PROCESSES IN SPACE

LUCA CARDELLI   PHILIPPA GARDNER

Microsoft Research Cambridge   *Imperial College London*

ABSTRACT. We introduce a geometric process algebra based on affine geometry, with the aim of describing the concurrent evolution of geometric structures in 3D space. We prove a relativity theorem stating that algebraic equations are invariant under rigid body transformations.

KEYWORDS: Process Algebra, Affine Geometry

1. INTRODUCTION

Context and Aims. Process Algebra provides a fundamental study of interacting computing systems. From initial work based on simple handshake interaction, the field has expanded to model systems with dynamic interaction patterns [10], static spatial distribution [6], and nested, topologically-changing distribution [2]. In applications to biological systems, interaction and distribution are sufficient to characterize well-mixed chemical systems subdivided into nested compartments, as are commonly found in cellular biology. There are many situations, however, both in computer systems (in robotics and sensor networks) and in biological systems (in growth and development) where a geometrical distance is also necessary beyond purely topological organization.

One of the key motivating examples of the π-calculus [10], for example, is the handover protocol of mobile phones: a mobile phone is connected to a fixed tower, and receives a new frequency to connect to a different tower. In actuality, the handover is based on the relative distance (signal power) between the mobile device and the fixed towers, and hence the protocol depends on geometry. The motivating examples of the Dpi calculus [6] and the Ambient Calculus [2] involve movement through space, but lack a notion of distance. More challenging examples can be found in developmental biology, which deals with the dynamic spatial arrangements of cells, and with forces and interactions between them. Many computational approaches have been developed for modeling geometric systems, including Cellular Automata, extended L-systems [12], and graph models, but few cover complex geometry, dynamic interaction, and dynamic organization together. In particular, the richness of interaction present in Process Algebra, is not found in other approaches.

We therefore start from Process Algebra and we extend it towards geometrical modeling, taking inspiration from a well-developed body of formal work in developmental biology. Concretely, we develop a calculus of processes located in 3-dimensional geometric space: 3π. While it may seem in principle logical to 'add a position (and possibly a velocity) to each process', naive attempts result in awkward formal systems with too many features: coordinates, position, velocity, identity, extent, force, collision detection, communication range, and so on. In addition, application areas such as developmental biology are challenging in that the coordinate space is not fixed: it effectively expands, moves, and warps as an organism is developing, making approaches based on fixed coordinate systems or grids awkward. Our aim is thus to begin incorporating flexible and general geometric capabilities in Process Algebra, and among those we must certainly count the geometric transformations of space.

Other important concepts are present in this paper in embryonic form only. Time flow is needed in addition to position to define speed, acceleration and force. Although there is causality in 3π, there is no time, and the forces we can express are therefore not quantitative. This can be fixed by adapting any standard addition of time flow from the π-calculus to 3π, e.g. synchronous or stochastic. With this approach, the time dimension remains with the evolution of processes, and not with a geometric notion of space-time. Geometric objects, and the spatial extent of an object, are not represented: each process is a point, and its spatial extent can be encoded only by convention through the patterns of interactions with other processes. Spatial extent is an important consideration, but it would entail potentially unbounded complexity, which again is likely to be application specific. Membrane-based abstractions have been investigated in Process Algebra as extensions of π-calculus and could be usefully integrated with geometry, to express notions such as volume-dependent splitting of compartments. The most compelling applications will come from situations where communication, transformations, and forces/extents are connected, such as in developmental biology.

Contributions. In this paper, we introduce a geometric process algebra, called 3π, that combines the interaction primitives of the π-calculus with geometric transformations. In particular, we introduce a single new geometric construct, *frame shift*, which applies a 3-dimensional affine transformation to an evolving process. This calculus is sufficient to express many dynamic geometric behaviors, thanks to the combined power of Affine Geometry and Process Algebra. It remains a relatively simple π-calculus, technically formulated in a familiar way, with a large but standard and fairly orthogonal geometric subsystem. From a Process Algebra point of view, we add powerful geometric data structures and transformations; standard notions of process equivalence give rise to geometric invariants. From an Affine Geometry point of view, we add a notion of interacting agents performing geometric transformations.

Introducing 3π. During biological development, tissues expand, split and twist, and there is no fixed coordinate system that one can coherently apply. To capture examples such as these, it is natural to turn to *affine geometry*, which is the geometry of properties that are invariant under linear transformations and translations. Affine geometry already comprises a well-studied set of fundamental geometric primitives. Our challenge is to choose how the geometry relates to the processes living within it, by working out how to combine naturally these affine primitives with the primitives of the π-calculus. How should the position of a process be represented? How should a process move from one position to another? How should processes at different positions interact?

In 3π, processes have access to the standard affine basis consisting of the origin ✦ and the orthogonal unit vectors $\uparrow_x, \uparrow_y, \uparrow_z$; each process 'believes' this basis to be the true coordinate system. However, geometric data is interpreted relative to a global frame $\mathcal{A}$, which is an affine map. In particular, what a process believes to be the origin, ✦, is actually $\mathcal{A}(✦)$, and this is seen as the *actual location* of the process in the global frame. The true size and orientation of the basis vectors is also determined by $\mathcal{A}$, as they are interpreted as $\mathcal{A}(\uparrow_x), \mathcal{A}(\uparrow_y), \mathcal{A}(\uparrow_z)$. The global frame $\mathcal{A}$ is inaccessible to processes. Although processes can carry out *observations* that may reveal some information about $\mathcal{A}$, such as using dot product to compute the absolute size of $\uparrow_x$, they have no way to obtain other information, such as the value of $\mathcal{A}(✦)$.

Processes are positioned via a *frame shift* operation $M[P]$, which is our main addition to the π-calculus. If process $M[P]$ is in a global frame $\mathcal{A}$, and M evaluates to an affine map $\mathcal{B}$, then P is interpreted in the shifted global frame $\mathcal{A} \circ \mathcal{B}$. The process $M[P]|N[Q]$ therefore indicates that processes P and Q are in different frames, with P shifted by M and Q by N. Conversely, the process

$M[P]|M[Q] = M[P|Q]$ indicates that P and Q are in the same frame. Frame shift operations can also be nested, with the process $M[N_1[P]|N_2[Q]]$ indicating that P is in the frame shifted first by N_1 and then M, whereas Q is shifted by N_2 then M. Since M denotes a general affine map, frame shift is more than just a change of location: it generalizes the Dpi [6] notion of multiple discrete process locations to multiple process frames in continuous space. Processes interact by exchanging data messages consisting of channel names or geometric data; such interactions are not restricted by the distance between processes. Geometric data is evaluated in its current frame and transmitted 'by value' to the receiver. Consider an output $!x(\maltese)$ to input $?x(z)$ interaction on channel x:

$$P \quad \triangleq \quad M[!x(\maltese).Q]|N[?x(z).R] \; _{\mathcal{A}}\rightarrow M[Q]|N[R\{z\backslash\varepsilon\}]$$

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

$$P \quad = \quad !x(M[\maltese]).M[Q]|?x(z).N[R]$$

which distributes the frame shifts throughout the process, thus exposing the output and input for interaction. In addition to communication, processes can compare data values. If R is $z = \maltese.R'$ in our above example, then after interaction this process computes whether $\mathcal{A} \circ \mathcal{B}(\maltese) = \mathcal{A} \circ \mathcal{C}(\maltese)$, where $\mathcal{C}$ is the evaluation of N in $\mathcal{A}$, and evolves to R' only if the original output and input processes are at the same position.

Example 1. Distance between processes.

Let us assume that the global frame is just the identity map. Process P below is located at -1 on the x axis, because X applies a translation $T(-\uparrow_x)$ to it. Similarly, process Q is located at $+1$ on the x axis by Y. When P outputs its origin, the actual value being communicated is thus the point $\langle -1, 0, 0\rangle$: this is a *computed value* that is not subject to any further transformation. Process Q receives that value as x, and computes the size of the vector $x \dotminus \maltese$ obtained by a point difference. In the frame of Q that computation amounts to the size of the vector $\langle -1, 0, 0\rangle \dotminus \langle 1, 0, 0\rangle$, which is 2. Therefore, the comparison $\|x \dotminus \maltese\| = 2$ succeeds, and process R is activated, having verified that the distance between P and Q is 2.

$$
\begin{aligned}
X &= T(-\uparrow_x)[P] \quad \text{where} \quad P = !m(\maltese) \\
Y &= T(\uparrow_x)[Q] \quad\;\; \text{where} \quad Q = ?m(x).\|x \dotminus \maltese\| = 2.R
\end{aligned}
$$

Example 2. Force fields.

A force field is a process that repeatedly receives the location of an 'object' process (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. This 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. By nondeterministic interaction with multiple force fields, an object can be influenced by several of them. Here P^* is process replication ($P^* \equiv P \mid P^*$) for repeated interaction with the field channel f:

$$
\begin{aligned}
Force &= (?f(x,p).!x(M\{p\}))^* \qquad\quad f \text{ is the force field channel; } M\{p\} \text{is a map} \\
Object &= (\nu x)!f(x, \maltese).?x(Y).Y[Object]
\end{aligned}
$$

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

The ability to express force fields is important for modeling 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.

Example 3. Orthogonal bifurcation in lung development.

Lung development in mice is based on three splitting processes [9], which demonstrate a relatively simple example of a developmental process. We show how to represent the third process (orthogonal bifurcation, $Orth$), which is a proper 3D process of recursive tree growth, where bifurcations alternate between orthogonal planes.

$$
\begin{array}{rcl}
Orth & = & !c(\maltese).(M_{90}(\tfrac{\pi}{6})[Orth] \mid M_{90}(-\tfrac{\pi}{6})[Orth]) \\
M_{90}(\vartheta) & = & R(M(\vartheta)[\uparrow_y], \tfrac{\pi}{2}) \circ M(\vartheta) \\
M(\vartheta) & = & Sc(\tfrac{1}{2}) \circ R(\uparrow_z, \vartheta) \circ T(\uparrow_y)
\end{array}
$$

The output of the origin $\maltese$ to the c channel at each iteration provides a trace of the growing process that can be plotted. The transformation $M(\vartheta)$ applies a translation $T(\uparrow_y)$ by $\uparrow_y$, a rotation $R(\uparrow_z, \vartheta)$ by ϑ around $\uparrow_z$, and a uniform scaling $Sc(\tfrac{1}{2})$ by $\tfrac{1}{2}$. The transformation $M_{90}(\vartheta)$ first applies an $M(\vartheta)$ transformation in the XY plane, and then applies a further $90°$ rotation around the 'current' direction of growth, which is $M(\vartheta)[\uparrow_y]$, therefore rotating out of the XY plane for the next iteration. Opposite $30°$ rotations applied recursively to $Orth$ generate the branching structure; note that because of parallel composition ('$\mid$') the tree grows nondeterministically.

2. Processes

We introduce a process algebra, 3π, where 3-dimensional geometric data (points, vectors, and affine maps, as well as channel names) can be exchanged between processes, and where processes can be executed in different frames. This is a proper extension of π-calculus with by-value communication of geometric data Δ, data comparisons $\Delta = \Delta'.P$, and frame shifting $M[P]$. By-value communication over named channels is achieved via an evaluation relation $\Delta_{\mathcal{A}} \mapsto \varepsilon$, which evaluates a *data term* Δ to a *data value* ε relative to a global frame $\mathcal{A}$. The data comparison process $\Delta = \Delta'.P$ evaluates to P if Δ and Δ' evaluate to the same value. Frame shifting is the characteristic construct of 3π: the frame shift process $M[P]$ means running the process P in the global frame $\mathcal{A}$ shifted by the affine map obtained by evaluating M.

The syntax of processes depends on the syntax of data Δ, given in Section 3. For now, it is enough to know that each data term Δ has a *data sort* σ, where the channel variables $x_{\mathbf{c}} \in Var_{\mathbf{c}}$ have sort $\mathbf{c}$, and the sort of $M[\Delta]$ is the sort of Δ.

Definition 4. Syntax of Processes

$$
\begin{array}{rcll}
\Delta & ::= & x_{\mathbf{c}} \mid ... \mid M[\Delta] & \text{Data terms} \\
\pi & ::= & ?_\sigma x(x') \mid !_\sigma x(\Delta) \mid \Delta =_\sigma \Delta' & \text{Action terms} \\
P & ::= & 0 \mid \pi.P \mid P + P' \mid P|P' \mid (\nu x)P \mid P^* \mid M[P] & \text{Process terms}
\end{array}
$$

An action term π can be an *input* $?_\sigma x(x')$, an *output* $!_\sigma x(\Delta)$, or a *data comparison* $\Delta =_\sigma \Delta'$. The input and output actions are analogous to π-calculus actions, where the input receives a data value of sort σ along channel x which it binds to x', and the output sends the value of Δ with sort σ along x. Process interaction only occurs between inputs $?_\sigma$, and outputs $!_\sigma$ of the same sort σ. A comparison of two data terms of sort σ blocks the computation if the terms do not match when evaluated using $_\mathcal{A} \mapsto$. The syntax of actions is restricted by sorting constraints: the x in $?_\sigma x(x')$ and $!_\sigma x(\Delta)$ must have a channel sort $\mathbf{c}$; the x' in $?_\sigma x(x')$ must have sort σ; the Δ in $!_\sigma x(\Delta)$ must

have sort σ; and the Δ, Δ' in $\Delta =_\sigma \Delta'$ must have sort σ. We often omit sorting subscripts, and we assume that variables of distinct sorts are distinct.

Process terms look like standard π-calculus terms. We have the standard empty process 0, the action process $\pi.P$ for action π (when $\pi =?x(x')$, the x' binds any free x' in P), choice $P + P'$, parallel composition $P|P'$, channel restriction $(\nu x)P$ where x has sort $\mathbf{c}$ (the x binds any free x in P), and replication $P*$. In addition, we have the non-standard process frame shifting $M[P]$, which represents a shifted frame given by M. We shall see in Section 3 that channel variables do not occur in M; hence, in $(\nu x)M[P]$, there is no possibility that any variable in M is bound by x.

We now give a *reduction* relation on process terms, written $_\mathcal{A}{\to}$, which relates two processes relative to the global frame $\mathcal{A}$. Reduction depends on an evaluation relation $\Delta_\mathcal{A} \mapsto \varepsilon$ from data Δ to values ε in a global frame $\mathcal{A}$, discussed in Section 3. The reduction rules for process terms are simply the rules of a by-value π-calculus with data terms Δ. Data evaluation is used in the (*Red Comm*) and (*Red Cmp*) rules. Data comparison $\Delta =_\sigma \Delta'.P$ requires the data evaluation $\Delta_\mathcal{A} \curlyvee \Delta'$, meaning there is a data value ε such that $\Delta_\mathcal{A} \mapsto \varepsilon$ and $\Delta'_\mathcal{A} \mapsto \varepsilon$.

Definition 5. Reduction

(*Red Comm*)	$\Delta_\mathcal{A} \mapsto \varepsilon \;\Rightarrow\; !_\sigma x(\Delta).P + P' \mid ?_\sigma x(y).Q + Q'_\mathcal{A}{\to}P|Q\{y\backslash\varepsilon\}$
(*Red Cmp*)	$\Delta_\mathcal{A} \curlyvee \Delta' \;\Rightarrow\; \Delta =_\sigma \Delta'.P_\mathcal{A}{\to}P$
(*Red Par*)	$P_\mathcal{A}{\to}Q \;\Rightarrow\; P \mid R_\mathcal{A}{\to}Q \mid R$
(*Red Res*)	$P_\mathcal{A}{\to}Q \;\Rightarrow\; (\nu x)P_\mathcal{A}{\to}(\nu x)Q$
(*Red* $\equiv$)	$P' \equiv P, P_\mathcal{A}{\to}Q, Q \equiv Q' \;\Rightarrow\; P'_\mathcal{A}{\to}Q'$

There is nothing specific in these rules about the use of the global frame $\mathcal{A}$: this is simply handed off to the data evaluation relation. There is also no rule for process frame shifting, $M[P]$, which is handled next in the structural congruence relation.

In the standard 'chemical' formulation [1] of π-calculus, the *structural congruence* relation has the role of bringing actions 'close together' so that the communication rule (*Red Comm*) can operate on them. We extend this idea to bringing actions together even when they are initially separated by frame shifts, so that the standard (*Red Comm*) rule can still operate on them. Therefore, structural congruence, $\equiv$, consists of the normal π-calculus rules plus additional rules for frame shifting: the ($\equiv Map...$) rules. These map rules essentially enable us to erase frame shifts from the process syntax and to push them to the data. In this sense, process frame shift $M[P]$ is an illusion, or syntactic sugar, for a π-calculus with frame shift only on the data. However, frame shift is important for modularity because, without it, we would have to modify the process code to apply the frame to all the data items individually.

Definition 6. Structural Congruence (non-standard $\equiv Map$ rules)

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

Many other rules can be derived, e.g., for communication across frames shifts at different depths, and for data comparison inside a local frame. In summary, the application of the structural congruence rules allows us to 'flatten' the local frames so that the rules of reduction can be applied directly.

There still remains the issue of correctness, or plausibility, of the new structural congruence rules. This issue can be explored by analyzing the expected derived rules, as we briefly mentioned above, and by establishing general properties of the whole system, as done in Section 4.

We have not discussed recursion, which was used in the introductory examples. However, recursive definitions in π-calculus can be encoded, and this extends in 3π to recursive definitions under frame shift by the ability to communicate transformations.

3. Geometric Data

Geometric data consists of points, vectors and transformations, with the operations of affine geometry (see Appendix, and [5]). We are interested in three main groups of transformations over $\mathbb{R}^3$. The *General Affine Group* $GA(3)$ is the group of *affine maps* over $\mathbb{R}^3$, which include rotation, translation, reflection, and stretching of space, and are indicated by script letters $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$. Affine maps are presented as pairs $\langle A, p\rangle$ where A is 3×3 invertible matrix representing a linear transformation, and p is a point in $\mathbb{R}^3$. The *Euclidean Group* $E(3)$ is the subgroup of $GA(3)$ where $A^T = A^{-1}$: namely, it is the group of isometries of $\mathbb{R}^3$ consisting of rotations, translations and reflections. The *Special Euclidean Group* $SE(3)$ is the subgroup of $E(3)$ where $det(A) = 1$: namely, the *direct isometries* consisting of rotations and translations, but not reflections. Elements of $SE(3)$ are known as the *rigid body transformations*, preserving distances, angles, and handedness. An affine map $\mathcal{A}$ has a canonical associated affine frame, namely the frame $\mathcal{A}(\maltese),\mathcal{A}(\uparrow_x),\mathcal{A}(\uparrow_y),\mathcal{A}(\uparrow_z)$; we therefore refer to $\mathcal{A}$ itself as a frame.
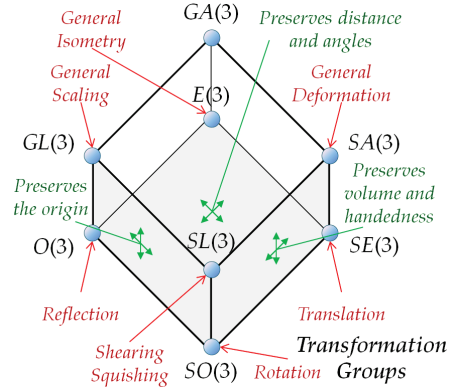
We next introduce data terms Δ and data values ε, and show how to compute data values relative to a global affine frame $\mathcal{A}$. Each data term and value has a sort $\sigma \in \Sigma = \{\mathbf{c}, \mathbf{a}, \mathbf{p}, \mathbf{v}, \mathbf{m}\}$, denoting channels, scalars, points, vectors, and maps respectively.

Definition 7. Data Values

The set of data values $\varepsilon \in Val$ is the union of the following five sets:
- $x_{\mathbf{c}} \in Val_{\mathbf{c}} \triangleq Var_{\mathbf{c}}$ are the channels;
- $b \in Val_{\mathbf{a}} \triangleq \mathbb{R}$ are the scalars;
- $q \in Val_{\mathbf{p}} \triangleq \mathbb{R}^3$ are the points, which we write $\langle x, y, z\rangle$;
- $w \in Val_{\mathbf{v}}$ are the vectors, a set isomorphic to $Val_{\mathbf{p}}$
  with a bijection $\uparrow: Val_{\mathbf{p}} \to Val_{\mathbf{v}}$ with inverse $\downarrow = \uparrow^{-1}$;
  elements of $Val_{\mathbf{v}}$ are written $\uparrow \langle x, y, z\rangle$;
- $\mathcal{A} \in Val_{\mathbf{m}} \triangleq \{\langle A, p\rangle \in \mathbb{R}^{3\times3} \times \mathbb{R}^3 \mid A^{-1} \text{ exists}\}$ are the affine maps.

Definition 8. Data Terms

Δ	$::=$	$x_{\mathbf{c}} \mid a \mid p \mid v \mid M \mid M[\Delta]$	Data
a	$::=$	$x_{\mathbf{a}} \mid r \mid f(a_i) \mid v \bullet v' \mid b$ $\qquad (i \in 1..arity(f))$	Scalars
p	$::=$	$x_{\mathbf{p}} \mid \maltese \mid v + p \mid q$	Points
v	$::=$	$x_{\mathbf{v}} \mid \uparrow_x \mid \uparrow_y \mid \uparrow_z \mid p - p' \mid a \cdot v \mid v + v' \mid v \times v' \mid w$	Vectors
M	$::=$	$x_{\mathbf{m}} \mid \langle a_{ij}, a_k\rangle \mid M \circ M' \mid M^{-1} \mid \mathcal{A}$ $\qquad (i,j,k \in 1..3)$	Maps

Data terms consist of *pure data terms*, which form the 'user syntax', additional *data values* $b,q,w,\mathcal{A}$, which are inserted during by-value substitutions resulting from process interaction, and variables $x_\sigma \in Var_\sigma$ of each sort $\sigma \in \Sigma$. Channels are regarded both as pure terms and values. Each term Δ has the appropriate sort σ; the sort of a data frame shift $M[\Delta]$ is the sort of Δ. The substitution $\Delta\{x\backslash\varepsilon\}$ distributes over the structure of Δ, with base cases $x\{x\backslash\varepsilon\} = \varepsilon$, $y\{x\backslash\varepsilon\} = y$ for $y \neq x$, and $\varepsilon'\{x\backslash\varepsilon\} = \varepsilon'$.

The scalar terms include the real number literals r, the dot product of vectors $v\bullet v'$, giving the ability to measure distances and angles, and basic functions $f(a_i)$, $i \in 1..arity(f)$, for real arithmetic and trigonometry. The point terms include the origin ($\maltese$) and the addition of a vector to a point. The vector terms include the unit vectors ($\uparrow_x$, $\uparrow_y$, $\uparrow_z$), point subtraction, the vector space operations ($\cdot$,+), and cross product $v \times v'$, which gives the ability to generate out-of-plane vectors, to measure areas and volumes, and to detect handedness.

The map terms include the base map terms $\langle a_{ij}, a_k \rangle$, composition, and inverse. In the term $\langle a_{ij}, a_k \rangle$ for $i, j, k \in 1..3$ the first 9 elements represent a 3×3 square matrix, and the last 3 elements represent a translation vector. We require the 3×3 matrix to be invertible, which is verified by a run-time check of the determinant.

The data term $M[\Delta]$ describes a *data frame shift*. Note that $M[\Delta] = \Delta$ is not always true even on scalars; e.g., $M[v\bullet v']$ is not the same as $v\bullet v'$ when M does not preserve distances. Hence, $M[\Delta]$ does not mean apply M to the data value produced by Δ; it means *shift frame* and evaluate the term Δ in the frame obtained from M and composed with the global frame.

The evaluation relation $\Delta_{\mathcal{A}}\mapsto\varepsilon$, describes the *computation* of a closed data term Δ to value ε, relative to global frame $\mathcal{A}$. The relation is a partial function defined by induction on the structure of terms. Most cases simply follow the structure of terms; the key rules are the evaluation of the origin in a frame: $\maltese_{\mathcal{A}}\mapsto\mathcal{A}(\langle 0,0,0 \rangle)$ (and similarly for the unit vectors like $\uparrow_x$ $_{\mathcal{A}}\mapsto\mathcal{A}(\uparrow \langle 1,0,0 \rangle)$), and the evaluation of frame shift: the value of $M[\Delta]$ in frame $\mathcal{A}$ is uniquely determined as the value of Δ in frame $\mathcal{A} \circ \mathcal{B}$, provided that the value of M in frame $\mathcal{A}$ is $\mathcal{B}$.

4. Process Observation and Equivalence

We establish the invariance of process behavior under certain transformations of the global frame. We base our results on *barbed congruence*, which is one of the most general notions of process equivalence in process algebra [4, 7, 10] and gives rise to a definition of algebraic process equation. For 3π, we relativize equations to affine frames, and investigate how the validity of the equality changes when shifting frames.

Barbed congruence is defined using barbs and observation contexts. Barbs identify what can be observed by the process environment; in our case, barbs are outputs on channels. Observation contexts define the process environment: different strengths of observation can be characterized by different classes of contexts. We choose to observe processes only by interaction on channels and by restricting the interaction channels. Therefore, we do not allow observation contexts that have the flavor of manipulating a whole process, like injecting a process into the observer's code, or injecting a process into a frame.

Definition 9. Barbed Congruence
- An *observation context* Γ is given by: $\Gamma ::= [] \mid P|\Gamma \mid \Gamma|P \mid (\nu x)\Gamma$, 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*: $P\downarrow_x \triangleq P \equiv (\nu y_1)..(\nu y_n)(!x(\Delta).P'|P'')$ with $x \neq y_1..y_n$.
- *$\mathcal{A}$-Barb on x*: $P_{\mathcal{A}}\Downarrow_x \triangleq \exists P'.P_{\mathcal{A}}\rightarrow^*P' \wedge P'\downarrow_x$.

○ *$\mathcal{A}$-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 $P_{\mathcal{A}} \rightarrow P'$ then there is Q' such that $Q_{\mathcal{A}} \rightarrow^* Q'$ and $P'\mathcal{R}Q'$, if $Q_{\mathcal{A}} \rightarrow Q'$ then there is P' such that $P_{\mathcal{A}} \rightarrow^* P'$ 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.

In order to state our theorems, we need *compatibility* relations, $\mathcal{A} \propto \Delta$ and $\mathcal{A} \propto P$, constraining the frame $\mathcal{A}$ by a simple analysis of the vector operators used in data Δ and process P. A closed data term is *affine* if it does not contain $v \bullet v'$ and $v \times v'$ subterms, *Euclidean* if it does not contain $v \times v'$ subterms, and *rigid* otherwise.

Definition 10. Frame and Group Compatibility
  ○ For $\mathcal{A} \in GA(3)$ and closed data term Δ, write $\mathcal{A} \propto \Delta$ ($\mathcal{A}$ compatible with Δ) iff:
  ○ if Δ contains $\bullet$ then $\mathcal{A} \in E(3)$;
  ○ if Δ contains $\times$ then $\mathcal{A} \in SE(3)$;
  ○ otherwise, no restriction on $\mathcal{A}$.
  ○ For group $G \subseteq GA(3)$ and closed data term Δ, write $G \propto \Delta$ iff $\forall \mathcal{A} \in G$. $\mathcal{A} \propto \Delta$. Write $\mathcal{A} \propto P$ and $G \propto P$ if $\mathcal{A}$ and G are compatible with all the data terms in P.

Hence we have: $GA(3) \propto \Delta$ implies Δ is affine; $E(3) \propto \Delta$ implies Δ is Euclidean; $SE(3) \propto \Delta$ implies Δ is rigid (i.e., $SE(3) \propto \Delta$ always).

We are normally interested only in equations between processes without computed values; we now restrict our attention to such process terms, which we call *pure* terms.

Definition 11. Pure Terms
A data term Δ and process term P is pure if it does not contain a value subterm ε of sort $\sigma \in \{\mathbf{a}, \mathbf{p}, \mathbf{v}, \mathbf{m}\}$. We use $\Delta^{\triangledown}$ and $P^{\triangledown}$ to denote such pure terms.

The invariance of equations between pure terms under certain maps is described by a relativity theorem. The key property is that G-equations are G-invariant, meaning that for a group G, the validity or invalidity of equations that are syntactically compatible with G is not changed by G transformations.

Definition 12. Equations and Laws
An equation is a pair of pure process terms $P^{\triangledown}, Q^{\triangledown}$, written $P^{\triangledown} = Q^{\triangledown}$. It is:
  ○ A *G-equation*, for $G \subseteq GA(3)$ iff $G \propto P^{\triangledown}$ and $G \propto Q^{\triangledown}$;
  ○ A *law in $\mathcal{A}$*, for $\mathcal{A} \in GA(3)$ iff $P^{\triangledown}{}_{\mathcal{A}} \approx Q^{\triangledown}$;
  ○ A *law in G*, for $G \subseteq GA(3)$ iff, $\forall \mathcal{A} \in G$ it is a law in $\mathcal{A}$;
  ○ *$\mathcal{B}$-invariant*, for $\mathcal{B} \in GA(3)$ iff $\forall \mathcal{A} \in GA(3)$ it is a law in $\mathcal{A}$ iff it is a law in $\mathcal{B} \circ \mathcal{A}$;
  ○ *G-invariant*, for $G \subseteq GA(3)$ iff $\forall \mathcal{B} \in G$ it is $\mathcal{B}$-invariant;
  ○ *Invariant across G*, for $G \subseteq GA(3)$ iff $\forall \mathcal{A}, \mathcal{B} \in G$ it is a law in $\mathcal{B}$ if it is a law in $\mathcal{A}$.

Theorem 13. *Relativity*
G-equations are G-invariant, and hence invariant across G.

For the three main transformation groups of interest, our theorem has the following corollaries:
  ○ $GA(3)$-equations (those not using $\bullet$ or $\times$) are $GA(3)$-invariant: that is, *affine equations are invariant under all maps.*

- $E(3)$-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 maps.*

Further, 'G-equations are invariant across G' can be read as 'G laws are the same in all G frames', in the same sense that one says that 'the laws of physics are the same in all inertial frames'. Thus we obtain:

- 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.*

For example, the Euclidean equation $(\uparrow_x \bullet \uparrow_x = 1.P^\triangledown) = P^\triangledown$ is a law in the $\mathcal{I}$ (identity) frame, and hence is a law in all Euclidean frames. Moreover, this equation may be valid or not in some initial frame (possibly a non-Euclidean one like a scaling $S(2 \bullet \uparrow_y)$), but its validity does not change under any further Euclidean transformation. Note also that this equation can be read from left to right as saying that $\uparrow_x \bullet \uparrow_x = 1.P^\triangledown$ computes to $P^\triangledown$. Hence equational invariance implies also computational invariance (but this only for computations from pure terms to pure terms, where any value introduced by communication is subsequently eliminated by data comparison).

As a second example, for any three points $p^\triangledown, q^\triangledown, r^\triangledown$, the affine equation $((q^\triangledown - p^\triangledown) + (r^\triangledown - q^\triangledown) = (r^\triangledown - p^\triangledown).P^\triangledown) = P^\triangledown$ is a law in the $\mathcal{I}$ frame, and so is a law in all frames; in fact it is the head-to-tail axiom of affine space.

As a third example, for any point $p^\triangledown$, the equation $(p^\triangledown = \maltese.P^\triangledown) = P^\triangledown$ is invariant under all translations (because all equations are invariant under rigid-body maps); hence, the comparison $p^\triangledown = \maltese$ gives the same result under all translations, and cannot be used to test the true value of the origin no matter how $p^\triangledown$ is expressed, as long as it is a pure term.

5. Conclusions

We have introduced 3π, an extension of the π-calculus based on affine geometry, to describe the concurrent evolution of geometric structures in 3D space. We have proved a relativity theorem stating that all algebraic equations are invariant under all 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. Moreover, 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. Finally, processes that use $\bullet$ but not $\times$ are invariant under all the isometries, meaning that they cannot observe whether they have been reflected. Therefore, these results describe the extent to which a process can observe its geometric frame, and describe the behavior of a process in different geometric frames.

References

[1] G. Berry, G. Boudol. The Chemical Abstract Machine. Proc. POPL'89, 81-94.
[2] L. Cardelli, A.D. Gordon. Mobile Ambients. Theoretical Computer Science, Special Issue on Coordination, D. Le Métayer Editor. Vol 240/1, June 2000. pp 177-213.
[3] H.S.M. Coxeter, Introduction to geometry, Wiley, 1961.
[4] C Fournet, G Gonthier. A Hierarchy of Equivalences for Asynchronous Calculi. Proc. 25th ICALP. LNCS 1443, 844-855. Springer 1998.

[5] J. Gallier. Geometric Methods and Applications for Computer Science and Engineering. Springer, 2001.

[6] M. Hennessy. A Distributed Pi-Calculus. Cambridge University Press, 2007.

[7] K. Honda, N. Yoshida. On Reduction-Based Process Semantics. Theoretical Computer Science, 152(2), pp. 437-486, 1995.

[8] M. John, R. Ewald, A.M. Uhrmacher. A Spatial Extension to the π Calculus. Electronic Notes in Theoretical Computer Science, 194(3) 133-148, Elsevier, 2008.

[9] R.J. Metzger, O.D. Klein, G.R. Martin, M.A. Krasnow. The branching programme of mouse lung development. Nature 453(5), June 2008.

[10] R. Milner. Communicating and Mobile Systems: The pi-Calculus. CUP, 1999.

[11] R. Milner, D. Sangiorgi. Barbed Bisimulation. In Proc. 19-the International Colloquium on Automata, Languages and Programming (ICALP '92), LNCS 623, Springer, 1992.

[12] P. Prusinkiewicz, A. Lindenmayer. The Algorithmic Beauty of Plants. Springer, 1991.

6. Appendix A: Geometry

6.1. **Vector Spaces and Automorphism Groups.** A *vector space* over a field F is a set V with operations $+ \in V \times V \to V$ (vector addition) and $\cdot \in F \times V \to V$ (scalar multiplication), such that $(V, +)$ is an abelian group, with identity the zero vector ϕ and inverse $-v$, and moreover: $a \cdot (v + w) = a \cdot v + a \cdot w$, $(a + b) \cdot v = a \cdot v + b \cdot v$, $(a \cdot b) \cdot v = a \cdot (b \cdot v)$, and $1 \cdot v = v$. Three-dimensional space, $\mathbb{R}^3$, is our basic vector space over the field of reals: the vectors are the points of $\mathbb{R}^3$, $+$ is coordinatewise addition, and $\cdot$ is coordinatewise multiplication. A *linear map* over a vector space V is an $f \in V \to V$ such that $f(v + w) = f(v) + f(w)$ and $f(a \cdot v) = a \cdot f(v)$; group axioms then ensure that it preserves also unit and inverse. $Lin(V)$ is the set of such linear maps. In *Euclidean spaces*, e.g. $\mathbb{R}^3$, one considers the ability to *measure*. This is achieved by extending the underlying vector space with the *dot product* of vectors, giving the ability to measure distances and angles, and with the *cross product* of vectors, giving the ability to generate out-of-plane vectors, to measure areas and volumes, and to detect handedness. Both dot and cross product are linear maps in each argument.

The *General Linear Group* $GL(V) \subseteq Lin(V)$ of a vector space V is the group of all the automorphisms (bijective linear maps) over V, i.e., invertible elements of $Lin(V)$. When studying subgroups of $GL(V)$, it is convenient to use linear algebra to represent the group elements. In particular, $GL(\mathbb{R}^3)$, the group of automorphisms of the $\mathbb{R}^3$ vector space, can be given as the group of invertible 3×3 matrices A in linear algebra, where matrix multiplication $A \cdot B$ is an operation over sizes $(n \times m) \times (m \times n) \to (m \times n)$. On matrices we use also A^T for transposition, $A + B$ for addition, $a \cdot A$ for scalar multiplication, and A^{-1} for inverse. With the elements $v \in \mathbb{R}^3$ interpreted as 1×3 (column) matrices, we obtain the required linearity properties from linear algebra: $A \cdot (v + v') = A \cdot v + A \cdot v'$ and $A \cdot (a \cdot v) = a \cdot (A \cdot v)$ for any scalar a. Note again that only the invertible, i.e. bijective, matrices are members of $GL(\mathbb{R}^3)$. The *Special Linear Group* $SL(\mathbb{R}^3)$ is the subgroup of matrices with determinant 1: as transformations these preserve volume and handedness.

The *General Affine Group* $GA(V)$ is the group of *affine vector maps* over V; these maps are presented as pairs $\langle A, u \rangle$ where $A \in GL(V)$, and where $u \in V$ is a translation vector. In particular, $GA(\mathbb{R}^3)$ is the affine group over the $\mathbb{R}^3$ vector space. We use 3×3 invertible matrices for A, with $\langle A, u \rangle(v) \triangleq A \cdot v + u$ for any $v \in \mathbb{R}^3$. Geometrically, affine vector maps transform straight lines into straight lines, and preserve ratios such as midpoints. The *Special Affine Group* $SA(\mathbb{R}^3)$ is the subgroup with matrices with determinant 1.

Concretely, we work always over the field $\mathbb{R}$ and the vector space $\mathbb{R}^3$, hence we abbreviate these groups as $GA(3)$, $SA(3)$, $GL(3)$, $SL(3)$.

For the next automorphisms groups we need to investigate some special matrices. An *orthogonal matrix* is a square matrix A such that $A^T = A^{-1}$ (and hence $A \cdot A^T = A^T \cdot A = id$, and also $det(A) =$

± 1). All orthogonal matrices are *isometries*, i.e., preserve distances, which can be seen as follows. The vector dot product (of column matrices) is defined as $v \bullet w \triangleq v^T \cdot w$, and $v^2 \triangleq v \bullet v$. If $A^T = A^{-1}$ we then have that $A \cdot v \bullet A \cdot w = (A \cdot v)^T \cdot (A \cdot w) = v^T \cdot A^T \cdot A \cdot w = v^T \cdot id \cdot w = v^T \cdot w = v \bullet w$. And also $(A \cdot v)^2 = v^2$. Distance in a vector space equipped with dot product is defined as $d(v, w) \triangleq \sqrt{(v-w)^2}$. For A orthogonal, we then have $d(A \cdot v, A \cdot w) = \sqrt{(A \cdot v - A \cdot w)^2} = \sqrt{(A \cdot (v-w))^2} = \sqrt{(v-w)^2} = d(v, w)$, that is, A preserves distances.

The *Orthogonal Group $O(3)$*, subgroup of $GL(3)$, is the group of *linear isometries* of $\mathbb{R}^3$, that is, the group of orthogonal matrices, which correspond to rotations and reflections around the origin. As we have just shown, members of $O(3)$ preserve dot product: $A \cdot v \bullet A \cdot w = v \bullet w$. The *Special Orthogonal Group $SO(3)$* contains only the direct linear isometries, that is, just the rotations. Members of $SO(3)$ distribute over cross product: $A \cdot v \times A \cdot w = A \cdot (v \times w)$ [13]. Intuitively that is because cross product can measure areas and handedness, but is insensitive to isometries that do not change handedness.

The *Euclidean Group $E(3)$*, subgroup of $GA(3)$, is the group of isometries of $\mathbb{R}^3$; its elements can be given as affine vector maps $\langle A, u \rangle$ where A is an orthogonal matrix (a rotation or reflection) and u is a translation vector. We have seen that members of $O(3)$ are isometries, but such $\langle A, u \rangle$ are too: for $A \in O(3)$ we have that $d(\langle A, u \rangle(v), \langle A, u \rangle(w)) = d(A \cdot v + u, A \cdot w + u) = \sqrt{(A \cdot v + u - (A \cdot w + u))^2} = \sqrt{(A \cdot v - A \cdot w)^2} = d(v, w)$. That is, all affine vector maps $\langle A, u \rangle$ where A is an orthogonal matrix are also isometries.

The subgroup $SE(3)$ of $E(3)$ of *direct isometries* excludes reflections; that is, the determinant of A must be 1. Elements of $SE(3)$ are then the *rigid body motions*, preserving handedness and distances.

The subgroup relation on the automorphism groups discussed so far forms a cube standing on the $SO(3)$ vertex, with $GA(3)$ at the top. Maps contained in the bottom faces of the cube have the following interpretation: the face below $E(3)$ preserves distances and angles; the face below $SA(3)$ preserves volumes and orientation; the face below $GL(3)$ preserves the origin. Various vertices of the cube hold the basic geometric transformations: rotation, translation, reflection, shearing, isotropic scaling, and volume-preserving squishing (non-orthogonal matrices with $det = 1$). There are many more automorphism groups; e.g., the group of pure translations, below $SE(3)$, the group of pure reflections, below $O(3)$, and the group of identities below all of them. However, the cube depicts the most studied automorphism groups, and a finer structure is not necessary for the study of geometric invariance properties, at least not in this paper.

We work in $GA(3)$ and its subgroups. For example, we regard an affine vector map $\langle A, u \rangle \in GA(3)$ as a member of $GL(3)$ when $u = 0$, and as a member of $E(3)$ when A is orthogonal, and further as a member of $O(3)$ when $u = 0$. We fix a representation of affine vector maps based on linear algebra.

6.2. **Affine Spaces and Affine Maps.** Affine geometry is intuitively the geometry of properties invariant under translation, rotation, reflection and stretching. It can be properly formulated by the notions of affine spaces and affine maps [3, 5].

Definition 14. Affine spaces
    An affine space is a triple (P, V, θ) where P is a set (of points), V is a vector space, and $\theta \in P \times P \to V$ is a function which characterizes 'the unique vector $\theta(p, q)$ from p to q'. The map θ must satisfy:
    (1) for each $p \in P$, $\theta_p \in P \to V = \lambda q.\theta(p, q)$ is a bijection;
    (2) the head-to-tail equation holds: $\theta(p, q) + \theta(q, r) = \theta(p, r)$.

Because of (1), P and V are isomorphic, but there is no canonical isomorphism. The vector $\theta(p,q)$ is sometimes called the point difference, written $q \dotminus p$. We also define vector-point addition as $\dotplus \in V \times P \to P = \lambda v, p.\theta_p^{-1}(v)$ (which is a *group action* of $(V,+)$ on P). The *affine space of free vectors* over P is a canonical affine space constructed over a set of points P that is also a vector space. It is common to take $V = P$ in such a construction. In our operational semantics, however, we need to distinguish between points and vectors; hence we take for V a set isomorphic but distinguishable from P. We focus on the space of free vectors over the points of $\mathbb{R}^3$. Note that $\mathbb{R}^3$ is also a vector space, with the null vector indicated by ϕ.

Definition 15. The affine space of free vectors over $\mathbb{R}^3$

The affine space of free vectors over $\mathbb{R}^3$ is $(\mathbb{R}^3, FV(\mathbb{R}^3), \Uparrow)$, where:

∘ The set of points of the affine space is $\mathbb{R}^3$.

∘ $FV(\mathbb{R}^3) \triangleq \{\phi\} \times \mathbb{R}^3$ is a vector space equipped with $\bullet$ and $\times$, given by the product structure.

∘ $\Uparrow \in \mathbb{R}^3 \times \mathbb{R}^3 \to FV(\mathbb{R}^3) \triangleq \lambda(p,q).\langle \phi, q - p \rangle$.

Auxiliary definitions and properties:

∘ $\Uparrow_p \triangleq \lambda(q). \Uparrow (p,q)$ is a bijection for each p.

∘ $q \dotminus p \triangleq \Uparrow (p,q) = \langle \phi, q - p \rangle$

∘ $v \dotplus p \triangleq \Uparrow_p^{-1}(v)$ with $\langle \phi, q \rangle \dotplus p = q + p$

∘ $\uparrow \triangleq \Uparrow_\phi$ and $\downarrow \triangleq \uparrow^{-1}$ are linear maps, with $\uparrow p = \langle \phi, p \rangle$, and $\downarrow \langle \phi, p \rangle = p$.

The set $\{\phi\} \times \mathbb{R}^3$ can be seen also as the set of canonical representatives of *free vectors* (equivalence classes of vectors with the same size and orientation), and can be explained as the vectors rooted at the origin.

Affine vector maps of the form $\lambda v.f(v) + u \in V \to V$ with $f \in Lin(V)$ are common in the literature of automorphism groups, as presented in Section 6.1. Affine point maps of the form $\lambda q.f(q \dotminus o) \dotplus p \in P \to P$ instead are common in the literature of affine spaces [5]. Confusingly, they are both called just 'affine maps'. Bijective point and vector maps form groups under function composition, identity, and inverse, and these groups are related by a group isomorphism: for each choice of origin o, just like there is an isomorphism θ_o between points P and vectors V, there is also a group isomorphism $\psi_o = \lambda h.\theta_o \circ h \circ \theta_o^{-1}$ between the group of bijective affine point maps with origin o, and the group of bijective affine vector maps $GA(V)$. The isomorphism transforms a point map that maps a point p seen as a vector $p \dotminus o$ to the point $f(p \dotminus o) \dotplus q$, into a vector map that maps the vector $p \dotminus o$ to the vector $f(p \dotminus o) + (q \dotminus o)$, which when rooted at the origin leads to the point $(f(p \dotminus o) + (q \dotminus o)) \dotplus o = f(p \dotminus o) \dotplus q$. Up to this group isomorphism, we consider affine point maps (then called just affine maps in the body of this paper) as members of $GA(V)$.

Affine point maps over the affine space of free vectors over $\mathbb{R}^3$ are denoted by script letters $\mathcal{A}, \mathcal{B}, \mathcal{C}, ...$ and are represented as pairs $\mathcal{A} = \langle A, q \rangle$. They are applied to points p to obtain transformed points $A \cdot p + q$, and are extended to vectors $v = \Uparrow (p,q)$ by taking $\mathcal{A}(\Uparrow (p,q)) = \Uparrow (\mathcal{A}(p), \mathcal{A}(q))$, which means that $\mathcal{A}(v) = (\uparrow \circ A \circ \downarrow)(v)$, where the translation components cancel: this reflects the fact that v are 'free' vectors, invariant under translation. These rules for applying maps, and the rules for composing and inverting maps, are given in Definition 16.

Definition 16. Affine point maps

$\mathcal{A} \in GA(3)$ means $\mathcal{A} = \langle A, p \rangle$ where $\uparrow \circ A \circ \downarrow \in GL(3)$;   that is, $det(A) \neq 0$.

$\mathcal{A} \in E(3)$ means $\mathcal{A} = \langle A, p \rangle$ where $\uparrow \circ A \circ \downarrow \in O(3)$;   that is, $A^T = A^{-1}$.

$\mathcal{A} \in SE(3)$ means $\mathcal{A} = \langle A, p \rangle$ where $\uparrow \circ A \circ \downarrow \in SO(3)$;   that is, $det(A) = 1$.

$$\forall q \in \mathbb{R}^3, \forall \mathcal{A} \in GA(3). \qquad \mathcal{A}(q) \quad = \quad \langle A, p \rangle(q) \quad \triangleq \quad A \cdot q + p \qquad \in \mathbb{R}^3$$
$$\forall v \in FV(\mathbb{R}^3), \forall \mathcal{A} \in GA(3). \qquad \mathcal{A}(v) \quad = \quad \langle A, p \rangle(v) \quad \triangleq \quad (\uparrow \circ A \circ \downarrow)(v) \qquad \in FV(\mathbb{R}^3)$$
$$\forall \mathcal{A}, \mathcal{B} \in G \text{ subgroup of } GA(3). \quad \mathcal{A} \circ \mathcal{B} \quad = \quad \langle A, p \rangle \circ \langle B, q \rangle \quad \triangleq \quad \langle A \cdot B, A \cdot q + p \rangle \quad \in G$$
$$\forall \mathcal{A} \in G \text{ subgroup of } GA(3). \qquad \mathcal{A}^{-1} \quad = \quad \langle A, p \rangle^{-1} \qquad \triangleq \quad \langle A^{-1}, -A^{-1} \cdot p \rangle \quad \in G$$

It should be noted that this definition can be formulated as a theorem in a general treatment of the groups of affine vector maps and affine point maps, and their representation in terms of linear algebra. For conciseness, we take it here as a given.

The following proposition collects all the geometric facts needed in Theorem 22.

Proposition 17. *Distribution laws of affine point maps*

1) $\forall p, q \in \mathbb{R}^3, \mathcal{A} \in GA(3).$ $\qquad \mathcal{A}(q) \dot{-} \mathcal{A}(p) = \mathcal{A}(q \dot{-} p) \qquad \in FV(\mathbb{R}^3)$

2) $\forall v \in FV(\mathbb{R}^3), p \in \mathbb{R}^3, \mathcal{A} \in GA(3).$ $\quad \mathcal{A}(v) \dot{+} \mathcal{A}(p) = \mathcal{A}(v \dot{+} p) \qquad \in \mathbb{R}^3$

3) $\forall v, w \in FV(\mathbb{R}^3), \mathcal{A} \in GA(3).$ $\qquad \mathcal{A}(v) + \mathcal{A}(w) = \mathcal{A}(v + w) \quad \in FV(\mathbb{R}^3)$

4) $\forall a \in R, v \in FV(\mathbb{R}^3), \mathcal{A} \in GA(3).$ $\quad a \cdot \mathcal{A}(v) = \mathcal{A}(a \cdot v) \qquad \in FV(\mathbb{R}^3)$

5) $\forall v, w \in FV(\mathbb{R}^3), \mathcal{A} \in E(3).$ $\qquad \mathcal{A}(v) \bullet \mathcal{A}(w) = v \bullet w \qquad \in R$

6) $\forall v, w \in FV(\mathbb{R}^3), \mathcal{A} \in SE(3).$ $\qquad \mathcal{A}(v) \times \mathcal{A}(w) = \mathcal{A}(v \times w) \quad \in FV(\mathbb{R}^3)$

Proof. Let $\mathcal{A} = \langle A, r \rangle$. By Definition 16: if $\mathcal{A} \in GA(3)$, then the vector map $\uparrow \circ A \circ \downarrow \in GL(3)$ is a linear map; if $\mathcal{A} \in E(3)$, then $\uparrow \circ A \circ \downarrow \in O(3)$; and if $\mathcal{A} \in SE(3)$ then $\uparrow \circ A \circ \downarrow \in SO(3)$. Recall that if $f \in O(3)$ then $f(v) \bullet f(w) = v \bullet w$, and if $f \in SO(3)$ then $f(v) \times f(w) = f(v \times w)$ [13].

1) $\langle A, r \rangle(q) \dot{-} \langle A, r \rangle(p) = A \cdot q + r \dot{-} A \cdot p + r = \langle \phi, (A \cdot q + r) - (A \cdot p + r) \rangle = \langle \phi, A \cdot (q - p) \rangle = (\uparrow \circ A \circ \downarrow) \cdot \langle \phi, q - p \rangle = \langle A, r \rangle(q \dot{-} p).$

2) $\langle A, r \rangle(\langle \phi, q \rangle) \dot{+} \langle A, r \rangle(p) = (\uparrow \circ A \circ \downarrow)(\langle \phi, q \rangle) \dot{+} A \cdot p + r = \langle \phi, A \cdot q \rangle \dot{+} A \cdot p + r = A \cdot (q + p) + r = \langle A, r \rangle(q + p) = \langle A, r \rangle(\langle \phi, q \rangle \dot{+} p).$

3) $\langle A, r \rangle(\langle \phi, p \rangle) + \langle A, r \rangle(\langle \phi, q \rangle) = (\uparrow \circ A \circ \downarrow)(\langle \phi, p \rangle) + (\uparrow \circ A \circ \downarrow)(\langle \phi, q \rangle) = \uparrow (A \cdot p) + \uparrow (A \cdot q) = \uparrow (A \cdot p + A \cdot q) = \uparrow (A \cdot (p + q)) = (\uparrow \circ A \circ \downarrow)(\langle \phi, p + q \rangle) = (\uparrow \circ A \circ \downarrow)(\langle \phi, p \rangle + \langle \phi, q \rangle) = \langle A, r \rangle(\langle \phi, p \rangle + \langle \phi, q \rangle).$

4) $a \cdot \langle A, r \rangle(\langle \phi, p \rangle) = a \cdot (\uparrow \circ A \circ \downarrow)(\langle \phi, p \rangle) = a \cdot \uparrow (A \cdot p) = \uparrow (a \cdot A \cdot p) = \uparrow (A \cdot (a \cdot p)) = (\uparrow \circ A \circ \downarrow)(\langle \phi, a \cdot p \rangle) = \langle A, r \rangle(a \cdot \langle \phi, p \rangle).$

5) $\langle A, r \rangle(v) \bullet \langle A, r \rangle(w) = (\uparrow \circ A \circ \downarrow)(v) \bullet (\uparrow \circ A \circ \downarrow)(w) = v \bullet w$ since $\uparrow \circ A \circ \downarrow \in O(3)$.

6) $\langle A, r \rangle(v) \times \langle A, r \rangle(w) = (\uparrow \circ A \circ \downarrow)(v) \times (\uparrow \circ A \circ \downarrow)(w) = (\uparrow \circ A \circ \downarrow)(v \times w)$ since $\uparrow \circ A \circ \downarrow \in SO(3)$ $= \langle A, r \rangle(v \times w).$ $\qquad \square$

References

[13] F. Jones. Vector Calculus. Chapter 7: Cross Product. (Unpublished book; available at http://www.owlnet.rice.edu/~fjones/chap7.pdf.)

7. Appendix B: Proofs

The full definition of Structural Congruence is as follows:

Definition 18. Structural Congruence

$$(\equiv Refl) \qquad P \equiv P$$
$$(\equiv Symm) \qquad P \equiv Q \Rightarrow Q \equiv P$$
$$(\equiv Tran) \qquad P \equiv Q, Q \equiv R \Rightarrow P \equiv R$$

$(\equiv Act)$ $P \equiv P' \Rightarrow \pi.P \equiv \pi.P'$

$(\equiv Sum)$ $P \equiv P', Q \equiv Q' \Rightarrow P + Q \equiv P' + Q'$

$(\equiv Par)$ $P \equiv P', Q \equiv Q' \Rightarrow P \mid Q \equiv P' \mid Q'$

$(\equiv Res)$ $P \equiv P' \Rightarrow (\nu x)P \equiv (\nu x)P'$

$(\equiv Repl)$ $P \equiv P' \Rightarrow P^* \equiv P'^*$

$(\equiv Map)$ $P \equiv P' \Rightarrow M[P] \equiv M[P']$

$(\equiv Map\ Cmp)$ $M[\Delta =_\sigma \Delta'.P] \equiv M[\Delta] =_\sigma M[\Delta'].M[P]$

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

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

$(\equiv Map\ Sum)$ $M[P + Q] \equiv M[P] + M[Q]$

$(\equiv Map\ Par)$ $M[P \mid Q] \equiv M[P] \mid M[Q]$

$(\equiv Map\ Res)$ $M[(\nu x)P] \equiv (\nu x)M[P]$

$(\equiv Map\ Comp)$ $M[N[P]] \equiv (M \circ M[N])[P]$

$(\equiv Sum\ Comm)$ $P + Q \equiv Q + P$

$(\equiv Sum\ Assoc)$ $(P + Q) + R \equiv P + (Q + R)$

$(\equiv Sum\ Zero)$ $P + 0 \equiv P$

$(\equiv Par\ Comm)$ $P \mid Q \equiv Q \mid P$

$(\equiv Par\ Assoc)$ $(P \mid Q) \mid R \equiv P \mid (Q \mid R)$

$(\equiv Par\ Zero)$ $P \mid 0 \equiv P$

$(\equiv Res\ Zero)$ $(\nu x)0 \equiv 0$

$(\equiv Res\ Sum)$ $(\nu x)(P + Q) \equiv P + (\nu x)Q$ $(x \notin fv_{\mathbf{c}}(P))$

$(\equiv Res\ Par)$ $(\nu x)(P \mid Q) \equiv P \mid (\nu x)Q$ $(x \notin fv_{\mathbf{c}}(P))$

$(\equiv Res\ Res)$ $(\nu x)(\nu y)P \equiv (\nu y)(\nu x)P$

$(\equiv Repl\ Zero)$ $0^* \equiv 0$

$(\equiv Repl\ Par)$ $(P \mid Q)^* \equiv P^* \mid Q^*$

$(\equiv Repl\ Copy)$ $P^* \equiv P \mid P^*$

$(\equiv Repl\ Repl)$ $P^{**} \equiv P^*$

The basic operators over the data values are given in Definition 19. Note that there are similar operations on different domains: for example, $+$ between reals, $+$ between vectors, and $\dotplus$ between vectors and points. Note also that vector mapping ignores the translation component p (or rather, it cancels when applied to the end points of v); this is the sense in which vectors are 'free': invariant under translation.

Definition 19. Operations on Points, Vectors, and Maps

$\langle x, y, z \rangle \dot{-} \langle x', y', z' \rangle \;\triangleq\; \uparrow \langle x - x', y - y', z - z' \rangle$ point subtraction

$\uparrow \langle x, y, z \rangle \dotplus \langle x', y', z' \rangle \;\triangleq\; \langle x + x', y + y', z + z' \rangle$ point translation

$a \cdot \uparrow \langle x, y, z \rangle \;\triangleq\; \uparrow \langle a \cdot x, a \cdot y, a \cdot z \rangle$ vector scaling

$\uparrow \langle x, y, z \rangle + \uparrow \langle x', y', z' \rangle \;\triangleq\; \uparrow \langle x + x', y + y', z + z' \rangle$ vector addition

$\uparrow \langle x, y, z \rangle \bullet \uparrow \langle x', y', z' \rangle \;\triangleq\; x \cdot x' + y \cdot y' + z \cdot z'$ dot product

$$\uparrow \langle x, y, z \rangle \times \uparrow \langle x', y', z' \rangle \;\triangleq\; \uparrow \langle y \cdot z' - z \cdot y', z \cdot x' - x \cdot z', x \cdot y' - y \cdot x' \rangle \quad \text{cross product}$$

$$\langle A, p \rangle (q) \;\triangleq\; A \cdot q + p \qquad\qquad\qquad\qquad\qquad\qquad \text{point mapping}$$
$$\langle A, p \rangle (v) \;\triangleq\; (\uparrow \circ A \circ \downarrow)(v) \qquad\qquad\qquad\qquad\qquad \text{vector mapping}$$
$$\langle A, p \rangle \circ \langle A', p' \rangle \;\triangleq\; \langle A \cdot A', A \cdot p' + p \rangle \qquad\qquad\qquad \text{map composition}$$
$$\langle A, p \rangle^{-1} \;\triangleq\; \langle A^{-1}, -A^{-1} \cdot p \rangle \qquad\qquad\qquad\qquad \text{map inverse}$$

In Definition 20 we define the relation $\Delta_{\mathcal{A}} \mapsto \varepsilon$, which describes the computation of a closed data term Δ to value ε, relative to global frame $\mathcal{A}$. The relation $_{\mathcal{A}}\mapsto$ is a partial function, described in operational style for ease of induction. The key rule is $(Frame\ Shift)$: when computation encounters a frame shift $M[\Delta]$, the value of $M[\Delta]$ in frame $\mathcal{A}$ is uniquely determined as the value of Δ in frame $\mathcal{A} \circ \mathcal{B}$, provided that the value of M in frame $\mathcal{A}$ is $\mathcal{B}$.

Definition 20. Computation of closed data terms in a frame $\mathcal{A}$

$(Scalar\ Real)$	$r_{\mathcal{A}} \mapsto b$	if literal r represents $b \in Val_{\mathbf{a}}$
$(Scalar\ Arith)$	$a_{i\,\mathcal{A}} \mapsto b_i \Rightarrow f(a_i)_{\mathcal{A}} \mapsto f(b_i) \quad i \in 1..arity(f)$	if $b_i \in Val_{\mathbf{a}}$, $f(b_i)$ defined
$(Scalar\ Dot)$	$v_{\mathcal{A}} \mapsto w, v'_{\mathcal{A}} \mapsto w' \Rightarrow v \bullet v'_{\mathcal{A}} \mapsto w \bullet w'$	if $w, w' \in Val_{\mathbf{v}}$
$(Point\ Origin)$	$\maltese_{\mathcal{A}} \mapsto \mathcal{A}(\langle 0, 0, 0 \rangle)$	
$(Point\ Move)$	$v_{\mathcal{A}} \mapsto w, p_{\mathcal{A}} \mapsto q \Rightarrow v + p_{\mathcal{A}} \mapsto w \dotplus q$	if $w \in Val_{\mathbf{v}}$, $q \in Val_{\mathbf{p}}$
$(Vect\ Unit_x)$	$\uparrow_{x\ \mathcal{A}} \mapsto \mathcal{A}(\uparrow \langle 1, 0, 0 \rangle)$	
$(Vect\ Unit_y)$	$\uparrow_{y\ \mathcal{A}} \mapsto \mathcal{A}(\uparrow \langle 0, 1, 0 \rangle)$	
$(Vect\ Unit_z)$	$\uparrow_{z\ \mathcal{A}} \mapsto \mathcal{A}(\uparrow \langle 0, 0, 1 \rangle)$	
$(Vect\ Sub)$	$p_{\mathcal{A}} \mapsto q, p'_{\mathcal{A}} \mapsto q' \Rightarrow p - p'_{\mathcal{A}} \mapsto q \dotminus q'$	if $q, q' \in Val_{\mathbf{p}}$
$(Vect\ Scale)$	$a_{\mathcal{A}} \mapsto b, v_{\mathcal{A}} \mapsto w \Rightarrow a \cdot v_{\mathcal{A}} \mapsto b \cdot w$	if $b \in Val_{\mathbf{a}}$, $w \in Val_{\mathbf{v}}$
$(Vect\ Add)$	$v_{\mathcal{A}} \mapsto w, v'_{\mathcal{A}} \mapsto w' \Rightarrow v + v'_{\mathcal{A}} \mapsto w + w'$	if $w, w' \in Val_{\mathbf{v}}$
$(Vect\ Cross)$	$v_{\mathcal{A}} \mapsto w, v'_{\mathcal{A}} \mapsto w' \Rightarrow v \times v'_{\mathcal{A}} \mapsto w \times w'$	if $w, w' \in Val_{\mathbf{v}}$
$(Map\ Given)$	$a_{ij\,\mathcal{A}} \mapsto b_{ij}, a_{k\,\mathcal{A}} \mapsto b_k \Rightarrow \langle a_{ij}, a_k \rangle_{\mathcal{A}} \mapsto \langle b_{ij}, b_k \rangle$	if $b_{ij}, b_k \in Val_{\mathbf{a}}$, $det(b_{ij}) \neq 0$
$(Map\ Comp)$	$M_{\mathcal{A}} \mapsto \mathcal{B}, M'_{\mathcal{A}} \mapsto \mathcal{B}' \Rightarrow M \circ M'_{\mathcal{A}} \mapsto \mathcal{B} \circ \mathcal{B}'$	if $\mathcal{B}, \mathcal{B}' \in Val_{\mathbf{m}}$
$(Map\ Inv)$	$M_{\mathcal{A}} \mapsto \mathcal{B} \Rightarrow M^{-1}_{\mathcal{A}} \mapsto \mathcal{B}^{-1}$	if $\mathcal{B} \in Val_{\mathbf{m}}$
$(Frame\ Shift)$	$M_{\mathcal{A}} \mapsto \mathcal{B}, \Delta_{\mathcal{A} \circ \mathcal{B}} \mapsto \varepsilon \Rightarrow M[\Delta]_{\mathcal{A}} \mapsto \varepsilon$	if $\mathcal{B} \in Val_{\mathbf{m}}$
$(Value)$	$\varepsilon_{\mathcal{A}} \mapsto \varepsilon$	if $\varepsilon \in Val_{\mathbf{m}}$

Most of these rules express a straightforward correspondence between the syntactic operations on data terms and semantic operations on values. It is easy to check that terms of sort σ compute to elements of Val_{σ}. Note that the rules $(Point\ Origin)$ and $(Vect\ Unit)$ make essential use of the current frame. The rules $(Scalar\ Arith)$ and $(Map\ Given)$ are partial: they can cause 'divide by zero', 'zero determinant', and other errors. However, $(Map\ Inv)$ is always defined because if $M_{\mathcal{A}} \mapsto \mathcal{B}$, then $\mathcal{B}$ must be invertible by $(Map\ Given)$. The $(Frame\ Shift)$ rule has already been discussed. The $(Value)$ rule normally comes into play after a by-value substitution due to process interaction: a value that was already evaluated in some frame is not further evaluated in the current frame. Moreover, since $Val_{\mathbf{c}} = Var_{\mathbf{c}}$, the $(Value)$ rule covers also the evaluation of channels to themselves; that is, $x_{\mathbf{c}\,\mathcal{A}} \mapsto x_{\mathbf{c}}$.

In the formulation of our results we also require the notion of $\mathcal{C}(\Delta)$, which is the application of the map $\mathcal{C}$ to all the value subterms of Δ:

Definition 21. Map Application on Data

For $\mathcal{C} = \langle A, p \rangle \in Val_\mathbf{m}$, define

$$
\begin{array}{llll}
\mathcal{C}(\varepsilon) & \triangleq & A \cdot \varepsilon + p & \text{if } \varepsilon \in Val_\mathbf{p} & \text{(on points)} \\
\mathcal{C}(\varepsilon) & \triangleq & (\uparrow \circ A \circ \downarrow)(\varepsilon) & \text{if } \varepsilon \in Val_\mathbf{v} & \text{(on vectors)} \\
\mathcal{C}(\varepsilon) & \triangleq & \varepsilon & \text{if } \varepsilon \in Val_\mathbf{a} \cup Val_\mathbf{m} \cup Val_\mathbf{c} & \text{(on scalars, maps, and channels)}
\end{array}
$$

$\mathcal{C}(\Delta)$ is the term obtained by replacing all the value subterms ε of Δ with $\mathcal{C}(\varepsilon)$.

The choices in this definition are simply explained by examples. Consider the term $\Delta =\uparrow \langle 1, 0, 0 \rangle + $ ✠, containing the fixed value $\uparrow \langle 1, 0, 0 \rangle$, and the relative origin ✠, with reductions (by $(Value)$ and $(Point\ Origin)$):

$$
\begin{array}{llll}
\Delta & =\uparrow \langle 1, 0, 0 \rangle + ✠ & _{\mathcal{A}}\mapsto & \uparrow \langle 1, 0, 0 \rangle + \mathcal{A}(\langle 0, 0, 0 \rangle) \\
\mathcal{C}(\Delta) & = \mathcal{C}(\uparrow \langle 1, 0, 0 \rangle) + ✠ & _{\mathcal{C} \circ \mathcal{A}}\mapsto & \mathcal{C}(\uparrow \langle 1, 0, 0 \rangle) + (\mathcal{C} \circ \mathcal{A})(\langle 0, 0, 0 \rangle)
\end{array}
$$

That is, for $\varepsilon =\uparrow \langle 1, 0, 0 \rangle + \mathcal{A}(\langle 0, 0, 0 \rangle)$, we have:

$$
\Delta_{\mathcal{A}}\mapsto\varepsilon \quad \text{and} \quad \mathcal{C}(\Delta)_{\mathcal{C} \circ \mathcal{A}}\mapsto\mathcal{C}(\varepsilon)
$$

Similarly, $\mathcal{B}[✠]_{\mathcal{A}}\mapsto(\mathcal{A} \circ \mathcal{B})(\langle 0, 0, 0 \rangle)$ and $\mathcal{C}(\mathcal{B}[✠]) = \mathcal{B}[✠] \,_{\mathcal{C} \circ \mathcal{A}}\mapsto (\mathcal{C} \circ \mathcal{A} \circ \mathcal{B})(\langle 0, 0, 0 \rangle) = \mathcal{C}((\mathcal{A} \circ \mathcal{B})(\langle 0, 0, 0 \rangle))$, where $\mathcal{C}(\mathcal{B}) = \mathcal{B}$ because maps $\mathcal{B}$ are arrays of reals, and like reals are not affected by mapping. This suggests the general form of our next theorem: $\mathcal{C}(\Delta)_{\mathcal{C} \circ \mathcal{A}}\mapsto$ means applying an extra $\mathcal{C}$ separately to the values inside Δ via $\mathcal{C}(\Delta)$ (which are then not modified by the $(Value)$ rule), and to the other terms inside Δ via $_{\mathcal{C} \circ \mathcal{A}}\mapsto$. The proof of Theorem 22 uses geometric facts that are derived in Appendix 1.

Theorem 22. *Global Frame Shift for Data*
$$
\mathcal{C} \propto \Delta, \Delta_{\mathcal{A}}\mapsto\epsilon \quad \Rightarrow \quad \mathcal{C}(\Delta)_{\mathcal{C} \circ \mathcal{A}}\mapsto\mathcal{C}(\epsilon)
$$

Proof. The proof is by mutual induction on the derivation of $\Delta_{\mathcal{A}}\mapsto\varepsilon$; that is, by induction on the conjunction of 5 statements for the 5 sorts σ of Δ, as given in the 5 cases below.

When $\Delta = \varepsilon$, the ε of the various sorts fall into the respective subcases. Since all these subcases are handled equally, we show the $(Value)$ case first:

Rule $(Value)$: Show that $\mathcal{C} \propto \varepsilon, \varepsilon_{\mathcal{A}}\mapsto\varepsilon \Rightarrow \mathcal{C}(\varepsilon)_{\mathcal{C} \circ \mathcal{A}}\mapsto\mathcal{C}(\varepsilon)$, for ε of any sort. Then, by $(Value)$ $\mathcal{C}(\Delta) = \mathcal{C}(\varepsilon)_{\mathcal{C} \circ \mathcal{A}}\mapsto\mathcal{C}(\varepsilon)$.

Case $(\sigma = \mathbf{c})$: Show that $\mathcal{C} \propto \Delta, \Delta_{\mathcal{A}}\mapsto x_\mathbf{c} \Rightarrow \mathcal{C}(\Delta)_{\mathcal{C} \circ \mathcal{A}}\mapsto x_\mathbf{c}$. Then, $\Delta_{\mathcal{A}}\mapsto x_\mathbf{c}$ is the consequent of Rule $(Value)$ or:

Rule $(Frame\ Shift)$: $M_{\mathcal{A}}\mapsto\mathcal{B}, \Delta'\mathcal{A} \circ B\mapsto x_\mathbf{c} \Rightarrow M[\Delta']_{\mathcal{A}}\mapsto x_\mathbf{c}$. Since $\mathcal{C} \propto M[\Delta']$, we have $\mathcal{C} \propto M$ and $\mathcal{C} \propto \Delta'$. It follows that $\mathcal{C}(M)_{\mathcal{C} \circ \mathcal{A}}\mapsto\mathcal{B}$ and $\mathcal{C}(\Delta')_{\mathcal{C} \circ \mathcal{A} \circ \mathcal{B}}\mapsto x_\mathbf{c}$ (by induction). Hence $\mathcal{C}(M[\Delta']) = \mathcal{C}(M)[\mathcal{C}(\Delta')]_{\mathcal{C} \circ \mathcal{A}}\mapsto x_\mathbf{c}$ by $(Frame\ Shift)$.

Case $(\sigma = \mathbf{a})$: Show that $\mathcal{C} \propto a, a_{\mathcal{A}}\mapsto b \Rightarrow \mathcal{C}(a)_{\mathcal{C} \circ \mathcal{A}}\mapsto b$. Then $a_{\mathcal{A}}\mapsto b$ is the consequent of Rule $(Value)$ or one of the rules:

Rule $(Scalar\ Real)$: $r_{\mathcal{A}}\mapsto b$. Then $\mathcal{C}(r) = r_{\mathcal{C} \circ \mathcal{A}}\mapsto b$ (by $(Scalar\ Real)$).

Rule $(Scalar\ Arith)$: $a_{i\mathcal{A}}\mapsto b_i \Rightarrow f(a_i)_{\mathcal{A}}\mapsto f(b_i)$ with $i \in 1..arity(f)$. Since $f(a_i)_{\mathcal{A}}\mapsto f(b_i)$ we know that $f(b_i)$ is defined. Then $\mathcal{C}(f(a_i)) = f(\mathcal{C}(a_i))_{\mathcal{C} \circ \mathcal{A}}\mapsto f(b_i)$ (by induction and $(Scalar\ Arith)$).

Rule (*Scalar Dot*): $v_\mathcal{A}\mapsto w, v'_\mathcal{A}\mapsto w' \Rightarrow v\bullet v'_\mathcal{A}\mapsto w\bullet w'$, and $\mathcal{C} \in E(3)$. $\mathcal{C}(v\bullet v') = \mathcal{C}(v)\bullet\mathcal{C}(v')_{\mathcal{C}\circ\mathcal{A}}\mapsto\mathcal{C}(w)\bullet\mathcal{C}(w')$ (by induction and (*Scalar Dot*)) $= w\bullet w'$ (by Prop. 5.2–4).

Rule (*FrameShift*)($\sigma = \mathbf{a}$): $M_\mathcal{A}\mapsto\mathcal{B}, a'_{\mathcal{A}\circ\mathcal{B}}\mapsto b \Rightarrow M[a']_\mathcal{A}\mapsto b$. Since $\mathcal{C} \propto M[a']$, we have $\mathcal{C} \propto M$ and $\mathcal{C} \propto a'$. It follows that $\mathcal{C}(M)_{\mathcal{C}\circ\mathcal{A}}\mapsto\mathcal{B}$ and $\mathcal{C}(a')_{\mathcal{C}\circ\mathcal{A}\circ\mathcal{B}}\mapsto b$ (by induction). Hence $\mathcal{C}(M[a']) = \mathcal{C}(M)[\mathcal{C}(a')]_{\mathcal{C}\circ\mathcal{A}}\mapsto b$ by (*Frame Shift*).

Case ($\sigma = \mathbf{p}$): Show that $\mathcal{C} \propto p, p_\mathcal{A}\mapsto q \Rightarrow \mathcal{C}(p)_{\mathcal{C}\circ\mathcal{A}}\mapsto\mathcal{C}(q)$. Then $p_\mathcal{A}\mapsto q$ is the consequent of Rule (*Value*) or one of the rules:

Rule (*Point Origin*): $\maltese_\mathcal{A}\mapsto\mathcal{A}(\langle 0,0,0\rangle)$. $\mathcal{C}(\maltese) = \maltese_{\mathcal{C}\circ\mathcal{A}}\mapsto(\mathcal{C} \circ \mathcal{A})(\langle 0,0,0\rangle)$ (by (*Point Origin*)) $= \mathcal{C}(\mathcal{A}(\langle 0,0,0\rangle))$

Rule (*Point Move*): $v_\mathcal{A}\mapsto w, p'_\mathcal{A}\mapsto q' \Rightarrow v + p'_\mathcal{A}\mapsto w \dotplus q'$. $\mathcal{C}(v+p') = \mathcal{C}(v)+\mathcal{C}(p')_{\mathcal{C}\circ\mathcal{A}}\mapsto\mathcal{C}(w) \dotplus \mathcal{C}(q')$ (by induction and (*Point Move*)) $= \mathcal{C}(w \dotplus q')$ (by Prop. 5.2–4).

Rule (*Frame Shift*)($\sigma = \mathbf{p}$):$M_\mathcal{A}\mapsto\mathcal{B}, p'_{\mathcal{A}\circ\mathcal{B}}\mapsto q \Rightarrow M[p']_\mathcal{A}\mapsto q$. Since $\mathcal{C} \propto M[p']$, we have $\mathcal{C} \propto M$ and $\mathcal{C} \propto p'$. It follows that $\mathcal{C}(M)_{\mathcal{C}\circ\mathcal{A}}\mapsto\mathcal{B}$ and $\mathcal{C}(p')_{\mathcal{C}\circ\mathcal{A}\circ\mathcal{B}}\mapsto\mathcal{C}(q)$ (by induction). Hence $\mathcal{C}(M[p']) = \mathcal{C}(M)[\mathcal{C}(p')]_{\mathcal{C}\circ\mathcal{A}}\mapsto\mathcal{C}(q)$ by (*Frame Shift*).

Case ($\sigma = \mathbf{v}$): Show that $\mathcal{C} \propto v, v_\mathcal{A}\mapsto w \Rightarrow \mathcal{C}(v)_{\mathcal{C}\circ\mathcal{A}}\mapsto\mathcal{C}(w)$. Then $v_\mathcal{A}\mapsto w$ is the consequent of Rule (*Value*) or one of the rules:

Rule (*Vect Unit*): $\uparrow_x {}_\mathcal{A}\mapsto\mathcal{A}(\langle 1,0,0\rangle)$. $\mathcal{C}(\uparrow_x) =\uparrow_x {}_{\mathcal{C}\circ\mathcal{A}}\mapsto(\mathcal{C} \circ \mathcal{A})(\langle 1,0,0\rangle)$ (by (*Vect Unit*)) $= \mathcal{C}(\mathcal{A}(\langle 1,0,0\rangle))$. Similarly for $\uparrow_y$ and $\uparrow_z$.

Rule (*Vect Sub*): $p_\mathcal{A}\mapsto q, p'_\mathcal{A}\mapsto q' \Rightarrow p - p'\mapsto q \dotdiv q'$. $\mathcal{C}(p - p') = \mathcal{C}(p) - \mathcal{C}(p')_{\mathcal{C}\circ\mathcal{A}}\mapsto\mathcal{C}(q) \dotdiv \mathcal{C}(q')$ (by induction and (*Vect Sub*)) $= \mathcal{C}(q \dotdiv q')$ (by Prop. 5.2–4).

Rule (*Vect Scale*): $a_\mathcal{A}\mapsto b, v'_\mathcal{A}\mapsto w' \Rightarrow a \cdot v'_\mathcal{A}\mapsto b \cdot w'$. $\mathcal{C}(a \cdot v') = \mathcal{C}(a) \cdot \mathcal{C}(v')_{\mathcal{C}\circ\mathcal{A}}\mapsto b \cdot \mathcal{C}(w')$ (by induction and (*Vect Scale*)) $= \mathcal{C}(b \cdot w')$ (by Prop. 5.2–4).

Rule (*Vect Add*): $v'_\mathcal{A}\mapsto w', v''_\mathcal{A}\mapsto w'' \Rightarrow v' + v''\mapsto w' + w''$. $\mathcal{C}(v' + v'') = \mathcal{C}(v') + \mathcal{C}(v'')_{\mathcal{C}\circ\mathcal{A}}\mapsto\mathcal{C}(w') + \mathcal{C}(w'')$ (by induction and (*Vect Add*)) $= \mathcal{C}(w' + w'')$ (by Prop. 5.2–4).

Rule (*Vect Cross*): $v'_\mathcal{A}\mapsto w', v''_\mathcal{A}\mapsto w'' \Rightarrow v' \times v''\mapsto w' \times w''$, and $\mathcal{C} \in SE(3)$. $\mathcal{C}(v' \times v'') = \mathcal{C}(v') \times \mathcal{C}(v'')_{\mathcal{C}\circ\mathcal{A}}\mapsto\mathcal{C}(w') \times \mathcal{C}(w'')$ (by induction and (*Vect Cross*)) $= \mathcal{C}(w' \times w'')$ (by Prop. 5.2–4).

Rule (*Frame Shift*)($\sigma = \mathbf{v}$): $M_\mathcal{A}\mapsto\mathcal{B}, v'_{\mathcal{A}\circ\mathcal{B}}\mapsto w \Rightarrow M[v']_\mathcal{A}\mapsto w$. Since $\mathcal{C} \propto M[v']$, we have $\mathcal{C} \propto M$ and $\mathcal{C} \propto v'$. It follows that $\mathcal{C}(M)_{\mathcal{C}\circ\mathcal{A}}\mapsto\mathcal{B}$ and $\mathcal{C}(v')_{\mathcal{C}\circ\mathcal{A}\circ\mathcal{B}}\mapsto\mathcal{C}(w)$ (by induction). Hence $\mathcal{C}(M[v']) = \mathcal{C}(M)[\mathcal{C}(v')]_{\mathcal{C}\circ\mathcal{A}}\mapsto\mathcal{C}(w)$ by (*Frame Shift*).

Case ($\sigma = \mathbf{m}$): Show that $\mathcal{C} \propto M, M_\mathcal{A}\mapsto\mathcal{B} \Rightarrow \mathcal{C}(M)_{\mathcal{C}\circ\mathcal{A}}\mapsto\mathcal{B}$. Then $M_\mathcal{A}\mapsto\mathcal{B}$ is the consequent of Rule (*Value*) or one of the rules:

Rule (*Map Given*): $a_{ij}{}_\mathcal{A}\mapsto b_{ij}, a_{k}{}_\mathcal{A}\mapsto b_k \Rightarrow \langle a_{ij}, a_k\rangle_\mathcal{A}\mapsto\langle b_{ij}, b_k\rangle$, for $i,j,k \in 1..3$ and $det(b_{ij}) \neq 0$. Then $\mathcal{C}(\langle a_{ij}, a_k\rangle) = \langle\mathcal{C}(a_{ij}), \mathcal{C}(a_k)\rangle_{\mathcal{C}\circ\mathcal{A}}\mapsto\langle b_{ij}, b_k\rangle$ (by induction and (*Map Given*))

Rule (*Map Comp*): $M'_\mathcal{A}\mapsto\mathcal{B}', M''_\mathcal{A}\mapsto\mathcal{B}'' \Rightarrow M' \circ M''_\mathcal{A}\mapsto\mathcal{B}' \circ \mathcal{B}''$. We have $\mathcal{C}(M' \circ M'') = \mathcal{C}(M') \circ \mathcal{C}(M'')_{\mathcal{C}\circ\mathcal{A}}\mapsto\mathcal{B}' \circ \mathcal{B}''$ (by induction and (*Map Comp*)).

Rule (*Map Inv*): $M'_\mathcal{A}\mapsto\mathcal{B}' \Rightarrow M'^{-1}{}_\mathcal{A}\mapsto\mathcal{B}'^{-1}$. We have $\mathcal{C}(M'^{-1}) = \mathcal{C}(M')^{-1}{}_{\mathcal{C}\circ\mathcal{A}}\mapsto\mathcal{B}'^{-1}$ (by induction and (*Map Inv*)).

Rule (*Frame Shift*)($\sigma = \mathbf{m}$): $M'_\mathcal{A}\mapsto D, M''_{\mathcal{A}\circ D}\mapsto\mathcal{B} \Rightarrow M'[M'']_\mathcal{A}\mapsto\mathcal{B}$. Since $\mathcal{C} \propto M'[M'']$, we have $\mathcal{C} \propto M'$ and $\mathcal{C} \propto M''$. It follows that $\mathcal{C}(M')_{\mathcal{C}\circ\mathcal{A}}\mapsto D$ and $\mathcal{C}(M'')_{\mathcal{C}\circ\mathcal{A}\circ D}\mapsto\mathcal{B}$ (by induction). Hence $\mathcal{C}(M'[M'']) = \mathcal{C}(M')[\mathcal{C}(M'')]_{\mathcal{C}\circ\mathcal{A}}\mapsto\mathcal{B}$ by (*Frame Shift*). $\qquad\square$

We now give a local frame shift result on processes that is the exact analog of the $(Frame\ Shift)$ rule on data given in Definition 20. This result uses all the $(\equiv Map...)$ rules in the structural congruence relation, except for the $(\equiv Map\ Comp)$ rule. The result depends on data computation only in using the $(Frame\ Shift)$ and $(Map\ Comp)$ rules. It would therefore hold for any data sublanguages and data computation rules which were compatible with these rules. Recall that process reduction, $P_\mathcal{A} \to Q$, was introduced in Definition 5.

Theorem 23. *Local Frame Shift*
  $M_\mathcal{A} \mapsto \mathcal{B}, P_{\mathcal{A} \circ \mathcal{B}} \to Q \Rightarrow M[P]_\mathcal{A} \to M[Q]$

Proof. The proof is by induction on the derivation of P $_{\mathcal{A} \circ \mathcal{B}} \to$ Q.

Rule $(Red\ Comm)$: $\Delta_{\mathcal{A} \circ \mathcal{B}} \mapsto \varepsilon \Rightarrow !_\sigma x(\Delta).P' + P'' \mid ?_\sigma x(y).Q' + Q''_{\mathcal{A} \circ \mathcal{B}} \to P' \mid Q'\{y \backslash \varepsilon\}$. From $M_\mathcal{A} \mapsto \mathcal{B}$, we obtain $M[\Delta]_\mathcal{A} \mapsto \varepsilon$ by $(Frame\ Shift)$. By $(Red\ Comm)$ we than have: $!_\sigma x(M[\Delta]).M[P']$ $+M[P''] \mid ?_\sigma x(y).M[Q'] + M[Q'']_\mathcal{A} \to M[P'] \mid M[Q']\{y \backslash \varepsilon\}$. Since $M_\mathcal{A} \mapsto \mathcal{B}$, we know that M is closed. Hence, for any variable y, we have $M[Q']\{y \backslash \varepsilon\} = M[Q'\{y \backslash \varepsilon\}]$. Therefore, $M[!_\sigma x(\Delta).P' + P'' \mid$ $?_\sigma x(y).Q' + Q'']_\mathcal{A} \to M[P' \mid Q'\{y \backslash \varepsilon\}]$ by $(\equiv Map\ Sum)$, $(\equiv Map\ Out)$, $(\equiv Map\ In)$, $(\equiv Map\ Par)$ and $(Red \equiv)$.

Rule $(Red\ Cmp)$: $\Delta_{\mathcal{A} \circ B} \curlyvee \Delta' \Rightarrow \Delta =_\sigma \Delta'.P'_{\mathcal{A} \circ \mathcal{B}} \to P'$. Since $M_\mathcal{A} \mapsto \mathcal{B}$, we have $M[\Delta]_\mathcal{A} \curlyvee M[\Delta']$ by $(Frame\ Shift)$, so from $(Red\ Cmp)$ we obtain $M[\Delta] =_\sigma M[\Delta'].M[P']_\mathcal{A} \to M[P']$. Therefore $M[\Delta =_\sigma \Delta'.P']_\mathcal{A} \to M[P']$ by $(\equiv Map\ Cmp)$ and $(Red \equiv)$.

Rule $(Red\ Par)$: $P'_{\mathcal{A} \circ \mathcal{B}} \to Q' \Rightarrow P' \mid R_{\mathcal{A} \circ \mathcal{B}} \to Q' \mid R$. By induction $M[P']_\mathcal{A} \to M[Q']$, hence $M[P'] \mid M[R]_\mathcal{A} \to M[Q'] \mid M[R]$ by $(Red\ Par)$ and $M[P' \mid R]_\mathcal{A} \to M[Q' \mid R]$ by $(\equiv Map\ Par)$ and $(Red \equiv)$.

Rule $(Red\ Res)$: $P'_{\mathcal{A} \circ \mathcal{B}} \to Q' \Rightarrow (\nu x)P'_{\mathcal{A} \circ \mathcal{B}} \to (\nu x)Q'$. By induction $M[P']_\mathcal{A} \to M[Q']$, hence $(\nu x)M[P']_\mathcal{A} \to (\nu x)M[Q']$ by $(Red\ Res)$ and $M[(\nu x)P']_\mathcal{A} \to M[(\nu x)Q']$ by $(\equiv Map\ Res)$ and $(Red \equiv)$.

Rule $(Red \equiv)$: $P \equiv P', P'_{\mathcal{A} \circ \mathcal{B}} \to Q', Q' \equiv Q \Rightarrow P_{\mathcal{A} \circ \mathcal{B}} \to Q$. By $(\equiv Map)$, $M[P] \equiv M[P']$ and $M[Q'] \equiv M[Q]$. By induction $M[P']_\mathcal{A} \to M[Q']$. Hence $M[P]_\mathcal{A} \to M[Q]$ by $(Red \equiv)$. $\qquad\square$

The $(\equiv Map\ Comp)$ rule is not used in the proof of the theorem. This indicates that we might restrict ourselves to a Dpi style calculus without the nesting of frames. In our nested calculus, the derived reduction for nested process frame, using Theorem 23 twice, is:

  $M_\mathcal{A} \mapsto \mathcal{B}, N_{\mathcal{A} \circ \mathcal{B}} \mapsto \mathcal{C}, P_{\mathcal{A} \circ \mathcal{B} \circ \mathcal{C}} \to Q \Rightarrow N[P]_{\mathcal{A} \circ \mathcal{B}} \to N[Q] \Rightarrow M[N[P]]_\mathcal{A} \to M[N[Q]]$

In a non-nested calculus, we could emulate this reduction, from the same assumptions, by:

  $M_\mathcal{A} \mapsto \mathcal{B}, N_{\mathcal{A} \circ \mathcal{B}} \mapsto \mathcal{C}, P_{\mathcal{A} \circ \mathcal{B} \circ \mathcal{C}} \to Q \Rightarrow M[N]_\mathcal{A} \mapsto \mathcal{C}$
  $\Rightarrow M \circ M[N]_\mathcal{A} \mapsto \mathcal{B} \circ \mathcal{C} \Rightarrow (M \circ M[N])[P]_\mathcal{A} \to (M \circ M[N])[Q]$

using $(Frame\ Shift)$, $(Map\ Comp)$ and Theorem 23. In other words, if we had neither $(\equiv Map\ Comp)$ nor nested process frames, we could still emulate $M[N[P]] by (M \circ M[N])[P]$. But with 3 nested process frames, we end up with 3 nested frames on the maps. Hence we would still need to handle nested frames at least on the data.

We show that we can shift process reductions to different frames. A shifted process does not reduce to exactly the same process as in the original version, e.g. changing from Q to $\mathcal{C}(Q)$, but those differences have no effect on process traces (under the usual $\propto$ assumptions). That is, differences due to value substitutions in different frames can then cancel out because data comparisons remove the values from the terms. The $\propto$ relation extends to processes in the obvious way: $\mathcal{C} \propto P$ holds if

and only if $\mathcal{C} \propto \Delta$ holds for all data subterms Δ of P, where $\mathcal{C} \propto \Delta$ is given in Definition 10. $\mathcal{C}(P)$ is the process obtained by replacing all the value subterms ε of P with $\mathcal{C}(\varepsilon)$.

Lemma 24. *Congruence Mapping*
$$P \equiv Q \;\Rightarrow\; \mathcal{C}(P) \equiv \mathcal{C}(Q)$$

Proof. The proof is by induction on the derivation of P ≡ Q. The interesting rules are the (≡ Map ...) rules; we look at two of them.

Rule (≡ *Map*): $P' \equiv Q' \;\Rightarrow\; M[P'] \equiv M[Q']$. By induction $\mathcal{C}(P') \equiv \mathcal{C}(Q')$, hence $\mathcal{C}(M)[\mathcal{C}(P')] \equiv \mathcal{C}(M)[\mathcal{C}(Q')]$ by (≡ *Map*), that is $\mathcal{C}(M[P']) \equiv \mathcal{C}(M[Q'])$.

Rule (≡ *Map In*): $M[?_\sigma x(y).P'] \equiv ?_\sigma x(y).M[P']$ $(y \notin fv(M))$. Then $y \notin fv(\mathcal{C}(M))$, and we have $\mathcal{C}(M[?_\sigma x(y).P']) = \mathcal{C}(M)[?_\sigma x(y).\mathcal{C}(P')] \equiv ?_\sigma x(y).\mathcal{C}(M)[\mathcal{C}(P')] = \mathcal{C}(?_\sigma x(y).M[P'])$ by (≡ *Map In*). $\qquad \square$

The $\propto$ relation is extended to the process syntax in the obvious way: $\mathcal{A} \propto P$ holds if $\mathcal{A} \propto \Delta$ holds for all data subterms Δ of P, where $\mathcal{A} \propto \Delta$ is given in Definition 4.1–2.

Lemma 25. $P \equiv Q \;\Rightarrow\; (\mathcal{A} \propto P \;\Leftrightarrow\; \mathcal{A} \propto Q)$

Proof. The proof is by induction on the derivation of the derivation of $P \equiv Q$.

Rule(≡ *Symm*): $Q \equiv P \;\Rightarrow\; P \equiv Q$. Then by induction we have that $Q \equiv P \;\Rightarrow\; (\mathcal{A} \propto Q \;\Leftrightarrow\; \mathcal{A} \propto P)$ and hence $\mathcal{A} \propto P \;\Leftrightarrow\; \mathcal{A} \propto Q$.

Rule(≡ *Map*): $P \equiv Q \;\Rightarrow\; M[P] \equiv M[Q]$. Then by induction we have$(\mathcal{A} \propto P \;\Leftrightarrow\; \mathcal{A} \propto Q)$, hence $(\mathcal{A} \propto M[P] \;\Leftrightarrow\; \mathcal{A} \propto M[Q])$.

The other cases are routine because of the same data subterms on both sides. $\qquad \square$

Lemma 26. $\mathcal{B} \propto P, P_\mathcal{A} \rightarrow Q \;\Rightarrow\; \mathcal{B} \propto Q$

Proof. Reduction does not introduce new subterms, except for (*Red Comm*) where the result follows from $\mathcal{B} \propto \varepsilon$ and $\mathcal{B} \propto Q \;\Rightarrow\; \mathcal{B} \propto Q\{y\backslash\varepsilon\}$, and for (*Red* ≡) where the result follows from Lemma 25. $\qquad \square$

To motivate the theorem, assume the data computation $\Delta_\mathcal{A} \mapsto \varepsilon$ which, by (*Red Comm*), implies the process reduction:

$$!c(\Delta) \mid ?c(x).x = \varepsilon'_\mathcal{A} \rightarrow \varepsilon = \varepsilon'$$

Also assume $\mathcal{C} \propto \Delta$, so we have $\mathcal{C}(\Delta)_{\mathcal{C} \circ \mathcal{A}} \mapsto \mathcal{C}(\varepsilon')$ by Theorem 22. Hence by (*Red Comm*):

$$!c(\mathcal{C}(\Delta)) \mid ?c(x).x = \mathcal{C}(\varepsilon')_{\mathcal{C} \circ \mathcal{A}} \rightarrow \mathcal{C}(\varepsilon) = \mathcal{C}(\varepsilon')$$

and since $\mathcal{C}(!c(\Delta) \mid ?c(x).x = \varepsilon') = !c(\mathcal{C}(\Delta)) \mid ?c(x).x = \mathcal{C}(\varepsilon')$ and $\mathcal{C}(\varepsilon = \varepsilon') = \mathcal{C}(\varepsilon) = \mathcal{C}(\varepsilon')$, we have:

$$\mathcal{C}(!c(\Delta) \mid ?c(x).x = \varepsilon')_{\mathcal{C} \circ \mathcal{A}} \rightarrow \mathcal{C}(\varepsilon = \varepsilon')$$

For this example we have shown that $P_\mathcal{A} \rightarrow Q \Rightarrow \mathcal{C}(P)_{\mathcal{C} \circ \mathcal{A}} \rightarrow \mathcal{C}(Q)$. Although P has to be replaced by $\mathcal{C}(P)$ in the shifted frame, the process shape P remains unchanged up to the embedded values. Moreover the change does not affect data comparisons in that, if the comparison $\varepsilon = \varepsilon'$ succeeds in $\mathcal{A}$, then the comparison $\mathcal{C}(\varepsilon) = \mathcal{C}(\varepsilon')$ succeeds in $\mathcal{C} \circ \mathcal{A}$. This example suggests the statement of the following theorem.

Theorem 27. *Global Frame Shift for Processes*
$\mathcal{C} \propto P, P_\mathcal{A} \to Q \Rightarrow \mathcal{C}(P)_{\mathcal{C} \circ \mathcal{A}} \to \mathcal{C}(Q)$

Proof. The proof is by induction on the derivation of $P_\mathcal{A} \to Q$.

Rule (Red Comm): $\Delta_\mathcal{A} \mapsto \varepsilon \Rightarrow !_\sigma x(\Delta).P' + P'' \mid ?_\sigma x(y).Q' + Q''_\mathcal{A} \to P'|Q'\{y \backslash \varepsilon\}$, $\mathcal{C} \propto l.h.s..$ By Theorem 22, $\mathcal{C} \propto P', \Delta_\mathcal{A} \mapsto \varepsilon \Rightarrow \mathcal{C}(\Delta)_{\mathcal{C} \circ \mathcal{A}} \mapsto \mathcal{C}(\varepsilon)$. Hence, we can produce the following instance of (*Red Comm*): $!_\sigma x(\mathcal{C}(\Delta)).\mathcal{C}(P') + \mathcal{C}(P'') \mid ?_\sigma x(y).\mathcal{C}(Q') + \mathcal{C}(Q'')_{C \circ \mathcal{A}} \to \mathcal{C}(P')|\mathcal{C}(Q')\{y \backslash \mathcal{C}(\varepsilon)\}$. Since $\mathcal{C}(Q')\{y \backslash \mathcal{C}(\varepsilon)\} = \mathcal{C}(Q'\{y \backslash \varepsilon\})$, it follows that $\mathcal{C}(!_\sigma x(\Delta).P' + P'' \mid ?_\sigma x(y).Q' + Q'')_{C \circ \mathcal{A}} \to \mathcal{C}(P' \mid Q'\{y \backslash \varepsilon\})$.

Rule (Red Cmp): $\Delta_\mathcal{A} \curlyvee \Delta' \Rightarrow (\Delta =_\sigma \Delta'.Q)_\mathcal{A} \to Q$, with $\mathcal{C} \propto (\Delta =_\sigma \Delta'.Q)$. By Theorem 22, since $\mathcal{C} \propto \Delta =_\sigma \Delta'$ and $\exists \varepsilon.\Delta_\mathcal{A} \mapsto \varepsilon$ and $\Delta'_\mathcal{A} \mapsto \varepsilon$, we have that $\exists \varepsilon' = \mathcal{C}(\varepsilon).\mathcal{C}(\Delta)_{\mathcal{C} \circ \mathcal{A}} \mapsto \varepsilon'$ and $\mathcal{C}(\Delta')_{\mathcal{C} \circ \mathcal{A}} \mapsto \varepsilon'$; hence $\mathcal{C}(\Delta)_{\mathcal{C} \circ \mathcal{A}} \curlyvee \mathcal{C}(\Delta')$. Therefore, by (*Red Cmp*) we obtain $\mathcal{C}(\Delta) =_\sigma \mathcal{C}(\Delta').\mathcal{C}(Q)_{\mathcal{C} \circ \mathcal{A}} \to \mathcal{C}(Q)$. It follows that $\mathcal{C}(\Delta =_\sigma \Delta'.Q)_{\mathcal{C} \circ \mathcal{A}} \to \mathcal{C}(Q)$.

Rule (Red Par): $P'_\mathcal{A} \to Q' \Rightarrow P' \mid R_\mathcal{A} \to Q' \mid R$, with $\mathcal{C} \propto P' \mid R$. By induction, since $\mathcal{C} \propto P'$, we have $\mathcal{C}(P')_{C \circ \mathcal{A}} \to \mathcal{C}(Q')$. Hence by (*Red Par*), $\mathcal{C}(P') \mid \mathcal{C}(R)_{C \circ \mathcal{A}} \to \mathcal{C}(Q') \mid \mathcal{C}(R)$, that is, $\mathcal{C}(P' \mid R)_{\mathcal{C} \circ \mathcal{A}} \to \mathcal{C}(Q' \mid R)$.

Rule (Red Res): $P'_\mathcal{A} \to Q' \Rightarrow (\nu x)P'_\mathcal{A} \to (\nu x)Q'$, with $\mathcal{C} \propto (\nu x)P'$. By induction, since $\mathcal{C} \propto P'$, we have $\mathcal{C}(P')_{\mathcal{C} \circ \mathcal{A}} \to \mathcal{C}(Q')$. Hence by (*Red Res*) $(\nu x)\mathcal{C}(P')_{\mathcal{C} \circ \mathcal{A}} \to (\nu x)\mathcal{C}(Q')$, that is, $\mathcal{C}((\nu x)P')_{\mathcal{C} \circ \mathcal{A}} \to \mathcal{C}((\nu x)Q')$.

Rule (Red $\equiv$): $P \equiv P', P'_\mathcal{A} \to Q', Q' \equiv Q \Rightarrow P_\mathcal{A} \to Q$, with $\mathcal{C} \propto P$. By Lemma 25, we have $\mathcal{C} \propto P, P \equiv P' \Rightarrow \mathcal{C} \propto P'$. By induction, we have $\mathcal{C} \propto P', P'_\mathcal{A} \to Q' \Rightarrow \mathcal{C}(P')_{\mathcal{C} \circ \mathcal{A}} \to \mathcal{C}(Q')$. By Lemma 24, we have $\mathcal{C}(P) \equiv \mathcal{C}(P')$ and $\mathcal{C}(Q') \equiv \mathcal{C}(Q)$. Hence, $\mathcal{C}(P)_{\mathcal{C} \circ \mathcal{A}} \to \mathcal{C}(Q)$ by (*Red $\equiv$*). □

The following theorem establishes that barbed congruence is preserved under frame shift.

Theorem 28. *Global Frame Shift for Barbed Congruence*
$\mathcal{C} \propto P, Q, P_\mathcal{A} \approx Q \Rightarrow \mathcal{C}(P)_{\mathcal{C} \circ \mathcal{A}} \approx \mathcal{C}(Q)$

Proof. Proof Consider the relation $\mathcal{R} = \{\langle \mathcal{C}(P), \mathcal{C}(Q) \rangle \mid P_\mathcal{A} \approx Q\}$. We show that $\mathcal{R}$ is an $(\mathcal{A} \circ \mathcal{C})$-candidate relation. The statement then follows since if $P_\mathcal{A} \approx Q$ then $\mathcal{C}(P)\mathcal{R}\mathcal{C}(Q)$ and $\mathcal{C}(P)_{\mathcal{A} \circ \mathcal{C}} \approx \mathcal{C}(Q)$. Fact: $P \downarrow_x$ if and only if $\mathcal{C}(P) \downarrow_x$.

1) Consider any $\langle \mathcal{C}(P), \mathcal{C}(Q) \rangle$ in $\mathcal{R}$ with $P_\mathcal{A} \approx Q$. If $\mathcal{C}(P) \downarrow_x$ then $P \downarrow_x$. Since $P_\mathcal{A} \approx Q$ and $P \downarrow_x$, we have $Q_\mathcal{A} \Downarrow_x$; that is, $\exists Q'.Q_\mathcal{A} \to^* Q' \wedge Q' \downarrow_x$. By Theorem 27 and Lemma 26 we have $\mathcal{C}(Q)_{\mathcal{C} \circ \mathcal{A}} \to^* \mathcal{C}(Q')$. Moreover $Q' \downarrow_x$ implies $\mathcal{C}(Q') \downarrow_x$, and hence $\mathcal{C}(Q)_{\mathcal{C} \circ \mathcal{A}} \Downarrow_x$. The converse is similar.

2) Consider any $\langle \mathcal{C}(P), \mathcal{C}(Q) \rangle$ in $\mathcal{R}$ with $P_\mathcal{A} \approx Q$. If $\mathcal{C}(P)_{\mathcal{C} \circ \mathcal{A}} \to P''$ then, by Theorem 27, $\mathcal{C}^{-1}(\mathcal{C}(P))_{\mathcal{C}^{-1} \circ \mathcal{C} \circ \mathcal{A}} \to \mathcal{C}^{-1}(P'')$; that is, $P_\mathcal{A} \to P' = \mathcal{C}^{-1}(P'')$. Since $P_\mathcal{A} \approx Q$, there is Q' such that $Q_\mathcal{A} \to^* Q'$ and $P'_\mathcal{A} \approx Q'$. Hence, by Theorem 27, there is $Q'' = C(Q')$ such that $C(Q)_{\mathcal{C} \circ \mathcal{A}} \to^* Q''$. Rewrite $P'_\mathcal{A} \approx Q'$ as $\mathcal{C}^{-1}(P'')_\mathcal{A} \approx \mathcal{C}^{-1}(Q'')$; then, by definition of $\mathcal{R}$, $\mathcal{C}(\mathcal{C}^{-1}(P'')) \mathcal{R} \mathcal{C}(\mathcal{C}^{-1}(Q''))$; that is, $P''\mathcal{R}Q''$. We have shown that if $\mathcal{C}(P)_{\mathcal{C} \circ \mathcal{A}} \to P''$ then there is Q'' such that $\mathcal{C}(Q)_{\mathcal{C} \circ \mathcal{A}} \to^* Q''$ and $P''\mathcal{R}Q''$. The converse is similar.

3) Consider any $\langle \mathcal{C}(P), \mathcal{C}(Q) \rangle$ in $\mathcal{R}$ with $P_\mathcal{A} \approx Q$. For any observation context Γ, $\mathcal{C}^{-1}(\Gamma)$ is an observation context, and hence we have that $\mathcal{C}^{-1}(\Gamma)[P]_\mathcal{A} \approx \mathcal{C}^{-1}(\Gamma)[Q]$. By definition of $\mathcal{R}$, we then have that $\mathcal{C}(\mathcal{C}^{-1}(\Gamma)[P]) \mathcal{R} \mathcal{C}(\mathcal{C}^{-1}(\Gamma)[Q])$, that is $\Gamma[\mathcal{C}(P)] \mathcal{R} \Gamma[\mathcal{C}(Q)]$. □

Theorem 29. *Relativity (Theorem 13)*
G-equations are G-invariant, and hence invariant across G.

Proof. Take $\mathcal{A} \in GA(3)$ and $\mathcal{B} \in G \subseteq GA(3)$, and assume that $P^{\triangledown} = Q^{\triangledown}$ is a law in $\mathcal{A}$, that is, $P_{\mathcal{A}}^{\triangledown} \approx Q^{\triangledown}$. By Theorem 28, since $\mathcal{B} \propto P^{\triangledown}, Q^{\triangledown}$, we have $\mathcal{B}(P^{\triangledown})_{\mathcal{B} \circ \mathcal{A}} \approx \mathcal{B}(Q^{\triangledown})$. But $P^{\triangledown}, Q^{\triangledown}$ are pure, so we obtain $P_{\mathcal{B} \circ \mathcal{A}}^{\triangledown} \approx Q^{\triangledown}$ and hence $P^{\triangledown} = Q^{\triangledown}$ is a law in $\mathcal{B} \circ \mathcal{A}$. Conversely, assume $P^{\triangledown} = Q^{\triangledown}$ is a law in $\mathcal{B} \circ \mathcal{A}$, that is $P_{\mathcal{B} \circ \mathcal{A}}^{\triangledown} \approx Q^{\triangledown}$. By Theorem 28, since $\mathcal{B}^{-1} \propto P^{\triangledown}, Q^{\triangledown}$, we have $\mathcal{B}^{-1}(P^{\triangledown})_{\mathcal{B}^{-1} \circ \mathcal{B} \circ \mathcal{A}} \approx \mathcal{B}^{-1}(Q^{\triangledown})$. Again, $P_{\mathcal{A}}^{\triangledown} \approx Q^{\triangledown}$, and $P^{\triangledown} = Q^{\triangledown}$ is a law in $\mathcal{A}$. We have shown that G-equations are G-invariant. Assume $P^{\triangledown} = Q^{\triangledown}$ is a G-equation, and hence G-invariant, and take $\mathcal{A}, \mathcal{B} \in G$. If $P^{\triangledown} = Q^{\triangledown}$ is a law in $\mathcal{A}$ then, since $\mathcal{B} \circ \mathcal{A}^{-1} \in G$, it is also a law in $\mathcal{B} \circ \mathcal{A}^{-1} \circ \mathcal{A}$ by definition of G-invariance, and hence it is a law in $\mathcal{B}$. We have shown that G-equations are invariant across G. $\qquad\square$