rightarrow prop(\Sigma^{'})$ as follows
  - prop(f)(beg)=beg
  - if $a \in \Sigma$ then prop(f)(a)=f(a)
  - $prop(f)(\neg\phi) = (not\ prop(f)(\phi))$
  - $prop(f)(\phi_1 \supset \phi_2) = (prop(f)(\phi_1) \supset prop(f)(\phi_2)$
  - $prop(f)(\phi_1 \supset \phi_2) = (prop(f)(\phi_1) U prop(f)(\phi_2)$

More Categories

- $THEO_{LTL}$ is the category of theories with:
  - objects: $< \Sigma, \Phi >$, where $\Phi = c_{\Sigma}(\Phi)$.
  - morphism $f :< \Sigma, \Phi > \rightarrow < \Sigma', \Phi' >$ is a sig. morph. $f : \Sigma \rightarrow \Sigma'$ such that $prop(f)(\Phi) \subseteq \Phi'$.
-
  - $PRES_{LTL}$ is the category of presentation with:
    - objects: $< \Sigma, \Phi >$,
    - morphism $f :< \Sigma, \Phi > \rightarrow < \Sigma', \Phi' >$ is a sig. morph. $f : \Sigma \rightarrow \Sigma'$ such that $prop(f)(c_{\Sigma}(\Phi)) \subseteq c_{\Sigma'}(\Phi')$.
- $THEO_{LTL}$ is a category
  - $prop(f;g)(\Phi) = prop(g)prop(f)(\Phi)$
  - $(f;g)$ is a theory morphism
- $PRES_{LTL}$ is a category

$THEO_{LTL}$ is a subcategory of $PRES_{LTL}$

- Every theory is a presentation.
- Given a theory $f : (\Sigma, \Phi) \longrightarrow (\Sigma^{'}, \Phi^{'})$, we need to prove $prop(f)(c_{\Sigma}(\Phi) \subseteq_{c_{\Sigma}} (\Phi^{'})$
  - As f is a theory morphism, $prop(f)(\Phi) \subseteq \Phi^{'}$ and Φ and $\Phi^{'}$ are closed, $c(\Phi) = \Phi$ and $c(\Phi^{'}) = \Phi^{'}$

Category:CLOSURE

- A closure system is a pair $< L, c >$ where L is a set and $c : 2^L \to 2^L$ is a total function satisfying the following properties:
  - Reflexivity: for every $\Phi \subseteq L,\ \Phi \subseteq c(\Phi)$.
  - Monotonicity: for every $\Phi, \Gamma \subseteq L,\ \Phi \subseteq \Gamma$ implies $c(\Phi) \subseteq c(\Gamma)$.
  - Idempotence: for every $\Phi \subseteq L,\ c(c(\Phi)) \subseteq c(\Phi)$.
- Objects in CLOSURE are closure systems
- morphisms $f :< L, c > \to < L', c' >$ are the maps $f : L \to L'$ such that $f(c(\Phi)) \subseteq c'(f(\Phi))$ for all $\Phi \subseteq L$.

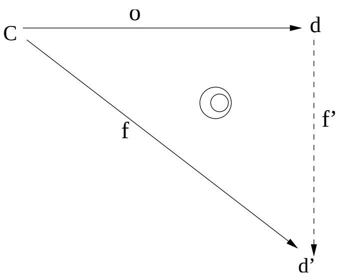
Reflective & Co-Reflective sub categories

Consider a closure systems (L, c).
$\Phi \subseteq L$ is closed iff $\Phi = c(\Phi)$.

- THEO is the category of theories with:
  - objects: closed subset of L
  - morphism: Inclusions.
- SPRES is the category of strict presentation with:
  - objects: subsets of L
  - morphism: Inclusions.
- PRES is the category of presentation with:
  - objects: subsets of L
  - morphism: by the preorder $\Phi \leq \Gamma$ iff $c(\Phi) \subseteq c(\Gamma)$

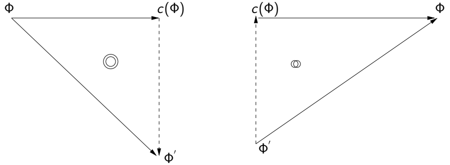
Categories: Reflective sub-categories

- D be a Reflective sub-category of a category C iff every C-object c, reflection for c is C-Morphism $o : c \rightarrow d$ such that for any C-morphism $f : c \rightarrow d'$ where d' is a D-Morphism , there is a unique D-morphism $f' : d \rightarrow d'$ such that f=o;f'



Reflective & Co-Reflective sub categories

- THEO is a full subcategory of PRES and SPRES.
- SPRES is a subcategory of PRES.
  It is an immediate consequence of the monotonicity
- THEO is a reflective subcategory of PRES and SPRES.
- SPRES is not co-reflective subcat of PRES.



- THEO is a coreflective subcategory of PRES.
- THEO is not a coreflective subcategory of SPRES.
- SPRES is a coreflective subcategory of PRES.

Functor between SET and MON

- SET is a category.
  - Objects in SET are sets
  - Morphisms in SET are total functions
- MON is a category.
  - Objects in MON are $(List(S), \diamond, [])$
  - List(S)is free monoid generated by S.
  - Morphism in MON are monoid homomorphism.
- functor LIST maps $SET \rightarrow List$ (which is object part of functor)
- functor LIST maps $f : S \rightarrow S^{'}$ to a function $LIST(f) : List(S) \rightarrow List(S^{'})$ (which forms morphism part of functor)
- Given a list $L = [s_1, s_2, \cdots, s_n]$ maps f over the elements of $list$:
  $LIST(f)(L) = f^*(L) = [f(s_1), f(s_2), \cdots, f(s_n)]$

Functor between SET and MON [conti...]

- f^* is a homomorphism
  $f^*([]) = []$
  $f^*(L \diamond L') = f^*(L) \diamond f^*(L')$
  $f^*([s]) = [f(s)]$
- Any total function between sets induces the monoid homomorphism between the corresponding monoids.
  PROOF
  - Preservation of Identities
    - $List(id_s)(L) = [id_s(s_1), \cdots, id_s(s_1)]$
    - $= [s_1, \cdots s_n] = L$
    - $= id_{List(s)}(L)$
  - Preservation of composition