A Pi-Calculus Model of the CD95 Receptor Mediated Pathway of Apoptosis

Jan Michael C. Yap
Department of Computer Science, UPDiliman
Melchor Hall, UP Diliman
Quezon City
jcyap@cs.engg.upd.edu.ph

ABSTRACT

In this paper, a process algebra for mobile and communicating process known as pi-calculus is used to model a certain pathway of a signal transduction process known as apoptosis, a regulated form of cell death. Mapping of the biochemical system components and processes to their pi-calculus equivalents was based on the guidelines described in a paper by Aviv Regev *et al* [2]. Qualitative analysis of the model done through specification matching and reachability analysis shows its consistency with the target system.

Keywords

Pi-Calculus, Apoptosis

1. INTRODUCTION

Mathematical and computer science methods, usually applied *in silico*, are increasingly gaining ground as tools for modeling biochemical systems [1, 2, 9]. Such models provide information as to the outcomes and parameters of certain processes and components within a biochemical system without resorting to the usual *in vivo* and *in vitro* experiments. In the realm of pathological sciences, *in silico* experiments could prove useful in that model simulations could provide insights as to how each of the components, as well as the whole system, function when there are imbalances in the system, possibly attributed to parasitic infection or internal abnormalities. These could be carried out by introducing perturbations to the normal values of either the amount of the components or the rates at which certain reactions are carried out or even the combination of both within the model [1, 2].

One such biochemical system that is given much attention nowadays is a cell activity known as *apoptosis*. Apoptosis is essentially a form of organized cell suicide, usually due to pathogenic infections or extreme imbalances occurring within a cell [6]. The utmost importance understanding the dynamics of such phenomenon is most apparent in the fight against cancer. It was found that cancer cells are abnormal cellular growth and such abnormality is attributed to the fact that these cells continuously live and proliferate: as such, they never undergo apoptotic activity despite the ill nature of the cell [5]. Thus, scientists are trying to find ways of inducing apoptotic activity within cancer cells as a means to mitigate and hamper the spread of such a disease. Wet laboratory experiments are usually employed to be able to develop certain drugs and even non-pharmacogenical methods to battle cancer. It would also be noted though that *in silico* methods are also gaining ground in the search and development of the cure as they prove to be less costly, in terms of both material and time.

2. APOPTOSIS

As an organism grows older, cells not only proliferate in order to adapt and adjust to the growth of the organism, but some of them also undergo a sort of necessary demise. Though it may seem strange, this is a very natural process - a process of "programmed cell death" called *apoptosis*. Apoptosis is a technical term used to describe the morphological processes that follows a sequence of controlled steps leading to locally and temporally defined controlled cellular self-destruction [6].

The apoptotic process is actually of utmost biological importance. During development many cells are produced in excess which eventually undergo programmed cell death and thereby contribute to sculpturing many organs and tissues. Also cells of an adult organism constantly undergo physiological cell death which must be balanced with proliferation in order to maintain homeostasis in terms of constant cell numbers. And lastly, in the event that there are damaged or potentially hazardous cells, apoptosis can help alleviate the danger by deleting the infected members [6].

Basically, there are 2 pathways through which apoptosis proceed based on where apoptotic signals originate: the *extrinsic* pathway and *intrinsic* pathway. Just recently, a third pathway termed as endoplasmic reticulum (ER) pathway was discovered, but details of which will not be discussed here. In this paper, the extrinsic pathway will be the sole focus.

Extrinsic apoptosis signalling is mediated by the activation of so called "death receptors" which are cell surface receptors that transmit apoptotic signals after ligation with specific ligands. Death receptors belong to the *tumor necrosis fac-*

tor receptor (TNFR) gene superfamily, including TNFR-1, Fas/CD95, and the TRAIL receptors DR-4 and DR-5. Subsequent signalling is mediated by the cytoplasmic part of the death receptor which contains a conserved sequence termed the *death domain* (DD). Adapter molecules like FADD or TRADD themselves possess their own DDs by which they are recruited to the DDs of the activated death receptor, thereby forming the so-called *death inducing signalling complex* (DISC). In addition to its DD, the adaptor FADD also contains a *death effector domain* (DED) which through homotypic DED-DED interaction sequesters procaspase-8 to the DISC. The local concentration of several procaspase-8 molecules at the DISC leads to their autocatalytic activation and release of active caspase-8 [6].

After the autocatalytic activation of procaspase-8 molecules finishes, active caspase-8 then processes downstream effector caspases which subsequently cleave specific substrates resulting in cell death. There are two types of extrinsic pathways identified based on how the downstream cleavage of caspases, usually termed as *caspase cascade*, proceeds: type I and type II [6].

Type I caspase cascade occurs when there is a direct and usually caspase-dependent activation of the caspase-3, -6, and -7, immediately leading to the onslaught of cell degradation and apoptosis. Type II activity on the other hand is when the signal coming from the activated receptor does not generate a caspase signalling cascade strong enough for execution of cell death on its own. In this case, the signal needs to be amplified via mitochondria-dependent apoptotic pathways. The link between the caspase signalling cascade and the mitochondria is provided by the Bcl-2 family member Bid. Bid is cleaved by caspase-8 and in its truncated form (tBID) translocates to the mitochondria where it acts in concert with the proapoptotic Bcl-2 family members Bax and Bak to induce the release of cytochrome c and other mitochondrial proapoptotic factors into the cytosol. Cytosolic cytochrome c is binding to monomeric Apaf-1 which then, in a dATP-dependent conformational change, oligomerizes to assemble the apoptosome, a complex of wheel-like structure with 7-fold symmetry, that triggers the activation of the initiator procaspase-9). Activated caspase-9 subsequently initiates a caspase cascade involving downstream effector caspases such as caspase-3, caspase-7, and caspase-6, ultimately resulting in cell death [6].

In [9], Bentele and his group studied one of the most well-known extrinsic apoptotic pathways - the CD95 induced apoptosis. CD95 is a member of the death receptor family, a subfamily of the TNFR superfamily. Cross-linking of CD95 either with its natural ligand, CD95L, or with agonistic antibodies, such as antiAPO-1, induces apoptosis in sensitive cells. The model was initiated with a structured information model of CD95-induced apoptosis by reconstructing the network topology of CD95-induced apoptosis by critically searching databases and the literature. Molecules and reactions directly or indirectly interacting with the known components of this pathway were incorporated leading to a model with approximately 70 molecules, 80 reactions, and

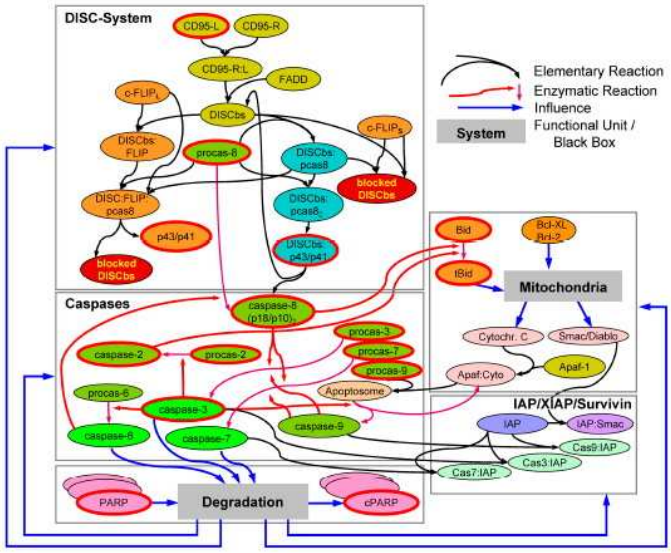

Figure 1: CD95-Induced Apoptosis Structured Information Model

more than 120 unknown parameters. To reduce the complexity of the model without sacrificing essential components of the network, subunits of different information qualities were incorporated: reactions with well-understood biochemical mechanisms, e.g., those of the DISC-system or of the caspases, were modeled mechanistically. For all other interactions, "black boxes" were introduced, defined by their experimentally observed inputoutput behavior. Subsystems were also identified and an iterative and adaptive process of decomposing the complete system into subsystems is then done. The iterative process led to the construction of a *structured information model*, wherein subsystems and their subsequent decompositions are based on the mathematical description (via differential equations) of the mechanistic parts and possible black boxes that may be encountered from a subsystem. The resulting model of CD95-induced apoptosis detailed would provide the basis of the pi-calculus model.

3. PI-CALCULUS

The notion of π-calculus was introduced by Robin Milner, Joachim Parrow, and David Walker in their seminal paper detailing a calculus that would describe computer processes and the interactions they undergo with each other [3]. Formally, π-calculus can be defined as such [3] (NOTE: Some notations were adopted from [4] and [2]):

Let N be an infinite set of *names*, with u, v, w,... ranging over the set. Let K be a set of *agent identifiers* each with an *arity* - an integer ≥ 0, with A, B, C,... ranging over the set. Let P, Q, R,... range over *agents* or *process expressions*, which can be any as follows:

1. A *summation* or *mutual exclusive operation* [1] $\sum_{i \in I} P_i =$

$P_1 + P_2 + P_3 + ...$, where each P_i is a process competing for execution with all other P_is. This means that if for some $i \in I$, P_i was chosen for execution, all other P_js, $j \neq i$, would be no longer be considered for execution. There is also a notion for an empty summation, or inaction denoted by **0**.

2. *Prefix* or *guard forms*, e.g. $\overline{y}$(x), y(x) denoting atomic actions between two or more processes. $\overline{y}$(x) is called a *negative prefix* wherein $\overline{y}$ may be thought of as an output port of an agent through which x can be passed from. y(x) is is called a *positive prefix* wherein y may be thought of as an input port of an agent through which x is a "place holder" to which names can be passed into. Also, a notion of a *silent prefix* τ denoting a silent action or delay is also introduced.

3. *Sequential actions*, e.g. P.Q denoting that Q is executed after P has finished execution.

4. *Composition* or *parallel execution* [1], e.g. $P|Q|R$ denoting concurrent or parallel execution of agents P, Q, and R.

5. *Restriction*, e.g. (new x)P denoting that ports x and $\overline{x}$ are exclusively for use by agents in P.

6. *Match*, e.g. $[x = y]$P, denoting execution of agent P if name x is equal to y, or **0** otherwise. The comparisons can also be extended to accommodate inequality (i.e. greater than (x > y) or less than (x < y)) comparisons.

7. *Parametrized agent*, e.g. $A(y_i)_{i \in I}$, where the parameters y_i is a name used within A.

8. A notion of a set of *free names of P*, denoted by FN(P), where all names included in this set are names in P not by bounded to a positive prefix or restriction.

4. MAPPING COMPONENTS AND PROCESSES OF BIOCHEMICAL SYSTEMS TO PI-CALCULUS

Applying π-calculus to model biochemical systems would have the calculus view molecular networks as mobile communication systems, where molecules play the part of computational processes, and their molecular interaction could be tagged as communication [1]. Moreso, essential properties general biochemical systems and events are covered by taking advantage of the concepts and operations presented by the calculus [2].

Molecules or substances are modeled as pi-calculus processes, and their reaction / binding sites are modeled as channels. A biochemical system is duly represented as concurrent processes (of which could also be assigned to another process). So if we are to define a system consisting of molecules A, B, and C, then we could model it as

A := ...
B := ...
C := ...
System := A | B | C

To model reactions between substances, processes that can interact are assigned complementary input and output channels. Suppose that there is a new substance formed after a reaction, then such is modeled by simulating a sequential action wherein a process of another kind (representing the new substance formed) is then created after successful communication between channels. For example, we have two substances A and B reacting to form substance C, then we have

A := $\bar{x}$()
B := x().C
C := ...

In the event that a reaction would only entail a change in the molecular state of a substance, there are two possible ways of modeling this in pi-calculus. The first one is via the *non-mobile* manner wherein a process representing the altered state is created after the reaction. For example, let A and B be two reacting substances wherein the reaction would only alter A. Via the non-mobile approach, we define another process, Altered_A representing the altered state of A and the resulting model would be

A := x().Altered_A
B := $\bar{x}$()
Altered_A := ...

In the above example, the Altered_A process will be created upon successful interaction between A and B via channel x.

The second manner via the *mobile* approach would not require the creation of a new process, but will rely on a change of name. This is done by assigning a free name as a temporary channel, and have the prior channel receive a message or name that would dictate the assignment of the temporary channel. For example, using the previous A and B processes in the non-mobile mode, we redefine the processes as

A := x(y).$\bar{y}$()
B := $\bar{x}$(z)

In the above example, B can interact with A via x. But this time, B passes a name x through the channel which would then be received by its complementary end in A (via its input argument y). Assuming successful interaction, every reference to y in A is effectively a reference to the name passed by B through x (in this case z). In that regard, A's state is changed based on the name received by the process via the preceding channel.

The summary of the mappings [2] is shown in Table 1.

Biomolecular entity	**π-calculus entity**
Molecular species	Process species
Molecular population	System of concurrent processes
Complementary motif	Complementary input and output occurrences types on the same channel name
Motif occurrence in molecule or domain	Communication offers on channels in process
Biomolecular event	**π-calculus event**
Specific interaction on complementary motif	Specific communication on complementary channels
Outcome following interaction	Communication prefix preceding process creation or other communication
Reconstitution following interaction	Communication prefix preceding process recreation
Changed molecular state following interaction	Communication prefix preceding creation of new process
Modification of motifs	Non-mobile approach: re-creation of a different type of process during interaction with a different channel set Mobile approach: Message (a tuple of channel names) sent in communication to replace channels in the receiving process

Table 1: Guidelines for abstract biomolecular systems to π-calculus

Apoptosis generally is a biochemical system, but specifically, it would categorically be called a signal transduction (ST) system. Given this idea, it is also noteworthy that π-calculus can easily be scaled to model ST networks. For ST systems, functional signaling domain would be handled as processes, component residues of domains as communication channels, and molecular interaction and modification will be treated as communication and subsequent change of channel names [1].

5. PI-CALCULUS MODEL OF THE CD-95 RECEPTOR MEDIATED APOPTOSIS

The whole apoptotic pathway is divided into 5 major subsystems: Caspase system, Mitochondrial module, DISC system, IAP system, and the Degradation blackbox. These five major subsystems are modeled as concurrent processes to form the apoptotic pathway system:

Apoptotic_pathway() := (DISC_system() | Caspase_system() | Mitochondria_module() | IAP_System() | Degradation_blackbox())

5.1 Caspase system

Procaspase2() := $\overline{cleave\_procas2\_caspase3}(rel)$.
Bound_Procaspase2(rel)

Bound_Procaspase2(rel) := rel().Caspase2()

Caspase2() := (new release_caspase2)
($\overline{cleave\_bid\_caspase2}(release\_caspase2)$.
$\overline{\text{Bound\_Caspase2(release\_caspase2)}}$) + $\overline{destroy\_molecule}()$ +
$\overline{destroy\_active\_caspase}()$

Bound_Caspase2(rel) = $\overline{rel}()$.Caspase2() + $\overline{destroy\_molecule}()$

Procaspase3() := $\overline{cleave\_procas3\_caspase8}(rel)$.
$\overline{\text{Bound\_Procaspase3(rel)}}$
+ $\overline{cleave\_procas3\_caspase9}(rel)$.Bound_Procaspase3(rel)

Bound_Procaspase3(rel) := rel().Caspase3()

Caspase3() := (new release_caspase3)
(cleave_procas2_caspase3(release_caspase3).
Bound_Caspase3(release_caspase3) +
cleave_apoptosome_caspase3(release_caspase3).
Bound_Caspase3(release_caspase3) +
cleave_procas6_caspase3(release_caspase3).
Bound_Caspase3(release_caspase3) +
influence_degradation_caspase3(release_caspase3).
Bound_Caspase3(release_caspase3)) +
(new uninhibit_caspase3)
(inhibit_caspase3(uninhibit_caspase3).
$\overline{\text{Bound\_Caspase3(uninhibit\_caspase3)}}$ +
$\overline{destroy\_molecule}()$ + $\overline{destroy\_active\_caspase}()$)

Bound_Caspase3(rel) := $\overline{rel}()$.Caspase3() + $\overline{destroy\_molecule}$

Procaspase6() := $\overline{cleave\_procas6\_caspase3}(rel)$.
Bound_Procaspase6(rel)

Bound_Procaspase6(rel) := rel().Caspase6()

Caspase6() := (new release_caspase6)
(cleave_procas8_caspase6(release_caspase6).
Bound_Caspase6(release_caspase6)) +
$\overline{\text{influence\_degradation\_caspase6}}()$ +
$\overline{destroy\_molecule}()$ + $\overline{destroy\_active\_caspase}()$

Bound_Caspase6(rel) := $\overline{rel}()$.Caspase6() + $\overline{destroy\_molecule}()$

Procaspase7() := $\overline{cleave\_procas7\_caspase8}(rel)$.
$\overline{\text{Bound\_Procaspase7(rel)}}$ +
$\overline{cleave\_procas7\_caspase9}(rel)$.Bound_Procaspase7(rel)

Bound_Procaspase7(rel) := rel().Caspase7()

Caspase7() := (new release_caspase7)
(influence_degradation_caspase7(release_caspase7).
Bound_Caspase7(release_caspase7)) +
(new uninhibit_caspase7)
(inhibit_caspase7(uninhibit_caspase7).
Bound_Caspase7(uninhibit_caspase7)) +
$\overline{destroy\_molecule}$() + $\overline{destroy\_active\_caspase}$()

Bound_Caspase7(rel) := $\overline{rel}$().Caspase7() + $\overline{destroy\_molecule}$()

Caspase8() := $\overline{destroy\_molecule}$() + $\overline{destroy\_active\_caspase}$)()
+ (new release_caspase8)
(cleave_bid_caspase8(release_caspase8).
Bound_Caspase8(release_caspase8) +
cleave_procas3_caspase8(release_caspase8).
Bound_Caspase8(release_caspase8) +
cleave_procas7_caspase8(release_caspase8).
Bound_Caspase8(release_caspase8))

Bound_Caspase8(rel) := $\overline{rel}$().Caspase8() + $\overline{destroy\_molecule}$()

Procaspase9() := bind_procas9_Apaf_Cyto()

Caspase9() := $\overline{destroy\_molecule}$() + $\overline{destroy\_active\_caspase}$()
+ (new release_caspase9)
(cleave_procas3_caspase9(release_caspase9).
Bound_Caspase9(release_caspase9) +
cleave_procas7_caspase9(release_caspase9).
Bound_Caspase9(release_caspase9)) +
(new uninhibit_caspase9)
(inhibit_caspase9(uninhibit_caspase9).
Bound_Caspase9(uninhibit_caspase9))

Bound_Caspase9(rel) := $\overline{rel}$().Caspase9() + $\overline{destroy\_molecule}$()

Caspase_system() := (Procaspase2() | Procaspase3() | Procaspase6() | Procaspase7() | Procaspase9())

5.2 Mitochondria module

TBid() := enter_mitochondria_tBid + $\overline{destroy\_molecule}$()

Bid() := $\overline{cleave\_bid\_caspase2}(rel)$.Bound_Bid(rel) +
$\overline{cleave\_bid\_caspase8}(rel)$.Bound_Bid(rel) +
$\overline{destroy\_molecule}$()

Bound_Bid(rel) := rel().TBid()

Bcl_XL_2() := enter_mitochondria_Bcl_XL_2 +
$\overline{destroy\_molecule}$()

Cytochrome_C() := bind_CytochromeC_Apaf1() +
$\overline{destroy\_molecule}$()

Smac_Diablo() := (new uninhibit_Smac_Diablo)
(inhibit_Smac_Diablo(uninhibit_Smac_Diablo).
Bound_Smac_Diablo(uninhibit_Smac_Diablo))
+ $\overline{destroy\_molecule}$()

Bound_Smac_Diablo(rel) := $\overline{rel}$().Smac_Diablo() +
$\overline{destroy\_molecule}$()

Mitochondria(tbid, bcl) :=
$\overline{enter\_mitochondria\_Bcl\_XL\_2}$().
Assess_TBid_Bcl_Concentrations(tbid, bcl + 1)
+ $\overline{enter\_mitochondria\_tBid}$().
Assess_TBid_Bcl_Concentrations(tbid + 1, bcl)

Assess_TBid_Bcl_Concentrations(tbid, bcl) := [tbid > bcl]
(Cytochrome_C() | Smac_Diablo()) + [tbid < bcl] Mitochondria(tbid, bcl)

Apaf_1() := $\overline{bind\_CytochromeC\_Apaf1}$().
Apaf1_CytochromeC() + $\overline{destroy\_molecule}$()

Apaf1_CytochromeC() := $\overline{bind\_procas9\_Apaf\_Cyto}$().
Apoptosome() +
(Cytochrome_C() | Apaf_1()) + $\overline{destroy\_molecule}$()

Apoptosome() := $\overline{cleave\_apoptosome\_caspase3}(rel)$.
Bound_Apoptosome(rel)+ (Apaf1_CytochromeC() | Caspase9())
+ $\overline{destroy\_molecule}$()

Bound_Apoptosome(rel) := rel().(Apaf1_CytochromeC() | Caspase9())

Mitochondria_module() := (Bid() | Bcl_XL_2() | Mitochondria(0, 0) | Apaf_1())

5.3 DISC system

CD95_L() := bind_CD95_L_R() + $\overline{destroy\_molecule}$()

CD95_R() := $\overline{bind\_CD95\_L\_R}$().CD95_LR() +
$\overline{destroy\_molecule}$()

CD95_LR() := bind_CD95_LR_FADD() + $\overline{destroy\_molecule}$()

FADD() := $\overline{bind\_CD95\_LR\_FADD}$().DISCbs() +
$\overline{destroy\_molecule}$()

DISCbs() := bind_DISCbs_procas8() + bind_DISCbs_cFLIPL()

+ bind_DISCbs_cFLIPS() + $\overline{destroy\_molecule}$()

CFLIPS() := $\overline{bind\_DISCbs\_cFLIPS}$().
Blocked_DISCbs_via_S() +
$\overline{bind\_DISCbs\_procas8\_cFLIPS}$().
Blocked_DISCbs_via_S() +
$\overline{destroy\_molecule}$()

Blocked_DISCbs_via_S() := ()

CFLIPL() := $\overline{bind\_DISCbs\_cFLIPL}$().
DISCbs_CFLIPL() +
$\overline{bind\_DISCbs\_procas8\_cFLIPL}$().
DISCbs_CFLIPL_Procas8() +
$\overline{destroy\_molecule}$()

DISCbs_CFLIPL() := bind_DISCbs_cFLIPL_procas8() +
$\overline{destroy\_molecule}$()

DISCbs_CFLIPL_Procas8() :=
(block_DISCbs_produce_p41_p43().P41_P43() |
$\overline{block\_DISCbs\_produce\_p41\_p43}$().
Blocked_DISCbs_via_L())

Blocked_DISCbs_via_L() := ()

P41_P43() := ()

DISCbs_p43_p41() := τ.(Caspase8() | DISCbs()) +
$\overline{destroy\_molecule}$

DISCbs_Procas8_2() := τ.DISCbs_p43_p41()

DISCbs_Procas8() := bind_DISCbs_procas8_procas8() +
bind_DISCbs_procas8_cFLIPL() +
bind_DISCbs_procas8_cFLIPS() +
$\overline{destroy\_molecule}$()

Procaspase8() := $\overline{bind\_DISCbs\_procas8}$().
DISCbs_Procas8() +
$\overline{bind\_DISCbs\_procas8\_procas8}$().
DISCbs_Procas8_2() +
$\overline{bind\_DISCbs\_cFLIPL\_procas8}$().
DISCbs_CFLIPL_Procas8() +
$\overline{cleave\_procas8\_caspase6}(rel)$.
Bound_Procaspase8(rel) +
$\overline{destroy\_molecule}$()

Bound_Procaspase8(rel) := rel().Caspase8()

DISC_system() := (CD95_L() | CD95_R() | FADD() | Procaspase8() | CFLIPL() | CFLIPS())

5.4 IAP system

IAP() := $\overline{inhibit\_caspase3}(rel)$.Bound_IAP(rel) +
$\overline{inhibit\_caspase7}(rel)$.Bound_IAP(rel) +
$\overline{inhibit\_caspase9}(rel)$.Bound_IAP(rel) +
$\overline{inhibit\_Smac\_Diablo}(rel)$.Bound_IAP(rel) +
$\overline{destroy\_molecule}$()

Bound_IAP(rel) := rel().IAP() + $\overline{destroy_molecule}$()

IAP_System() := IAP()

5.5 Degradation blackbox

PARP() := $\overline{cleave\_PARP\_caspase3}(rel)$.
Bound_PARP(rel) +
$\overline{cleave\_PARP\_caspase7}(rel)$.Bound_PARP(rel) +
$\overline{destroy\_molecule}$()

Bound_PARP(rel) := rel().CPARP()

CPARP() := $\overline{destroy\_cPARP}$()

Degradation() := (destroy_active_caspase().Degradation() |
($\overline{influence\_degradation\_caspase6}$().
(Apoptotic_Activity() | Degradation()) +
$\overline{influence\_degradation\_caspase7}(rel)$.
(cleave_PARP_caspase7(rel) | Apoptotic_Activity() | Degradation()) +
$\overline{influence\_degradation\_caspase3}(rel)$.
(cleave_PARP_caspase3(rel) | Apoptotic_Activity() | Degradation())))

Apoptotic_Activity() := destroy_cPARP().
Apoptotic_Activity() +
destroy_molecule().Apoptotic_Activity()

Degradation_blackbox() := (PARP() | Degradation())

6. QUALITATIVE ANALYSIS OF THE MODEL

To ensure the functionality of the pi-calculus model, we proceed by proving the correctness and timeliness of the response of the processes within the system being modeled. Jane Hillston cited certain methods through which process algebra models such as the one done using pi-calculus could be validated qualitatively [7]. Two methods will be used to validate the apoptosis model created: reachability analysis and specification matching .

6.1 Reachability analysis

The goal of the reachability analysis of a process algebra model is to check whether certain processes within the system is reachable given a defined starting point. Performing reachability analysis in pi-calculus models entailed tracing *process transitions*. Process transitions are the result of"simulating" the communications of the channels among processes. Each processes then undergo transitions due to the subsequent transactions through and among their channels [4]. As of the creation of this paper, there were no programs found to perform reachability analysis, and as such, the process transitions were traced manually.

In the apoptosis model, the starting point would be the reaction of the CD-95 ligand and receptor. The two endpoint processes would be the processes representing DISC blockage and apoptotic activity initiation.

For the DISC blockage, there are four process transitions identified, with two major pathway categories (one via C-FLIP L and one via C-FLIP S):
s

1. Via C-FLIP L

   (a) CD95_L() | CD95_R() → CD95_LR()
   CD95_LR () | FADD() → DISCbs()
   DISCbs() | CFLIPL() → DISCbs_CFLIPL()
   DISCbs_CFLIPL() | Procaspase8() →
   DISCbs_CFLIPL_Procas8()
   DISCbs_CFLIPL_Procas8() →
   P41_43() | Blocked_DISCbs_via_L()

   (b) CD95_L() | CD95_R() → CD95_LR()
   CD95_LR () | FADD() → DISCbs()
   DISCbs() | Procaspase8() → DISCbs_Procas8()
   DISCbs_Procas8() | CFLIPL() →
   DISCbs_CFLIPL_Procas8()
   DISCbs_CFLIPL_Procas8() →
   P41_43() | Blocked_DISCbs_via_L()

2. Via C-FLIP S

   (a) CD95_L() | CD95_R() → CD95_LR()
   CD95_LR () | FADD() → DISCbs()
   DISCbs() | CFLIPS() → Blocked_DISCbs_via_S()

   (b) CD95_L() | CD95_R() → CD95_LR()
   CD95_LR () | FADD() → DISCbs()
   DISCbs() | Procaspase8() → DISCbs_Procas8()
   DISCbs_Procas8() | CFLIPS() →
   Blocked_DISCbs_via_S()

For the process transitions leading to apoptotic activity, a total of 30 were identified. Six process transitions adhering to Type I pathways were identified, the most direct or the "shortest" (i.e. the path with the least processes involved) is detailed below:

- CD95_L() | CD95_R() → CD95_LR()
  CD95_LR () | FADD() → DISCbs()
  DISCbs() | Procaspase8() → DISCbs_Procas8()
  DISCbs_Procas8() | Procaspase8() → DISCbs_Procas8_2()
  DISCbs_Procas8_2() → DISCbs_p43_p41()
  DISCbs_p43_p41() → DISCbs() | Caspase8()
  Caspase8() | Procaspase3() →
  Bound_Caspase8() | Bound_Procaspase3()
  Bound_Caspase8() | Bound_Procaspase3() → Caspase8()
  | Caspase3()
  Caspase3() | Degradation() → Apoptotic_Activity()

Twenty-two process transitions corresponding to Type II pathways were also identified in the model. The most direct one is detailed below:

- CD95_L() | CD95_R() → CD95_LR()
  CD95_LR () | FADD() → DISCbs()
  DISCbs() | Procaspase8() → DISCbs_Procas8()
  DISCbs_Procas8() | Procaspase8() → DISCbs_Procas8_2()
  DISCbs_Procas8_2() → DISCbs_p43_p41()
  DISCbs_p43_p41() → DISCbs() | Caspase8()
  Caspase8() | Bid() → Bound_Caspase8() | Bound_Bid()
  Bound_Caspase8() | Bound_Bid() → Caspase8() | TBid()
  TBid() | Mitochondria() → Assess_TBid_Bcl_Concentrations()
  Assess_TBid_Bcl_Concentrations() →
  Smac_Diablo()| Cytochrome_C()
  Cytochrome_C() | Apaf_1() → Apaf1_CytochromeC()
  Apaf1_CytochromeC() | Procaspase9() → Apoptosome()
  Apoptosome() | Caspase3() → Bound_Apoptosome() |
  Bound_Caspase3()
  Bound_Apoptosome() | Bound_Caspase3() →
  Apaf1_CytochromeC() | Caspase9()
  Caspase9() | Procaspase3() →
  Bound_Caspase9() | Bound_Procaspase3()
  Bound_Caspase9() | Bound_Procaspase3() →
  Caspase9() | Caspase3()
  Caspase3() | Degradation() → Apoptotic_Activity()

6.2 Specification matching

The idea of specification matching is to see if the process definitions in a process algebra model is consistent with the expected activities defined in the system being modeled. To do this, we make a walk through the activities in the apoptotic process, and see the pi-calculus counterpart of those activities.

We first take a look at the DISC system, which is the starting point of the whole apoptotic process. The DISC system is composed of the following substances: the CD95 ligand, the receptor, FADD, C-FLIP$_L$, C-FLIP$_S$, and procaspase 8. All of these exist with some initial non-zero amount and undergo certain reactions. The pi-calculus equivalent of the DISC system is as such:

DISC_system() := (CD95_L() | CD95_R() | FADD() | Procaspase8() | CFLIPL() | CFLIPS())

The CD95 ligand is represented as CD95_L process the receptor as CD95_R, FADD as FADD, C-FLIP$_L$ as CFLIPL, and C-FLIP$_S$ as CFLIPS. These processes are taken to be running concurrently.

The whole apoptotic process begins when the CD95 ligand ligates the receptor to form the CD95L-R complex. In the pi-calculus model, this is the representation of the CD95 ligand and its receptor:

CD95_L() := $\overline{bind\_CD95\_L\_R}()$ +
$\overline{destroy\_molecule}()$

CD95_R() := bind_CD95_L_R().CD95_LR() +
$\overline{destroy\_molecule}()$

Here, the CD95_L process issues a signal through the channel bind_CD95_L_R. The CD95_R listens from that channel, and once a signal is received, proceeds to act as the process CD95_LR.

After the L-R complex has been created, it now binds with FADD to form the DISC binding site. The corresponding pi-calculus processes for the L-R complex and FADD are:

CD95_LR() := $\overline{bind\_CD95\_LR\_FADD}()$ +
$\overline{destroy\_molecule}()$

FADD() := bind_CD95_LR_FADD().DISCbs() +
$\overline{destroy\_molecule}()$

The process CD95_LR sends a signal through the bind_CD95_LR_FADD channel. On the other end of the channel, the FADD process waits for the signal, and once the signal is received, proceeds to act out the process DISCbs.

The DISC binding site can possibly react with three different substances: C-FLIP$_L$, C-FLIP$_S$, and procaspase 8. Each of these substances compete for binding with the DISC binding site. Here is the pi-calculus representation of the DISC binding site given the specification:

DISCbs() := bind_DISCbs_procas8() + bind_DISCbs_cFLIPL()
+ bind_DISCbs_cFLIPS() + $\overline{destroy\_molecule}()$

The DISCbs process listens through three channels: bind_DISCbs_procas8() representing interaction with procaspase 8, bind_DISCbs_cFLIPL() representing interaction with C-FLIP$_L$, and bind_DISCbs_cFLIPS() representing interaction with C-FLIP$_S$. If a signal reaches first from any of these channels, then it will proceed to act out the next action succeeding that channel and disregard signals from the other channels.

Assuming C-FLIP$_L$ successfully binds with the DISC binding site first, it then forms a DISC-FLIP complex. This DISC-FLIP complex can then bind with procaspase 8 and later produces a substance called p41/43 together with the blocked DISC binding site.

CFLIPL() := $\overline{bind\_DISCbs\_cFLIPL}()$.
DISCbs_CFLIPL() +
$\overline{bind\_DISCbs\_procas8\_cFLIPL}()$.
DISCbs_CFLIPL_Procas8() +
$\overline{destroy\_molecule}()$

DISCbs_CFLIPL() := bind_DISCbs_cFLIPL_procas8() +
$\overline{destroy\_molecule}()$

DISCbs_CFLIPL_Procas8() :=
(block_DISCbs_produce_p41_p43().P41_P43() |
$\overline{block\_DISCbs\_produce\_p41\_p43}()$.
Blocked_DISCbs_via_L())

Blocked_DISCbs_via_L() := ()

P41_P43() := ()

The DISCbs process as seen a while ago listens from the bind_DISCbs_cFLIPL channel. If it does receive a signal first on that channel, it then acts out the DISCbs_CFLIPL process, which listens through the bind_DISCbs_cFLIPL_procas8. The Procaspase8 process (representing procaspase 8) can send a signal through that channel as shown below:

Procaspase8() := $\overline{bind\_DISCbs\_procas8}()$.
DISCbs_Procas8() +
$\overline{bind\_DISCbs\_procas8\_procas8}()$.
DISCbs_Procas8_2() +
$\overline{bind\_DISCbs\_cFLIPL\_procas8}()$.
DISCbs_CFLIPL_Procas8() +
$\overline{cleave\_procas8\_caspase6}(rel)$.
Bound_Procaspase8(rel) +
$\overline{destroy\_molecule}()$

Now assuming successful transmission, the DISCbs_CFLIPL process now proceeds to act out the process DISCbs_CFLIPL_Procas8. That process in turn will act out the Blocked_DISCbs_via_L and P41_P43 processes, both "dead end" processes.

Another substance, C-FLIP$_S$ can also bind with the DISC

binding site. Successful bind would block the DISC binding site and prevent it from reacting with other substances any further.

CFLIPS() := $\overline{bind\_DISCbs\_cFLIPS}()$.
Blocked_DISCbs_via_S() +
$\overline{bind\_DISCbs\_procas8\_cFLIPS}()$.
Blocked_DISCbs_via_S() +
$\overline{destroy\_molecule}()$

Blocked_DISCbs_via_S() := ()

The CFLIPS process can send a signal to the DISCbs process via the bind_DISCbs_ cFLIPS channel. If the signal is successfully received by the DISCbs process, it will then act as the Blocked_DISCbs_via_S process, another "dead end" process.

If procaspase 8 successfully binds with DISC binding site, they would then form a DISC-procaspase 8 complex. The complex can be inhibited by either C-FLIP$_L$ or C-FLIP$_S$ or bind yet again with another instance of procaspase 8 to form the DISC-procaspase 8_2 complex.

In the pi-calculus model, it was shown that the Procaspase8 process can interact with DISCbs via the bind_DISCbs_procas8 channel. A successful communication would result in the acting out of a new process:

DISCbs_Procas8() := bind_DISCbs_procas8_procas8() +
bind_DISCbs_procas8_cFLIPL() +
bind_DISCbs_procas8_cFLIPS() +
$\overline{destroy\_molecule}()$

The DISCbs_Procas8 process can interact with the CFLIPL process (via the bind_DISCbs_procas8_cFLIPL channel) or the CFLIPS process (via the bind_DISCbs_ procas8_cFLIPS channel). Successful interaction would ultimately lead to dead end processes. But if it receives a signal first from another instance of the Procaspase8 process via the bind_DISCbs_procas8_procas8 channel, then it becomes the process:

DISCbs_Procas8_2() := τ.DISCbs_p43_p41()

After some time, the DISC-procaspase 8_2 complex will form the DISC-p43/41 complex. The complex later degrade to two substances: a DISC binding site and caspase 8.

In pi-calculus once again, after some delay the DISCbs_Procas8_2 now becomes the DISCbs_p43_p41. The process is defined as such:

DISCbs_p43_p41() := τ.(Caspase8() | DISCbs()) +
$\overline{destroy\_molecule}$

After some delay, the DISCbs_p43_p41 process then acts out two processes that will run concurrently: Caspase8 and DISCbs. The Caspase8 process is defined as:

Caspase8() := $\overline{destroy\_molecule}() + \overline{destroy\_active\_caspase})()$
+ (new release_caspase8)
(cleave_bid_caspase8(release_caspase8).
Bound_Caspase8(release_caspase8) +
cleave_procas3_caspase8(release_caspase8).
Bound_Caspase8(release_caspase8) +
cleave_procas7_caspase8(release_caspase8).
Bound_Caspase8(release_caspase8))

Now that caspase 8 has been formed, the caspase subsystem now comes into play. The molecules involved in this particular subsystem are procaspases 2, 3, 6, 7, and 9, which can be transformed into caspases after some interaction. The pi-calculus model represents this particular subsystem as such:

Caspase_system() := (Procaspase2() | Procaspase3() | Procaspase6() | Procaspase7() | Procaspase9())

Within the caspase subsystem, the formed caspase 8 may interact with either procaspase 3 or procaspsae 7 to form caspase 3 or caspase 7 respectively. It should be noted that when a caspase reacts with a substance, the reaction is enzymatic. The caspase is the enzyme, while the substance it reacts with is the substrate.

Here are the pi-calculus equivalents of procaspase 3 and 7, as well as caspase 3 and 7:

Bound_Caspase8(rel) := $\overline{rel}()$.Caspase8() + $\overline{destroy\_molecule}()$

Procaspase3() := $\overline{cleave\_procas3\_caspase8}(rel)$.
Bound_Procaspase3(rel)
+ $\overline{cleave\_procas3\_caspase9}(rel)$.Bound_Procaspase3(rel)

Bound_Procaspase3(rel) := rel().Caspase3()

Caspase3() := (new release_caspase3)
(cleave_procas2_caspase3(release_caspase3).
Bound_Caspase3(release_caspase3) +
cleave_apoptosome_caspase3(release_caspase3).
Bound_Caspase3(release_caspase3) +
cleave_procas6_caspase3(release_caspase3).
Bound_Caspase3(release_caspase3) +
influence_degradation_caspase3(release_caspase3).
Bound_Caspase3(release_caspase3)) +
(new uninhibit_caspase3)

(inhibit_caspase3(uninhibit_caspase3).
Bound_Caspase3(uninhibit_caspase3)) +
$\overline{destroy\_molecule}() + \overline{destroy\_active\_caspase}())$

Procaspase7() := $\overline{cleave\_procas7\_caspase8}(rel)$.
Bound_Procaspase7(rel) +
$\overline{cleave\_procas7\_caspase9}(rel)$.Bound_Procaspase7(rel)

Bound_Procaspase7(rel) := rel().Caspase7()

Caspase7() := (new release_caspase7)
(influence_degradation_caspase7(release_caspase7).
Bound_Caspase7(release_caspase7)) +
(new uninhibit_caspase7)
(inhibit_caspase7(uninhibit_caspase7).
Bound_Caspase7(uninhibit_caspase7)) +
$\overline{destroy\_molecule}() + \overline{destroy\_active\_caspase}()$

The process Caspase8 can either interact with the Procaspase3 process via the channel cleave_procas3_caspase8 or Procaspase7 via the channel cleave_procas3_caspase7. Since the reaction is enzymatic, the definitions follow the guideline depicted in section 4 of the previous chapter about mapping enzymatic reactions to pi-calculus. The enzyme process (Caspase8) upon successful interaction with the substrate process (Procaspase3 or Procaspase7) then proceed to act as a bound enzyme process (Bound_Caspase8) and the substrate process as a bound substrate process (Bound_Procaspase3 or Bound_Procaspase7). The release channel is defined via the restriction operator and passed via the channel used to bind the enzyme and the substrate. This is to ensure that the bound enzyme process will release the correct product process based on the corresponding substrate process it communicated with. This same principle is applied to all other enzymatic interactions. Once a signal has been passed and received through the release channel, the enzyme process (Caspase8) is restored while a product process (Caspase3 or Caspase7) is now created.

Still focusing within the caspase subsystem but now going through the path from caspase 3, the resulting caspase can cleave procaspase 2 or procaspase 6. The cleave would result in producing either caspase 2 or caspase 6. Again, the reaction is enzymatic.

The pi-calculus representations of the aforementioned processes are:

Bound_Caspase3(rel) := $\overline{rel}()$.Caspase3() +
$\overline{destroy\_molecule}$

Procaspase2() := $\overline{cleave\_procas2\_caspase3}(rel)$.
Bound_Procaspase2(rel)

Bound_Procaspase2(rel) := rel().Caspase2()

Caspase2() := (new release_caspase2)
($\overline{cleave\_bid\_caspase2}(release\_caspase2)$.
Bound_Caspase2(release_caspase2)) + $\overline{destroy\_molecule}()$ +
$\overline{destroy\_active\_caspase}()$

Procaspase6() := $\overline{cleave\_procas6\_caspase3}(rel)$.
Bound_Procaspase6(rel)

Bound_Procaspase6(rel) := rel().Caspase6()

Caspase6() := (new release_caspase6)
(cleave_procas8_caspase6(release_caspase6).
Bound_Caspase6(release_caspase6)) +
influence_degradation_caspase6() +
$\overline{destroy\_molecule}() + \overline{destroy\_active\_caspase}()$

From the definition of the Caspase3 process, it may interact with the Procaspase2 and Procaspase3 processes via the cleave_procas2_caspase3 and cleave_procas6_caspase3 channels respectively. As the reaction is enzymatic, the same principle as discussed in the case of the Caspase8-Procaspase3/7 interaction is applied.

The resulting caspase 6 can enzymatically bind with procaspase 8 to form caspase 8 yet again. This is represented in pi-calculus as an interaction via the cleave_procas8_ caspase6(release_caspase6 channel. Once a successful interaction through that channel has been done, both processes go through the bound process state (Bound_Caspase6 and Bound_Procaspase8) and later split up into the enzyme process (Caspase6) and the product process (Caspase8).

Caspase 2 and caspase 8 both can interact with a particular substance: Bid. Bid is an integral part of the type II pathway, and thus the mitochondria now is in focus. The mitochondria module consists of the molecules Bid, BclXL/2, and Apaf-1 as well the mitochondria itself. The pi-calculus model of the mitochondria module is thus defined:

Mitochondria_module() := (Bid() | Bcl_XL_2() | Mitochondria(0, 0) | Apaf_1())

Bid is cleaved by either caspase 2 or 8 to form the substance tBid. tBid enters the mitochondria. Another substance that enters the mitochondria is the BclXL/2 molecule. The mitochondria then checks if the amount of tBid that entered exceeds the amount of BclXL/2. When the tBid amount is more than that of Bcl XL/2, the mitochondria would then release stored amounts of Cytochrome C and Smac Diablo [9].

Bid() := $\overline{cleave\_bid\_caspase2}(rel)$.Bound_Bid(rel) +

$\overline{cleave\_bid\_caspase8}(rel)$.Bound_Bid(rel) + $\overline{destroy\_molecule}()$

Bound_Bid(rel) := rel().TBid()

TBid() := enter_mitochondria_tBid + $\overline{destroy\_molecule}()$

Bcl_XL_2() := enter_mitochondria_Bcl_XL_2 + $\overline{destroy\_molecule}()$

Cytochrome_C() := bind_CytochromeC_Apaf1() + $\overline{destroy\_molecule}()$

Smac_Diablo() := (new uninhibit_Smac_Diablo)
(inhibit_Smac_Diablo(uninhibit_Smac_Diablo).
Bound_Smac_Diablo(uninhibit_Smac_Diablo))
+ $\overline{destroy\_molecule}()$

Mitochondria(tbid, bcl) :=
$\overline{enter\_mitochondria\_Bcl\_XL\_2}()$.
Assess_TBid_Bcl_Concentrations(tbid, bcl + 1)
+ $\overline{enter\_mitochondria\_tBid}()$.
Assess_TBid_Bcl_Concentrations(tbid + 1, bcl)

Assess_TBid_Bcl_Concentrations(tbid, bcl) := [tbid > bcl]
(Cytochrome_C() | Smac_Diablo()) + [tbid < bcl] Mitochondria(tbid, bcl)

As seen above, the Bid process can interact with either Caspase8 or Caspase2. Interaction with either of the two would form the TBid process. Both TBid and the Bcl_XL_2 processes can interact with the Mitochondria process via the enter_ mitochondria_tBid channel and enter_mitochondria_Bcl_XL_2 channel respectively. Note here that the Mitochondria has two parameters, tbid and bcl. The two parameters represent the amount of TBid and Bcl_XL_2 entry respectively. Initially, both are set to zero (as seen in the definition of the mitochondria module). Once a signal has been received from one of the channels, the Mitochondria proceeds to act out the auxiliary process, Assess_TBid_Bcl_Concentrations(tbid, bcl). The parameters tbid and bcl represent the number of successful process interactions of TBid and Bcl_XL_2 resprectively. When a signal from the enter_mitochondria_tBid has been received first, an increment to the parameter tbid is made. When the signal comes from enter_mitochondria_Bcl_XL_2, an increment to the parameter bcl is made. The Assess_TBid_Bcl_Concentrations will proceed to one of the two competing processes via the match operator. If the value of tbid is greater than bcl ($[tbid > bcl]$), then the processes Cytochrome_C and Smac_Diablo are acted out concurrently. If the value of bcl is greater than tbid ($[tbid < bcl]$), then the Mitochondria process would still be carried out still keeping track of the current values of tbid and bcl.

Once Cytochrome C and Smac Diablo have been released, Cytochrome C can bind with Apaf-1 to form the Apaf-1-Cytochrome C complex. The complex can degrade back to its original components, or may bind with procaspase 9 to form an apoptosome. The apoptosome can have an enzymatic reaction with caspase 3 to form Caspase 9 and Apaf-1-Cytochrome C complex [9].

The implementations in pi-calculus are pretty much straightforward:

Procaspase9() := bind_procas9_Apaf_Cyto()

Apaf_1() := $\overline{bind\_CytochromeC\_Apaf1}()$.
Apaf1_CytochromeC() + $\overline{destroy\_molecule}()$

Apaf1_CytochromeC() := $\overline{bind\_procas9\_Apaf\_Cyto}()$.
Apoptosome() +
(Cytochrome_C() | Apaf_1()) + $\overline{destroy\_molecule}()$

Apoptosome() := $\overline{cleave\_apoptosome\_caspase3}(rel)$.
Bound_Apoptosome(rel)+ (Apaf1_CytochromeC() | Caspase9())
+ $\overline{destroy\_molecule}()$

Bound_Apoptosome(rel) := rel().(Apaf1_CytochromeC() | Caspase9())

Caspase 9 produced from the cleavage of apoptosome can aid in the production of executioner caspases. It can cleave procaspases 3 and 7 to produce caspase 3 and 7 respectively. Here is the pi-calculus process representing caspase 9:

Caspase9() := $\overline{destroy\_molecule}()$ + $\overline{destroy\_active\_caspase}()$
+ (new release_caspase9)
(cleave_procas3_caspase9(release_caspase9).
Bound_Caspase9(release_caspase9) +
cleave_procas7_caspase9(release_caspase9).
Bound_Caspase9(release_caspase9)) +
(new uninhibit_caspase9)
(inhibit_caspase9(uninhibit_caspase9).
Bound_Caspase9(uninhibit_caspase9))

Regulation of apoptosis is the responsibility of the IAP. IAP can bind with Smac Diablo, caspases 3, 7, and 9 to inhibit their reactive capabilities. It should be noted though that such inhibition is reversible.

The IAP system in the pi-calculus model described in the previous chapter is just composed of the IAP process.

IAP_System() := IAP()

The IAP process can interact with several other processes, namely Caspase3, Caspase7, Caspase9, and Smac_Diablo.

IAP() := $\overline{inhibit\_caspase3}(rel)$.Bound_IAP(rel) +
$\overline{inhibit\_caspase7}(rel)$.Bound_IAP(rel) +
$\overline{inhibit\_caspase9}(rel)$.Bound_IAP(rel) +
$\overline{inhibit\_Smac\_Diablo}(rel)$.Bound_IAP(rel) +
$\overline{destroy\_molecule}()$

Bound_IAP(rel) := rel().IAP() + $\overline{destroy\_molecule}()$

The inhibition by IAP is reversible, i.e. the hold of IAP on the bound substance will be relinquished after some time. This is the reason why there is a rel channel associated with the Bound_IAP process: a signal coming from the channel would represent the relinquishing of the hold of IAP over the inhibited substance.

Finally, the last component of the apoptotic pathway is the degradation module. It is composed of the PARP molecule and the degradation mechanism itself. The pi-calculus equivalent is modeled as such:

Degradation_blackbox() := (PARP() | Degradation())

Inside the degradation blackbox, PARP can be cleaved by either caspase 3 or caspase 7 to produce cPARP. Concurrently, the cleave would also influence / activate apoptotic activity. Caspase 6 can also influence apoptosis, but this time without cleaving PARP. Degradation of other molecules would proceed once apoptosis has been activated. Degradation of cPARP is defined separately. The degradation mechanism also accounts for the steady decay of active caspases [9].

PARP() := $\overline{cleave\_PARP\_caspase3}(rel)$.
Bound_PARP(rel) +
$\overline{cleave\_PARP\_caspase7}(rel)$.Bound_PARP(rel) +
$\overline{destroy\_molecule}()$

Bound_PARP(rel) := rel().CPARP()

CPARP() := $\overline{destroy\_cPARP}()$

Degradation() := (destroy_active_caspase().Degradation() |
($influence\_degradation\_caspase6$().
(Apoptotic_Activity() | Degradation()) +
$\overline{influence\_degradation\_caspase7}(rel)$.
(cleave_PARP_caspase7(rel) | Apoptotic_Activity() | Degradation()) +
$\overline{influence\_degradation\_caspase3}(rel)$.
(cleave_PARP_caspase3(rel) | Apoptotic_Activity() | Degradation())))

Apoptotic_Activity() := destroy_cPARP().
Apoptotic_Activity() +
destroy_molecule().Apoptotic_Activity()

In the pi-calculus representation above, the Degradation process can have interactions with three processes: Caspase3, Caspase6, and Caspase7. Notice that the process will be recreated upon successful interaction with any of the processes to establish persistence of the mechanism. Successful interaction with any of the three channels would proceed to the activation of the Apoptotic_Activity process. An additional action, cleave_PARP_caspase3 for interaction with the Caspase3 process or cleave_PARP_caspase7 for interaction with the Caspase7 process, is also concurrently defined to represent interaction of the processes with PARP. The interaction would again represent enzymatic reaction. The Apoptotic_Activity has a channel associated with it, destroy_molecule that can interact with all other molecules to represent molecule degradation. The destroy_cPARP channel is solely for interaction with the CPARP process. Persistence of the activity is represented by the recreation of the process after interaction through the channel.

7. CONCLUSION

Indeed, pi-calculus can provide another means by which a biochemical system such as the signal transduction pathway of receptor mediated apoptosis can be modeled. It has been shown through the reachability analysis and specification matching that the expected functionality of the pi-calculus model for the apoptotic pathway as presented is ensured. A good extension of this project would be to conduct quantitative verification of the model through simulations (using available pi-calculus simulators such as BioSpi or SPiM) or through methods described in [7, 8]. Simulations to be carried out are recommended to use the parameters (or the modification of such) described in [9] and [10] for consistency.

8. REFERENCES

[1] A. Regev, W. Silverman, and E. Shapiro, "Representing Biomolecular Processes with Computer Process Algebra: π-Calculus programs of Signal Transduction Pathways," 2000.

[2] A. Regev and E. Shapiro, "The π-calculus as an abstraction for biomolecular systems," 2004.

[3] R. Milner, J. Parrow, and D. Walker, "A Calculus of Mobile Processes (Part I and II)," 1990.

[4] R. Milner, *Communicating and Mobile Systems: the π-Calculus*, Cambridge University Press, Cambridge, UK, 1999.

[5] G. Karp, *Cell and Molecular Biology Concepts and Experiments*, 2nd edition, John Wiley and Sons, Inc., USA, 1999.

[6] A. Gewies, "Introduction to Apoptosis," *ApoReview*, 2003.

[7] J. Hillston, "Process Algebras for Quantitative Analysis" (Lecture Slides), University of Edinburgh, June 28, 2005.

[8] J. Hillston, "Quantitative Evaluation of Biological Systems" (Lecture Slides), University of Edinburgh,

June 28, 2005.

[9] M. Bentele, I. Lavrik, M. Ulrich, S. Stößer, D.W. Heermann, H. Kalthoff, P.H Krammer, and R. Eils, "Mathematical modeling reveals threshold mechanism in CD-95 induced apoptosis," *The Journal of Cell Biology*, vol. 166, 2004.

[10] Online Supplemental Material - Bentele et. al., http://www.jcb.org/cgi/doi/10.1083/jcb.200404158.