# A Process Model of Actin Polymerisation

Luca Cardelli [1]

*Microsoft Research*
*7 JJ Thomson Avenue, CB3 0FB, Cambridge, UK*

Emmanuelle Caron

*Centre for Molecular Microbiology and Infection*
*Imperial College, London, UK*

Philippa Gardner

*Department of Computing*
*Imperial College, London, UK*

Ozan Kahramanoğulları [2]

*Department of Computing &*
*Centre for Integrative Systems Biology*
*Imperial College, London, UK*

Andrew Phillips

*Microsoft Research*
*7 JJ Thomson Avenue, CB3 0FB, Cambridge, UK*

**Abstract**

Actin is the monomeric subunit of actin filaments which form one of the three major cytoskeletal networks in eukaryotic cells. Actin dynamics, be it the polymerisation of actin monomers into filaments or the reverse process, plays a key role in many cellular activities such as cell motility and phagocytosis. There is a growing number of experimental, theoretical and mathematical studies on the components of actin polymerisation and depolymerisation. However, it remains a challenge to develop compositional models of actin dynamics, e.g., by using differential equations. In this paper, we propose compositional process algebra models of actin polymerisation, and present a geometric representation of these models that allows to generate movies reflecting their dynamics.

*Keywords:* actin, stochastic $\pi$-calculus, process modelling, geometric plotting

[1] The authors would like to thank anonymous referees for useful comments and suggestions. Cardelli acknowledges support of a visiting professorship at Imperial College. Gardner acknowledges support of a Microsoft Research Cambridge/Royal Academy of Engineering Senior Fellowship. Kahramanoğulları acknowledges support of the UK Biotechnology and Biological Sciences Research Council through the Centre for Integrative Systems Biology at Imperial College (grant BB/C519670/1).

[2] To whom correspondence should be addressed. Email: ozank@doc.ic.ac.uk

*This paper is electronically published in*
*Electronic Notes in Theoretical Computer Science*
*URL:* www.elsevier.nl/locate/entcs

# 1 Introduction

Actin is the monomeric subunit of actin filaments which form one of the three major cytoskeletal networks in eukaryotic cells. Actin dynamics, be it the polymerisation of actin monomers into filaments or the reverse process, plays a key role in many cellular activities. For example, during cell motility and phagocytosis, protrusive activity is due to the growth of actin filaments close to the cell membrane.

There is a growing number of experimental, theoretical and mathematical studies on the components of actin polymerisation and depolymerisation (see, e.g., [17]). However, it remains a challenge to develop models that can test hypotheses regarding the mechanisms of actin dynamics, especially when these models need to be composed with models established for other biological components. In particular, models of complexation as in actin meshwork growth by means of differential equations is a difficult task (see, e.g., [1,12]). This is because of the necessity in these models to treat every possible filament length as a different species, and thus describe its behaviour with a distinct equation. The situation becomes even more complicated in these models when different states of monomers, such as being bound to ATP or ADP, are considered.

Process algebra are languages which have originally been designed to formally describe complex reactive computer systems. Due to the resemblance between these and biological systems, process algebra have been recently used to model biological systems [22]. In these languages, typically, each sub-component is described separately, together with its interaction channels. Through these channels, it can send and receive information to other processes. In a biological setting, where processes are biological species, these interactions find an interpretation as, e.g. complexation of proteins or phosphorylation of specific sites. Potentially interacting and non-interacting components can be put together to build increasingly complex systems and each sub-system can be altered locally (without modifying other components) when modification is needed. This constitutes the compositionality of the approach as a distinguishing feature in contrast to, e.g., differential equations.

Stochastic pi-calculus is a process algebra where stochastic rates are imposed on processes. By using the SPiM tool (stochastic pi-calculus machine), we can run computer simulations that display the change in time in the populations of the different species of the system being modelled [19]. We give compositional process models of actin polymerisation. Our models incrementally reflect the different levels of complexity in the actin dynamics with respect to the capabilities of actin monomers. By running simulations on these models, we observe the behaviour of a system of actin filaments as the emergent behaviour of a collection of actin filaments in a meshwork, acting in concert. As the first contribution of the paper, we thus demonstrate how filaments built from monomers can be modelled compositionally as processes with ease, also when monomers have different states or binding capabilities. Another contribution of the paper is our extension to the SPiM tool with geometric plotting capabilities which we use to plot growing actin filaments: with this extension, we now have a way of simulating not only the populations of biochemical species, but also the evolution of their spatial distribution over time with respect to the parameters of the individual species. This novel form of computational modelling lays the

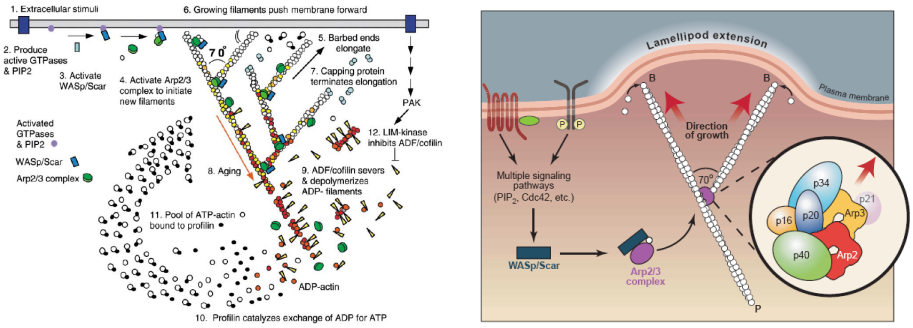



Fig. 1. Left: Actin growth as a response to extracellular stimuli [21]. Right: the Arp2/3 complex can bind to the side of pre-existing actin filaments and thereby lead to the formation of branches at a 70 degrees angle [25].

foundation for building models that reflect the behaviour of the modelled systems in the geometric space.

We model each actin monomer as a stochastic pi-calculus process. Each monomer binds/unbinds at its barbed end to/from another monomer at its pointed end. Monomers first form dimers, then polymers, resulting in linear actin filaments. The process model of the Arp2/3 complex binds to the pointed end of an actin monomer to nucleate the daughter filament. On the other end, Arp2/3 complex binds to the mother actin filament. We can then run simulations to observe the collective behaviour with different numbers of monomers and filaments evolving in time where filaments are built and unbuilt. We then parametrise filaments by coordinates which denote abstract locations for a geometric plotting. This results in movies of actin polymerisation that are generated with respect to the simulations with our models. Our models of actin polymerisation should ultimately provide an understanding of the mechanisms underlying actin-dependent cellular events such as cell motility and phagocytosis. They can potentially also serve as a template for similar complexation models.

## 2 The Role of Actin in Cells

Actin is one of the most abundant proteins in cells. This globular, 43-kDa protein can assemble into long polymers called actin filaments or filamentous (F-) actin. In eukaryotes, actin filaments provide mechanical support to cells and tissues. Furthermore, dynamic changes in the length of actin filaments, in other words actin polymerisation and depolymerisation, are essential for many aspects of cell function. Specifically, remodelling of the length and organisation of actin filaments is required for all functions that involve changes in cell shape, including cell motility, the division of one cell into two daughter cells and the protrusion of parts of the cells, for example the projection of axon and dendrites by neuronal cells and the capture of microorganisms during phagocytosis [21].

Actin monomers can self-assemble into helical, F-actin in vitro in the presence of ATP. The actin filaments are polarised, with a fast growing barbed end and a slow growing pointed end. This polarisation stems from the intrinsic polarity of actin monomers. In cells, actin polymerisation is highly regulated, firstly through the interaction of actin monomers and polymers with a variety of actin-binding

proteins (e.g., monomer-trapping proteins; filament capping and severing proteins); and secondly in response to the activation of intracellular signalling pathways by external stimuli (see Figure 1).

The rate-limiting step in actin polymerisation in vitro is the formation of the first complex of two to three monomers (the nucleus), also known as the nucleation phase. In eukaryotes, three actin nucleators have so far been identified: spire, formins and the Arp2/3 complex [6]. The Arp2/3 complex was the first actin nucleation factor identified. It is composed of 7 proteins, two of which the Actin-Related Proteins (Arp) 2 and 3- are thought to interact in such a way that they resemble an actin nucleus. The Arp2/3 complex can speed up the growth of actin filaments in vitro; it can also bind to the side of pre-existing actin filaments and thereby lead to the formation of branches at a 70 degrees angle (see Figure 1). The nucleating and branching activities of Arp2/3 are tightly regulated intracellularly, to ensure proper spatio-temporal control of actin polymerisation. The function of the Arp2/3 complex is activated at the end of intracellular signalling pathways that involve the Rho-family GTP-binding proteins Rac and Cdc42 and their WASP (Wiskott-Aldrich Syndrome Protein)-family interactors [2].

Although cells that lack Arp2/3 are impaired in a range of actin-dependent functions, it is clear that there are other critical regulators of actin dynamics. In addition to the other actin nucleator indicated above, it is worth remembering that actin filaments can be organised into different types of higher degree structures, for example bundles and contractile acto-myosin cables.

## 3 Actin Growth as Process Interaction

We use the stochastic $\pi$-calculus as our modelling language. In the following, we give a brief introduction to stochastic $\pi$-calculus.

### *3.1 Biological Processes as Computations*

In the stochastic $\pi$-calculus, models are built by composing atomic processes. Each process has a precise description of what actions it can take. Once a biological system has been modelled using these basic components, the model can be stochastically simulated in order to predict the evolution of the system over time. In this paper, the simulations are performed using the Stochastic Pi Machine (SPiM) [3] [19] which serves as a platform for implementing stochastic $\pi$-calculus processes and for running machine simulations. We use the SPiM notation throughout this paper.

Processes are viewed as the choice between zero or more processes. A process with $n$ choices is written as `let P = do P1 or ... or Pn`. If there is only one choice, we write `let P = P1`. A process can perform an input `?x(m);Q` or output `!x(n);Q` on a channel `x` or perform a delay, written as `delay@r;Q`, where `r` is a real number value denoting the rate of an exponential distribution and `Q` is the continuation process. Complementary input and output actions interact by means of hand shake operations on channels declared with the syntax `new`. The operator `new x@r:t P` creates a fresh channel `x` of rate `r` to be used in the process `P` where

[3] `http://research.microsoft.com/~aphillip/spim/`

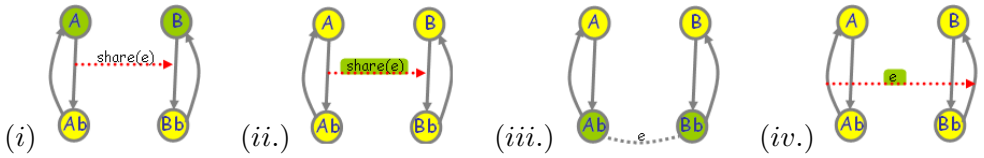



Fig. 2. Graphical representation of the evolution of the A and B interaction model. Processes A and B coexist (*i.*), and they can interact on channel share (*ii.*). When they interact, A sends the private channel $e$, and B receives it. This way, they evolve to processes Ab and Bb, respectively, which share the private channel $e$, resembling a covalent bond between two bio-chemical species (*iii.*). By interacting on channel $e$, they evolve back to the processes A and B, respectively (*iv.*).

`t` is the type of the channel `x` which can be, e.g., `chan(chan,chan)` denoting that the channel can pass the names of two channels. When a process is prefixed with a declaration of a fresh channel, that channel remains private to the process and does not conflict with any other channel. A process can be the empty process, written `()`. Two process components `P` and `Q` can be combined using parallel composition `P | Q`. This constitutes the basic form of compositionality which allows to compose processes in order to gradually build bigger models.

We can model complexation of biological species using processes [22]. Let us see this on the following example: consider the situation where the biological processes A and B can interact to form AB complex, which can return to the state where A and B coexist. We can depict this as the reaction $\mathsf{A} + \mathsf{B} \rightleftharpoons \mathsf{AB}$ . This reaction is coded in SPiM as follows:

```
new share@1.0:chan(chan)
let A() = ( new e@1.0: chan() !share(e); Ab(e) )
and Ab(e:chan) = !e; A()
let  B() = ?share(e); Bb(e) and Bb(e:chan) = ?e; B()
```

The first line of the code states that there is a channel `share` which takes another channel as argument. The second and third line state that the process `A` can interact on channel `share` and broadcast the private channel `e`, and then evolve to process `Ab`, which can send a message on channel `e` and evolve to `A`. The fourth line states that process `B` can receive a message on channel `share`, and then evolve to `Bb` which can receive a message on channel `e` and evolve to `B`.

Figure 2 shows a run of a cycle of this reaction in the style of the graphical representation of the SPiM language. There, the system is represented as two processes that interact over shared channels. Green marks indicate the current event.

### *3.2 A Simple Polymer Model*

We give a simple model of a polymer as the complexation of a potentially unbounded number of single monomers as in [3]. In our process representation of the monomers, each state of a monomer is given by a process: there are the states Af (free), Al (bound on the left), Ar (bound on the right), and Ab (bound on both sides). Each monomer can move between these states by interacting with another monomer. Actin monomers can in fact polymerise and depolymerise at both ends. However, the depolymerisation at the barbed end and the polymerisation at the pointed end are very slow. Because of this, in this subsection, we consider a polymer model, which can grow only at the barbed end (left) and shrink only at the pointed end

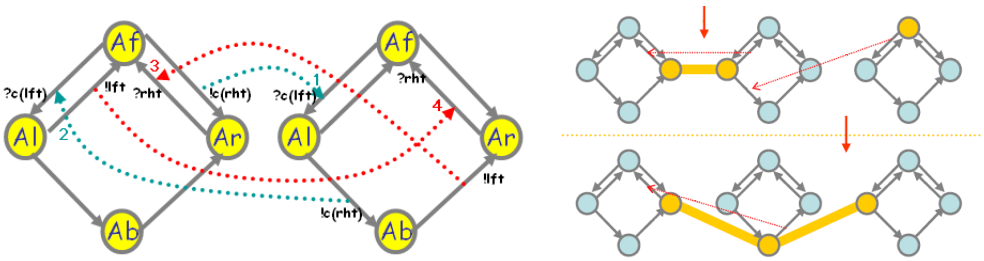
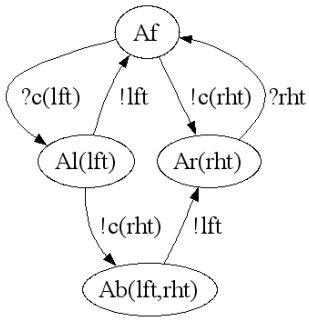



Fig. 3. A graphical representation of a simple actin polymerisation process model together with a schematic representation of some possible configurations of monomers.

(right).

In Figure 3 on the left-hand-side, there are the graphical representations of the processes for two monomers. Their possible interactions are depicted as dashed arrows: the arrows 1 and 2 are the association interactions whereas the arrows 3 and 4 are the disassociation interactions. The SPiM code for this model is as follows:

```
new c@0.116:chan(chan)
let Af() = (new rht@0.0027:chan
  do ?c(lft); Al(lft)
  or !c(rht); Ar(rht) )
and Al(lft:chan) = (new rht@0.0027:chan
  do !lft; Af()
  or !c(rht); Ab(lft,rht) )
and Ar(rht:chan) = ?rht; Af()
and Ab(lft:chan, rht:chan) = !lft; Ar(rht)
```




This code describes the situation depicted in Figure 3: (1) a process `Af` can interact with another process `Af`, and as a result of this one of them evolves to process `Al` and the other one evolves to process `Ar`. This describes the association of two monomers forming a dimer. (2) A process `Af` can also interact with a process `Al`, and as a result of this it evolves to process `Al` whereas `Al` evolves to process `Ab`. This describes the association of monomers at the barbed (left) end of the actin filament. (3) A process `Ar` can dissociate from `Ab` by interacting on a name private to both processes, and as a result of this `Ar` evolves to process `Af` whereas `Ab` evolves to process `Ar`. This describes the disassociation of a monomer from the

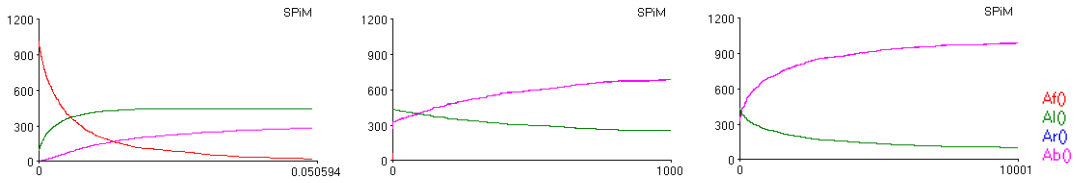



Fig. 4. The result of a simulation with the model depicted in Figure 3 at time 0.05 sec., 1000.0 sec and 10000.0 sec, with 1200 monomer processes at the beginning of the simulation.

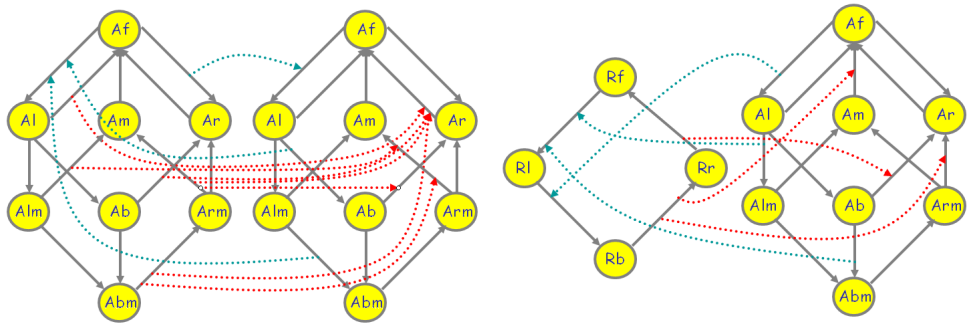



Fig. 5. A graphical representation of an actin monomer and Arp2/3 process models with their possible interactions. A denotes the actin monomer in its eight different states, and R denotes Arp2/3 protein in its four different states.

pointed (right) end of the actin filament. (4) A process `Al` can dissociate from a process `Ar`, and as a result of this both of them evolve to process `Af`. This describes the disassociation of a dimer to two monomers. The right-hand-side in Figure 3 is a schematic representation of some possible configurations of monomers, with thick lines joining their current states and representing their current associations.

The plots in Figure 4 show three views of the same simulation run with 1200 monomers, at 0.05 sec., 1000.0 sec., and 10000.0 sec.. The association rate ($11.6\mu M^{-1}s^{-1}$) is set with respect to ATP-actin and the disassociation rate ($0.27s^{-1}$) is set with respect to ADP-actin as given in [1,11] with a factoring constant of 100 (see, e.g., [26]). Because the units of the rates are given in $\mu M$, the initial number of 1200 monomers simulates the concentration of $1200\mu M$ of actin monomers, which is the concentration reported in [1,11].

### *3.3 Branching Polymer with Processes*

The localised actin polymerisation close to cell membrane depends on the addition of actin monomers at the barbed end (in contrast to the pointed end) of the filaments, and also on the generation of daughter branches on mother actin filaments (see Figure 1). The branching formation is initiated by Arp2/3 (actin related proteins) complex on the sides of existing mother actin filaments. The Arp2/3 complex anchors the pointed end of the future daughter filament to the mother filament as the free barbed end of the daughter grows away from the complex.

In order to model this branching in the filaments, we extend the monomer model depicted in Figure 3 with an additional binding site, that we call *mid*, for a branching process R which can bind to a monomer at this site. Then other monomers can bind to R to form the daughter filament. Thus, each monomer has 3 binding sites: left, right and mid. As before, the monomers which are free, bound on left, bound on right and bound on both sides are denoted with Af, Al, Ar and Ab. A monomer which is bound on mid is denoted with Am. Similarly monomers which are bound on mid and also bound on either left or right, or bound on both left and right are denoted with Alm, Arm and Abm. A graphical representation of this model is depicted in Figure 5. Similar to the model in Figure 3, in this model the filaments can grow only at the barbed end (right) and shrink at the pointed end (left). The

SPiM code of this model is given in Appendix B.

The plots in Figure 6 show three views of the same simulation run with 1200 monomers (Af) and 30 Arp2/3 processes (Rf), at 0.05 sec., 1000.0 sec. and 10000.0 sec.. The rates of the model are set as in [1,11], with a factoring similar to the model depicted in Figure 3.

### *3.4 An Alternative Simpler Branching Polymer Model*

In the model of Subsection 3.3, introduction of a new binding site results in 8 states for the monomers. This is because each of the previously available 4 states of the monomer are extended with a capability of binding to the process R. In this subsection, we introduce an alternative model by restricting the binding of the process R only to the monomer in the bound state (Ab). This results in the model depicted in Figure 7. The SPiM code of this model is given in Appendix C.

The plots in Figure 8 show three views of the same simulation with this model run with 1200 monomers (Af) and 30 Arp2/3 processes (Rf), at 0.05 sec., 1000.0 sec. and 10000.0 sec.. The rates of the model are set as in [1,11], with a factoring similar to the model depicted in Figure 3.

## 4 Actin as a Process

The models that we discussed so far focus on the polymerisation aspect of the actin monomers. However, actin filaments are formed as a result of more complex biochemical systems acting in concert. In this section, with the aim of getting closer to the actual biological systems that we are modelling, we extend our models with further aspects of the actin monomers and filaments.

### *4.1 Actin Monomers that Grow and Shrink at both Ends*

In all the models presented above, the filaments can grow only at the barbed end and shrink at the pointed end. However, actin filaments can grow and shrink at both ends, although the shrinking at the barbed end and the growth at the pointed

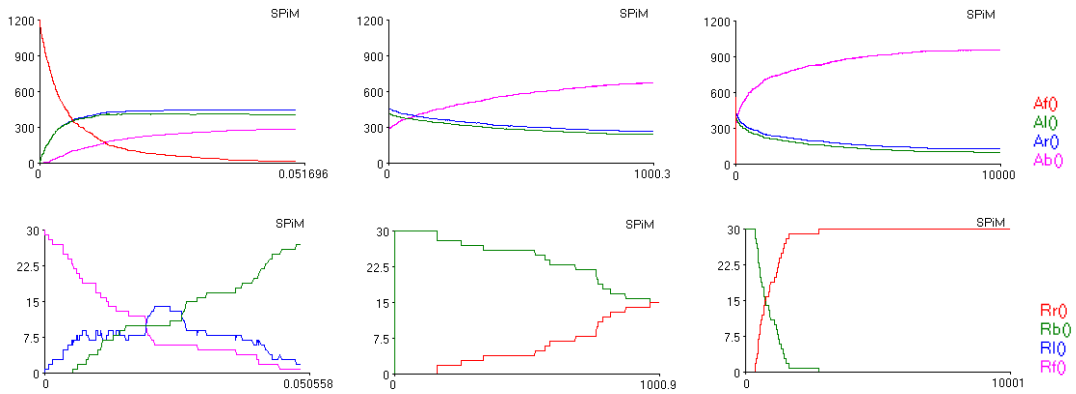



Fig. 6. The result of a simulation with the model depicted in Figure 5 at time 0.05 sec., 1000.0 sec and 10000.0 sec, with 1200 monomer and 30 Arp2/3 protein processes at the beginning of the simulation. The first row plots the Af, Al, Ar and Ab monomers. The second row plots the Rr, Rb, Rl and Rf.

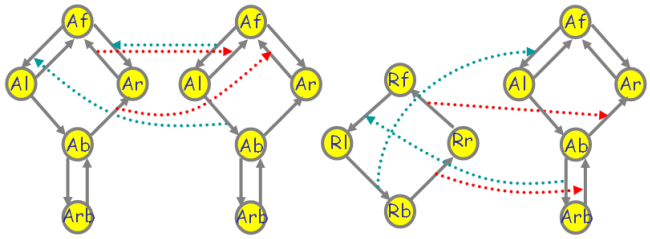



Fig. 7. Graphical representations of a model of a monomer and Arp2/3 process which simplifies the model depicted in Figure 5.

end are much slower. Because of this, we now extend our model in such a way that both growing and shrinking of the filaments at both ends are possible. Extending the linear (non-branching) filament model this way results in a model depicted in the left of Figure 9. The SPiM code for this model is given in Appendix D. Then by extending this model with a process for the Arp2/3 protein, as in the model of Figure 5, results in the middle model depicted in Figure 9. The SPiM code for this model is given in Appendix E.

By setting the kinetic rates with respect to those for ATP-actin given in [11], we run simulations. The plots in Figure 10 and Figure 11 show three views of the simulations with these models run with 1200 monomers (Af) and 30 Arp2/3 processes (Rf), at 0.05 sec., 1000.0 sec. and 10000.0 sec..

### *4.2 Capping Proteins*

In actin dependent events such as cell motility and phagocytosis, mechanisms of control for the actin assembly are essential. The barbed end of an actin filament is the site for rapid actin polymerisation in cells, so altering the availability of free actin filament barbed ends provides a regulation mechanism for the actin assembly. Capping of the barbed ends by capping proteins is a mechanism, which reduces the rate of drawdown on the pool of unpolymerised actin. The free end of the new filament elongates until a capping protein becomes available. Then, capping proteins bind with a high rate to barbed ends and terminate the growth. As a result of this, each filament grows only transiently (see, e.g., [24,14]).

We model the capping protein as a process which can bind to Al representing the barbed end of the filament. This results in the right-most monomer model depicted in Figure 9. There, Cf and Cb denote the free and bound capping protein. The

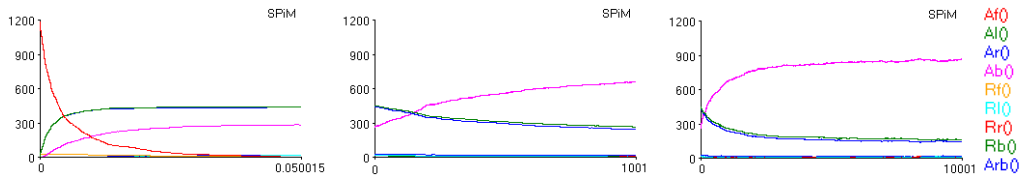



Fig. 8. The result of a simulation with the model depicted in Figure 7 at time 0.05 sec., 1000.0 sec and 10000.0 sec, with 1200 actin monomer and 30 Arp2/3 protein processes at the beginning of the simulation.

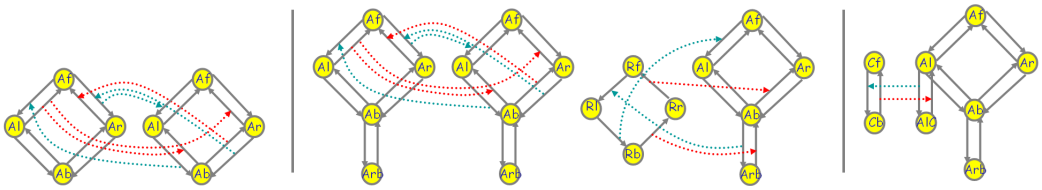


Fig. 9. Graphical representations of models extending the models in previous Subsections. The left and middle models extend the models in Figure 3 and Figure 7 with the capability of growing and shrinking both on left-end and right-end. The right model extends the model in the middle with a process that models capping protein.

SPiM code for this model is given in Appendix F.

We adopt the rate and concentration data on capping proteins given in [1] to our model and run simulations. The plots in Figure 12 show three views of these simulation with these models run with 1200 monomers (Af) 30 Arp2/3 processes (Rf) and 100 capping protein processes Cf, at 0.05 sec., 1000.0 sec. and 10000.0 sec..

### *4.3 An Actin Model with Multiple Layers*

A model that explains the mechanisms that are involved in actin assembly can be further extended with the role of ATP/ADP. Filamental or monomeric actin are bound to ATP molecules which can hydrolyse to ADP-$P_i$-actin which can then evolve to ADP-actin by dissociating the phosphate. Actin subunits in branched network hydrolyse their bound ATP quickly, but dissociate the phosphate slowly. Dissociation of phosphate initiates disassembly reactions, which then promote severing and dissociation of ADP-actin monomers from filament ends (see, e.g., [20,21]).

By using the model introduced in Subsection 4.2 as a single layer of an actin

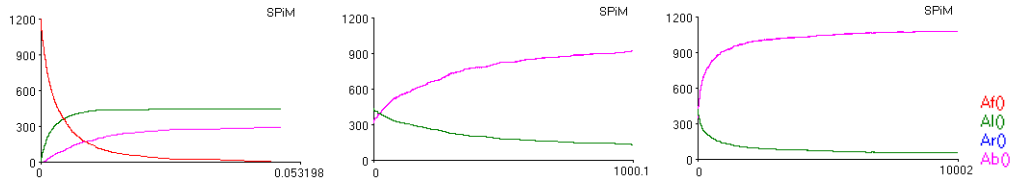



Fig. 10. The result of a simulation with the left model depicted in Figure 9 at time 0.05 sec., 1000.0 sec and 10000.0 sec, with 1200 actin monomer and 30 Arp2/3 protein processes at the beginning of the simulation.

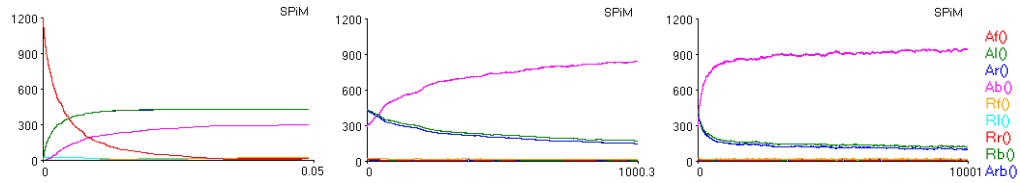



Fig. 11. The result of a simulation with the middle model depicted in Figure 9 at time 0.05 sec., 1000.0 sec and 10000.0 sec, with 1200 actin monomer and 30 Arp2/3 protein processes at the beginning of the simulation.

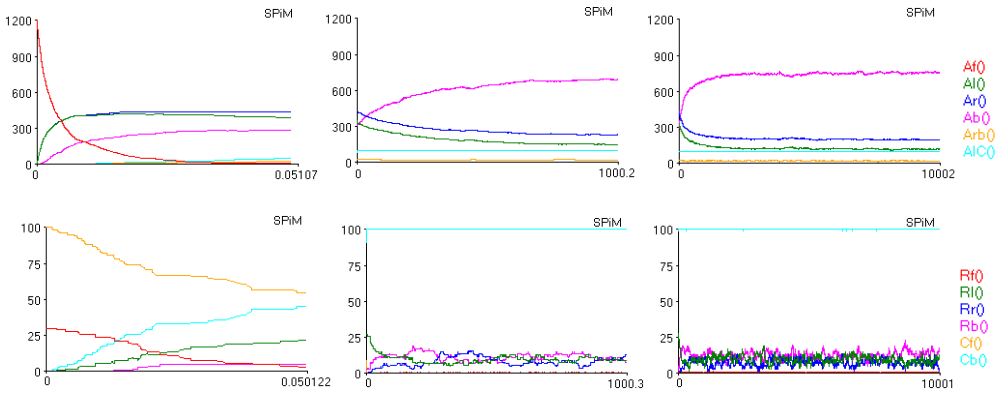



Fig. 12. The result of a simulation with the right model depicted in Figure 9, at time 0.05 sec., 1000.0 sec and 10000.0 sec, with 1200 actin monomer, 30 Arp2/3 protein processes and 100 capping protein processes at the beginning of the simulation. The first row plots the Af, Al, Ar, Ab, Arb and AlC monomers. The second row plots the Rf, Rl, Rr, Rb, Cf and Cb.

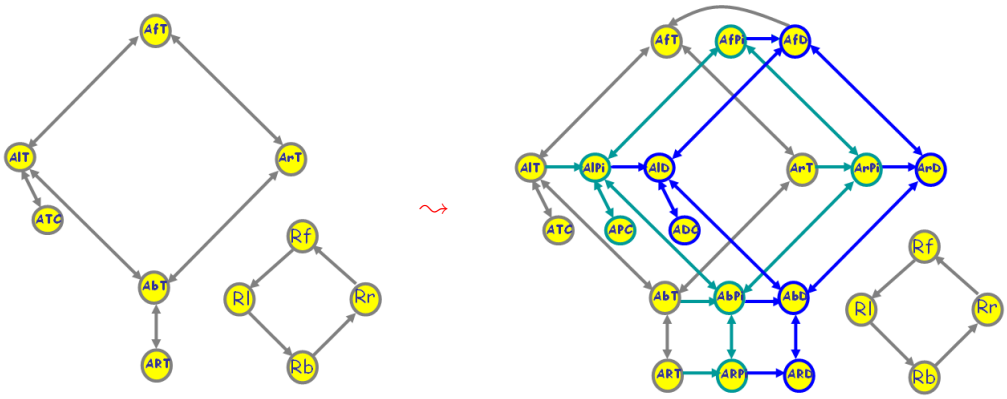



Fig. 13. Graphical representation of an actin model with multiple layers modelling ATP-actin, ADP-P$_i$-actin and ADP-actin.

monomer, we construct a three layered model of actin monomers as depicted in Figure 13 to reflect the role of ATP/ADP. In this model, AfT, AlT, ArT, AbT denote the ATP-actin in its free and left and right bounds. AfPi and AfD denote the free ADP-P$_i$-actin and ADP-actin, respectively. We denote their left and right bound forms analogously as before. Similarly, ATC, APC, and ADC denote the monomers that are bound to a capping protein process. In this model, ADP-actin can hydrolyse to ADP-P$_i$-actin and ADP-P$_i$-actin can dissociate its phosphate to become ADP-actin. We assume that the exchange to ATP actin is quick in the free monomer and we reflect this assumption also in the structure of our model by not allowing the hydrolysis of free ATP-actin. The SPiM code for this model is given in Appendix G.

By setting the kinetic rates with respect to those for ATP-actin, ADP-P$_i$-actin and ADP-actin given in [11], we run simulations. The plots in Figure 14 show three views of these simulation with these models run with 1200 monomers (Af), 30 Arp2/3 processes (Rf) and 100 capping protein processes (Cf), at 0.05 sec., 1000.0

sec. and 10000.0 sec. These numbers of processes simulate the concentrations of the corresponding biochemical species with respect to those reported in [1] with a factoring constant of 100.

## 5 Geometric Plotting

The output of the simulations of our process models display the change in number of the processes for the biochemical species that they represent varying over the course of the simulation. However, it is often beneficial also to have a geometric representation of the processes evolving in time. For example, when models of actin polymerisation in the context of cell motility and phagocytosis are considered, their geometric representation gains importance in analysing these systems (see, e.g., [18,13]). Because the change in the actin filament meshwork in geometric space due to their polymerisation and depolymerisation is important, their geometric representation becomes a desired feature of the models.

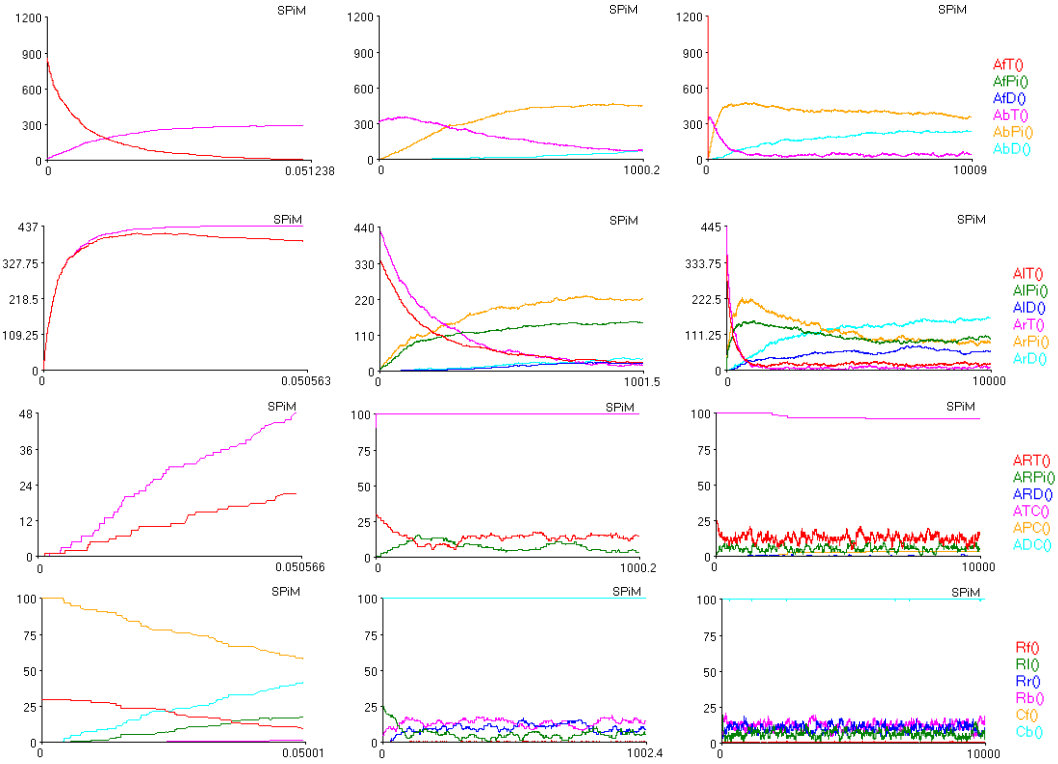



Fig. 14. The result of a simulation with the model depicted in Figure 13 at time 0.05sec., 1000.0sec and 10000.0sec, with 1200 actin monomers with three states ATP-actin, ADP-Pi-actin, and ADP-actin together with 30 Arp2/3 protein processes and 100 capping protein processes at the beginning of the simulation. The four rows of plots display the change in the population of different species of the same simulation. The first row plots the AfT, AfPi, AfD, AbT, AbPi and AbD monomers. The second row plots the AIT, AIPi, AID, ArT, ArPi and ArD monomers. The third row plots the ART, ARPi, ARD, ATC, APC and ADC monomers. The fourth row plots the Rf, Rl, Rr, Rb, Cf and Cb.

### *5.1 Encoding the Geometric Data*

In order to be able to visualise the actin filaments which are constructed by our process models, we extend our process models with coordinate parameters. The free actin processes (Af) do not have coordinate parameters, because they are assumed to be free in the cytosol. However, all the bound actin monomers are equipped with a coordinate parameter. When a free monomer binds to a filament, the free monomer evolves to a bound state, while receiving the coordinate information from the filament that it binds to. As an example for this, consider the following SPiM code:

```
let Af() = ?c(x,y,el); Al(x,y + 1.0, el)
and Al(x:float, point:float,lft:chan) =
   ( new rht@lam:chan  !c(x,point,rht); Ab(x,point,lft,rht) )
```

When the processes Af and Al interact over the channel c, the process Af receives the $x$- and $y$-coordinates of the process Al. Then, Af evolves into the state Al, while recording its coordinates as $(x, y + 1)$. Al evolves into the state Ab, keeping its coordinates unchanged, because its position in space does not change, but its state changes.

When we are modelling branching filaments, we adopt this idea to the processes Rf that represent the free Arp2/3 proteins in the cytosol. However, in this case, we need two more parameters which give the vector for the direction of the growth. Thus, each bounded monomer process has parameters for its location in space and and also parameters for its growth direction. When a filament grows along its axis, its direction vector remains unaltered. However, in case of branching, the direction vector of the daughter filament is updated with respect to a rotation matrix for 70 degrees. This is because the angle between a mother actin filament and a daughter filament is measured as 70 degrees [25]. As an example for this, consider the following SPiM code:

```
...
and Arb(x:float, y:float,
        x1:float, y1:float, lft:chan, rht:chan) =
         ( new e@lam:chan do !r(x,y,
 (x1 * 0.34) + (y1 * 0.94 ), (x1 * ( - 0.94)) + (y1 * 0.34),    e);
                       Arbb(x,y, x1, y1, e, lft, rht)
                          or !r(x,y,
 (x1 * 0.34) - (y1 * 0.94 ), (x1 * 0.94) + (y1 * 0.34),         e);
                       Arbb(x,y, x1, y1, e, lft, rht) )
...
let Rf() = ?r(x,y,x1,y1,er); Rl(x,y,x1,y1,er)
...
```

Upon interaction over the channel r, the process Arb sends its coordinate to process Rf together with its direction vector rotated with a 70 degrees rotation matrix. Because actin filaments take a rotating helical shape, the model stochastically chooses between 70 or -70 degrees with equal probability in order to be able to model the growth of a tree-shaped filament in 2D.

$$\texttt{()} \oplus L \triangleq L \tag{1}$$

$$\texttt{X(n)} \oplus L \triangleq \texttt{X(n)} :: L \quad \text{if} \quad \texttt{X(m)} = \texttt{do P1 or ... or PN} \tag{2}$$

$$\texttt{X(n)} \oplus L \triangleq \texttt{P} \oplus L \quad \text{if} \quad \texttt{X(m)} = \texttt{P} \neq \texttt{do P1 or ... or PN} \tag{3}$$

$$\texttt{P1|...|PN} \oplus L \triangleq \texttt{P1} \oplus ... \oplus \texttt{PN} \oplus L \tag{4}$$

$$\texttt{new x@r:t P} \oplus L \triangleq \texttt{P\{ x:=y\}} \oplus L \quad \text{if} \quad y \text{ isfresh} \tag{5}$$

Fig. 15. Expanding a process `P` into a list of products `X1(m1)` :: ... :: `XN(mN)`

In our simulations the monomers are assumed to be freely diffusing until they become bound to polymers. Each bound monomer is parameterised by a set of coordinates that represent its current location. These coordinates do not have any effect on the rate of interaction of the monomers. Thus, our geometric models remain consistent with the hypotheses of the Gillespie algorithm. It is also important to note that in the structure of our model, we do not allow the interaction of the monomers at the two ends of a filament. This way, we prevent loops that would result in wrong coordinates.

### *5.2 SPiM Extensions*

In order to enable geometric plotting of SPiM models, we extend the SPiM tool. This extension consists of two parts, one for outputting the simulation results as event traces and one for filtering and plotting these events.

The original algorithm for choosing the next reaction at each step of a simulation is described in detail in [19]. At each step, this algorithm chooses a reaction from the set of possible reactions that can be one of the two types: a delay or an interaction between an output and an input. We modified the simulator such that it outputs these reactions. Then, each reaction denotes an event to be plotted: the reactants of an event denote those processes that need to be removed from the plot, while the products denote the processes that need to be added to the plot at each time step.

The following events, which can be seen as ground instances of rewriting rules similar to those used in rule-based modelling approaches (see, e.g., [7,8,9]), are examples to the output of the simulator for geometric plotting. From these events, we extract the processes that are relevant for the geometric plotting together with their first two parameters which give their coordinates. By treating the left-hand-side and right-hand-side of an event as negative and positive effects of an event, we then plot the evolution of the system on the coordinate system.

```
0.214211814049 Af() Al(59.188584,47.660896,0.762146,0.623574,rht~5391)
  --> Arb(59.188,47.660,0.762,0.623,rht~5391,rht~5653) Al(59.950,48.284,0.762,0.623,rht~5653)
0.214216818695 Rf() Arb(32.3,58.3,0.77,0.63,rht~5183,rht~5215)
  --> Arbb(32.3,58.3,0.77,0.63,e~5655,rht~5183,rht~5215) Rl(32.3,58.3,0.9702,0.196,e~5655)
0.214229554433 Af() Al(40.9556,18.988,0.9702,0.196,rht~5029)
  --> Arb(40.955,18.98,0.97,0.196,rht~502,rht~5657) Al(39.985,18.792,0.97,0.196,rht~5657)
```

These events are generated as follows. In the case of a delay `delay@r;P` executed by a process `X(m)`, there is a single reactant `X(m)` with products `P`. Since `P` can be an arbitrary process, an additional function is needed to convert this process to a list of products `X1(m1)` :: ... :: `XN(mN)`. The conversion is done using an expansion function $\texttt{P} \oplus L$, which adds a process `P` to a list $L$, as defined in Figure 15.

The rules assume that each choice of actions `do P1 or ... or PN` is associ-

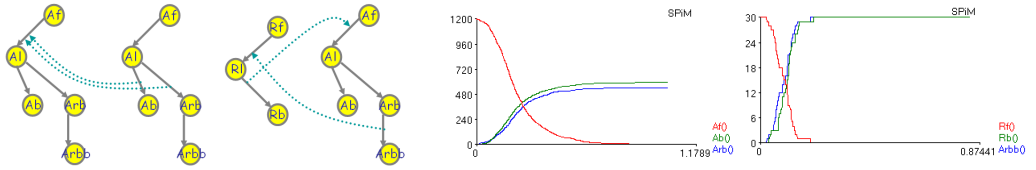



Fig. 16. Graphical representation of an actin model for geometric plotting and the result of a simulation with it.

ated with a corresponding process definition `X(m)`. This constraint is enforced by the original simulator as described in [19]. It is straightforward to prove that the expansion is compatible with the structural congruence rules of the calculus, and therefore preserves the correctness of the simulator. The expanded list of products is then added to the simulator by adapting the simulation algorithm of [19].

In the case of an interaction between an output `!x(n);P1` executed by a process `X1(m1)` and an input `?x(m);P2` executed by a process `X2(m2)`, there are two reactants `X1(m1)` and `X2(m2)` with product `P1|P2{ m:=n}`. The same expansion rules of Figure 15 are used to convert this into a list of products.

### *5.3 An Example Model for Geometric Plotting*

We give a simple model of actin polymerisation as depicted in Figure 16. In this model, because the plotting is constrained by the size of a screen, we model only the association of the monomers, but not their dissociation. [4] The SPiM code of this model is given in Appendix H. Figure 17 displays screen-shots from a movie generated by this model by using the extension of the SPiM tool.

## 6 Discussion

By using process algebra techniques, we have constructed different models of actin polymerisation, while experimenting with different structures of models, and sim-

[4] This constraint can be easily lifted by considering a smaller plotting resolution or introducing dynamic boundaries to the plots.

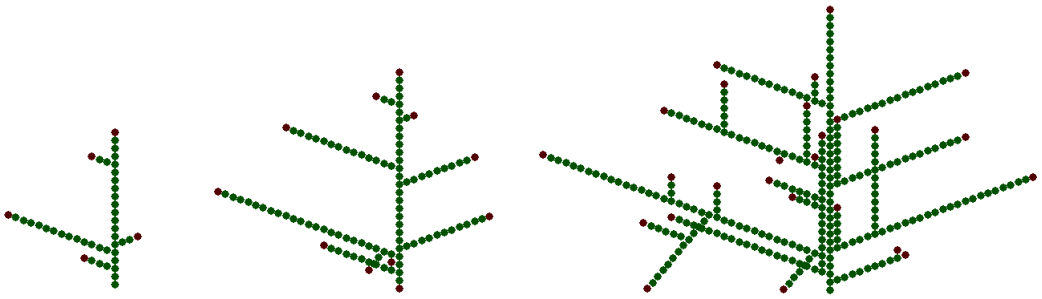


Fig. 17. Screen shots from the movie generated by the model depicted in Figure 16 which demonstrates the growth of an actin filament in time.

plifying and extending them at will. Due to to the compositionality of the process algebra approach, our models bring about a flexibility and freedom in construction of models, in contrast to models of these systems constructed by using other stochastic techniques and differential equations (see, e.g., [1,12,17]).

Our models are compositional in the sense that we can describe a single component of the system as a module of internal states, and then run multiple instances of this component in parallel in order to simulate the emergence of polymers of arbitrary lengths [5]. This contrasts with previous models of actin dynamics where each polymer of a given length needed to be modelled explicitly. Furthermore, our approach also allows coordinates to be attached to the monomers in order to observe the emergent structure of the polymers.

It is important to note that our plotting extension does not affect the simulation dynamics, and is therefore mainly a visualisation extension to the simulator. However, in order to make full use of this extension we also need to extend our models with coordinate parameters, and explicitly program how these coordinates can change over time. The fact that little modification was required to the simulator in order to output events as well as concentrations demonstrates the expressive power of pi-calculus as a programming language. In a such a setting, the main challenge is determining how to program local changes in coordinates such that the global geometric properties of a system are accurately reproduced.

A closely related approach to process algebra for representing biological components and their internal states consists in using rewriting rules to model the same systems [7,8,9]; in fact, the two approaches can be formally related [10]. Rule-based approaches are arguably higher-level because they do not need to represent the internal state space of each component, but have to rely on complex translations to produce the equivalent of process algebra expressions that can then be simulated. Only recently, sophisticated techniques have been developed to directly execute rule-based representations [16].

Influential work on adding geometric capabilities to discrete system descriptions goes back to extensions of L-Systems [23]. There is now a growing interest in adding geometric capabilities to process algebras, because of the obvious need to represent spatial information in many biological models; see for example [15].

The differential equation models of actin assembly often rely on many assumptions about the structure of the filaments. Because they require a great number of equations in order to reflect the filament structure and also lower-level mechanisms involved in these systems, their execution and analysis is expensive in terms of computational resources in comparison to higher-level models. Because process algebra models reflect the interaction of individual components with other components, in our models, we are able to lift the restrictions that are imposed on differential equation models and concentrate on the interactions of each component at a higher-level setting. This is because the behaviour of the system arises as the emergent behaviour of the components interacting with each other, as in the actual biological systems that we are modelling.

For example, one of the assumptions that is made in [1] is that the hydrolysis from ATP-actin to ADP-actin can happen only at the ends of the filaments, although hydrolysis in actin filaments is random and not restricted to the ends of the

filaments. In our model the hydrolysis to ADP-actin is not restricted to the ends of the filaments.

The mechanisms underlying actin polymerisation are complex and involve many components such as ADF/cofilin, which contributes to ADP-actin dissociation, and the profilin which is the nucleotide exchange factor for actin that catalyses the exchange of ADP for ATP [21]. Compositional construction of our models should demonstrate how such components can be included at will. Furthermore, actin polymerisation is a key component of many cellular activities such as cell motility and phagocytosis. We believe that our models can be modified and integrated into models of these other systems with the aim of generating hypothesis for wet-lab experiments. Ongoing work includes integrating an analysis on actin-filament-length, also by correlating the number of bound monomers in the filaments and monomers at the barbed-end of the filaments in our models. Another topic of future work is combining the models presented here with those in [4] to obtain models of signalling in phagocytosis resulting in actin remodelling. The geometric plotting tool should also contribute to the analysis of the models such as those for cell motility and phagocytosis. We believe that it is also useful as a debugging tool, to give a precise description on the events of the simulations.

## References


[1] Jonathan B. Alberts and Garrett M. Odell. In silico reconstitution of listeria propulsion exhibits nano-saltation. *PLOS Biology*, 2:2054–2066, 2004.

[2] Guillaume Bompard and Emmanuelle Caron. Regulation of WASP/WAVE proteins: making a long story short. *The Journal of Cell Biology*, 166(7):957–962, 2004.

[3] Luca Cardelli. Artificial biochemistry. In *Algorithmic Bioproceses*, LNCS. Springer, 2008. to appear.

[4] Luca Cardelli, Philippa Gardner, and Ozan Kahramanoğulları. A process model of rho GTP-binding proteins in the context of phagocytosis. In N. Cannata and E. Merelli, editors, *Proceedings of the First Workshop "From Biology To Concurrency and back (FBTC 2007)", September 2007*, volume 194 of *ENTCS*, pages 87–102. Elsevier, 2008.

[5] Luca Cardelli and Gianluigi Zavattaro. On the computational power of biochemistry. In *Proceedings of Third International Conference on Algebraic Biology*, LNCS. Springer, 2008. to appear.

[6] Ekta Seth Chhabra and Henry N. Higgs. The many faces of actin: matching assembly factors with cellular structures. *Nature Cell Biology*, 9:1110–1121, 2007.

[7] Vincent Danos, Jérome Feret, Walter Fontana, Russell Harmer, and Jean Krivine. Rule-based modelling of cellular signalling. In L. Caires and V. T. Vasconcelos, editors, *Concurrency Theory, Proceedings of the 18th Int. Conf., CONCUR 2007*, volume 4703 of *LNCS*, pages 17–41. Springer, 2007.

[8] Vincent Danos, Jerome Feret, Walter Fontana, and Jean Krivine. Scalable simulation of cellular signaling networks. In Zhong Shao, editor, *Proceeding of 5th Asian Symposium, APLAS 2007*, volume 4807 of *LNCS*, pages 139–157. Springer, 2007.

[9] Vincent Danos, Jerome Feret, Walter Fontana, and Jean Krivine. Abstract interpretation of cellular signalling networks. In F. Logozzo, D. Peled, and L. D. Zuck, editors, *Verification, Model Checking, and Abstract Interpretation, Proceedings of the 9th International Conference, VMCAI 2008*, volume 4905 of *LNCS*, pages 83–97. Springer, 2008.

[10] Vincent Danos and Cosimo Laneve. Formal molecular biology. *Theoretical Computer Science*, 325(1):69–110, 2004.

[11] I. Fujivara, D. Vavylonis, and T.D.Pollard. Polymerization kinetics of ADP- and ADP-Pi actin determined by fluoresence microscopy. *Proc Natl Acad Sci U S A*, 104(21):8827–8832, 2007.

[12] Jifeng Hu, Anastasios Matzavinos, and Hans G. Othmer. A theoretical approach to actin filament dynamics. *Journal of Statistical Physics*, 128(1/2):111–138, 2007.

[13] J.H Iwasa and R. D. Mullins. Spatial and temporal relationships between actin-filament nucleation, capping, and disassembly. *Current Biology*, 17:395–406, 2007.

[14] Aron B. Jaffe and Alan Hall. Dynamic changes in the length distribution of actin filaments during polymerization can be modulated by barbed end capping proteins. *Cell Motility Cytoskeleton*, 61:1–8, 2005.

[15] Mathias John, Roland Ewald, and Adeline M. Uhrmacher. A spatial extension to the pi calculus. In N. Cannata and E. Merelli, editors, *Proceedings of the First Workshop "From Biology To Concurrency and back (FBTC 2007)", September 2007*, volume 194 of *ENTCS*, pages 133–148. Elsevier, 2008.

[16] Jean Krivine. An exact and scalable stochastic simulation algorithmm. In *Proceedings of the Int. Conference of Computational Methods in Sciences and Engineering, ICCMSE 2007*, 2007.

[17] Alex Mogilner and George Oster. Force generation by actin polymerization II: The elastic ratchet and tethered filaments. *Biophysical Journal*, 84:1591–1605, 2003.

[18] Alex Mogilner and George Oster. Cell motility driven by actin polymerization. *Biophysical Journal*, 71:3030–3045, 2008.

[19] Andrew Phillips and Luca Cardelli. Efficient, correct simulation of biological processes in the stochastic pi-calculus. In *Computational Methods in Systems Biology*, volume 4695 of *LNCS*, pages 184–199. Springer, 2007.

[20] Thomas D. Pollard. Regulation of actin filament assembly by Arp2/3 complex and formins. *The Annual Review of Biophysics and Biomolecular Structure*, 36:451–477, 2007.

[21] Thomas D. Pollard and Gary G. Borisy. Cellular motility driven by assembly and disassembly of actin filaments. *Cell*, 112:453–465, 2003.

[22] C. Priami, A. Regev, E. Shapiro, and W. Silverman. Application of a stochastic name-passing calculus to representation and simulation of molecular processes. *Information Processing Letters*, 80, 2001.

[23] Prusinkiewicz, Przemyslaw, Lindenmayer, and Aristid. *The Algorithmic Beauty of Plants*. Springer, 1990.

[24] D. A. Schafer, P. B. Jennings, and J. A. Cooper. Dynamics of capping protein and actin assembly in vitro: uncapping barbed ends by polyphosphoinositides. *The Journal of Cell Biology*, 135:169–179, 1996.

[25] Alan Weeds and Sharon Yeoh. Action at the Y-branch. *Science*, 294:1660–1661, 2001.

[26] O. Wolkenhauer, M. Ullah, W. Kolch, and Cho K.H. Modeling and simulation of intracellular dynamics: choosing an appropriate framework. *IEEE Trans. Nanobioscience*, 3:200–207, 2004.

# Appendix A

The SPiM code of the model depicted in Figure 3: a model for a simple monomer of filaments which grow at the right-end and shrink at the left-end.

```
directive sample 1.0
directive plot Af(); Al(); Ar(); Ab()

val mu = 0.116      (* association *)
val lam = 0.0027    (* dissociation *)
new c@mu:chan(chan)

let Af() =
  ( new rht@lam:chan
    do ?c(lft); Al(lft)
    or !c(rht); Ar(rht) )

and Al(lft:chan) =
   ( new rht@lam:chan
    do !lft; Af() or
       !c(rht); Ab(lft,rht) )

and Ar(rht:chan) = ?rht; Af()

and Ab(lft:chan, rht:chan) = !lft; Ar(rht)

run 1200 of Af()
```

# Appendix B

The SPiM code of the model depicted in Figure 5: monomer for a branching filament with a process for Arp2/3.

```
directive sample 10000.0
directive plot Af(); Al(); Ar(); Ab();
              Rr(); Rb() ; Rl(); Rf()

val mu = 0.116
val alp = 0.116      (* association *)
val lam = 0.0027     (* dissociation *)

new c@mu:chan(chan) new r@alp:chan(chan)

let Af() = ( new rht@lam:chan
             do ?c(lft); Al(lft)
             or !c(rht); Ar(rht) )

and Al(lft:chan) =
   ( new rht@lam:chan  new mid@lam:chan
    do !lft; Af()
    or !c(rht); Ab(lft,rht)
    or !r(mid); Alm(lft,mid))

and Ar(rht:chan) = ?rht; Af()

and Ab(lft:chan, rht:chan) =
      ( new mid@lam:chan
       do !lft; Ar(rht)
       or !r(mid); Abm(lft,rht, mid) )

and Alm(lft:chan,mid:chan) =
       ( new rht@lam:chan
           do !c(rht); Abm(lft,rht,mid)
```

```
          or !lft; Am(mid) )

and Abm(lft:chan, rht:chan, mid:chan) =
        !lft; Arm(rht,mid)

and Arm(rht:chan, mid:chan) =
       do ?rht; Am(mid) or !mid; Ar(rht)

and Am(mid:chan) = !mid; Af()

let Rf() = ?r(lft); Rl(lft)

and Rl(lft:chan) = ?c(rht); Rb(lft,rht)

and Rb(lft:chan,rht:chan) = ?lft; Rr(rht)

and Rr(rht:chan) = ?rht; Rf()

run 1200 of Af()
run 30 of Rf()
```

# Appendix C

The SPiM code of the model depicted in 5: monomer for a branching filament with a process for Arp2/3. Simplified version of the previous model.

```
directive sample 10000.0
directive plot Af(); Al(); Ar(); Ab();
               Rf(); Rl(); Rr(); Rb();
               Arb()

val mu = 0.166       (* association *)
val lam = 0.0027     (* dissociation *)

new c@mu:chan(chan)
new d@mu:chan(chan)
new r@mu:chan(chan)

let Af() =
  ( new rht@lam:chan
    do ?c(lft); Al(lft)
    or ?d(er); Al(er)
    or !c(rht); Ar(rht) )

and Al(lft:chan) =
  ( new rht@lam:chan
    do !lft; Af()
    or !c(rht); Ab(lft,rht) )

and Ar(rht:chan) = ?rht; Af()

and Ab(lft:chan, rht:chan) =
         ( new e@lam:chan
           do !lft; Ar(rht)
           or !r(e); Arb(e, lft, rht) )

and Arb(e:chan,lft:chan,rht:chan) =
           ?e; Ab(lft,rht)

let Rf() =
    ?r(el); Rl(el)

and Rl(e:chan) =
     ( new er@lam:chan
        !d(er); Rb(e,er) )

and Rb(e:chan,er:chan) = !e; Rr(er)

and Rr(er:chan) =  ?er; Rf()

run 1200 of Af()
run 30 of Rf()
```

# Appendix D

The SPiM code of the left-most model depicted in Figure 9: simple monomer of filaments which grow and shrink at both ends.

```
directive sample 10000.0
directive plot Af(); Al(); Ar(); Ab()

val mu1 = 0.166
val mu2 = 0.013

val lam1 = 0.014
val lam2 = 0.008


new c@mu1:chan(chan)
new d@mu2:chan(chan)

let Af() =
  ( new rht@lam2:chan
    do ?c(lft); Al(lft)
    or ?d(rht); Ar(rht)
    or !c(rht); Ar(rht) )

and Al(lft:chan) =
  ( new rht@lam1:chan
     do !lft; Af()
     or !c(rht); Ab(lft,rht) )

and Ar(rht:chan) =
   ( new lft@lam2:chan
     do !d(lft); Ab(lft,rht)
     or ?rht; Af() )

and Ab(lft:chan, rht:chan) =
          do !lft; Ar(rht)
          or !rht; Al(lft)

run 1200 of Af()
```

# Appendix E

The SPiM code of the middle model depicted in Figure 9: simple monomer of filaments which grow and shrink at both ends together with an Arp2/3 protein process model.

```
directive sample 10000.0
directive plot Af(); Al(); Ar(); Ab(); Arb();
               Rf(); Rl(); Rr(); Rb()

val mu1 = 0.166
val mu2 = 0.013

val lam1 = 0.014
val lam2 = 0.008

new c@mu1:chan(chan)
new d@mu2:chan(chan)
new e@mu1:chan(chan)

let Af() =
  ( new rht@lam2:chan
    do ?c(lft); Al(lft)
    or ?d(rht); Ar(rht)
    or !c(rht); Ar(rht) )

and Al(lft:chan) =
  ( new rht@lam1:chan
     do !lft; Af()
     or !c(rht); Ab(lft,rht) )
```

```
and Ar(rht:chan) =
   ( new lft@lam2:chan
     do !d(lft); Ab(lft,rht)
     or ?rht; Af() )

and Ab(lft:chan, rht:chan) =
      ( new mid@lam2:chan
        do !lft; Ar(rht)
        or !rht; Al(lft)
        or !e(mid); Arb(lft,rht,mid) )

and Arb(lft:chan, rht:chan, mid:chan) =
         ?mid; Ab(lft,rht)

let Rf() =
    ?e(lft); Rl(lft)

and Rl(lft:chan) =
     ( new rht@lam1:chan
       !c(rht); Rb(lft,rht) )

and Rb(mid:chan,rht:chan) = !mid; Rr(rht)

and Rr(rht:chan) =  ?rht; Rf()

run 1200 of Af()
run 30 of Rf()
```

## Appendix F

The SPiM code of the right-most model depicted in Figure 9: simple monomer of filaments which grow and shrink at both ends together with an Arp2/3 protein process and capping protein process.

```
directive sample 0.05
directive plot Af(); Al(); Ar(); Ab();
               Arb(); AlC();
               Rf(); Rl(); Rr(); Rb();
               Cf(); Cb()

val mu1 = 0.166
val mu2 = 0.013

val lam1 = 0.014
val lam2 = 0.008

val alpha = 0.03
val beta = 0.000004

new c@mu1:chan(chan)
new d@mu2:chan(chan)
new e@mu1:chan(chan)
new p@alpha:chan(chan)

let Af() =
  ( new rht@lam2:chan
    do ?c(lft); Al(lft)
    or ?d(rht); Ar(rht)
    or !c(rht); Ar(rht) )

and Al(lft:chan) =
  ( new rht@lam1:chan
    new cap@beta:chan
     do !lft; Af()
     or !c(rht); Ab(lft,rht)
     or !p(cap); AlC(lft,cap) )

and AlC(lft:chan,cap:chan) =
      !cap; Al(lft)

and Ar(rht:chan) =
   ( new lft@lam2:chan
     do !d(lft); Ab(lft,rht)
     or ?rht; Af() )

and Ab(lft:chan, rht:chan) =
      ( new mid@lam2:chan
        do !lft; Ar(rht)
        or !rht; Al(lft)
        or !e(mid); Arb(lft,rht,mid) )

and Arb(lft:chan, rht:chan, mid:chan) =
         ?mid; Ab(lft,rht)

let Cf() = ?p(cap); Cb(cap)

and Cb(cap:chan) = ?cap; Cf()

let Rf() =
    ?e(lft); Rl(lft)

and Rl(lft:chan) =
     ( new rht@lam1:chan
       !c(rht); Rb(lft,rht) )

and Rb(mid:chan,rht:chan) = !mid; Rr(rht)

and Rr(rht:chan) =  ?rht; Rf()

run 1200 of Af()
run 30 of Rf()
run 100 of Cf()
```

## Appendix G

The SPiM code of the model depicted in Figure 13: complex monomers of filaments which grow and shrink at both ends together with an Arp2/3 protein process and capping protein process. Each monomer is in one of the states ATP-actin, ADP-Pi-actin or ADP-actin.

```
directive sample 0.05
directive plot AfT(); AfPi(); AfD();
               AlT(); AlPi(); AlD();
               ArT(); ArPi(); ArD();
               AbT(); AbPi(); AbD();
               ART(); ARPi(); ARD();
               ATC(); APC(); ADC();
               Rf(); Rl(); Rr(); Rb();
               Cf(); Cb()

val mu1 = 0.116
val mu2 = 0.013

val lam1 = 0.014
val lam2 = 0.008

val mu3 = 0.034
val mu4 = 0.0011

val lam3 = 0.002
val lam4 = 0.002

val mu5 = 0.029
val mu6 = 0.0014

val lam5 = 0.054
val lam6 = 0.0025

val alpha = 0.03
val beta = 0.000004

val atp2pi = 0.003
val pi2adp = 0.0003
```

```
val fast = 1.0
val faster = 1000.0

new c@mu1:chan(chan)
new d@mu2:chan(chan)
new e@mu1:chan(chan)

new c1@mu3:chan(chan)
new d1@mu4:chan(chan)
new e1@mu3:chan(chan)

new c2@mu5:chan(chan)
new d2@mu6:chan(chan)
new e2@mu5:chan(chan)

new p@alpha:chan(chan)

let AfT() =
  ( new rht@lam2:chan
    do ?c(lft); AlT(lft)
    or ?d(rht); ArT(rht)
    or !c(rht); ArT(rht) )

and AlT(lft:chan) =
  ( new rht@lam1:chan
    new cap@beta:chan
     do !lft; AfT()
     or !c(rht); AbT(lft,rht)
     or !p(cap); ATC(lft,cap)
     or delay@atp2pi; AlPi(lft) )

and ATC(lft:chan,cap:chan) =
      !cap; AlT(lft)

and ArT(rht:chan) =
   ( new lft@lam2:chan
     do !d(lft); AbT(lft,rht)
     or ?rht; AfT()
     or delay@atp2pi; ArPi(rht) )

and AbT(lft:chan, rht:chan) =
      ( new mid@lam2:chan
         do !lft; ArT(rht)
         or !rht; AlT(lft)
         or !e(mid); ART(lft,rht,mid)
         or delay@atp2pi; AbPi(lft, rht) )

and ART(lft:chan, rht:chan, mid:chan) =
          do ?mid; AbT(lft,rht)
          or delay@atp2pi; ARPi(lft,rht,mid)

(* ###################################### *)

and AfPi() =
  ( new rht@lam4:chan
    do ?c1(lft); AlPi(lft)
    or ?d1(rht); ArPi(rht)
    or !c1(rht); ArPi(rht)
    or delay@fast; AfD() )

and AlPi(lft:chan) =
  ( new rht@lam3:chan
    new cap@beta:chan
     do !lft; AfPi()
     or !c1(rht); AbPi(lft,rht)
     or !p(cap); APC(lft,cap)
     or delay@pi2adp; AlD(lft) )

and APC(lft:chan,cap:chan) =
      !cap; AlT(lft)

and ArPi(rht:chan) =
   ( new lft@lam4:chan
     do !d1(lft); AbPi(lft,rht)
     or ?rht; AfPi()
     or delay@pi2adp; ArD(rht) )

and AbPi(lft:chan, rht:chan) =
      ( new mid@lam4:chan
         do !lft; ArPi(rht)
         or !rht; AlPi(lft)
         or !e1(mid); ARPi(lft,rht,mid)
         or delay@pi2adp; AbD(lft, rht) )

and ARPi(lft:chan, rht:chan, mid:chan) =
          do ?mid; AbPi(lft,rht)
          or delay@pi2adp; ARD(lft,rht,mid)

(* ###################################### *)

and AfD() =
  ( new rht@lam6:chan
    do ?c2(lft); AlD(lft)
    or ?d2(rht); ArD(rht)
    or !c2(rht); ArD(rht)
    or delay@faster; AfT() )

and AlD(lft:chan) =
  ( new rht@lam5:chan
    new cap@beta:chan
     do !lft; AfD()
     or !c2(rht); AbD(lft,rht)
     or !p(cap); ADC(lft,cap) )

and ADC(lft:chan,cap:chan) =
      !cap; AlD(lft)

and ArD(rht:chan) =
   ( new lft@lam5:chan
     do !d2(lft); AbD(lft,rht)
     or ?rht; AfD() )

and AbD(lft:chan, rht:chan) =
      ( new mid@lam6:chan
         do !lft; ArD(rht)
         or !rht; AlD(lft)
         or !e2(mid); ARD(lft,rht,mid))

and ARD(lft:chan, rht:chan, mid:chan) =
           ?mid; AbD(lft,rht)

(* ###################################### *)

let Cf() = ?p(cap); Cb(cap)

and Cb(cap:chan) = ?cap; Cf()

let Rf() =
    ?e(lft); Rl(lft)

and Rl(lft:chan) =
     ( new rht@lam1:chan
        do !c(rht); Rb(lft,rht)
        or !c1(rht); Rb(lft,rht)
        or !c2(rht); Rb(lft,rht) )

and Rb(mid:chan,rht:chan) = !mid; Rr(rht)

and Rr(rht:chan) =  ?rht; Rf()

run 1200 of AfT()
run 30 of Rf()
run 100 of Cf()
```

# Appendix H

The SPiM code of the model depicted in Figure 16: monomers that only grow while branching. Bound monomer processes pass geometric plotting data to the monomers that bind to the filaments.

```
directive sample 10.0
directive plot Af(); Ar(); Ab();
               Rf(); Rb(); Arb(); Arbb()
directive debug

val lam = 0.0027      (* dissoc *)
```

```
val mu = 0.116        (* assoc *)

new c@mu:chan(float,float,float,float,chan)
new r@mu:chan(float, float, float, float, chan)

let Af() =
 ?c(x, y, x1, y1, lft);
          Al(x + x1, y + y1, x1,y1, lft)

and Al(x:float, y:float,
          x1: float, y1:float, lft:chan) =
 ( new rht@lam:chan
    do !c(x, y, x1, y1, rht);
        Ab(x,y,x1,y1,lft,rht)
    or !c(x, y, x1, y1, rht);
        Arb(x,y, x1, y1,lft,rht))

and Ab(x:float, y:float,
     x1:float, y1: float, lft:chan, rht:chan) = ()

and Arb(x:float, y:float,
  x1:float, y1:float, lft:chan, rht:chan) =
    ( new e@lam:chan
       do !r(x,y,
         (x1 * 0.34) + (y1 * 0.94 ),
         (x1 * ( - 0.94)) + (y1 * 0.34), e);
             Arbb(x,y, x1, y1, e, lft, rht)
       or !r(x,y,
         (x1 * 0.34) - (y1 * 0.94 ),
         (x1 * 0.94) + (y1 * 0.34), e);
             Arbb(x,y, x1, y1, e, lft, rht) )

and Arbb(x:float,y:float, x1:float, y1:float,
      e:chan,lft:chan,rht:chan) = ()

let Rf() = ?r(x,y,x1,y1,er); Rl(x,y,x1,y1,er)

and Rl(x:float,y:float, x1:float,y1:float,el:chan) =
   ( new e@lam:chan
        !c(x,y,x1,y1,e); Rb(x,y,x1,y1,el,e) )

and Rb(x:float, y:float,
      x1:float, y1:float, el:chan, e:chan) = ()

run 1 of ( new e@1.0:chan Al(20.0,5.0, 0.0 ,1.0, e))
run 1 of ( new e@1.0:chan Al(50.0,5.0, 0.0 ,1.0, e))
run 1 of ( new e@1.0:chan Al(60.0,5.0, 0.0 ,1.0, e))
run 1 of ( new e@1.0:chan Al(100.0,5.0, 0.0 ,1.0, e))

run 1200 of Af()
run 30 of Rf()
```


Fig. 18. Graphical representations of the models in Appendix A, Appendix B, Appendix C, Appendix D and Appendix E with explicit interaction channels.