Available online at www.sciencedirect.com

SciVerse ScienceDirect

Electronic Notes in Theoretical Computer Science

Electronic Notes in Theoretical Computer Science 284 (2012) 23–34

www.elsevier.com/locate/entcs

Towards Abstraction-Based Verification of Shape Calculus

F. Buti, M. Callisto De Donato, F. Corradini, M.R. Di Berardini, E. Merelli, L. Tesei[1]

School of Science and Technology, Computer Science Division
University of Camerino
Via Madonna delle Carceri 9, 62032 Camerino, Italy

Abstract

The Shape Calculus is a bio-inspired timed and spatial calculus for describing 3D geometrical shapes moving in a space. Shapes, combined with a behaviour, form 3D processes, i.e., individual entities able to bind with other processes on compatible spatial channels and to split over previously established bonds. Due to geometrical space, timed behaviours, a wide degree of freedom in defining motion laws and usual non-determinism, 3D processes typically exhibits an infinite behaviour that prevents any decidable analysis. Shape Calculus models are currently used only for simulation and, thus, validation of models and hypothesis testing. In this work we introduce a complementary, and synergetic, way of using the calculus for systems biology purposes: we define a first abstract interpretation that can be used to verify untimed and unspatial safety properties of a given model. Such an abstraction focuses on the possible interactions that, during the evolution of the system, can occur among processes yielding new composed processes and, thus, new species. Other possible abstract domains for the verification of more expressive properties are also discussed.

Keywords: Abstract interpretation, Process algebra, Spatiality, Systems Biology

1 Introduction

In the context of the challenges raised by Systems Biology to several disciplines, computer scientists, among others, have started to contribute trying to adapt models and languages designed originally for the design and the analysis of hardware/software systems to biological systems. This adaptation process has revealed that some of the languages, although general-purpose, needed to be expanded with concepts and characteristics typical of biological modelling. One of these features is surely space, considered both in a topological and a geometrical way. For instance in [3], authors outline a modelling and simulation approach which covers not only the simulation of individual entities moving in space but also the stochastic spatial simulation at the

[1] Email: {federico.buti,massimo.callisto,flavio.corradini,mariarita.diberardini,emanuela.merelli,luca.tesei}@unicam.it

1571-0661 © 2012 Elsevier B.V. Open access under CC BY-NC-ND license.
doi:10.1016/j.entcs.2012.05.013

population level and the combination of the two. In [11], instead, the rule-based approach is extended to take in account position, orientation and geometric structure of molecules in combinatorially complex chemical reaction systems.

In this context, the Shape Calculus [2,1] was proposed as a very rich language to describe mainly, but not only, biological phenomena. The main characteristics of this calculus are that it is spatial - with a geometric notion of a 3D space - and it is shape-based, i.e., entities have geometric simple or complex shapes that affect the possible interactions with other entities. In the Shape Calculus we consider *3D processes* consisting of entities with a 3D shape and a dynamic behaviour, situated in a 3D virtual environment. 3D processes move accordingly to a personalized (to each process) motion law, collide and possibly bind each other and compound new 3D processes. Thus, a network of interacting 3D processes typically exhibits *infinite* behaviours. On the one hand the expressive power of the calculus is very high and it embeds natively features that are typical of biomodels. On the other hand, this richness and infiniteness prevents any application of analysis techniques existing for untimed and/or unspatial systems. Currently, Shape Calculus models are used as a base for a related simulation environment, called BioShape [5], that is used for model definition, simulation and validation [4,7] as well as for hypothesis testing and uniform multi-scale simulations [6].

Orthogonal to simulation-based analysis techniques, a more formal, and possibly synergetic with simulation, approach to the study of Shape Calculus models is that of formal verification. In this work we investigate the application of the *abstract interpretation* framework [9,8] on the Shape Calculus to reduce the complexity of dynamics and, thus, to obtain the decidability of verification of properties. As a first step, we consider an abstract domain in which time, movements and space are abstracted in order to focus only on all possible bounds and splits that can occur among processes. This permits to prove, by performing a finite fixpoint iteration, untimed and unspatial safety properties such as "a species formed by the binding of glucose and ATP can never be generated in the evolution of the system".

The paper is structured as follows: Section 2 introduces the main concepts of the Shape Calculus, Section 3 describes the proposed abstract interpretation, finally Section 4 discusses other possible abstractions to verify more expressive properties.

2 Shape Calculus: a calculus for moving shapes

Let $\mathbb{P}, \mathbb{V} = \mathbb{R}^3$ be the sets of positions and velocities, resp., in a *global* three dimensional coordinate system. We also assume relative coordinate systems, the *local* coordinate system, that will always be w.r.t. a certain shape S with origin in a reference point **p** (the *centre* of S). The local coordinate system allows us to express parts of the shape independently from its actual global position. Given $p \in \mathbb{P}$ expressed in global coordinates and $V \subseteq \mathbb{P}$ a set of points expressed in a local coordinate system whose origin is p, the function $\mathsf{global}(\mathrm{V}, \mathrm{p}) = V + p = \{x + p \mid x \in V\}$ denotes V w.r.t. the global coordinates.

2.1 Shapes

Any 3D shape can be approximated - with arbitrary precision - by composing *basic shapes* i.e. "glueing" shapes on common surface. Basic shapes can be spheres, cones, cylinders or convex polyhedra.

Definition 2.1 (3D shapes) The set $\mathbb{S}$ of *3D shapes*, ranged over by $S, S', \cdots$, is generated by the grammar: $S ::= \sigma \mid S \langle X \rangle S$ where σ is a **basic shape**. A basic shape is defined by the tuple $\sigma = \langle V, m, \mathbf{p}, \mathbf{v} \rangle$ where $V \subseteq \mathbb{P}$, $m \in \mathbb{R}^+$ is the mass, $p \in \mathbb{P}$ the centre of mass and $v \in \mathbb{V}$ the velocity of σ. If $S = \sigma = \langle V, m, \mathbf{p}, \mathbf{v} \rangle$, we define $\mathcal{P}(S) = V$, $m(S) = m$, $\mathcal{R}(S) = \mathbf{p}$, $\mathbf{v}(S) = \{\mathbf{v}\}$ to be, resp., the set of points, mass, reference point and velocity of σ. $\mathcal{B}(S) \subset V$ is the set of points on the surface of σ. $S = S_1 \langle X \rangle S_2$ is a **compound shape** [2] where $X \subseteq \mathbb{P}$. We let $\mathcal{P}(S) = \mathcal{P}(S_1) \cup \mathcal{P}(S_2)$, $m(S) = m(S_1) + m(S_2)$, $\mathcal{R}(S) = \big(m(S_1) \cdot \mathcal{R}(S_1) + m(S_2) \cdot \mathcal{R}(S_2)\big) / \big(m(S_1) + m(S_2)\big)$ [3] and $\mathbf{v}(S) = \mathbf{v}(S_1) \cup \mathbf{v}(S_2)$ [4]. Finally we define the *boundary* of a compound shape S as the set of global points $\mathcal{B}(S) = (\mathcal{B}(S_1) \cup \mathcal{B}(S_2)) \backslash \{x \in \mathbb{P} \mid x \text{ is interior of } \mathcal{P}(S_1) \langle X \rangle \mathcal{P}(S_2)\}$.

Continuous trajectories of shapes are approximated with a polygonal chain [10] and velocities are updated on the vertices of the chain, instead of continuously update them. Let $\mathbb{T} \in \mathbb{R}_0^+$ be the time domain. We divide $\mathbb{T}$ into an infinite sequence of time steps t_i s.t. $t_0 = 0$ and $t_i \leq t_{i-1} + \Delta$ for all $i > 0$, where Δ is called *movement time step* and depends on the desired degree of approximation. The updating of velocities is performed by exploiting a function $\mathsf{steer} : \mathbb{T} \to (\mathbb{S} \hookrightarrow \mathbb{V})$ [5] that describes how the velocity of all existing shapes, at each time t, is changed. Both velocity update and evolution of shapes are represented as an update of the shape tuples.

In some situations, the duration of a time step can be shorter than Δ since collisions can occur before the end of the time step. These collisions must be resolved and the whole system must re-adapt itself to the new situation through a collision response mechanism (see [2] for more details).

2.2 Behaviour of shapes

The *internal behaviour* of a shape is described as a variant of TCCS [12] where basic actions provide information about binding capabilities and split possibilities. Let $\Lambda = \{a, b, \cdots\}$ be a countably infinite set of *channels names* and $\overline{\Lambda} = \{\overline{a} \mid a \in \Lambda\}$ the corresponding *co-channels names* with $\overline{\overline{a}} = a$ for each a. Elements in $\mathcal{A} = \Lambda \cup \overline{\Lambda}$ are ranged over by $\alpha, \beta, \cdots$.

Binding capabilities are represented by *channels*, i.e. pairs $\langle \alpha, X \rangle$ where $\alpha \in \mathcal{A}$ is a name and X is a *surface of contact*. Intuitively, X is a subset of the boundary of a shape where the channel is active and, thus, bindings are enabled on it. Names introduce a notion of compatibility: if $\beta = \overline{\alpha}$ and $X \cap Y \neq \emptyset$ then $\langle \alpha, X \rangle$ and $\langle \beta, X \rangle$

[2] In this paper we consider only compound shapes that are well-formed according to [1].

[3] Again for simplicity, we use the centre of mass as the reference point.

[4] Well-formed shapes must have a singleton as set of velocities.

[5] Given a time instant $t \in \mathbb{T}$, $\mathsf{steer}\, t\, S$ is undefined iff shape S does not exist at time t.

$$\textsc{Pref}_a \frac{\mu \in \mathcal{C} \cup \omega(\mathcal{C})}{\mu.B \xrightarrow{\mu} B} \qquad \textsc{Del}_a \frac{B \xrightarrow{\mu} B'}{\epsilon(0).B \xrightarrow{\mu} B'} \qquad \textsc{Sum}_a \frac{B_1 \xrightarrow{\mu} B_1'}{B_1 + B_2 \xrightarrow{\mu} B_1'}$$

$$\textsc{Str}_1 \frac{L = \{\langle \alpha, X\rangle\}}{\rho(L).B \xrightarrow{\rho(\alpha,X)} B} \qquad \textsc{Str}_2 \frac{L = \{\langle \alpha, X\rangle\} \cup L' \;\; L' \neq \emptyset}{\rho(L).B \xrightarrow{\rho(\alpha,X)} \rho(L').B} \qquad \textsc{Str}_3 \frac{B \xrightarrow{\rho(\alpha,X)} B'}{\rho(L).B \xrightarrow{\rho(\alpha,X)} \rho(L).B'}$$

$$\textsc{Nil}_t \frac{}{\mathsf{nil} \overset{t}{\rightsquigarrow} \mathsf{nil}} \qquad \textsc{Pref}_t \frac{\mu \in \mathcal{C} \cup \omega(\mathcal{C})}{\mu.B \overset{t}{\rightsquigarrow} \mu.B} \qquad \textsc{Str}_t \frac{}{\rho(L).B \overset{t}{\rightsquigarrow} \rho(L).B}$$

$$\textsc{Sum}_t \frac{B_1 \overset{t}{\rightsquigarrow} B_1' \quad B_2 \overset{t}{\rightsquigarrow} B_2'}{B_1 + B_2 \overset{t}{\rightsquigarrow} B_1' + B_2'} \qquad \textsc{Del}_t \frac{t' \geq t}{\epsilon(t').B \overset{t}{\rightsquigarrow} \epsilon(t' - t).B} \qquad \textsc{Del}_c \frac{B \overset{t}{\rightsquigarrow} B'}{\epsilon(t').B \overset{t+t'}{\rightsquigarrow} B'}$$

Table 1
Functional and Temporal behaviour of $\mathbb{B}$'s terms

are *compatible*, otherwise they are *incompatible*. Compatibility is used to determine if a collision between two shapes is elastic (channels not compatible) or inelastic (otherwise).

We also introduce two different kinds of actions, $\omega(\alpha, X)$ and $\rho(\alpha, X)$, to represent weak and strong splits of shape bonds, respectively. With an abuse of notation, two strong-split actions $\rho(\alpha, X)$ and $\rho(\beta, Y)$ are compatible if so are the channels $\langle \alpha, X\rangle$ and $\langle \beta, Y\rangle$. We will see that a synchronization between multiple pairs of compatible strong-split actions correspond to a strong-split operation. Split operations behave differently w.r.t. *time passing*: enabled strong-splits forbid time passing, while weak-splits can be arbitrarily delayed.

Let $\mathcal{C}$ be the set of all channels, $\omega(\mathcal{C}) = \{\omega(\alpha, X) \,|\, \langle \alpha, X\rangle \in \mathcal{C}\}$ and $\rho(\mathcal{C}) = \{\rho(\alpha, X) \,|\, \langle \alpha, X\rangle \in \mathcal{C}\}$ be the sets of *weak-split actions* and *strong-split actions*, resp. Our processes perform *atomic* actions belonging to the set $\mathsf{Act} = \mathcal{C} \cup \omega(\mathcal{C}) \cup \rho(\mathcal{C})$ whose elements are ranged over by $\mu, \mu', \cdots$. We finally assume a countably infinite collection $\mathcal{K}$ of *process name* or *process constants*.

Definition 2.2 (Shape behaviours) The set of *shape behaviours*, denoted by $\mathbb{B}$, is generated by the following grammar:

$$B ::= \mathsf{nil} \;\big|\; \langle \alpha, X\rangle.B \;\big|\; \omega(\alpha, X).B \;\big|\; \rho(L).B \;\big|\; \epsilon(t).B \;\big|\; B + B \;\big|\; K$$

where $\langle \alpha, X\rangle \in \mathcal{C}$, $L \subseteq \mathcal{C}$ (non-empty) whose elements are pairwise incompatible, $t \in \mathbb{T}$ and $K \in \mathcal{K}$.

As usual the nil operator can only let time pass without limits[6]. $\langle \alpha, X\rangle.B$ and $\omega(\alpha, X).B$ are (action-)prefixing known from CCS. $\langle \alpha, X\rangle.B$ exhibits a binding capability along the channel $\langle \alpha, X\rangle$, while $\omega(\alpha, X).B$ models the behaviour of a shape that, before evolving in B, wants to split a single bond established via the channel $\langle \alpha, X\rangle$. $\rho(L).B$ is the strong-split operator; it can evolve in B only if *all* strong split actions $\rho(\alpha, X)$ with $\langle \alpha, X\rangle \in L$ can be performed simultaneously. The other operators are the same as given in [12].

Rules in Table 1 define a weak[7] temporal transition relation $\overset{t}{\rightsquigarrow} \subseteq (B \times B)$ for

[6] A trailing nil will often be omitted; e.g. $\langle \alpha, X\rangle.\omega(\alpha, X)$ abbreviates $\langle \alpha, X\rangle.\omega(\alpha, X).\mathsf{nil}$.

[7] This weak relation is used when giving temporal semantics to 3D processes. It will become a real time passing if and only if a strong split of a compound process, which is considered urgent, is not enabled.

$$\textsc{Basic}_t\ \frac{B \overset{t}{\rightsquigarrow} B'}{S[B] \overset{t}{\rightsquigarrow} (S+t)[B']} \qquad \textsc{Comp}_t\ \frac{P \overset{t}{\rightsquigarrow} P' \quad Q \overset{t}{\rightsquigarrow} Q' \quad X' = X + (t \cdot \mathbf{v}(P))}{P\,\langle a, X\rangle\,Q \overset{t}{\rightsquigarrow} P'\,\langle a, X'\rangle\,Q'}$$

$$\textsc{Basic}_c\ \frac{B \xrightarrow{\langle\alpha,X\rangle} B' \quad Y = \mathsf{global}(X, \mathcal{R}(S))}{S[B] \xrightarrow{\langle\alpha,Y\rangle} S[B']} \qquad \textsc{Basic}_s\ \frac{B \xrightarrow{\rho(\alpha,X)} B' \quad Y = \mathsf{global}(X, \mathcal{R}(S))}{S[B] \xrightarrow{\rho(\alpha,Y)} S[B']}$$

$$\textsc{Comp}_s\ \frac{P \xrightarrow{\rho(\alpha,Y)} P'}{P\,\langle a, X\rangle\,Q \xrightarrow{\rho(\alpha,Y)} P'\,\langle a, X\rangle\,Q} \qquad \textsc{Comp}_w\ \frac{P \xrightarrow{\omega(\alpha,Y)} P'}{P\,\langle a, X\rangle\,Q \xrightarrow{\omega(\alpha,Y)} P'\,\langle a, X\rangle\,Q}$$

$$\textsc{Comp}_c\ \frac{P \xrightarrow{\langle\alpha,Y\rangle} P' \qquad Y \subseteq \mathcal{B}(P\,\langle a, X\rangle\,Q)}{P\,\langle a, X\rangle\,Q \xrightarrow{\langle\alpha,Y\rangle} P'\,\langle a, X\rangle\,Q} \qquad \textsc{StrPar}\ \frac{P \overset{\rho(b,Y)}{\Rightarrow} P'}{P\,\langle a, X\rangle\,Q \overset{\rho(b,Y)}{\Rightarrow} P'\,\langle a, X\rangle\,Q}$$

$$\textsc{StrSync}\ \frac{P \xrightarrow{\rho(\alpha,X_p)} P' \quad Q \xrightarrow{\rho(\overline{\alpha},X_q)} Q' \quad \alpha \in \{a, \overline{a}\} \quad X = X_p \cap X_q}{P\,\langle a, X\rangle\,Q \overset{\rho(a,X)}{\Rightarrow} P'\,\langle a, X\rangle\,Q'}$$

Table 2
Functional and temporal behaviour of 3DP-terms

$t \in \mathbb{T}$ and the action transition relation $\xrightarrow{\mu} \subseteq (B \times B)$ for $\mu \in Act$. Most of the temporal rules are those provided in [12]. In our case rules $\textsc{Pref}_t$ and $\textsc{Str}_t$ state that processes like $\langle\alpha, X\rangle.B$, $\omega(\alpha, X).B$ and $\rho(L).B$ can be arbitrarily weak-delayed. Regarding functional rules, the only worth noting are $\textsc{Str}_1$ and $\textsc{Str}_2$, defining strong-split behaviours. If $\langle\alpha, X\rangle \in L$ then $\rho(L).B$ can do a $\rho(\alpha, X)$-action and evolves either in B (if $L = \{\langle\alpha, X\rangle\}$) or in $\rho(L\backslash\{\langle\alpha, X\rangle\}).B$ (otherwise). Rule $\textsc{Str}_3$ is needed to handle arbitrarily nested terms, e.g. $\rho(\{\langle a, X\}).\rho(\{\langle\overline{b}, Y\}).B$. Other rules are as expected. For brevity, symmetric rules and rules for process variables have been omitted.

2.3 3D processes and their semantics

Behaviours and shapes are compounded to create *3D processes*, the basic building blocks of a Shape Calculus network.

Definition 2.3 (3D processes) The set 3DP of *3D processes* is generated by the grammar $P ::= S[B] \mid P\,\langle a, X\rangle\,P$, where $S \in \mathbb{S}$, $B \in \mathbb{B}$, $a \in \Lambda$ and $X \subseteq \mathbb{P}$ non-empty. The shape of each $P \in \mathsf{3DP}$ is defined by induction as follows: $\mathsf{shape}(S[B]) = S$, $\mathsf{shape}(P\,\langle a, X\rangle\,Q) = \mathsf{shape}(P)\,\langle X\rangle\,\mathsf{shape}(Q)$.
We also write $\mathsf{steer}\,t\,P$ to denote $P[\![\mathsf{steer}\ t\,\mathsf{shape}(P)]\!]$.

Rules in Table 2 define the temporal transition relation $\overset{t}{\rightsquigarrow} \subseteq (\mathsf{3DP} \times \mathsf{3DP})$ for $t \in \mathbb{T}$ and the functional transition relation $\xrightarrow{\mu} \subseteq (\mathsf{3DP} \times \mathsf{3DP})$ for $\mu \in \mathsf{Act}$. Essentially, a 3D process inherits its behaviour from the $\mathbb{B}$-terms defining its internal behaviour, but now sites of binding capabilities and split actions are expressed w.r.t. a global coordinate system (see rules $\textsc{Basic}_c$ and $\textsc{Basic}_s$). Note that rule $\textsc{Basic}_w$, omitted, is similar to $\textsc{Basic}_s$ replacing the $\rho()$-action with the $\omega()$-action. Other symmetric rules are omitted. Rules StrSync and StrPar define the transition relations $\overset{\rho(a,X)}{\Rightarrow} \subseteq (\mathsf{3DP} \times \mathsf{3DP})$ for strong-split of compatible channels [8]. Recall that

[8] Replacing $\rho(-)$ with $\omega(-)$ we obtain $\overset{\omega(a,X)}{\Rightarrow} \subseteq (\mathsf{3DP} \times \mathsf{3DP})$.

strong-split operations require that *all* the enabled strong-splits have been performed *together* before time passes further. In such a case we say that a process $P \in 3DP$ is able to *complete a reaction*, written $P \searrow$. According to [1] we restrict the timed operational semantics of 3D processes and we say that $P \xrightarrow{t} Q$ iff $P \overset{t}{\rightsquigarrow} Q$ and either $P \overset{\rho}{\not\rightarrow}$ or $P \not\searrow$.

Definition 2.4 (Networks of 3D processes) The set $\mathbb{N}$ of *networks of 3D processes* (3D networks, for short) is generated by the grammar $N ::= \mathsf{Nil} \mid P \mid N \parallel N$.

We now provide a sketch of 3D networks semantics. A full description can be found in [1]. 3D networks can perform actions ρ, ω, κ and t. ρ, ω represent strong- and weak-split operations, κ corresponds to a collision detection and response event and t is time passing. The first two actions define transitions $N \xrightarrow{\nu} N'$ [9] where $\nu \in \{\rho, \omega\}$ after which new 3D processes are created from the splitting of bounds [10]. A transitions $N \xrightarrow{\kappa} M$ represents collision response, executed when collisions are detected within the current Δ. Note that all collisions, elastic and inelastic, are resolved simultaneously in N. We assume that an inelastic collision always replaces the original pair of processes with the resulting 3D compounded process. The new configuration will be $N' = M \parallel P_1\langle a, X\rangle Q_1$ if $\exists\ P, Q \in N$ s.t. $P \xrightarrow{\langle \alpha, X_a\rangle} P_1, Q \xrightarrow{\langle \overline{\alpha}, X_b\rangle} Q_1, \alpha \in \{a, \overline{a}\}, X \in X_a \cap X_b, X \neq \emptyset$. We used M to denote the rewriting of N without P and Q. From the considerations above, the time step $N \xrightarrow{t} N'$ with $t \in \mathbb{T}$ and $0 \leq t \leq \Delta$ can be either $t = \Delta$ if no collision is detected or $t = t' < \Delta$ where t' is the time of collision detection. After each such time step, the steer function updates shape velocities.

We now give a very simple example that has only the purpose of showing the features of the calculus without any particular biological outcome. Note that, by now, the more promising biological applications of the Shape Calculus is at the cell/tissue level [4].

Example 2.5 (First Glycolysis step) The glycolysis pathway is part of the process by which individual cells produce and consume nutrient molecules. Here we focus on the first reaction:

$$\textit{glucose, ATP} \rightleftharpoons \textit{glucose-6-phosphate, ADP, } H^+$$

The 3D processes $S_{GLC}[\mathsf{GLC}]$, $S_{ATP}[\mathsf{ATP}]$ and $S_{HEX}[\mathsf{HEX}]$ represents the reactants. Their shapes are approximations of public available 3D models and their behaviours are defined as follows:

$$\mathsf{HEX} = \langle \mathsf{atp}, X_{ha}\rangle.\mathsf{HA} + \langle \mathsf{glc}, X_{hg}\rangle.\mathsf{HG},$$
$$\mathsf{HA} = \omega(\mathsf{atp}, X_{ha}).\mathsf{HEX} + \epsilon(t_h).\langle \mathsf{glc}, X_{hg}\rangle.\rho(\{\langle \mathsf{atp}, X_{ha}\rangle, \langle \mathsf{glc}, Y_{hg}\rangle\}).\mathsf{HEX},$$
$$\mathsf{HG} = \omega(\mathsf{glc}, X_{hg}).\mathsf{HEX} + \epsilon(t_h).\langle \mathsf{atp}, X_{ha}\rangle.\rho(\{\langle \mathsf{atp}, X_{ha}\rangle, \langle \mathsf{glc}, Y_{hg}\rangle\}).\mathsf{HEX},$$

[9] If $\nu = \rho$, $N \xrightarrow{\nu} N'$ represents strong-split operations from P contained in N and $P \searrow$.

[10] In [1] the function $\mathsf{split} : \mathsf{3DP} \times \wp(\mathcal{C}) \to \mathbb{N}$ is used to "physically" finalize splitting of 3D processes. split is inductively defined on a process $P \in 3DP$ and a set of channels $C \in \mathcal{C}$ as $\mathsf{split}(P, C) = \mathsf{split}(R, C) || \mathsf{split}(Q, C)$ if $P = R\langle a, X\rangle Q$ and $\langle a, X\rangle \in C$, P otherwise.

$$\mathsf{ATP} = \langle \overline{\mathsf{atp}}, X_{ah} \rangle.(\epsilon(t_a).\rho(\{\langle \overline{\mathsf{atp}}, X_{ah} \rangle\}).\mathsf{ADP} + \omega(\overline{\mathsf{atp}}, X_{ah}).\mathsf{ATP})$$
$$\mathsf{GLC} = \langle \overline{\mathsf{glc}}, X_{gh} \rangle.(\epsilon(t_g).\rho(\{\langle \overline{\mathsf{glc}}, X_{gh} \rangle\}).\mathsf{G6P} + \omega(\overline{\mathsf{glc}}, X_{gh}).\mathsf{GLC}$$

The Hexokinase process evolves by binding with one instance of each metabolite. Once it has connected with both of them and all delays are consumed, a strong-split action is possible, i.e., the reaction occurs. Glucose and ATP evolves to glucose-6-phosphate and ADP, which we assume to be, for simplicity, nil.

3 Abstract interpretation of Shape Calculus

Let us consider a 3D network whose components are several (a finite number) instances of the processes described in Example 2.5, i.e., they are 3D processes located in different positions and with different velocity vectors. Such a network evolves in many different configurations due to the steer function, the timing behaviour and the non-determinism of interactions and splittings. In this section we provide an abstraction of the Shape Calculus where a *set* of 3D networks is abstracted to a set of 3D abstract processes. The concrete domain that we consider is the power set of the set of all 3D networks.

Definition 3.1 (Concrete Domain) The concrete domain is the complete lattice $(\wp(\mathbb{N}), \subseteq, \cup, \cap, \{\}, \mathbb{N})$ where $\wp$ is the power set operator, set union is the least upper bound (lub), set intersection is the greatest lower bound (glb), $\{\}$ is the bottom element and $\mathbb{N}$ is the top element.

We deal with *abstract* 3D processes, i.e., 3D processes without position and velocity. In the following we use the notation $\mathbb{S}^\sharp$, $\mathbb{B}^\sharp$, and $\mathsf{3DP}^\sharp$ to denote the set of abstract shapes, abstract behaviours and abstract 3D processes, respectively. A basic shape $\sigma = \langle V, m, p, v \rangle$ is abstracted to $\sigma^\sharp = \langle V, m, \odot, \odot \rangle$ where $\odot$ represents any set of points in the global coordinate system. Inductively on the syntactic structure, $S\langle X \rangle S$ is abstracted to $S^\sharp \langle \odot \rangle S^\sharp$ and a 3D processes $P\langle a, X \rangle Q$ is abstracted to $P^\sharp \langle a, \odot \rangle Q^\sharp$. In a similar fashion, all time delays $\epsilon(t)$ are collapsed to $\epsilon(\cdot)$ which represents a zero delay.

As usual, the abstraction is formalised by means of abstraction functions α and concretization functions γ. In Table 3 we define the abstraction functions $\alpha_S : \mathbb{S} \to \mathbb{S}^\sharp$, $\alpha_B : \mathbb{B} \to \mathbb{B}^\sharp$ and $\alpha_P : \wp(\mathbb{N}) \to \wp(\mathsf{3DP}^\sharp)$ to abstract shapes, behaviours and 3D processes, respectively. Note that α_B does not abstract from local binding sets X. In fact, in the abstraction we are only interested in all possible binding capabilities between abstract 3D processes. The abstract semantics introduced later on will consider a pair of channel compatible if they belong to the same type, abstracting from the region of contact. Thus, retaining X does not effect the abstract interpretation in any case. Also note that temporal delays are zeroed since we are considering all the possible bindings at any time. Finally, the abstraction function of a network or a set of networks returns a set of abstract 3D processes. It is important to remark that several instances of the same entity can be contained in a network. For instance, in the Example 2.5 we would have several HEX as well as

$\alpha_S(\sigma)$	$=$	$\langle V, m, \odot, \odot\rangle$
$\alpha_S(S\langle X\rangle T)$	$=$	$\alpha_S(S)\langle\odot\rangle\alpha_S(T)$
$\alpha_B(\mathsf{nil})$	$=$	$\mathsf{nil}^\sharp$
$\alpha_B(\langle a, X\rangle.B)$	$=$	$\langle a, X\rangle.\alpha_B(B)$
$\alpha_B(B_1 + B_2)$	$=$	$\alpha_B(B_1) + \alpha_B(B_2)$
$\alpha_B(\omega(a, X).B)$	$=$	$\omega(a, X).\alpha_B(B)$
$\alpha_B(\rho(a, X).B)$	$=$	$\rho(a, X).\alpha_B(B)$
$\alpha_B(\epsilon(t).B)$	$=$	$\epsilon(\cdot).\alpha_B(B)$
$\alpha_B(\rho(L).B)$	$=$	$\rho(L).\alpha_B(B)$
$\alpha_P(S[B])$	$=$	$\alpha_S(S)[\alpha_B(B)]$
$\alpha_P(P\langle a, X\rangle Q)$	$=$	$\alpha_P(P)\langle a, \odot\rangle\alpha_P(Q)$
$\alpha_P(\|_{i\in I}\ P_i)$	$=$	$\bigcup_{i\in I}\{\alpha(P_i)\}$

Table 3
Abstraction functions

several *GLC* and *ATP* to represent different *concentrations* over the space. After abstracting from space and velocity, all the instances of a same 3D process collapse to a unique abstract 3D process. Hence, elements of the abstract domain are *sets* of abstract 3D processes.

Definition 3.2 (Abstract Domain) The abstract domain is denoted A and it is the complete lattice $(\wp(\mathsf{3DP}^\sharp), \subseteq, \cup, \cap, \{\}, \mathsf{3DP}^\sharp)$.

Table 4 shows the corresponding concretization functions that produce all the possible elements of the concrete domain in terms of absolute position in space, velocity and number of processes instances in the environment.

For γ_S we assume to limit the velocity of shapes by a *maximal velocity* v_{max}. This is needed to guarantee the consistency of the collision detection system. We remark that γ_S generates from a compounded abstract shape a set of concrete compounded shapes such that the intersection of their boundaries is not empty. Although the set contains also interpenetrating shapes, i.e. not well-formed [1], this does not affect the correctness of the abstraction. The same assumptions are used in $\gamma_P(P^\sharp\langle a^\sharp, \odot\rangle Q^\sharp)$ in order to obtain the set of instances of a compounded abstract 3D process. Finally, $\gamma_P(\{P_1^\sharp, \ldots, P_n^\sharp\})$ generates the set of all possible sets of concrete 3D networks. Each set differs in its cardinality and in the number of concrete instances of each abstract 3D process. Note that the cardinality can be zero, thus in some of the generated concrete networks some species are not present.

Proposition 3.3 *(Galois insertion) Let $a \in \mathsf{3DP}^\sharp$ and $C \in \wp(\mathbb{N})$. α and γ forms a Galois insertion: i) α and γ are monotonic, ii) $C \subseteq \gamma(\alpha(C))$, iii) $\alpha(\gamma(a)) = a$.*

The abstract semantics is given substituting each concrete rule with its abstract version. Because of their similarity, we use the same rules defined in the previous section, except those refined in Table 5. In such a case the concrete syntax is substituted by the abstract one introduced before. Accordingly, the abstract rules will define abstract transition relations that are the same given in the concrete semantic but marked with symbol $\sharp$. We remark that rules in Table 5 are needed

$\gamma_S(\sigma^\sharp)$	$=$	$\{\sigma = \langle V, m, p, v\rangle \mid p \in \mathbb{P}, 0 \leq \|v\| \leq v_{max}\}$
$\gamma_S(S^\sharp\langle\odot\rangle T^\sharp)$	$=$	$\{S\langle Y\rangle T \mid S \in \gamma_S(S^\sharp), T \in \gamma_S(T^\sharp),$ $Y \subseteq \mathcal{B}(S) \cap \mathcal{B}(T), Y \neq \emptyset\}$
$\gamma_B(\mathsf{nil}^\sharp)$	$=$	$\{\mathsf{nil}\}$
$\gamma_B(\langle a, X\rangle.B^\sharp)$	$=$	$\{\langle a, X\rangle.B \mid B \in \gamma_B(B^\sharp)\}$
$\gamma_B(B_1^\sharp + B_2^\sharp)$	$=$	$\{B_1 + B_2 \mid B_1 \in \gamma_B(B_1^\sharp), B_2 \in \gamma_B(B_2^\sharp)\}$
$\gamma_B(\omega(a, X).B^\sharp)$	$=$	$\{\omega(a).B \mid B \in \gamma_B(B^\sharp)\}$
$\gamma_B(\rho(a, X).B^\sharp)$	$=$	$\{\rho(a, X).B \mid B \in \gamma_B(B^\sharp)\}$
$\gamma_B(\epsilon(\cdot).B^\sharp)$	$=$	$\{\epsilon(t).B \mid t \in \mathbb{T}, B \in \gamma_B(B^\sharp)\}$
$\gamma_B(\rho(L).B^\sharp)$	$=$	$\{\rho(L).B \mid B \in \gamma_B(B^\sharp)\}$
$\gamma_P(S^\sharp[B^\sharp])$	$=$	$\{S[B] \mid S \in \gamma_S(S^\sharp), B \in \gamma_B(B^\sharp)\}$
$\gamma_P(P^\sharp\langle a^\sharp, \odot\rangle Q^\sharp)$	$=$	$\{P\langle a, X\rangle Q \mid P \in \gamma_P(P^\sharp), Q \in \gamma_P(Q^\sharp),$ $X \subseteq \mathcal{B}(\mathsf{shape}(P)) \cap \mathcal{B}(\mathsf{shape}(Q)), X \neq \emptyset\}$
$\gamma_P(\{P_1^\sharp, \ldots, P_n^\sharp\})$	$=$	$\wp(\{N \in \mathbb{N} \mid N = (\|_{i=1}^n\ (\|_{P \in U_i}\ P))\})$ where $\forall i \in \{1, \ldots, n\}\ (U_i \in \wp(\gamma_P(P_i^\sharp)) \wedge U_i$ finite$)$

Table 4
Concretization functions

to abstract away from space and velocity. Rules $\textsc{Basic}_c^\sharp$ and $\textsc{Basic}_s^\sharp$ simply return a channel without information on the region of contact. Indeed, rule $\textsc{StrSync}_2$ now considers two channels compatible if and only if they belong to the *same* name type. Symmetric rules and rules for weak-split have been omitted. In the latter case, it is enough to replace ρ-action with ω-action. Finally, note that timing rules are retained although time is zeroed. These rules are only used for the proof of local correctness. They will not create circularities because, generating the same abstract processes once applied, they do not affect the abstract fixpoint iteration. Other rules that are similar to the concrete semantics behave as expected.

The interactions of abstract 3D processes are the focus of our abstraction. An abstract network $N^\sharp = \{P_1^\sharp, \ldots, P_n^\sharp\}$ performs the same type of transitions of a corresponding concrete one. Thus, we consider similar transition relations $\xrightarrow{t}_\sharp$, $\xrightarrow{\nu}_\sharp$ and $\xrightarrow{\kappa}_\sharp$. In the abstract domain temporal transitions do not change the network, i.e., for each generic abstract network $N^\sharp$ we have that $N^\sharp \xrightarrow{t}_\sharp N^\sharp$. The transition relation $\xrightarrow{\nu}_\sharp$ remains the same for $\nu = \omega$ as well as in case of strong-splits. However, given the semantics of $\xrightarrow{t}_\sharp$ (see Section 2), in the abstract domain a process is always able to complete a reaction, i.e. $P \searrow$, if all the involved splits are enabled, no matter of delays. In case of interactions due to collisions, $N_1^\sharp \xrightarrow{\kappa}_\sharp N_2^\sharp$, we define

$$N_2^\sharp = N_1^\sharp \cup \{R^\sharp \mid \exists\ P^\sharp, Q^\sharp \in N_1^\sharp \colon P^\sharp \xrightarrow{\langle\alpha,\odot\rangle}_\sharp P_1^\sharp, Q^\sharp \xrightarrow{\langle\overline{\alpha},\odot\rangle}_\sharp Q_1^\sharp, \alpha \in \{a, \overline{a}\} \wedge R^\sharp = P_1^\sharp\langle a, \odot\rangle Q_1^\sharp\}$$

It can be proven that for each local operator, that is to say, for each rule defining the concrete semantics, the abstract version is a correct approximation of the concrete one. Thus, by the general results of abstract interpretation, we get that the defined abstraction is globally correct. In order to perform a verification we need to instruct a fixpoint iteration. Let N be the network we want to test:

$$\textsc{Del}_a^\sharp \frac{B^\sharp \xrightarrow{\mu}_\sharp B'^\sharp}{\epsilon(\cdot).B^\sharp \xrightarrow{\mu}_\sharp B'^\sharp} \qquad \textsc{Del}_t^\sharp \frac{}{\epsilon(\cdot).B^\sharp \overset{t}{\rightsquigarrow}_\sharp \epsilon(\cdot).B^\sharp} \qquad \textsc{Del}_c^\sharp \frac{B^\sharp \overset{t}{\rightsquigarrow}_\sharp B'^\sharp}{\epsilon(\cdot).B^\sharp \overset{t'}{\rightsquigarrow}_\sharp B'^\sharp}$$

$$\textsc{Comp}_t^\sharp \frac{P^\sharp \overset{t}{\rightsquigarrow}_\sharp P'^\sharp \quad Q^\sharp \overset{t}{\rightsquigarrow}_\sharp Q'^\sharp}{P^\sharp \langle a, \odot\rangle Q^\sharp \overset{t}{\rightsquigarrow}_\sharp P'^\sharp \langle a, \odot\rangle Q'^\sharp} \qquad \textsc{Comp}_c^\sharp \frac{P^\sharp \xrightarrow{\langle a, \odot\rangle}_\sharp P'^\sharp}{P \langle a, \odot\rangle Q^\sharp \xrightarrow{\langle a, \odot\rangle}_\sharp P'^\sharp \langle a, \odot\rangle Q^\sharp}$$

$$\textsc{Basic}_c^\sharp \frac{B^\sharp \xrightarrow{\langle a, X\rangle}_\sharp B'^\sharp}{S^\sharp[B^\sharp] \xrightarrow{\langle a, \odot\rangle}_\sharp S^\sharp[B'^\sharp]} \qquad \textsc{Basic}_s^\sharp \frac{B^\sharp \xrightarrow{\langle a, X\rangle}_\sharp B'^\sharp}{S^\sharp[B^\sharp] \xrightarrow{\rho(\alpha, \odot)}_\sharp S^\sharp[B'^\sharp]}$$

$$\textsc{StrSync}^\sharp \frac{P^\sharp \xrightarrow{\rho(\alpha, \odot)}_\sharp P'^\sharp \quad Q^\sharp \xrightarrow{\rho(\overline{\alpha}, \odot)}_\sharp Q'^\sharp \quad \alpha \in \{a, \overline{a}\}}{P^\sharp \langle a, \odot\rangle Q^\sharp \overset{\rho(a, \odot)}{\Rightarrow} P'^\sharp \langle a, \odot\rangle Q'^\sharp}$$

Table 5
Abstract Rules for $\mathbb{B}^\sharp$ and $\mathsf{3DP}^\sharp$ terms

- $F \uparrow^0 (N) = \alpha(\{N\})$
- $F \uparrow^n (N) = \bigcup_{M^\sharp \in \{N^\sharp \mid F\uparrow^{n-1}(N) \xrightarrow{x}_\sharp N^\sharp,\ x \in \{\omega, \rho, \kappa, t\}\}} M^\sharp$

Note that even if the behaviours of the involved abstract 3D processes are finite state processes, then the possible number of different abstract networks is not guaranteed to be finite. Indeed, some processes can be defined that mimic unbounded polymers, for instance a compound process that can always be compounded with an existing species and then continues to do so. Thus, it is not always guaranteed that the fixpoint iteration stops in a finite number of steps. If this happens, however, we can look at the obtained set of abstract processes and can conclude that if a species is not present, then it can never appear in any of the actual concrete traces. Further abstractions are required to guarantee the termination in the general case.

Example 3.4 (Glucose/ATP bond) Consider again Example 2.5. We are now interested in verifying if it is possible for a glucose and an ATP molecule to interact, i.e., bind. Considering any concrete network with some instances of the 3D processes defined in the example, the abstraction will always be the set of the following three abstract processes:

$$ATP^\sharp = S_a^\sharp[\mathsf{ATP}^\sharp] \ , \ GLC^\sharp = S_g^\sharp[\mathsf{GLC}^\sharp] \text{ and } HEX^\sharp = S_a^\sharp[\mathsf{HEX}^\sharp]$$

The contained behaviour is almost identical to the concrete one due to the definition of α_B. We can now apply the abstract transition relations presented above. At each step only the changing or newly created processes are shown for the sake of brevity. First step is the execution of a collision transition. We obtain the possible combination of the Hexokinase with the two metabolites:

$$\{HEX^\sharp[\mathsf{XG}^\sharp] \ \langle glc, \odot\rangle \ GLC^\sharp[\epsilon(\cdot).\rho(\{\langle \overline{\mathsf{glc}}, X_{gh}\rangle\}).\mathsf{nil} + \omega(\overline{\mathsf{glc}}, X_{gh}).\mathsf{GLC}^\sharp],$$
$$HEX^\sharp[\mathsf{XA}^\sharp] \ \langle atp, \odot\rangle \ ATP^\sharp[\epsilon(\cdot).\rho(\{\langle \overline{\mathsf{atp}}, X_{ah}\rangle\}).\mathsf{nil} + \omega(\overline{\mathsf{atp}}, X_{ah}).\mathsf{ATP}^\sharp]\}$$

Now both processes can weakly split on the newly created bold and get back to original unbound processes (see $\mathsf{XG}^\sharp/\mathsf{XA}^\sharp$) or execute a delay. Since time is abstracted away, we can execute the subsequent actions. Thus, at this stage we can weakly split the current enable bond or bind with ATP (glucose respectively). In both cases we obtain the same compound:

$$\{GLC^\sharp[\epsilon(\cdot).\rho(\{\langle\overline{\mathsf{glc}}, X_{ah}\rangle\}).\mathsf{nil} + \omega(\overline{\mathsf{glc}}, X_{ah}).\mathsf{GLC}^\sharp]\ \langle glc, \odot\rangle$$
$$HEX^\sharp[\rho(\{\langle\overline{\mathsf{atp}}, X_{ah}\rangle, \langle\overline{\mathsf{glc}}, X_{gh}\rangle\})]\ \langle atp, \odot\rangle$$
$$ATP^\sharp[\epsilon(\cdot).\rho(\{\langle\overline{\mathsf{atp}}, X_{ah}\rangle\}).\mathsf{nil} + \omega(\overline{\mathsf{atp}}, X_{ah}).\mathsf{ATP}^\sharp]\}$$

According to the concrete domain glc-hex-atp would not be able to complete a reaction, i.e. glc-hex-atp $\not\searrow$ due to the delays of both GLC and ATP. This is not the case of the abstract domain in which the strong-split action can be executed. Thus we obtain also $\{GLC^\sharp[\mathsf{nil}^\sharp], ATP^\sharp[\mathsf{nil}^\sharp]\}$. No further transition for these processes can be executed, so fixpoint is reached. Hence, we can conclude, for instance, that GLC and ATP processes cannot interact directly between each other.

4 Conclusions and Future work

In this paper we have applied abstract interpretation to the Shape Calculus, a bio-inspired calculus, with the aim to check properties that would be impossible to check in the concrete algebra, due to the variability induced by motion laws, space, time and interaction-related non-determinism. In particular we were interested in proving untimed and unspatial safety properties. The use of the abstract interpretation framework is based on the abstraction of spatial information from the shapes and on the temporal abstraction of shapes behaviours. This extension simplifies the semantics of the calculus and drastically reduces the state space of a given 3D network. Concerning future work, on the one hand a further abstraction must be defined in order to always guarantee the termination of the current abstraction when processes behaving like unbounded polymers are present. On the other hand, a refinement of the current abstraction in which time is retained is certainly of interest. This would permit the verification of quantitative timed safety properties like "glucose-6-phosphate will never be produced before 20 milliseconds". The long-term objective is to construct a lattice of abstract domains permitting the checking of a large variety of different possibly quantitative properties, maybe sometimes abstracting only time and not space, or abstracting behaviours, but not motion.

References

[1] Bartocci, E., D. R. Cacciagrano, M. R. D. Berardini, E. Merelli and L. Tesei, *Timed operational semantics and well-formedness of shape calculus*, Sci. Ann. Comp. Sci. **20** (2010), pp. 32–52.

[2] Bartocci, E., F. Corradini, M. R. D. Berardini, E. Merelli and L. Tesei, *Shape calculus. a spatial mobile calculus for 3d shapes*, Sci. Ann. Comp. Sci. **20** (2010), pp. 1–31.

[3] Bittig, A. T., M. Jeschke and A. M. Uhrmacher, *Towards modelling and simulation of crowded environments in cell biology*, AIP Conference Proceedings **1281** (2010), pp. 1326–1329.
URL http://link.aip.org/link/?APC/1281/1326/1

[4] Buti, F., D. Cacciagrano, F. Corradini, E. Merelli, L. Tesei and M. Pani, *Bone Remodelling in BioShape*, Electronic Notes in Theoretical Computer Science **268** (2010), pp. 17–29, *Proc. of CS2Bio 2010.*

[5] Buti, F., D. R. Cacciagrano, F. Corradini, E. Merelli and L. Tesei, *BioShape: a spatial shape-based scale-independent simulation environment for biological systems*, Procedia Computer Science **1** (2010), pp. 827–835, *Proc. of 7th Int. Workshop on Multiphysics Multiscale Systems, ICCS 2010.*

[6] Buti, F., D. R. Cacciagrano, F. Corradini, E. Merelli and L. Tesei, *A uniform multiscale meta-model of BioShape*, Electronic Notes in Theoretical Computer Science (To appear 2011), *Proc. of Cs2Bio 2011, June 9th, Reykjavik, Iceland.*

[7] Cacciagrano, D., F. Corradini and E. Merelli, *Bone Remodelling: A Complex Automata-Based Model Running in BioShape*, in: *ACRI 2010*, LNCS **6350**, 2010, pp. 116–127.

[8] Cousot, P., *Abstract interpretation*, ACM Comput. Surv. **28** (1996), pp. 324–328.

[9] Cousot, P. and R. Cousot, *Abstract interpretation frameworks*, J. Log. Comput. **2** (1992), pp. 511–547.

[10] Ericson, C., "Real-time collision detection," Elsevier North-Holland, Inc., 2005.

[11] Gruenert, G., B. Ibrahim, T. Lenser, M. Lohel, T. Hinze and P. Dittrich, *Rule-based spatial modeling with diffusing, geometrically constrained molecules*, BMC Bioinformatics **11** (2010), p. 307.
URL http://www.biomedcentral.com/1471-2105/11/307

[12] Yi, W., *Real-time behaviour of asynchronous agents*, in: *CONCUR*, 1990, pp. 502–520.