Towards Formal Reasoning about Molecular Pathways in HOL

Sohaib Ahmad, Osman Hasan and Umair Siddique
School of Electrical Engineering and Computer Science (SEECS)
National University of Sciences and Technology (NUST), Islamabad, Pakistan
Email: {11mseesahmad,osman.hasan,umair.siddique}@seecs.nust.edu.pk

Abstract—**A molecular pathway primarily refers to a chain of chemical reactions within a cell and their analysis plays a vital role in developing effective drugs for various human infectious diseases, such as Cancer and Malaria. However, the existing cell pathway analysis techniques, such as paper-and-pencil proof methods, graph theory and Petri Nets, are either incapable of assuring accurate results or only deal with certain biological systems, which limits the usage of these techniques in the safety-critical field of human medicine. In this paper, we propose to use higher-order-logic theorem proving to accurately deduce results of biological reactions in a pathway. The proposed framework is primarily based on Z-Syntax, which is a formal language to model molecular reactions and is based on three logical operators and four inference rules. As a first step towards this goal, we formalize these operators and inference rules in higher-order logic and provide automated reasoning support for verifying molecular reactions using the HOL4 theorem prover. For illustration purposes, we present the formal verification of a reaction involving TP53 degradation.**

Keywords-**HOL4; Theorem Proving; Molecular Pathways; Z-Syntax.**

I. Introduction

Molecular biology is extensively used to construct models of biological processes in the form of networks or pathways, such as protein-protein interaction networks and signaling pathways[1]. These biological networks, usually referred to as biological regulatory networks (BRNs) or gene regulatory networks (GRNs)[2], are then analyzed based on the principles of molecular biology to understand the dynamics of complex living organisms. Such an analysis plays a vital role in investigating the treatment of various human infectious diseases and future drug design targets. For example, the analysis of BRNs has been used to predict treatment decisions for sepsis patients [3] and the prediction of protein folding kinetics [4] provides valuable insights for the design of therapeutic or preventative interventions for certain classes of diseases.

Even though several pathways of different biological mechanisms have been discovered, many yet remain undiscovered. Current pace of research may take several years to discover the remaining pathways and then more time would be required to understand them[1]. Predictive analysis provides us with a very powerful replacement to traditional research in pathways discovery. Consider a scenario in which start and end of a molecular pathway is known. Deductive approach will allow us to fill the missing data using forward and backward deduction on the start and end of the molecular pathway respectively[5]. This approach is the main focus of this paper.

Traditionally, the molecular pathways are found by researchers in the form of wet-lab experiments (e.g. [6], [7]), which are very slow and expensive and may not ensure accurate results due to the inability to accurately characterize the complex biological processes in an experimental setting. The other alternatives for deducing molecular pathways include paper-and-pencil proof methods (e.g. using boolean modeling or kinetic logic [8]) or computer-based techniques (e.g. [9]) for analyzing molecular biology problems. The manual proofs become quite tedious for the large systems, where the calculation of unknown parameters takes several hundred proof steps, and are thus prone to human errors. The computer-based methods mainly consists of Petri Nets [10] and graph theory [11], but these methods are not generic and hence grasp only certain areas of molecular biology [5].

Formal methods, which are computer based mathematical analysis techniques, have the potential to overcome the above mentioned analysis inaccuracies and their usage in molecular biology is being actively investigated these days [12]. The focus of most of this research has been upon using model checking [13] and Petri nets [10]. This new trend has shown promising results in many applications of molecular biology (e.g. [14], [15]). However, the inherent state-space explosion problem of [16] model checking limits the scope of this success to the systems where the biological entities can acquire a small set of possible levels.Theorem proving [17], which is the second most widely used formal method, does not suffer from the state-space explosion problem of model checking and, has also been advocated for conducting molecular biology based analysis [18]. The main idea behind theorem proving is to construct a computer based mathematical model of the given system and then verify the properties of interest about this model based on natural deduction. The foremost requirement for conducting the theorem proving based analysis of any system is to formalize the mathematical or logical foundations required to model and analyze that system in an appropriate logic. There have been several attempts to formalize the foundations of molecular biology. For example, the earliest axiomatization even dates back to 1937 [19] and other efforts related to the formalization of biology are presented in [20]. Recent formalizations, based on K-Calculus [21] and π-Calculus [22] also include some formal reasoning support for biological systems. But the understanding and utilization of these techniques is very cumbersome for a working biologist [23].

In order to develop a biologist friendly formal framework for automated deduction of molecular reactions, we propose to formalize the Z-Syntax [5] language in higher-order logic. Z-Syntax is a formal language that supports modeling and logical deductions about molecular pathways. The main strength of Z-Syntax is its biologist-centered nature as its operators and inference rules have been designed in such a way that the they are understandable by the biologists. However, to the best of our knowledge, the logical deductions about biological processes, expressed in Z-Syntax, can only be done manually based on the paper-and-pencil based approach. This limits the usage of Z-Syntax to smaller problems and also makes the deduction process error-prone due to the human involvement. We overcome this limitation by formalizing Z-Syntax and developing formal reasoning support for mechanized deduction of biological processes expressed in Z-Syntax. In particular, we formalize the foundational operators and inference rules of Z-Syntax in higher-order logic and also present a set of functions that can be used to model the biological reactions at molecular level in Z-Syntax and deduce its correctness using a sound theorem prover. To illustrate the practical effectiveness of our development, we use it to analyze the reaction involving TP53 degradation (a well known tumor suppressor).

For now our framework only handles static reactions but Z-Syntax fecilitates modeling the concept of time in molecular reactions [5] and thus the reaction kinetics. Therefore, by choosing higher-order logic, as our modeling framework, we have the flexibility to add the reaction kinetics using calculus formalizations [24].

II. Preliminaries

This section describes Z-Syntax, higher-order-logic theorem proving and HOL4 to facilitate the understanding of the paper.

A. *Z-Syntax*

Z-Syntax [5] exploits the analogy between biological processes and logical deduction. Some of the key features of Z-syntax are: 1) the ability to express molecular reactions in a mathematical way; 2) heuristic nature, i.e., if the conclusion of a reaction is known then one can deduce the missing data from the initialization data; 3) computer implementable semantics. Z-syntax consists of the following three operators:

Z-Interaction: The interaction of two molecules is expressed by the Z-Interaction ($*$) operator. In biological reactions, Z-interaction is not associative, meaning that: $(A * B) * C \neq A * (B * C)$

Z-Conjunction: The aggregate of same or different molecules (not necessarily interacting with each other) is formed using the Z-Conjunction ($\&$) operator. For example, a bag of n molecules can be represented as $A_1\&A_2\&A_3\cdots\&A_n$. Z-conjunction is fully associative, i.e., $(A\&B)\&C = A\&(B\&C)$. However, the logical idempotence property doesn't apply here, i.e., $A\&A \neq A$.

Z-Conditional: The occurrence of a net forward reaction under a given condition is expressed using the Z-Conditional ($\rightarrow$) operator. For example, a path from A to B under the condition C can be expressed as: $A \rightarrow B$ if there is a C that allows it.

Z-Syntax consists of four inference rules that play a vital role in deducing the outcomes of biological reactions:

Elimination of Z-conditional($\rightarrow$E): If $A \rightarrow B$ can be derived from C and A can be derived from D, then B can be derived from $C\&D$, i.e.;

$$if\ C \vdash (A \rightarrow B)\ and\ (D \vdash A)\ then\ (C\&D \vdash B) \quad (1)$$

Introduction of Z-conditional($\rightarrow$I): If B can be derived from $C\&A$, then $A \rightarrow B$ can be derived from C alone.

$$C\&A \vdash B\ then\ C \vdash (A \rightarrow B) \quad (2)$$

Elimination of Z-conjunction($\&$E): If $A\&B$ can be derived from C, then both A and B can be derived from C.

$$C \vdash (A\&B)\ then\ (C \vdash A)\ and\ (C \vdash B) \quad (3)$$

Introduction of Z-conjunction($\&$I): If A can be derived from C, and B can be derived from D, then the $A\&B$ can be derived from $C\&D$.

$$(C \vdash A)\ and\ (D \vdash B)\ then\ (C\&D) \vdash (A\&B) \quad (4)$$

Besides the regular formulas that can be derived based on the above mentioned operators and inferences rule, Z-Syntax also makes use of *Empirically Valid Formulae* (EVF). These EVFs basically represent the non-logical axioms of molecular biology and are assumed to be validated empirically in the lab.

It has been shown that a biological reaction can be mapped using the above mentioned three operators and their final outcomes can be derived using the above mentioned four inference rules [5]. For illustration purposes, the stepwise derivation of three biological reactions have also been presented in [5]. The main contribution of our paper is the formalization of the above mentioned operators and inference rules in higher-order logic so that the derivation of biological reactions can be done within the sound core of HOL4.

B. *Higher-order-Logic Theorem Proving and HOL4*

Higher-order logic is a system of deduction with precise semantics and due to its high expressiveness it can be used to model complex systems that can be expressed in a closed mathematical form. Interactive theorem proving is the field of computer science and mathematical logic concerned with computer based formal proof tools that require some sort of human assistance. The core of theorem provers usually consists of a handful of axioms and primitive inference rules. Soundness is assured as new theorems can only be verified based on these primitive axioms or inference rules or any other already verified theorems in this same theorem prover. Powerful mathematical techniques such as induction and abstraction are the strengths of theorem proving. These distinguishing characteristics make higher-order logic theorem proving a very flexible verification technique that can be used to accurately analyse almost all kinds of system.

HOL4 provides an interactive theorem proving environment for the construction of mathematical proofs in higher-order logic. In order to ensure secure theorem proving, the logic in the HOL system is represented in the strongly-typed functional programming language ML. The HOL core consists of only 5 basic axioms and 8 primitive inference rules, which are implemented as ML functions. The HOL theorem prover includes many proof assistants and automatic proof procedures to assist the user in directing the proof. The user interacts with a proof editor and provides it with the necessary tactics to prove goals while some of the proof steps are solved automatically by the automatic proof procedures. HOL4 has been widely used for the formal verification of software and hardware systems and many mathematical theories. We utilized the HOL4 theories of Booleans, pairs, lists and positive integers in our work.

III. Formalization of Z-Syntax

We model the molecules as variables of arbitrary data types (α) in our formalization of Z-syntax. A list of molecules ($\alpha\ list$) represents the Z-Interaction or a molecular reaction among the elements of the list. The Z-Conjunction operator forms a collection of non-reacting molecules and can now be formalized as a list of list of molecules ($\alpha\ list\ list$). This data type allows us to apply the Z-Conjunction operator between individual molecules (a list with a single element) or multiple interacting molecules (a list with multiple elements). The Z-Conditional operator is used to update the status of molecules, i.e., generate a new set of molecules based on the available EVFs (wet-lab verified reactions). Each EVF is modeled in our formalization as a pair ($\alpha\ list\ \#\ \alpha\ list\ list$) where the first element is a list of molecules ($\alpha\ list$) indicating the reacting molecules and the second element is a list of list of molecules ($\alpha\ list\ list$) indicating the resulting set of molecules after the reaction between the molecules of the first element of the pair has taken place. A collection of EVFs is represented as a list of EVFs ($(\alpha\ list\ \#\ \alpha\ list\ list) list$) in our formalization.

The elimination of Z-Conditional rule is the same as the elimination of implication rule (Modus Ponens) in classical logic and thus it can be directly handled by the HOL4 simplification and rewriting rules. Similarly, the introduction of Z-Conditional rule can also be inferred from the rules of classical logical and can be handled by the HOL4 system without the introduction of a new inference rule. The elimination of Z-Conjunction rule allows us to infer the presence of a single molecule from a bag of inferred molecules. In HOL we apply this rule at the end of the reaction to check if the desired molecule has been obtained. Based on our data types, described above, this rule can be formalized in HOL4 by returning a particular molecule from a list of molecules:

Definition 1: Elimination of Z-Conjunction Rule

```
⊢ ∀ L m.
z_conj_elim L m = if MEM m L then [m] else L
```

The function `z_conj_elim` has the data type ($\alpha\ list \rightarrow \alpha \rightarrow \alpha\ list$). The HOL4 function `MEM m L` returns true if `m` is a member of the list `L`. Thus, the above function returns the given element as a single element in a list if it is a member of the given list. Otherwise, it returns the argument list as is.

The introduction of Z-Conjunction rule along with Z-Interaction allows us to perform a reaction between any of the available molecules during the experiment. Based on our data types, this rule is equivalent to the append operation of lists. Thus, it can be formalized as the following HOL4 function:

Definition 2: Intro of Z-Conjunction and Z-Interaction

```
⊢ ∀ L m n.
z_conj_int L m n = FLAT [EL m L; EL n L]::L
```

The above definition has the data type ($\alpha\ list\ list \rightarrow num \rightarrow num \rightarrow\ \alpha\ list\ list$). The HOL4 functions `FLAT` and `EL` are used to flatten a list of list to a single list and return a particular element of a list, respectively. Thus, the function `z_conj_int` takes a list `L` and appends the list of two of its elements `m` and `n` on its head. Mostly the initial instances of the molecules, for which the introduction of Z-conjunction is applied, have to be deleted from the main list based on the laws of stoichiometry, which can be done as follows:

Definition 3: Element Deletion

```
⊢ ∀ L. del L 0 = TL L ∧
  ∀ L n. del L (n + 1) = HD L::del (TL L) n
⊢ ∀ L m n. z_del L m n = if m > n
                 then del (del L (m + 1)) (n+1)
                 else del (del L (n + 1)) (m+1)
```

Here, the function `del` recursively deletes the element with the given index from a given list. Whereas, the main function `z_del` uses this function to delete two elements with the given indices from a given list. Note that the indices are incremented while calling the function `del` in order to compensate for the appended element at the head in the function `z_conj_int`.

The above data types and definitions can be used to formalize any molecular pathway (which is expressible using Z-Syntax) and reason about its correctness within the sound core of the HOL4 theorem prover. However, each reasoning step in this process would have to be provided by the user, which is a very cumbersome exercise. In order to minimize this effort, we describe automated reasoning support for Z-Syntax next.

IV. Formal Reasoning support for Z-Syntax

Our main objective is to develop a framework that accepts a list of initial molecules and possible EVFs and allows the user to formally deduce the final outcomes of the corresponding biological experiment within the sound core of HOL4.

A. Framework

In this regard, we first develop a function that compares a particular combination of molecules with all the EVFs and upon finding a match introduces the newly formed molecule in the initial list and deletes the consumed instances.

Definition 4: EVF Matching

```
⊢ ∀ L E m n. z_EVF L E 0 m n =
    if FST (EL 0 E) = HD L
      then (T,z_del (TL L ++ SND (EL 0 E)) m n
      else (F,TL L) ∧
  ∀ L E p m n.
```

```
z_EVF L E (p + 1) m n =
    if FST (EL (p + 1) E) = HD L
      then
        (T,z_del (TL ++ SND (EL (p + 1) E)) m n
      else z_EVF L E p m n
```

Here, HD and TL returns head and tail of a list respectively. Similarly, FST and SND returns first and second element of a pair respectively. The data type of the function `z_EVF` is: (α *list list* $\rightarrow$ (α *list*#α *list list*) *list* $\rightarrow$ *num* $\rightarrow$ *num* $\rightarrow$ *num* $\rightarrow$ *bool* # α *list list*). The function `LENGTH` returns the length of a list. The function `z_EVF` takes a list of molecules `L` and recursively checks its head, or the top most element, against all elements of the EVF list `E`. If there is no match then the function returns a pair with its first element being false (`F`), indicating that no match occurred, and the second element equal to the tail of the input list `L`. Otherwise, if a match is found then the function replaces the head of list `L` with the second element of the EVF pair and deletes the matched elements from the initial list as these elements have already been consumed. This modified list is then returned along with a true (`T`) value, which acts as a flag to indicate an element replacement.

Next, in order to deduce the final outcome of the experiment, we have to call the function `z_EVF` recursively by placing all the possible combinations of the given molecules at the head of list `L` one by one. This can be done as follows:

Definition 5: Recursive Function for variable n in Def. 4

```
⊢ ∀ L E m. z_recur1 L E m 0 =
    z_EVF (z_conj_int L m 0) E
        (LENGTH E - 1) m 0 ∧
  ∀ L E m n.
z_recur1 L E m (n + 1) =
    if FST (z_EVF (z_conj_int L M (n + 1)) E
        (LENGTH E - 1) m (n + 1)) ⇔ T
      then z_EVF (z_conj_int L m (n + 1)) E
        (LENGTH E - 1) m (n + 1)
      else z_recur1 L E m n
```

Definition 6: Recursive Function for variable m in Def. 4

```
⊢ ∀ L E n.z_recur2 L E 0 n =
    if FST (z_recur1 L E 0 n) ⇔ T
      then (T,SND (z_recur1 L E 0 n))
      else (F,SND (z_recur1 L E 0 n)) ∧
  ∀ L E m n.
z_recur2 L E (m + 1) n =
    if FST (z_recur1 L E (m + 1) n) ⇔ T
      then (T,SND (z_recur1 L E (m + 1) n))
      else z_recur2 L E m (LENGTH L - 1)
```

Both the above functions have the same data type, i.e, (α *list list* $\rightarrow$ (α *list* #α *list list*) *list* $\rightarrow$ *num* $\rightarrow$ *num* $\rightarrow$ *num* $\rightarrow$ *bool* # *alpha list list*). The function `z_recur1` calls the function `z_EVF` after appending the combination of molecules indexed by variables `m` and `n` using the introduction of Z-Conjunction rule. Thus, the new combination is always placed on the top of the list, which is passed as the variable `L` to the function `z_EVF`. Moreover, we provide the length of the EVF list (`LENGTH E - 1`) as the variable `p` of the function `z_EVF` so that the new combination is compared to all the entries of the EVF list. It is also important to note that the function `z_recur1` terminates as soon as a match in the EVF list is found. This function also returns a flag indicating the status of this match as the first element of its output pair. The second function `z_recur2` checks this flag and if it is found to be true (meaning that a match in the EVF list is found) then it terminates by returning the output list of the function `z_recur1`. Otherwise, it continues with all the remaining values of the variable `m` recursively. In the first case, i.e., when a match is found, the above two functions have to be called all over again with the new list. These iterations will continue until there is no match found in the execution of functions `z_recur1` and `z_recur2`. This behavior can be modeled by the following recursive function:

Definition 7: Final Recursion Function for Z-Syntax

```
⊢ ∀ L E m n.
z_deduction_recur L E m n 0 = (T,L) ∧
  ∀ L E m n q.
z_deduction_recur L E m n (q + 1) =
    if FST (z_recur2 L E m n) ⇔ T
      then z_deduction_recur
        (SND (z_recur2 L E m n)) E
        (LENGTH (SND (z_recur2 L E m n)) - 1)
        (LENGTH (SND (z_recur2 L E m n)) - 1) q
      else (T,SND (z_recur2 L E
        (LENGTH L - 1) (LENGTH L - 1)))
```

The function `z_deduction_rec` also has the same data type as the above mentioned recursion functions. The variable of recursion in this function should be assigned a value that is greater than the total number of EVFs so that the application of none of the EVF is missed to guarantee correct operation. Similarly, the variables `m` and `n` above should be assigned the values of (`LENGTH L - 1`) to ensure that all the combinations of the list `L` are checked against the elements of the list of EVFs. Thus, the final deduction function for Z-Syntax can be expressed in HOL4 as follows:

Definition 8: Final Deduction Function for Z-Syntax

```
⊢ ∀ L E. z_deduction L E =
    SND (z_deduction_recur L E
    (LENGTH L - 1) (LENGTH L - 1) LENGTH E)
```

The data type of the function `z_deduction` is (α *list list* $\rightarrow$ (α *list* # *alpha list list*) *list* $\rightarrow$ *alpha list list*). It accepts the initial list of molecules and the list of valid EVFs and returns a list of final outcomes of the experiment under the given conditions. Next, the output of the function `z_deduction` has to be checked if the desired molecule is present in the list. This can be done by using the elimination of the Z-Conjunction rule given in Definition 1.

The formal definitions, presented in this section, allow us to recursively check all the possible combinations of the initial molecules against the first elements of given EVFs. In case of a match, the corresponding EVF can be applied and the process can restart again to find other possible matches from the new list of molecules. This process terminates when no more molecules are found to be reacting with each other and at this point we will have the list of post-reaction molecules.

The desired result can then be obtained from these molecules using the elimination of Z-Conjunction rule. The main benefit of the development, presented in this section, is that it facilitates automated reasoning about the molecular biological experiments within the sound core of a theorem, which will be demonstrated in the next section.

B. *Z_Syntax Tactic*

Definition 8 simplifies the process of defining molecular pathways. It effectively hides all the complexities of recursion functions and provides the biologist with a simple function which only requires an Initial Aggregate of molecules and a list of EVFs. Solving this function however requires sufficient knowledge of HOL4 and ML (which is highly unlikely for most biologists). In order to simplify the automation process of above defined pathway we also developed an automatic simplifier `Z_SYNTAX_SIMP` [25] that opens up the proof with a single iteration of the function `z_deduction` and works very efficiently with the proofs involving our functions. Mainly it returns the list of molecules with new products and removed reactants after each phase of reaction.

V. Case Study: TP53 Degradation

We utilize the proposed framework to analyze the TP53 degradation reaction [5]. The regulatory loop leading to the TP53 degradation is illustrated in figure 1, where the coloured circles denote the list of molecules given by biologists and the hollow circles depict the molecules that do not interact with any other molecule in a particular step of the reaction.

A. *Verification Goal*

A theorem representing the regulatory loop involving MDM2, MDM2 and TP53 and leading to TP53 degradation [5], can be described as follows:

$$\texttt{TP53 \& TP53 \& MDM2 \& U \& P} \vdash \texttt{d(TP53)}$$

In HOL, this biological reaction can be written as the following theorem using our reported framework:

```
⊢ DISTINCT [TP53;dTP53;P;U;iMDM2;MDM2] ⟹
z_conj_elim (z_deduction
[[TP53];[TP53];[iMDM2];[U];[P]]
[([TP53;MDM2],[[MDM2]]);
 ([MDM2;TP53],[[TP53;MDM2]]);
 ([TP53;MDM2;U],[[TP53;U];[MDM2]]);
 ([TP53;U;P],[[dTP53];[U];[P]])]) [dTP53]
                                    =[[dTP53]]
```

The `DISTINCT` function used in assumption of the above theorem ensures that all the molecule variables used in this theorem represent distinct molecules. The first list argument of the function `z_deduction` is the initial aggregate (IA) of molecules that are available for reaction and the second list argument of the function `z_deduction` represents the valid EVFs for this reaction. Thus, the function `z_deduction` would deduce the final list of molecules under these particular conditions. The function `z_conj_elim` will return the molecule `dTP53` if it is present in the post-reaction list of molecules, as described previously.

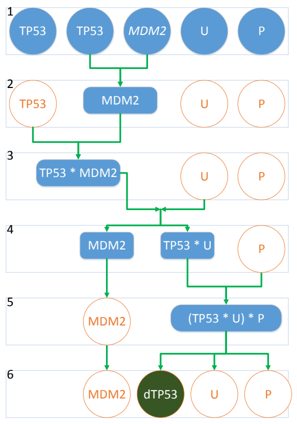

Fig. 1. Reaction representing Degradation of TP53

B. *Reaction Flow*

Figure 1 shows the step-wise reaction leading to d(TP53). The blue-coloured circles show the chemical interactions and green colour represents the desired product in the pathway, whereas each rectangle shows total number of molecules in the reaction at a given time. It is obvious from the figure that whenever a reaction yields a product, the reactants get consumed (no longer remain in the list) hence satisfying the stoichiometry of a reaction.

C. *Proof Script*

The script of TP53 Degradation Theorem is given below:

```
e(RW_TAC std_ss[DISTINCT,UNIQ_MEM,MEM]);
e(Z_SYNTAX_SIMP z_EVF_def );;
e(Z_SYNTAX_SIMP z_EVF_def );;
e(Z_SYNTAX_SIMP z_EVF_def );;
e(Z_SYNTAX_SIMP z_EVF_def );;
e(Z_SYNTAX_SIMP z_EVF_def );;
e(REWRITE_TAC [MEM,z_conj_elim_def]);
```

It is quite evident from the proof script of Theorem 1 that its proof steps can be completely automated and the proof can be done in one step as well. However, we have kept the reasoning process manual purposefully as this way the users can observe the status of the reaction at every iteration. For example, each application of `Z_SYNTAX_SIMP` on the reaction, depicted in Figure 1, would result in moving from a state n to $n + 1$.

Note that we can also record the time required to complete each iteration. For example, in case of above examples, time for each step is 2: 95.969s, 3: 51.829s, 4: 27.127s, 5: 28.062s, 6: 0.2130s, respectively. Our HOL4 proof script is available for download [25], and thus can be used for further developments and analysis of different molecular pathways.

VI. Application Perspectives

The verification presented in the above section, has been done by applying the Z-Syntax rules manually on paper in [5]. Even though the theorem (TP53 Degradation) is simple but its purpose is to enable the reader to grasp the functionality and potential of this automated reasoning. Our proposed formalization allows us to formally verify such theorems automatically within the sound core of HOL4 theorem prover. The evident benefit of our reasoning approach is its automatic nature as the user does not need to think about the proof steps and which EVFs to apply where. However, the most useful benefit of the proposed approach is its accuracy due to the inherent soundness of theorem proving. Thus, there is no risk of human error or wrong application of EVFs. We have shown that our framework is capable of modeling molecular reactions using Z-Syntax inference rules. Which means that given a set of possible EVFs, our formalism can derive a final aggregate **B** from an initial aggregate **A** automatically. If the framework fails to deduce B, it still provides the biologist with intermediate steps so that one can examine the reaction in detail and figure out the possible cause of failure. Though our method supports forward reasoning (making its way from reactants to products), but from its definitions it is easy to develop a formalism that offers backward reasoning support. In case of failure to reach final aggregate **B** we can use backward reasoning technique to start from **B** and try to reach **A**. It will allow one to predict the missing data in between the initial and final aggregate and reconstruct the path from **A** to **B**.

VII. Conclusion

In this paper, we present a biologist-friendly and generic formal reasoning for automatic mapping of molecular reactions. The proposed method inherits its biologist-friendly nature from the Z-Syntax language, which presents a deduction style formalism for molecular biology in the most biologist-centered way[5]. Moreover, the formal reasoning for some basic reactions can be handled automatically, which makes the proposed approach quite attractive. Finally, the proposed approach guarantees to provide accurate analysis due to the soundness of theorem proving. The reason behind using higher-order-logic theorem proving in our work is that though our framework does not support reaction kinetics or probabilistic nature of reactions for now, but this feature is supported by Z-Syntax. Hence, this feature can be incorporated in the proposed framework using formal reasoning support for differential equations and probabilistic reasoning.

The proposed work opens the doors to many new directions of research. Firstly, we are working on a GUI to add more user friendly features in our work. Moreover, we are also targeting some bigger case studies, such as Dysregulation of the cell cycle pathway during tumor progression [26] and Fanconi Anemia/Breast Cancer (FA/BRCA) pathway [27]. The formal verification of the functions developed in this paper for reasoning about molecular biological processes using HOL4 is also underway. Another interesting future direction is to leverage upon the high expressiveness of higher-order-logic and utilize calculus and differential theoretic reasoning to add reaction kinetics support in our formalism.

References

[1] "Biological Pathways," Jun. 2012. [Online]. Available: http://www.genome.gov/27530687

[2] F. C. e. a. G.Bernot, "Semantics of Biological Regulatory Networks," *Electr. Notes theoretical Computer Science*, vol. 180, no. 3, pp. 3–14, 2007.

[3] C. Langmead, "Generalized Queries and Bayesian Statistical Model Checking in Dynamic Bayesian Networks: Application to Personalized Medicine," in *Proc. of the 8th International Conference on Computational Systems Bioinformatics (CSB)*, 2009, pp. 201–212.

[4] C. Langmead and S. K. Jha, "Predicting Protein Folding Kinetics via Model Checking," 2007, pp. 252–264.

[5] G. Boniolo, M. D'Agostino, and P. Di Fiore, "Zsyntax: a Formal Language for Molecular Biology with Projected Applications in Text Mining and Biological Prediction," *PloS one*, vol. 5, no. 3, p. e9511, 2010.

[6] J. G. e. a. N.H. Hunt, "Immunopathogenesis of Cerebral Malaria," *Int. J. Parasitol*, vol. 36, no. 5, pp. 569–582, 2006.

[7] K. Hirayama, "Genetic Factors Associated with Development of Cerebral Malaria and Fibrotic Schistosomiasis," *Korean J. Parasitol*, 2002.

[8] R. Thomas, *Kinetic Logic: A Boolean Approach to the Analysis of Complex Regulatory Systems*. Springer-Verlag, 1979, vol. 29 Lecture Notes in Biomathematics.

[9] C. N. J.J Tyson and B. Novak, "The dynamics of Cell Cycle Regulation," *Bioessays*, vol. 24, no. 12, pp. 1095–1109, 2002.

[10] P. J. Goss and J. Peccoud, "Quantitative Modeling of Stochastic Systems in Molecular Biology by using Stochastic Petri Nets," *Proceedings of the National Academy of Sciences*, vol. 95, no. 12, pp. 6750–6755, 1998.

[11] J. Pospchal and V. Kvasnika, "Reaction graphs and a construction of reaction networks," *Theoretica chimica acta*, vol. 76, no. 6, pp. 423–435, 1990.

[12] N. Bonzanni, K. Feenstra, W. Fokkink, and E. Krepska, "What can Formal Methods bring to Systems Biology?" in *FM 2009: Formal Methods*, ser. Lecture Notes in Computer Science. Springer, 2009, vol. 5850, pp. 16–22.

[13] E. M. C. Jr., O. Grumberg, and D. A. Peled, *Model Checking*. The MIT Press, 1999.

[14] L. T. Fab. Corblin, E. Fanchon, "Applications of a Formal Approach to Decipher Discrete Genetic Networks," *BMC Bioinformatics*, vol. 11, p. 385, 2010.

[15] M. M. L. Paulevé and O. Roux, "Abstract Interpretation of Dynamics of Biological Regulatory Networks," *Electr. Notes Theor. Comput. Sci.*, vol. 272, pp. 43–56, 2011.

[16] R. Pelánek, "Fighting State Space Explosion: Review and Evaluation," in *Proc. of Formal Methods for Industrial Critical Systems (FMICS'08)*, 2008.

[17] J. Harrison, *Handbook of Practical Logic and Automated Reasoning*. Cambridge University Press, 2009.

[18] O. Wolkenhauer, D. Shibata, and M. Mesarovic, "The role of Theorem Proving in Systems Biology," *Journal of Theoretical Biology*, vol. 300, pp. 57–61, 2012.

[19] J. Woodger, A. Tarski, and W. Floyd, *The axiomatic method in biology*. The University Press, 1937.

[20] A. Zanardo and M. Rizzotti, "Axiomatization of genetics 2. Formal development," *J. Theor. Biol.*, vol. 118, no. 2, pp. 145–152, 1986.

[21] V. Danos and C. Laneve, "Formal Molecular Biology," *theoretical Computer Science*, vol. 325, no. 1, pp. 69–110, 2004.

[22] A. Regev and E. Shapiro, "Cells as computation," *Nature*, vol. 419, pp. 343–343, 2002.

[23] W. Fontana, "Systems Biology, Models, and Concurrency," *SIGPLAN Not.*, vol. 43, no. 1, pp. 1–2, Jan. 2008.

[24] J. Harrison, *Theorem Proving with the Real Numbers*. Springer-Verlag, 1998.

[25] S. Ahmad, "Automated Reasoning of Molecular Pathways in HOL - HOL Proof Script. http://save.seecs.nust.edu.pk/students/ahmad/holzsyntax.html," 2013.

[26] R. Maglietta, V. Liuzzi, E. Cattaneo, E. Laczko, A. Piepoli, A. Panza, M. Carella, O. Palumbo, T. Staiano, F. Buffoli, A. Andriulli, G. Marra, and N. Ancona, "Molecular Pathways Undergoing Dramatic Transcriptomic Changes During Tumor Development in the Human Colon," *BMC Cancer*, vol. 12, no. 1, p. 608, 2012.

[27] A. Rodríguez, D. Sosa, L. Torres, B. Molina, S. Frías, and L. Mendoza, "A Boolean Network Model of the FA/BRCA Pathway," *Bioinformatics*, vol. 28, no. 6, pp. 858–866, 2012.