Spatial Process Algebra for Developmental Biology

Luca Cardelli

Microsoft Research

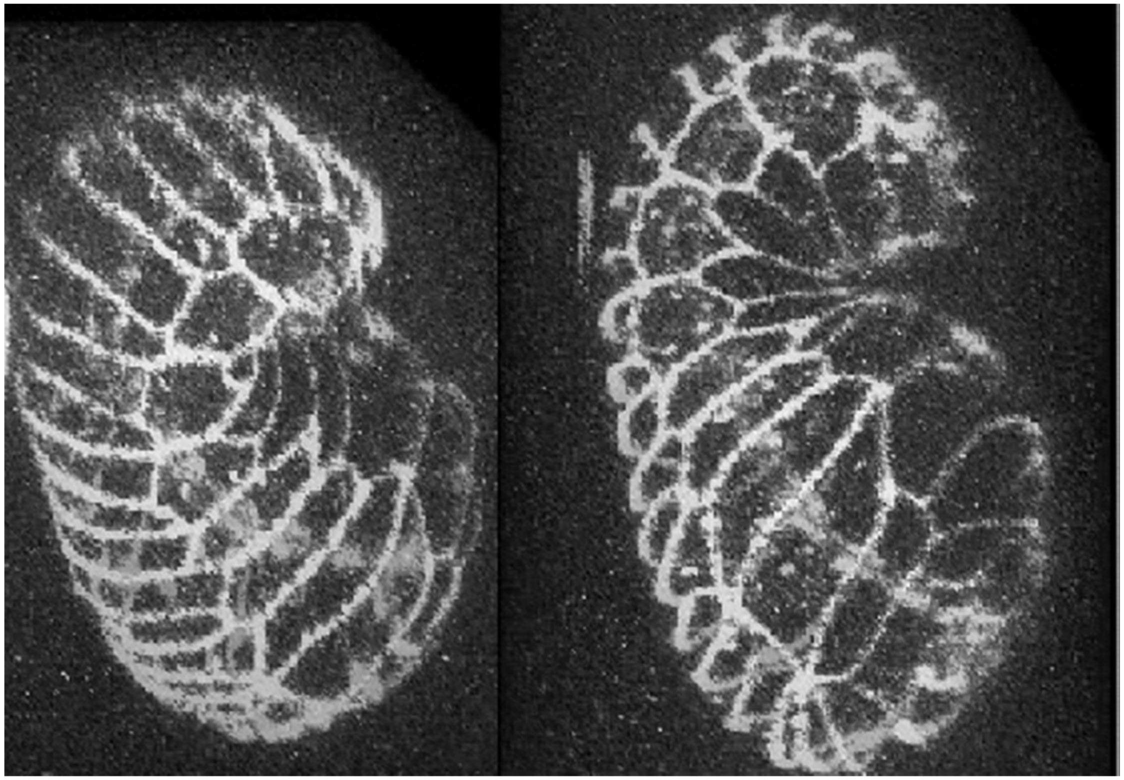
with Philippa Gardner

Imperial College London

2010-03-23 Torino

http://LucaCardelli.name

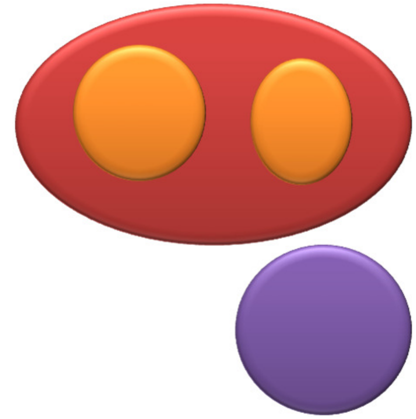
Introduction



4D movie of developing *C. Elegans* embryo from Mohler Lab

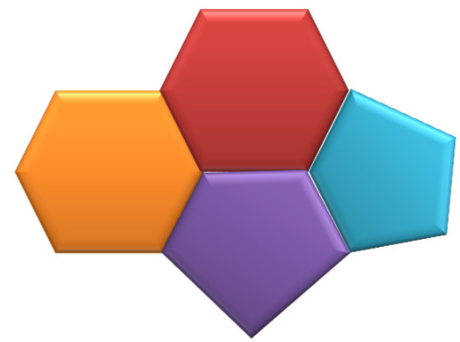
From Topology to Geometry

- Process Algebra and Membrane Computing
  - Successful in describing the hierarchical (*topological*) organization and transformation of complex systems.



- In many situations, however, *geometry* is necessary
  - Geometric constraints exist both at the subcellular level and at higher levels of cellular organization.
  - Developmental biology deals with dynamic spatial arrangements, and with forces and interactions.

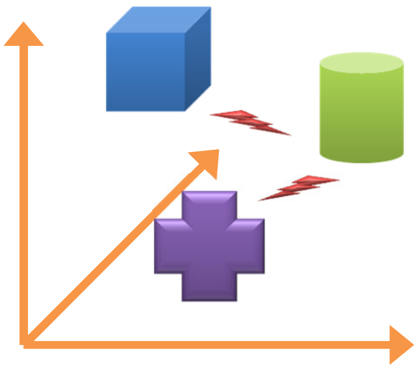
- While many discrete geometric approaches exists
  - Cellular Automata, and graph models.
  - Geometric extensions of L-systems.



- Few cover both key aspects of development
  - Rich geometry.
  - Rich communication.

3π

- We have developed a geometric process algebra
  - Processes located in 3-dimensional space.
  - With a kinetic component (change of position over time).
  - With rich communication capabilities (as usual in π).



- Easy you say: just ‘add a position to each process’
  - Naive attempts result in awkward formal systems with too many features: coordinates, position, velocity, identity, force, collision, communication…

- Developmental biology is geometrically peculiar
  - The coordinate space is not fixed: it effectively expands, moves, and warps as the organism develops.
  - Approaches based fixed grid or coordinate systems are awkward.

- Algorithmic Botany has shown the way:
  - Affine geometry: the geometry of properties invariant under linear transformations and translations.

Shifting Reference Frames

- 3π is a π-calculus extended with a single new process construction able to shift the frame of reference. (N.B.: 'shifting' means 'composing'.)

- A *frame shift*, consists of dynamically applying a 3-dimensional affine transformation to a whole evolving process.
  - This is sufficient to express many dynamic geometric behaviors, thanks to the combined power of Affine Geometry and Process Algebra.
  - The π-calculus remains relatively simple, technically formulated in a familiar way, with a large but standard and modular geometric subsystem.

- From a Process Algebra point of view:
  - 3π adds powerful geometric data structures and transformations.
  - Position and communication collude to model *forces*.

- From an Affine Geometry point of view:
  - Key geometric invariants become theorems of 3π: *relativity*.
  - These invariants ensure that processes can be freely transformed, and that they cannot 'cheat' geometrically.

It's Still About Nesting

- While containment is the key notion in topological models:

  n[P] a process P in a container with interface n

  - Algebraic rules for manipulating nested n[-] brackets.

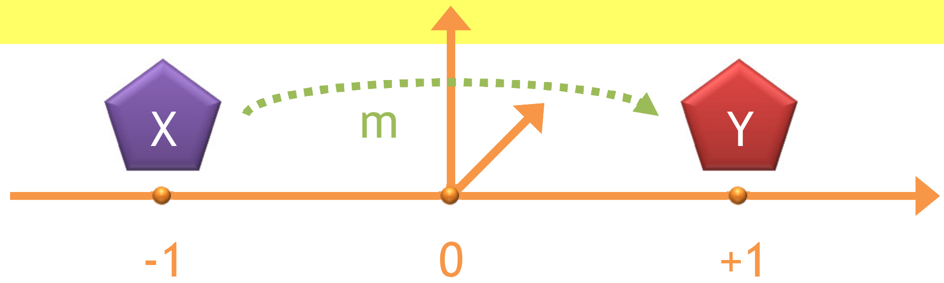
- Here frame shift is the key notion in geometric models:

  M[P] a process P transformed by an affine map M

  - Algebraic rules for manipulating nested M[-] brackets.

Examples

Ex: Distance Between Processes



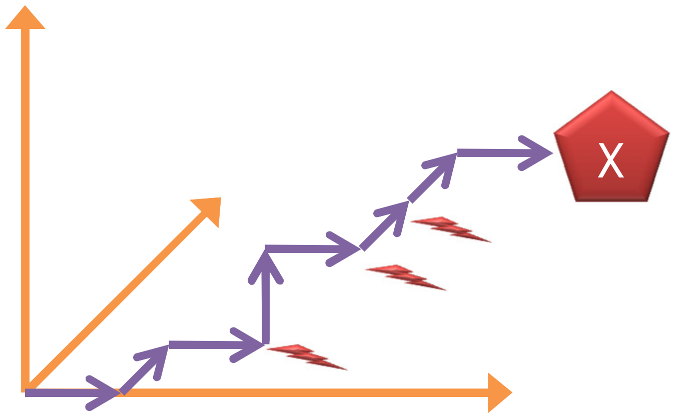
$X = T(-↥_x)[P]$ where $P = !m(⊹)$

$Y = T(↥_x)[Q]$ where $Q = ?m(x).\ \|x∸⊹\|{=}2.\ R$

- **Two processes on the X axis:**
  - Process P is relocated at -1 on the x axis, by a translation $T(-↥_x)$ with unit x vector $↥_x$. Process Q is relocated at +1 by a translation $T(↥_x)$.
  - When P outputs *its origin* ⊹ on m, the actual value sent is the point ⟨-1,0,0⟩.
  - Process Q receives that value as x, and computes the size of the *vector* x∸⊹. In the frame of Q that is the size of the vector ⟨-1,0,0⟩ ∸ ⟨1,0,0⟩, which is 2.
  - Therefore, the comparison $\|x∸⊹\|{=}2$ succeeds, and process R is activated, having verified that the distance between P and Q is 2.

- Note: the *affine basis* $⊹, ↥_x, ↥_y, ↥_z$ is available to each process, but is interpreted relatively to the *current frame* of the process.

Ex: Random Motion and Telemetry

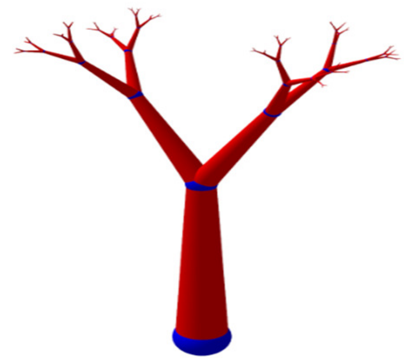


$$X = \tau.T(\hat{1}_x)[X] + \tau.T(\hat{1}_y)[X] + \tau.T(\hat{1}_z)[X] + !c(\div).X$$

- The recursive process X nonderministically applies unit translations to itself along the axes.
- It also nondeterminstically outputs its current origin on channel c, therefore producing a telemetry of its movements.

Ex: Orthogonal Bifurcation in Development

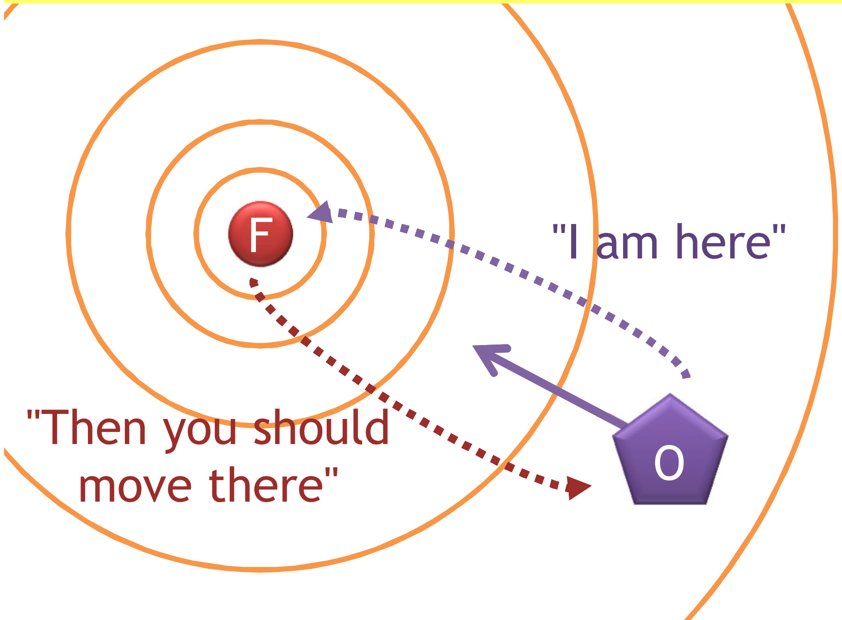
- Lung development in mouse is based on three splitting processes [9].
  - We show how to represent the third (orthogonal bifurcation, Orth).
  - Bifurcations alternate between orthogonal planes.



$$\mathrm{Orth} = !c(⊹).\ (\mathrm{M90}(\pi/6)[\mathrm{Orth}] \mid \mathrm{M90}(-\pi/6)[\mathrm{Orth}])$$
$$\mathrm{M90}(\theta) = \mathrm{R}(\mathrm{M}(\theta)[↥_y],\pi/2)\circ \mathrm{M}(\theta)$$
$$\mathrm{M}(\theta) = \mathrm{Sc}(½)\circ \mathrm{R}(↥_z,\theta)\circ \mathrm{T}(↥_y)$$

  - M(θ): applies a translation $T(↥_y)$ by $↥_y$, a rotation $R(↥_z,\theta)$ by θ around $↥_z$, and a uniform scaling Sc(½) by ½.
  - M90(θ): first applies an M(θ) transformation in the XY plane, and then applies a further 90° rotation around the ‘current’ direction of growth, which is $M(\theta)[↥_y]$, therefore rotating out of the XY plane for the next iteration.
  - Orth: Outputs the current origin ⊹ to the c channel at each iteration, providing a trace of the growing process that can be plotted. Opposite 30° rotations applied recursively to Orth itself generate the branching structure.

Ex: Force fields



Force = (?f(x,p). !x(M{p}))*
Object = (νx) !f(x,+). ?x(Y). Y[Object]

f is the force field channel; M{p} is a map

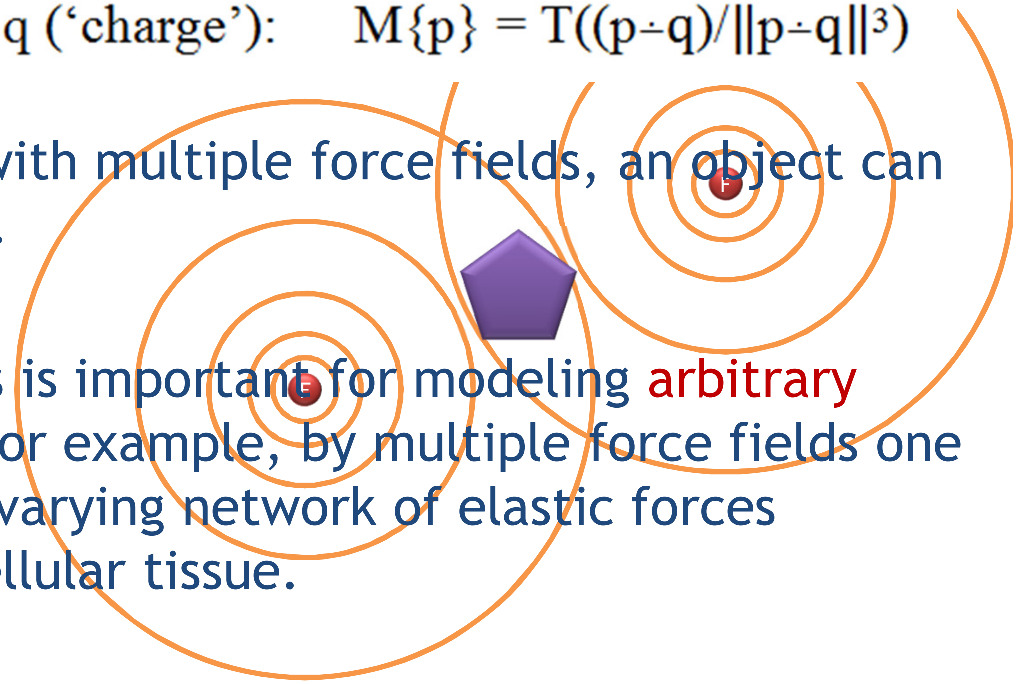
- A force field is a process F that receives the location of an ‘object’ process P (and, if appropriate, a representation of its mass or charge), and tells it how to move by a discrete step.

- The latter is done by replying to the object with a transformation that the object applies to itself.

Ex: Various Forces

- This force field transformation can depend on the distance between the object and the force field, and can easily represent inverse square and linear (spring) attractions and repulsions.

A uniform field ('wind'):	$M\{p\} = T(\hat{1}_x)$
A linear attractive field at q ('spring'):	$M\{p\} = T(\frac{1}{2}\cdot(q \dot{-} p))$
An inverse-square repulsive field at q ('charge'):	$M\{p\} = T((p \dot{-} q)/\|p \dot{-} q\|^3)$

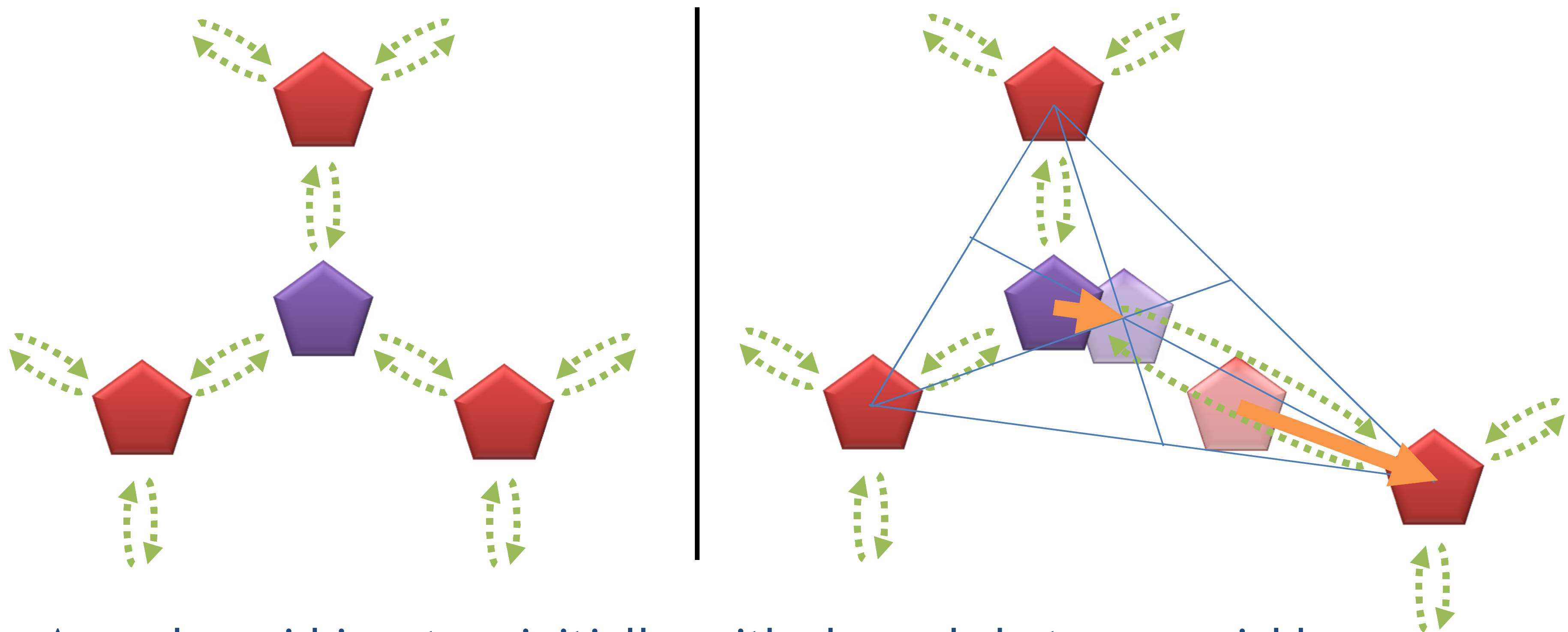


- By nondeterministic interaction with multiple force fields, an object can be influenced by several of them.

- The ability to express force fields is important for modeling arbitrary constraints in physical systems. For example, by multiple force fields one can set up an arbitrary and time-varying network of elastic forces between neighboring cells in a cellular tissue.

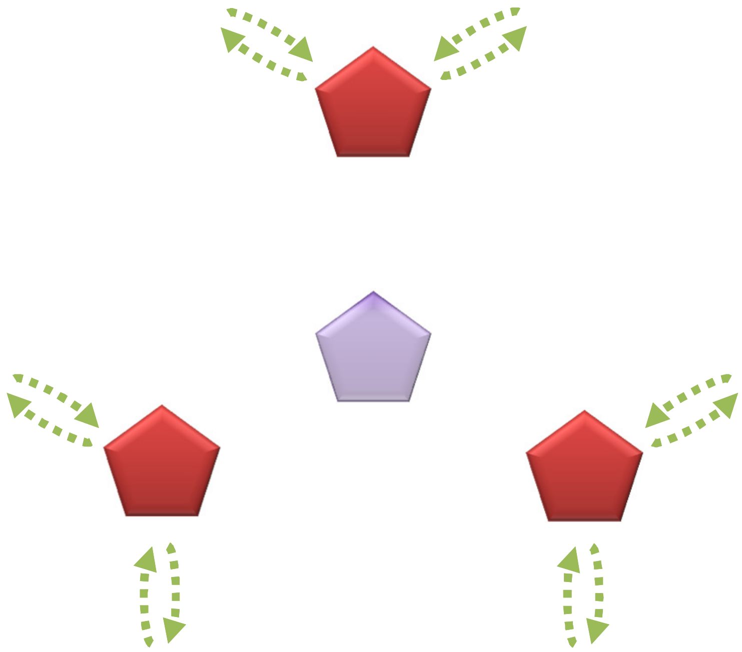
- Some forces have unlimited range: one should *not* restrict communication based on distance.

Ex: Self-Adjusting Grid



- A regular grid is set up initially, with channels between neighbors.
- Each process repeatedly polls the position of its neighbors, and moves to their geometric mean position.
- If one process moves the others adjust, eventually recreating a regular grid.

Ex: Self-Healing Grid



- If a process dies, the neighbors try to communicate via other paths to recreate a single process at their mean position.

- (These quickly become pretty tricky exercises in distributed programming.)

Design Decisions

Motivation

- Process algebras
  - Are used to study fundamental primitives of interaction between distributed, concurrent processes.
  - Used successfully to analyze distributed systems.
  - More recently, used for modeling interaction in biochemical systems.
  - Including topological (containment) interactions.

- Geometric process algebras
  - Combine the interaction primitives of process algebra with geometric transformations.
  - To model geometric properties in distributed systems.
  - And in particular to model geometric properties of biochemical systems.

- Biological systems
  - Provide complex examples of process interactions that depend on geometry, and of geometry that depends on interactions.
  - Such as the growth of tissues, the diffusion of signaling molecules, and the overlapping of chemical gradients.

What Kind of Geometry?

- During biological development, tissues expand, split and twist, and there is no fixed coordinate system that one can coherently apply.
- It is natural to turn to *affine geometry*, which is the geometry of properties that are invariant under linear transformations and translations.
- This has been used successfully, for example, to model plant development (geometric variants of L-systems).
- What's new here? We add communication to affine geometry, so as to model the fine coordination that is needed during development.

Will It Blend?

- How are we going to blend the geometry with the process algebra?
- Q1: How should the position of a process be represented?
- Q2: How should a process move from one position to another?
- Q3: How should processes at different positions interact?
- Q4: What theorems can we prove that blend properties of geometry and of process algebra, to show we have reached a smooth integration?

Q1: Processes in Space

- The location of a 3π process
  - Each process 'believes' to be located at the origin ⊹. *This convention avoids the complexity of introducing a separate syntactic notion of process location.*
  - The true location of a process is however given by $\mathcal{A}(\text{⊹})$, for the global frame $\mathcal{A}$ that the process is in.
  - The *spatial extent* of a process is not represented, except by its interactions with other processes.
  - Hence, different processes can be in different locations, *if* we have some way of manipulating the global frame.

- The scale and orientation of a 3π process
  - Each process 'believes' to be oriented in the orthogonal frame $↥_x, ↥_y, ↥_z$, and to have normal scale 1 along all three axis.
  - The true scale and orientation of a process is however given by $\mathcal{A}(↥_x), \mathcal{A}(↥_y), \mathcal{A}(↥_z)$, for the global frame $\mathcal{A}$ that the process is in.
  - Hence, different processes can have different scales and orientations, again *if* we have some way of manipulating the global frame.

Q1: *Active* Processes in Space

- More than geometric arrangements
  - We do not have just static arrangements of entities in space.
  - We have active processes that can compute, observe, communicate, and even move, transform, and apply forces.

- The observations of a 3π process
  - A process can carry out comparisons between computed values (observations), and change behavior based on the result.
  - Since computations are relative to a frame, these are observations in a frame, whose result may change from one frame to another. Very much like 'physics experiments'.
  - Some processes can observe absolute scale and absolute handedness, but never absolute position or orientation.

Q2: Manipulating the Global Frame

- A process can be placed in a new frame by a *frame shift* operation M[Q], where the *local frame* M denotes an affine map $\mathcal{B}$.
  - If a process M[Q] is in a global frame $\mathcal{A}$, then the process Q is in the shifted global frame $\mathcal{A}\circ\mathcal{B}$.
- Different parts of a process can be shifted differently.
  - The process M[Q] | N[R] indicates that processes Q and R are in different frames, with Q shifted by M and R by N.
  - Frame shift operations can be nested, with the process $M[N_1[Q] \mid N_2[R]]$ indicating that Q is in the frame shifted first by N_1 and then M, whereas R is shifted by N_2 then M.
  - Conversely, the process M[Q] | M[R] = M[Q | R] indicates that Q and R are in the same frame.
  - Frame shift can be used by a process to change its own global frame, by recursively applying a frame shift to itself.
- Frame shift is more than a change of location
  - Since M denotes a general affine map, it can change orientation scale, etc.

Q3: Communication Across Frames

- Processes interact by exchanging data over shared channels
  - Messages may contain geometric data or other communication channels.
  - Data is evaluated in its current geometric frame before transmission, and then transmitted ('by value') to the receiver.

- Process interaction is not restricted by distance
  - Because some forces have infinite range.
  - Because geometric entities are not always spheres!

- Communication can happen across local frames:

$$P \stackrel{\text{def}}{=} M[!x(\div).Q] \mid N[?x(z).R] \quad {}_{\mathcal{A}}\!\!\rightarrow \quad M[Q] \mid N[R\{z \backslash \varepsilon\}]$$

  - where M evaluates to $\mathcal{B}$ in the global frame $\mathcal{A}$ and $\div$ evaluates to $\varepsilon = \mathcal{A}\circ\mathcal{B}(\div)$. This interaction across frame shifts is achieved via the equality:

$$P = !x(M[\div]).M[Q] \mid ?x(z).N[R]$$

    which distributes the frame shifts throughout the process, thus exposing the output and input for interaction

Q4: What can a Process Know about its Frame?

- General process
  - One that can use all the operations of affine geometry on points and vectors, as well as dot and cross product.
- A general process cannot observe its true position.
  - It can make relative comparisons by point subtraction.
  - Point *addition*, though, violates translation invariance.
  - Affine geometry is axiomatized *without* point addition.
- A general process cannot observe its orientation.
  - It can measure angles by dot product.
  - It can measure volumes and handedness by cross product.
  - It can tell if it has been scaled by using dot product: $\|\uparrow_x\|=1$?
  - It can tell if it has been reflected by using cross product.

A general process is invariant under all rigid-body motions.

(Rotations and translations, not reflections.)

Q4: What can a Process Know about its Frame?

- Euclidean process
  - A general process that does not use cross product.
- A Euclidean process cannot observe its handedness
  - I.e. it cannot tell if it has been reflected.

A Euclidean process is invariant under all isometries.

(Distance-preserving maps, including reflections.)

- Affine process
  - A general process that does not use dot or cross product.
- An affine process cannot observe its size and angles
  - I.e. it cannot tell if it has been stretched.

An affine process is invariant under all affine maps.

(Linear maps plus translations.)

Processes

Syntax of Processes

- A pretty standard π-calculus syntax with data terms Δ (details later).
- Data can have one of five sorts σ = {scalar, point, vector, map, channel}

$\Delta ::= x_c \,\vdots\, \ldots \,\vdots\, M[\Delta]$	Data terms
$\pi ::= ?_\sigma x(x') \,\vdots\, !_\sigma x(\Delta) \,\vdots\, \Delta =_\sigma \Delta'$	Action terms
$P ::= 0 \,\vdots\, \pi.P \,\vdots\, P+P' \,\vdots\, P \mid P' \,\vdots\, (\nu x)P \,\vdots\, P^* \,\vdots\, M[P]$	Process terms

- The new addition is process frame shift M[P], where M is (a data term denoting) an affine map. And a similar data frame shift M[Δ].
- Process reduction is relative to a global frame; M[P] means running process P in a global frame that has been shifted by (composed with) M.

Reduction

- Process reduction is indexed by a global frame $\mathcal{A}$, which is an affine map: $\quad P \ {}_{\mathcal{A}}\!\!\rightarrow Q$
- Uses a data evaluation relation from data terms Δ to data values ε, similarly indexed by a global frame: $\quad \Delta \ {}_{\mathcal{A}}\!\!\rightarrowtail \varepsilon$

(Red Comm)	$\Delta \ {}_{\mathcal{A}}\!\!\rightarrowtail \varepsilon \ \Rightarrow \ !_{\sigma}x(\Delta).P + P' \mid ?_{\sigma}x(y).Q + Q' \ {}_{\mathcal{A}}\!\!\rightarrow \ P \mid Q\{y \backslash \varepsilon\}$
(Red Cmp)	$\Delta \ {}_{\mathcal{A}}\!\!\curvearrowright\!\curvearrowleft \Delta' \ \Rightarrow \ \Delta =_{\sigma} \Delta'.P \ {}_{\mathcal{A}}\!\!\rightarrow P$
(Red Par)	$P \ {}_{\mathcal{A}}\!\!\rightarrow Q \ \Rightarrow \ P \mid R \ {}_{\mathcal{A}}\!\!\rightarrow Q \mid R$
(Red Res)	$P \ {}_{\mathcal{A}}\!\!\rightarrow Q \ \Rightarrow \ (\nu x)P \ {}_{\mathcal{A}}\!\!\rightarrow (\nu x)Q$
(Red $\equiv$)	$P' \equiv P, \ P \ {}_{\mathcal{A}}\!\!\rightarrow Q, \ Q \equiv Q' \ \Rightarrow \ P' \ {}_{\mathcal{A}}\!\!\rightarrow Q'$

eliminate impure terms

introduce impure terms

- Otherwise, a completely standard π-calculus reduction relation
  - Communication is by-value (using $\Delta \ {}_{\mathcal{A}}\!\!\rightarrowtail \varepsilon$ for evaluation).
  - *Data comparison* $\Delta =_{\sigma} \Delta'$. generalizes 'channel matching' to all data sorts.
  - The global frame $\mathcal{A}$ is just handed down to the evaluation relation.
  - There is no new rule here about frame shift: it is handled via (Red $\equiv$).

Structural Congruence

- The now standard 'chemical' formulation of π-calculus

$$P \equiv Q$$

  - The *structural congruence* relation has the role of shuffling the syntax to bring communication actions 'close together' so that the communication rule (Red Comm) can operate on them.
  - We extend this idea to bringing communication actions together even when they are initially separated by frame shifts.

- A *major* technical simplification
  - Avoids explicit communication rules between processes in different frames.
  - This way, the fundamental (Red Comm) rule remain unchanged.
  - The standard (Red $\equiv$) rule, connecting reduction with structural congruence, also remains unchanged
  - N.B.: structural congruence is *not* indexed by a global frame.

Structural Congruence

(≡ Refl)	$P \equiv P$	(≡ Sum Comm)	$P+Q \equiv Q+P$
(≡ Symm)	$P \equiv Q \Rightarrow Q \equiv P$	(≡ Sum Assoc)	$(P+Q)+R \equiv P+(Q+R)$
(≡ Tran)	$P \equiv Q,\ Q \equiv R \Rightarrow P \equiv R$	(≡ Sum Zero)	$P+0 \equiv P$
(≡ Act)	$P \equiv P' \Rightarrow \pi.P \equiv \pi.P'$	(≡ Par Comm)	$P \mid Q \equiv Q \mid P$
(≡ Sum)	$P \equiv P',\ Q \equiv Q' \Rightarrow P+Q \equiv P'+Q'$	(≡ Par Assoc)	$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$
(≡ Par)	$P \equiv P',\ Q \equiv Q' \Rightarrow P \mid Q \equiv P' \mid Q'$	(≡ Par Zero)	$P \mid 0 \equiv P$
(≡ Res)	$P \equiv P' \Rightarrow (\nu x)P \equiv (\nu x)P'$	(≡ Res Zero)	$(\nu x)0 \equiv 0$
(≡ Repl)	$P \equiv P' \Rightarrow P^* \equiv P'^*$	(≡ Res Sum)	$(\nu x)(P+Q) \equiv P+(\nu x)Q \quad (x \notin fv_c(P))$
(≡ Map)	$P \equiv P' \Rightarrow M[P] \equiv M[P']$	(≡ Res Par)	$(\nu x)(P \mid Q) \equiv P \mid (\nu x)Q \quad (x \notin fv_c(P))$
(≡ Map Cmp)	$M[\Delta =_\sigma \Delta'.P] \equiv M[\Delta] =_\sigma M[\Delta'].M[P]$	(≡ Res Res)	$(\nu x)(\nu y)P \equiv (\nu y)(\nu x)P$
(≡ Map Out)	$M[!_\sigma x(\Delta).P] \equiv !_\sigma x(M[\Delta]).M[P]$	(≡ Repl Zero)	$0^* \equiv 0$
(≡ Map In)	$M[?_\sigma x(y).P] \equiv ?_\sigma x(y).M[P] \quad (y \notin fv_\sigma(M))$	(≡ Repl Par)	$(P \mid Q)^* \equiv P^* \mid Q^*$
(≡ Map Sum)	$M[P+Q] \equiv M[P]+M[Q]$	(≡ Repl Copy)	$P^* \equiv P \mid P^*$
(≡ Map Par)	$M[P \mid Q] \equiv M[P] \mid M[Q]$	(≡ Repl Repl)	$P^{**} \equiv P^*$
(≡ Map Res)	$M[(\nu x)P] \equiv (\nu x)M[P]$		
(≡ Map Comp)	$M[N[P]] \equiv (M \circ M[N])[P]$		

The Map Rules

- Used to distribute frame shift over the syntax of processes
  - Pushing frame shift 'inside' until it is applied only to data terms.
  - Several rules (Cmp,Sum,Par,Res) simply distribute a frame shift to subprocesses, and to data.
  - The (Comp) rule distributes a frame shift over another frame shift. Note that N must be evaluated in its original frame (inside M), so it is $M \circ M[N]$ instead of just $M \circ N$.

($\equiv$ Map Cmp)	$M[\Delta =_\sigma \Delta'.P] \equiv M[\Delta] =_\sigma M[\Delta'].M[P]$
($\equiv$ Map Out)	$M[!_\sigma x(\Delta).P] \equiv !_\sigma x(M[\Delta]).M[P]$
($\equiv$ Map In)	$M[?_\sigma x(y).P] \equiv ?_\sigma x(y).M[P]$ $(y \notin fv_\sigma(M))$
($\equiv$ Map Sum)	$M[P+Q] \equiv M[P]+M[Q]$
($\equiv$ Map Par)	$M[P \mid Q] \equiv M[P] \mid M[Q]$
($\equiv$ Map Res)	$M[(\nu x)P] \equiv (\nu x)M[P]$
($\equiv$ Map Comp)	$M[N[P]] \equiv (M \circ M[N])[P]$

  - The communication rules (Out,In) are key.

The Out,In Rules

- Pushing frame shift through an output action:
  - Performing an output of Δ in the global frame shifted by M, is the same as performing an output of Δ shifted by M in the global frame.

$$(\equiv \text{Map Out}) \qquad M[!_\sigma x(\Delta).P] \equiv !_\sigma x(M[\Delta]).M[P]$$

  - (In both cases continuing by reducing P in the global frame shifted by M.)

- Pushing frame shift through an input action:
  - Performing an input of y in the global frame shifted by M, is the same as performing an input of y in the global frame. (Why?)

$$(\equiv \text{Map In}) \qquad M[?_\sigma x(y).P] \equiv ?_\sigma x(y).M[P] \qquad (y \notin fv_\sigma(M))$$

  - Because communication is by-value: the *value* bound to y is evaluated in a frame, and that *value* then does not change no matter where it is received.
  - Note that this is the only rule where we cannot in general push the frame shift *out* (right to left), because of the side condition.

Geometric Data

Data Terms Δ (and Data Values ε)

$\Delta ::= x_c \;\vdots\; a \;\vdots\; p \;\vdots\; v \;\vdots\; M \;\vdots\; M[\Delta]$		Data
$a ::= r \;\vdots\; f(a_i) \;\vdots\; v \bullet v' \;\vdots\; x_a \;\vdots\; \boxed{a}$	$(i \in 1..arity(f))$	Scalars
$p ::= \dotplus \;\vdots\; v{+}p \;\vdots\; x_p \;\vdots\; \boxed{p}$		Points
$v ::= \uparrow_x \;\vdots\; \uparrow_y \;\vdots\; \uparrow_z \;\vdots\; p{-}p' \;\vdots\; a \cdot v \;\vdots\; v{+}v' \;\vdots\; v \times v' \;\vdots\; x_v \;\vdots\; \boxed{v}$		Vectors
$M ::= \langle a_{ij}, a_k \rangle \;\vdots\; M \circ M' \;\vdots\; M^{-1} \;\vdots\; x_m \;\vdots\; \boxed{\mathcal{A}}$	$(i,j,k \in 1..3)$	Maps
$\varepsilon ::= x_c \;\vdots\; a \;\vdots\; p \;\vdots\; v \;\vdots\; \mathcal{A}$		Values

- $\boxed{}$: impure terms (computed values inside terms)

Data Computation in a Frame

(Scalar Real)	$\mathrm{r}\ {}_{\mathcal{A}}{\mapsto}\ b$	if literal r represents $b \in Val_a$
(Scalar Arith)	$\mathrm{a_i}\ {}_{\mathcal{A}}{\mapsto}\ b_i \Rightarrow \mathrm{f(a_i)}\ {}_{\mathcal{A}}{\mapsto}\ f(b_i)$ $\mathrm{i} \in 1..arity(f)$	if $b_i \in Val_a$, $f(b_i)$ defined
(Scalar Dot)	$\mathrm{v}\ {}_{\mathcal{A}}{\mapsto}\ w,\ \mathrm{v'}\ {}_{\mathcal{A}}{\mapsto}\ w' \Rightarrow \mathrm{v} \bullet \mathrm{v'}\ {}_{\mathcal{A}}{\mapsto}\ w \bullet w'$	if $w, w' \in Val_v$
(Point Origin)	⊹ $\ {}_{\mathcal{A}}{\mapsto}\ \mathcal{A}(\langle 0,0,0 \rangle)$	
(Point Move)	$\mathrm{v}\ {}_{\mathcal{A}}{\mapsto}\ w,\ \mathrm{p}\ {}_{\mathcal{A}}{\mapsto}\ q \Rightarrow \mathrm{v} + \mathrm{p}\ {}_{\mathcal{A}}{\mapsto}\ w + q$	if $w \in Val_v$, $q \in Val_p$
(Vect Unit)	↥$_{\mathrm{x}}\ {}_{\mathcal{A}}{\mapsto}\ \mathcal{A}(\uparrow\langle 1,0,0 \rangle)$, ↥$_{\mathrm{y}}\ {}_{\mathcal{A}}{\mapsto}\ \mathcal{A}(\uparrow\langle 0,1,0 \rangle)$, ↥$_{\mathrm{z}}\ {}_{\mathcal{A}}{\mapsto}\ \mathcal{A}(\uparrow\langle 0,0,1 \rangle)$	
(Vect Sub)	$\mathrm{p}\ {}_{\mathcal{A}}{\mapsto}\ q,\ \mathrm{p'}\ {}_{\mathcal{A}}{\mapsto}\ q' \Rightarrow \mathrm{p} - \mathrm{p'}\ {}_{\mathcal{A}}{\mapsto}\ q \dot{-} q'$	if $q, q' \in Val_p$
(Vect Scale)	$\mathrm{a}\ {}_{\mathcal{A}}{\mapsto}\ b,\ \mathrm{v}\ {}_{\mathcal{A}}{\mapsto}\ w \Rightarrow \mathrm{a \cdot v}\ {}_{\mathcal{A}}{\mapsto}\ b \cdot w$	if $b \in Val_a$, $w \in Val_v$
(Vect Add)	$\mathrm{v}\ {}_{\mathcal{A}}{\mapsto}\ w,\ \mathrm{v'}\ {}_{\mathcal{A}}{\mapsto}\ w' \Rightarrow \mathrm{v} + \mathrm{v'}\ {}_{\mathcal{A}}{\mapsto}\ w + w'$	if $w, w' \in Val_v$
(Vect Cross)	$\mathrm{v}\ {}_{\mathcal{A}}{\mapsto}\ w,\ \mathrm{v'}\ {}_{\mathcal{A}}{\mapsto}\ w' \Rightarrow \mathrm{v} \times \mathrm{v'}\ {}_{\mathcal{A}}{\mapsto}\ w \times w'$	if $w, w' \in Val_v$
(Map Given)	$\mathrm{a_{ij}}\ {}_{\mathcal{A}}{\mapsto}\ b_{ij},\ \mathrm{a_k}\ {}_{\mathcal{A}}{\mapsto}\ b_k \Rightarrow \langle \mathrm{a_{ij}}, \mathrm{a_k} \rangle\ {}_{\mathcal{A}}{\mapsto}\ \langle b_{ij}, b_k \rangle$	if $b_{ij}, b_k \in Val_a$, $det(b_{ij}) \neq 0$
(Map Comp)	$\mathrm{M}\ {}_{\mathcal{A}}{\mapsto}\ \mathcal{B},\ \mathrm{M'}\ {}_{\mathcal{A}}{\mapsto}\ \mathcal{B}' \Rightarrow \mathrm{M} \circ \mathrm{M'}\ {}_{\mathcal{A}}{\mapsto}\ \mathcal{B} \circ \mathcal{B}'$	if $\mathcal{B}, \mathcal{B}' \in Val_m$
(Map Inv)	$\mathrm{M}\ {}_{\mathcal{A}}{\mapsto}\ \mathcal{B} \Rightarrow \mathrm{M}^{-1}\ {}_{\mathcal{A}}{\mapsto}\ \mathcal{B}^{-1}$	if $\mathcal{B} \in Val_m$
(Frame Shift)	$\mathrm{M}\ {}_{\mathcal{A}}{\mapsto}\ \mathcal{B},\ \Delta\ {}_{\mathcal{A} \circ \mathcal{B}}{\mapsto}\ \varepsilon \Rightarrow \mathrm{M}[\Delta]\ {}_{\mathcal{A}}{\mapsto}\ \varepsilon$	if $\mathcal{B} \in Val_m$
(Value)	$\varepsilon\ {}_{\mathcal{A}}{\mapsto}\ \varepsilon$	if $\varepsilon \in Val$

Observation

Relativity

- Relating process behavior and geometry: what can we prove?

- How does behavior change when changing frame?
  - Galilean relativity (about classical 4D-spacetime):

    The laws of physics are the same in all inertial frames.

    (inertial frame = rigid body motion at constant velocity)

    The laws of physics are algebraic laws (equations) that remain valid (observationally invariant) under some class of transformations.

  - 3π relativity (about 3D space):

    The laws of process algebra are the same in all rigid-body frames.

- What are the "laws of process algebra"?
  - All equations that are valid under observational equivalence!
  - An equation is a pair of *pure terms* P,Q, written $P = Q$.
  - A law in a frame $\mathcal{A}$ is a valid equation $P\ {}_{\mathcal{A}}{\approx}\ Q$ for all observations ($\approx$).

Relativity Theorem

- G-Equations
  - An equation $P \;_{\mathcal{A}}\!\approx Q$ is a *G*-equation (a syntactic condition)
    if *P*,*Q* do *not* use operators that (intuitively) can detect certain changes:
    - $GA(3)$- (Affine) Equations: P,Q contain neither $\bullet$ nor $\times$.
      If P,Q cannot measure, they are not affected by any $C \in GA(3)$:
      not affected by any affine map.
    - $E(3)$- (Euclidean) Equations: P,Q contain $\bullet$.
      If P,Q can measure distance, they are still not affected by any $C \in E(3)$:
      not affected by any isometry.
    - $SE(3)$- (General) Equations: P,Q contain $\times$ (and possibly $\bullet$).
      If P,Q can detect reflections, they are still not affected by any $C \in SE(3)$:
      not affected by any rigid-body motion.
  - An equation is *G*-invariant (a semantic property) if its validity does not change under any *G* shift (any $C \in G$ composed with $\mathcal{A}$).
  - An equation is invariant across *G* (a semantic property)
    if it is valid or not valid for all *G* frames.

- Theorem: Relativity
  - *G*-equations are *G*-invariant, and invariant across *G*.

Relativity Corollaries

- For the three main transformation groups *G* of interest:
  - *GA*(3)-equations (those not using • or $\times$) are *GA*(3)-invariant: that is, ***affine equations are invariant under all transformations***
  - *E*(3)-equations (those not using $\times$) are *E*(3)-invariant: that is, ***Euclidean equations are invariant under isometries***
  - *SE*(3)-equations (*all* equations, since *SE*(3) imposes no syntactic restrictions) are *SE*(3)-invariant: that is,
    all equations are invariant under rigid-body transformations.
    (Galilean Relativity)

- Further, *'G-equations are invariant across G'* can be read as *'G laws are the same in all G frames'*:
  - ***affine laws are the same in all frames***
  - ***Euclidean laws are same in all Euclidean frames***
  - ***all laws are the same in all rigid body frames.*** (Galilean Relativity)

[Technical] Observational Equivalence

- The notions of *external observer*, *observable behavior*, and *observational equivalence* are well characterized by *barbed congruence*.

- We adapt it to obtain observational equivalence in a frame: $P \ {}_{\mathcal{A}}\!\approx Q$
  - *P* and *Q* when located in frame $\mathcal{A}$ have the same observable actions.

Definition (Barbed Congruence)

- ***Observation Context***: An observation context Γ is given by:

  $\Gamma ::= [\,] \ \vdots \ P|\Gamma \ \vdots \ \Gamma|P \ \vdots \ (\nu x)\Gamma$ where $[\,]$ only occurs once in Γ.

  The process, $\Gamma[Q]$ is the process obtained by replacing the unique $[\,]$ in Γ with Q.
- ***Strong Barb on x***: $P{\downarrow_x} \ \stackrel{\text{def}}{=} \ P \equiv (\nu y_1)..(\nu y_n)\ (!x(\Delta).P' \mid P'')$ with $x \neq y_1..y_n$.
- ***${}_{\mathcal{A}}$Barb on x***: $P\ {}_{\mathcal{A}}{\Downarrow_x} \ \stackrel{\text{def}}{=} \ \exists P'.\ P \ {}_{\mathcal{A}}{\to^*}\ P' \wedge P'{\downarrow_x}$.
- ***${}_{\mathcal{A}}$Candidate Relation***: $\mathcal{R}$ is an ${}_{\mathcal{A}}$*candidate relation* iff for all $P\mathcal{R}Q$:

  (1) if $P{\downarrow_x}$ then $Q\ {}_{\mathcal{A}}{\Downarrow_x}$; conversely if $Q{\downarrow_x}$ then $P\ {}_{\mathcal{A}}{\Downarrow_x}$;

  (2) if $P \ {}_{\mathcal{A}}{\to}\ P'$ then there is Q' such that $Q \ {}_{\mathcal{A}}{\to^*}\ Q'$ and $P'\mathcal{R}Q'$,

  if $Q \ {}_{\mathcal{A}}{\to}\ Q'$ then there is P' such that $P \ {}_{\mathcal{A}}{\to^*}\ P'$ and $P'\mathcal{R}Q'$;

  (3) for all observation contexts Γ, we have $\Gamma[P]\ \mathcal{R}\ \Gamma[Q]$.
- ***${}_{\mathcal{A}}$Barbed Congruence***: ${}_{\mathcal{A}}\!\approx$ is the union of all ${}_{\mathcal{A}}$candidate relations, which is itself an ${}_{\mathcal{A}}$candidate relation.

[Technical] Main Theorem to prove Relativity

- **Theorem: Global Frame Shift for Barbed Congruence**

  If P,Q are oblivous to C (a syntactic condition) then:

  $$\mathrm{P} \ {}_{\mathcal{A}}\!\approx \mathrm{Q} \Rightarrow C(\mathrm{P}) \ {}_{C\circ\mathcal{A}}\!\approx C(\mathrm{Q})$$

  - I.e., if two processes P,Q are observationally equivalent in global frame $\mathcal{A}$ then they are observationally equivalent in global frame $C\circ\mathcal{A}$
  - Except that, if P/Q are impure, then C needs to be applied to the values inside by C(P/Q).

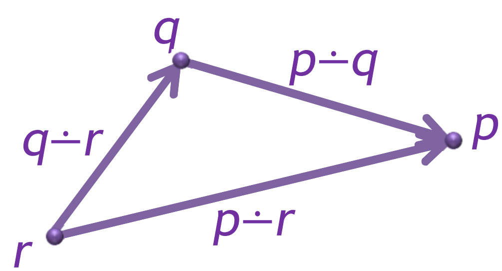
- The syntactic condition:
  - If P,Q contains neither • nor ×, then C can be any affine map.
  - If P,Q contain • then $C \in E(3)$
    (if P,Q can measure then C had better be an isometry).
  - If P,Q contains × then $C \in SE(3)$
    (if P,Q can detect reflections then C had better be a rigid-body motion).

Relativity Example 1

- For any three pure points p,q,r and pure process P, the affine equation:

$$((p \dot{-} q) + (q \dot{-} r) = (p \dot{-} r).\ P) \quad = \quad P$$

is a law in the id frame, and so by relativity is a law in all frames. In fact it is the head-to-tail axiom of affine space.



Relativity Example 2

- For any pure process P, the Euclidean equation:

  $$(\text{↥}_x \bullet \text{↥}_x = 1.\ P) \;=\; P$$

  is a law in the id frame, and hence is a law in all Euclidean frames.

- Note also that this equation can be read from left to right as saying that $\text{↥}_x \bullet \text{↥}_x = 1.P$ computes to P.
- Hence this computation gives the same result in all Euclidean frames.

Relativity Example 3

- For any pure point p and pure process P, the equation

  $$(p{=}\text{⊹}.\ P) \ = \ P$$

  is invariant under all translations (it remains valid iff it is valid).
- That's because *all* equations are invariant under *all* rigid-body maps, like translations.
- Hence, the comparison $p{=}$⊹ gives the same result under all translations, and cannot be used to test the true value of the origin no matter how p is expressed, as long as it is a pure term.

- Relativity implies that it is impossible to discover the value of the origin.

Relativity Summary

- All process equations are invariant under rigid body transformations (rotations and translations, not reflections), implying that no pure process can observe the location of the origin, nor the orientation of the basis vectors in the global frame.

- Processes that do not perform absolute measurements (via $\bullet$ and $\times$) are invariant under all affine transformations, meaning that they are also unable to observe the size of the basis vectors and the angles between them.

- Processes that use $\bullet$ but not $\times$ are invariant under all the isometries, meaning that they cannot observe whether they have been reflected.

Conclusions

Did It Blend?

- Q1: How should the position of a process be represented?
  - By the affine basis relative to a global frame.

- Q2: How should a process move from one position to another?
  - By frame shift.

- Q3: How should processes at different positions interact?
  - By *standard* π-calculus communication supported by frame-shift structural congruence rules. Communication not limited by distance.

- Q4: What theorems can we prove that blend properties of geometry and of process algebra, to show we have reached a smooth integration?
  - Relativity (invariances of barbed congruence under affine transformations).

Related Work

- Affine geometry is widely used in computer graphics; probably the most accessible reference for computer scientists is Gallier's book [5].
- It has been used in conjunction with L-Systems in very successful models of plant development [11]. However, L-systems are contextual term rewriting systems and, unlike 3π, do not have an intrinsic notion of interaction, which is important since biological development is regulated by sophisticated intra-cellular interactions.
- There is a solid body of work in functional computer graphics, but not so much in concurrent computer graphics.
- SpacePi [8] is an extension of π-calculus to model spatial dynamics in biological systems. Similar general aims to our work, but technically rather different. We do not restrict communication to a radius because that can be achieved by comparing data values, because some physical forces have infinite radius, and because geometric constraints on interaction are not necessarily of such a simple form (e.g., interaction restricted to adjacent cells of odd shapes).