Formalization of Zsyntax to Reason about Molecular Pathways in HOL4

Sohaib Ahmad[1], Osman Hasan[1], Umair Siddique[1], and Sofiéne Tahar[2]

[1] School of Electrical Engineering and Computer Science (SEECS)
National University of Sciences and Technology (NUST)
Islamabad, Pakistan
{`11mseesahmad,osman.hasan,umair.siddique`}@seecs.nust.edu.pk
[2] Department of Electrical and Computer Engineering
Concordia University
Montreal, Quebec, Canada
tahar@ece.concordia.ca

Abstract. The behavioral characterization of biological organisms is a fundamental requirement for both the understanding of the physiological properties and potential drug designs. One of the most widely used approaches in this domain is molecular pathways, which offers a systematic way to represent and analyze complex biological systems. Traditionally, such pathways are analyzed using paper-and-pencil based proofs and simulations. However, these methods cannot ascertain accurate analysis, which is a serious drawback for safety-critical applications (e.g., analysis of cancer cells and cerebral malarial network). In order to overcome these limitations, we recently proposed to formally reason about molecular pathways within the sound core of a theorem prover. As a first step towards this direction, we formally expressed three logical operators and four inference rules of Zsyntax , which is a deduction language for molecular pathways. In the current paper, we extend this formalization by verifying a couple of behavioral properties of Zsyntax based deduction using the HOL4 theorem prover. This verification not only ensures the correctness of our formalization of Zsyntax but also facilitates its usage for the formal reasoning about molecular pathways. For illustration purposes, we formally analyze a molecular reaction of the glycolytic pathway leading from D-Glucose to Fructose-1,6-bisphosphate.

1 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. The analysis of these biological networks, usually referred to as biological regulatory networks (BRNs) or gene regulatory networks (GRNs) [10], is based on the principles of molecular biology to understand the dynamics of complex living organisms. Moreover, the analysis of molecular pathways 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 recently used to predict treatment decisions for sepsis patients [15].

Traditionally, the molecular biology based analysis is carried out by biologists in the form of wet-lab experiments (e.g. [7, 13]). These experiments, despite being very slow and expensive, do not ensure accurate results due to the inability to accurately characterize the complex biological processes in an experimental setting. Other alternatives for deducing molecular reactions include paper-and-pencil proof methods (e.g. using Boolean modeling [27] or kinetic logic [28]) or computer-based techniques (e.g. [29]) for analyzing molecular biology problems. The manual proofs become quite tedious for large systems, where the calculation of unknown parameters takes several hundred proof steps, and are thus prone to human errors. The computer-based methods consist of graph theoretic techniques [21], Petri nets [11] and model checking [3]. These approaches have shown very promising results in many applications of molecular biology (e.g. [8, 14]). However, these methods are not generic and hence have been used to describe some specific areas of molecular biology [4]. Moreover, the inherent state-space explosion problem of model checking [20] limits the scope of this success only to systems where the biological entities can acquire a small set of possible levels.

Theorem proving [12], i.e., a widely used formal methods technique, does not suffer from the state-space explosion problem of model checking, and has also been advocated for conducting molecular biology based analysis [30]. 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 using deductive reasoning. 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 [31] and other efforts related to the formalization of biology are presented in [32, 25]. Recent formalizations, based on K-Calculus [6] and π-Calculus [22–24], also include some formal reasoning support for biological systems. But the understanding and utilization of these techniques is very cumbersome for a working biologist as highlighted by Fontana in [9].

In order to develop a biologist friendly formal deduction framework for reasoning about molecular reactions, we propose to formalize the Zsyntax [4] language in higher-order logic. Zsyntax is a formal language that supports modeling and logical deductions about any biological process. The main strength of Zsyntax is its biologist-centered nature as its operators and inference rules have been designed in such a way that they are understandable by the biologists. Traditionally, logical deductions about biological processes, expressed in Zsyntax , were done manually based on the paper-and-pencil based approach. This limits the usage of Zsyntax to smaller problems and also makes the deduction process error-prone due to the human involvement. As a first step towards overcoming this limitation, we formalized the logical operators and inference rules of Zsyntax in higher-order logic [2]. In the current paper, we build upon these formal definitions to verify a couple of key behavioral properties of Zsyntax based molecular pathways using the HOL4 theorem prover. The formal verification of these properties raises the confidence level in our definitions of Zsyntax operators and inference rules, which have complex interrelationships. Moreover, these formally verified properties can be used to facilitate the formal reasoning about chemical reactions at the molecular level. In order to illustrate the usefulness and effectiveness of our formalization for analyzing real-world problems in molecular biology, we present the formal analysis of a molecular reaction of the glycolytic pathway leading from D-Glucose to Fructose-1,6-bisphosphate [4].

Our current framework handles static reactions but it can be further extended to study the reaction kinetics [4] due to the flexibility of Zsyntax . The main motivation behind using higher-order-logic theorem proving in our work is to be able to leverage upon the high expressiveness of higher-order logic and thus reason about differential equations and probabilistic properties, which form an integral part of reaction kinetics. However, the scope of the current paper is on the formalization of Zsyntax based deduction calculus for molecular pathways but this formalization can later be extended to support reaction kinetics as well because it is done in a higher-order-logic theorem prover.

The rest of the paper is organized as follows: Section 2 provides an introduction to Zsyntax and the HOL4 theorem prover. The higher-order-logic formalization of Zsyntax operators and inference rules using HOL4 is described in Section 3. This is followed by the descriptions of the behavioral properties of Zsyntax along with their formal proof sketches in Section 4. The illustrative case study on the glycolytic pathway is presented in Section 4. We conclude the paper in Section 5 while highlighting some interesting potential applications of our work.

2 Preliminaries

2.1 Zsyntax

Zsyntax [4] exploits the analogy between biological processes and logical deduction. Some of the key features of Zsyntax 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. Zsyntax 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.

Z-Conjunction: The aggregate of same or different molecules (not necessarily interacting with each other) is formed using the Z-Conjunction (&) operator. Z-Conjunction is fully associative.

Z-Conditional: A path from A to B under the condition C is expressed using the Z-Conditional ($\rightarrow$) operator as: $A \rightarrow B$ if there is a C that allows it.

Zsyntax supports four inference rules, given in Table 1, that play a vital role in deducing the outcomes of biological reactions:

Table 1. Zsyntax Inference Rules

Inference Rules	**Definition**
Elimination of Z-conditional($\rightarrow$E)	`if` $C \vdash (A \rightarrow B)$ `and` $(D \vdash A)$ `then` $(C\&D \vdash B)$
Introduction of Z-conditional($\rightarrow$I)	$C\&A \vdash B$ `then` $C \vdash (A \rightarrow B)$
Elimination of Z-conjunction(&E)	$C \vdash (A\&B)$ `then` $(C \vdash A)$ `and` $(C \vdash B)$
Introduction of Z-conjunction(&I)	$(C \vdash A)$ `and` $(D \vdash B)$ `then` $(C\&D) \vdash (A\&B)$

Besides the regular formulas that can be derived based on the above mentioned operators and inference rule, Zsyntax 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 any biological reaction can be mapped and their final outcomes can be derived using the above mentioned three operators and four inference rules [4]. For example, consider a scenario in which three molecules A, B and C react with each other to yield another molecule Z. This can be represented as a Zsyntax theorem as follows:

$$A \ \& \ B \ \& \ C \vdash Z$$

The Z-Conjunction operator & is used to represent the given aggregate of molecules and then the inference rules from Table 1 are applied on these molecules along with some EVFs (chemical reactions verified in laboratories) to obtain the final product Z. For the above example, these EVFs could be:

$$A * B \rightarrow X \text{ and } X * C \rightarrow Z$$

meaning that A will react with B to yield X and X in return will react with C to yield the final product Z.

The main contribution of our paper is the formal verification of the Zsyntax based deduction method based on the higher-order-logic formalization of the above-mentioned operators and inference rules using the HOL4 theorem prover. This work will in turn facilitate the derivation of biological reactions within the sound core of HOL4.

2.2 HOL4 Theorem Prover

HOL4 is an interactive theorem prover developed at the University of Cambridge, UK, for conducting proofs in higher-order logic. It utilizes the simple type theory of Church [5] along with Hindley-Milner polymorphism [17] to implement higher-order logic. HOL4 has been successfully used as a verification framework for both software and hardware as well as a platform for the formalization of pure mathematics.

In order to ensure secure theorem proving, the logic in the HOL4 system is represented in the strongly-typed functional programming language ML [19]. An ML abstract data type is used to represent higher-order logic theorems and the only way to interact with the theorem prover is by executing ML procedures that operate on values of these data types. The HOL4 core consists of only 5 basic axioms and 8 primitive inference rules, which are implemented as ML functions. Soundness is assured as every new theorem must be verified by applying these basic axioms and primitive inference rules or any other previously verified theorems/inference rules.

A HOL4 theory is a collection of valid HOL4 types, constants, axioms and theorems, and is usually stored as a file in computers. Users can reload a HOL4 theory in the HOL4 system and utilize the corresponding definitions and theorems right away. Various mathematical concepts have been formalized and saved as HOL4 theories by the HOL4 users. We utilize the HOL4 theories of Booleans, arithmetics and lists extensively in our work. Table 2 provides the mathematical interpretations of some HOL4 symbols and functions frequently used in this paper.

Table 2. HOL4 Symbols and Functions

HOL Symbol	Standard Symbol	Meaning
$\wedge$	*and*	Logical *and*
$\vee$	*or*	Logical *or*
$\neg$	*not*	Logical *negation*
::	*cons*	Adds a new element to a list
++	*append*	Joins two lists together
`HD L`	*head*	Head element of list L
`TL L`	*tail*	Tail of list L
`EL n L`	*element*	n^{th} element of list L
`MEM a L`	*member*	True if a is a member of list L
`LENGTH L`	*length*	Length of list L
`FST`	fst (a, b) = a	First component of a pair
`SND`	snd (a, b) = b	Second component of a pair
`SUC n`	$n + 1$	Successor of a *num*

3 Formalization of Zsyntax

We modeled the molecules as variables of arbitrary data types (α) in our formalization of Zsyntax [2]. A list of molecules (α *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 (α *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 (α *list* # α *list list*) where the first element is a list of molecules (α *list*) indicating the reacting molecules and the second element is a list of list of molecules (α *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 ((α *list* # α *list list*)*list*) in our formalization.

The elimination of Z-Conditional rule is the same as the elimination of implication rule (Modus Ponens) in propositional 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 propositional logic and can be handled by the HOL4 system without the introduction of a new inference rule. The elimination of the Z-Conjunction rule allows us to infer the presence of a single molecule from an aggregate of inferred molecules. This rule is usually applied 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 (α *list* $\rightarrow \alpha \rightarrow \alpha$ *list*). 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 it 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.

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 (α *list list* $\rightarrow$ *num* $\rightarrow$ *num* $\rightarrow$ α *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.

Based on the laws of stoichiometry [4], the reacting molecules using the Z-Conjunction operator have to be deleted from the aggregate of molecules. The following function represents this behavior in our formalization:

Definition 3. Reactants Deletion
```
⊢ ∀ L m n. z_del L m n = if m > n
                then del (del L m) n
                else del (del L n) m
```

Here the function `del L m` deletes the element at index `m` of the list L and returns the updated list as follows:

Definition 4. Element Deletion

```
⊢ ∀ L. del L 0 = TL L ∧
  ∀ L n. del L (n + 1) = HD L::del (TL L) n
```

Thus, the function `z_del L m n` deletes the m^{th} and n^{th} elements of the given list `L`. We delete the higher indexed element before the lower one in order to make sure that the first element deletion does not effect the index of the second element that is required to be deleted. The above data types and definitions can be used to formalize any molecular pathway (which is expressible using Zsyntax) and reason about its correctness within the sound core of the HOL4 theorem prover.

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. 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 5. 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
```

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 equals 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.

Definition 6. Recursive Function for calling `z_EVF`

```
⊢ ∀ 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 data type of function z_deduction_recur is (α *list list* $\rightarrow$ (α *list* #α *list list*) *list* $\rightarrow$ *num* $\rightarrow$ *num* $\rightarrow$ *num* $\rightarrow$ *bool* # α *list list*). It accepts the list of molecules L and the list of EVFs E along with their corresponding indices m and n, respectively, and a recursion variable q. It returns a pair with the first element being a Boolean flag, which becomes true when there are no more remaining reactions, and the second element being the list of molecules representing the post-reaction state. The function z_decuction_recur recursively calls the function z_EVF for all possible molecule combinations using the function z_recur2, which in turn uses the function z_recur1 for this purpose. The arguments m and n of functions z_recur1 and z_recur2 are initialized with LENGTH L and the sole purpose of these functions is to exhaust all possible combinations of the variables m and n for the function z_conj_int, given in Definition 5. The formalization of the above mentioned functions and more details about their behavior can be obtained from [1, 2].

In order to model a complete experiment for a given list of molecules, the variable of recursion in the function z_deduction_recur 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. Similarly, the variables m and n of the function z_deduction_recur should be assigned the values of (LENGTH L - 1) to ensure that all combinations of the list L are checked against the elements of the list of EVFs. Thus, the final deduction function for Zsyntax can be expressed in HOL4 as follows:

Definition 7. Final Deduction Function for Zsyntax

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

The data type of function z_deduction is (α *list list* $\rightarrow$ (α *list* # α *list list*) *list* $\rightarrow$ α *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, by calling the function z_decuction_recur.

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 is applied by replacing the reacting molecules with their outcome in the molecule list and the process restarts 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, given in Definition 1. 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 prover.

4 Formal Verification of Zsyntax Properties

In order to ensure the correctness and soundness of our definitions, we use them to verify a couple of properties representing the most important characteristics of molecular reactions. The first property deals with the case when there is no combination of reacting molecules in the list of molecules and in this case we verify that after the Zsyntax based experiment execution both the pre and post-experiment lists of molecules are the same. The second property captures the behavior of the scenario when the given list of molecules contains only one set of reacting molecules and in this case we verify that after the Zsyntax based experiment execution the post-experiment list of molecules contains the product of the reacting molecules minus its reactants along with the remaining molecules provided initially. We represent these scenarios as formally specified properties in higher-order logic using our formal definitions, given in the previous section. These properties are then formally verified in HOL4.

4.1 Scenario 1: No Reaction

We verify the following theorem for the first scenario:

Theorem 1.

```
⊢ ∀ E L.
        ∼(NULL E) ∧ ∼(NULL L) ∧
        (∀ a m n. MEM a E ∧ m < LENGTH L ∧ n < LENGTH L
              ⇒ ∼MEM (FST a) [HD (z_conj_int L m n)])
        ⇒ z_deduction L E = L
```

The variables `E` and `L` represent the lists of EVFs and molecules, respectively. The first two assumptions ensure that both of these lists have to be non-empty, which are the pre-conditions for a molecular reaction to take place. The next conjunct in the assumption list of Theorem 1 represents the formalization of the no-reaction-possibility condition as according to this condition no first element of any pair in the list of EVFs `E` is a member of the head of the list formed by the function `z_conj_int`, which picks the elements corresponding to the two given indices (that range over the complete length of the list of molecules `L`) and appends them as a flattened single element on the given list `L`. This constraint is quantified for all variables `a`, `m` and `n` and thus ensures that no combination of molecules in the list `L` matches any one of the first elements of the EVF list `E`. Thus, under this constraint, no reaction can take place for the given lists `L` and `E`. The conclusion of Theorem 1 represents the scenario that the output of our formalization of Zsyntax based reaction would not make any change in the given molecule list `L` and thus verifies that under the no-reaction-possibility condition our formalization also did not update the molecule list.

The verification of this theorem is interactively done by ensuring the no-update scenario for all molecule manipulation functions, i.e., `z_EVF`, `z_recur1`, `z_recur2` and `z_deduction_recur`, under the no-reaction-possibility condition [1]. For example, the corresponding theorem for `z_EVF` function is as follows:

Theorem 2.

```
⊢ ∀ E L m n P.
        ∼(NULL E) ∧ ∼(NULL L) ∧ m < LENGTH L ∧ n < LENGTH L ∧
        P < LENGTH E ∧ (∀ a. MEM a E ⇒ ∼MEM (FST a) [HD L])
        ⇒ z_EVF L E P m n = (F,TL L)
```

The assumptions of above theorem ensure that both lists L and E are not empty and the arguments of the function z_EVF are bounded by the LENGTH of L and E. The last conjunct in the assumption list models the no-reaction-possibility condition in the context of the function z_EVF. The conclusion of the theorem states that no update takes place under the given conditions by ensuring that the function z_EVF returns a pair with the first element being F (False), representing no match, and the second element being equal to TL L, which is actually equal to the original list L since an element was appended on head of L by the parent function.

4.2 Scenario 2: Single Reaction

The second scenario complements the first scenario and caters for the case when a reaction is possible and we verify that the molecules list is indeed updated based on the outcomes of that reaction. In order to be able to track the reaction and the corresponding update, we limit ourselves to only one reaction in this scenario but since we verify a generic theorem (universally quantified) for all possibilities our result can be extended to cater for multiple reactions as well. The theorem corresponding to this scenario 2 is as follows:

Theorem 3.

```
⊢ ∀ E L z m' n'.
      ∼NULL E ∧ ∼NULL (SND (EL z E)) ∧ 1 < LENGTH L ∧
      m' ≠ n' ∧ m' < LENGTH L ∧ n' < LENGTH L ∧ z < LENGTH E ∧
      ALL_DISTINCT (L ++ SND (EL z E)) ∧
      (∀ a b. a ≠ b ⇒ FST (EL a E) ≠ FST (EL b E)) ∧
      (∀ K m n. m < LENGTH K ∧ n < LENGTH K ∧
      (∀ j. MEM j K ⇒ MEM j L ∨ ∃ q. MEM q E ∧ MEM j (SND q)) ⇒
        if (EL m K = EL m' L) ∧ (EL n K = EL n' L)
          then HD (z_conj_int K m n) = FST (EL z E)
          else ∀ a. MEM a E ⇒ FST a ≠ HD (z_conj_int K m n))
      ⇒ z_deduction L E = z_del (L ++ SND (EL z E)) m' n'
```

The first two assumptions ensure that neither the list E, i.e., the list of EVFs, nor the second element of the pair at index z of the list E is empty. Similarly, the third assumption ensures that the list L, i.e., the list of initial molecules, contains at least two elements. These constraints ensure that we can have at least one reaction with the resultant being available at index z of the EVF list. The next four assumptions ensure that the indices m' and n' are distinct and these along with the index z fall within the range of elements of their respective lists of molecules L or EVFs E. According to the next assumption, i.e., ALL DISTINCT

(L ++ SND (EL z E)), all elements of the list L and the resulting molecules of the EVF at index z are distinct, i.e., no molecule can be found two or more times in the initial list L or the post-reaction list E. The next assumption, i.e., ($\forall$ a b. a $\neq$ b $\Rightarrow$ FST (EL a E) $\neq$ FST (EL b E)), guarantees that all first elements of the pairs in list E are also distinct. Note that this is different from the previous condition since the list E contains pairs as elements and the uniqueness of the pairs does not ensure the uniqueness of its first elements. The final condition models the presence of only one pair of reactants scenario. According to the assumptions of this implication condition, the variable K is used to represent a list that only has elements from list L or the second elements of the pairs in list E. Thus, it models the molecules list in a live experiment. Moreover, the variables m and n represent the indices of the list K and thus they must have a value less than the total elements in the list K (since the first element is indexed 0 in the HOL4 formalization of lists). Now, if the indices m and n become equal to m' and n', respectively, then the head element of the z_conj_int K m n would be equal to FST of EL z E. Otherwise, for all other values of indices m and n, no combination of molecules obtained by HD(Z_conj_int K m n) would be equal to the first element of any pair of the list E. Thus, the if case ensures that the variables m' and n' point to the reacting molecules in the list of molecules L and the variable z points to their corresponding resultant molecule in the EVF list. Moreover, the else case ensures that there is only one set of reacting molecules in the list L. The conclusion of the theorem formally describes the scenario when the resulting element, available at the location z of the EVF list, is appended to the list of molecules while the elements available at the indices m' and n' of L are removed during the execution of the function z_deduction on the given lists L and E.

The proof of Theorem 3 is again based on verifying sub-goals corresponding to this scenario for all the sub-functions, i.e., z_EVF, z_recur1, z_recur2 and z_deduction_recur. The formal reasoning for all of these proofs involved various properties of the del function for a list element and some of the key theorems developed for this purpose in our development are given in Table 3 and more details can be found in [1].

The formalization described in this section consumed about 500 man hours and approximately 2000 lines of HOL4 code, mainly due to the undecidable nature of higher-order logic. However, this effort raises the confidence level on the correctness of our formalization of Zsyntax . This fact distinguishes our work from all the other formal methods based techniques used in the context of BRNs, where the deduction rules are applied without being formally checked. Moreover, our formally verified theorems can also be used in the formal analysis of molecular pathways. The assumptions of these theorems provide very useful insights about the constraints under which a reaction or no reaction would take place. To the best of our knowledge, this is the first time that properties, like Theorems 1 and 3, about a molecular pathway experiment have been formally verified. Thus, the identification of these properties and their formal verification both constitute contributions of this paper.

Table 3. Formally Verified Properties of the `del` Function

Signature	**Theorem**
`del_ASSOC_THM`	$\vdash \forall$ L E m. m $<$ LENGTH L $\Rightarrow$ del (L $++$ E) m $=$ del L m $++$ E
`del_LENGTH_THM`	$\vdash \forall$ L E m. m $<$ LENGTH L $\Rightarrow$ LENGTH (del L m) $=$ LENGTH L $-$ 1
`del_EL_THM`	$\vdash \forall$ L m n. m $<$ n $\wedge$ n $<$ LENGTH L $\wedge$ 1 $<$ LENGTH L $\Rightarrow$ EL m L $=$ EL m (del L n)
`del_DISTINCT_THM`	$\vdash \forall$ L n. n $<$ LENGTH L $\wedge$ ALL_DISTINCT L $\Rightarrow$ ALL_DISTINCT (del L n)
`del_MEM_THM`	$\vdash \forall$ L a m. m $<$ LENGTH L $\wedge$ MEM a (del L m) $\Rightarrow$ MEM a L
`del_NOT_MEM_THM`	$\vdash \forall$ L m. ALL_DISTINCT L $\wedge$ m $<$ LENGTH L $\Rightarrow\sim$ MEM (EL m L) (del L m)

5 Case Study: Pathway leading to Fructose-1,6-bisphosphate

Formation of Fructose-1,6-bisphosphate (F1,6P) is an intermediate step in glycolysis, i.e., a sequence of enzyme catalyzed reaction that breaks down glucose and forms pyruvate, which is then used to supply energy to living cells through the citric acid cycle [18]. In this section, we show how this pathway involving F1,6P can be formally verified in HOL4 using our formalization of Zsyntax .

The theorem representing the reaction of the glycolytic pathway leading from D-Glucose to F1,6P [4] can be described in classical Zsyntax format as follows:

```
Glc & HK & GPI & PFK & ATP & ATP ⊢ F1,6P
```

Using our formalization, this theorem can be defined in HOL4 as follows:

```
⊢ DISTINCT [Glc; HK; GPI; PFK; ATP; ADP; G6P; F6P; F16P] ⟹
(z_conj_elim (z_deduction [[Glc];[HK];[GPI];[PFK];[ATP];[ATP]]
                          [([Glc;HK],[[HK;Glc]]);
                           ([HK;Glc;ATP],[[HK];[G6P];[ADP]]);
                           ([G6P;GPI],[[F6P];[GPI]]);
                           ([F6P;PFK],[[PFK;F6P]]);
                           ([PFK;F6P;ATP],[[PFK];[F16P];[ADP]])] ) [F16P]
= [[F16P]]
```

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. The EVFs mentioned in the form of pairs and involving the molecules (G6P, F6P, etc.) are obtained from wet lab experiments, as reported in [4]. The `DISTINCT` function used above makes sure that all molecule variables (from initial aggregate and EVFs) used in this theorem represent distinct molecules. 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 F1,6P if it is present in the post-reaction list of molecules, as previously described.

Figure 1 shows the pathway leading to F1,6P in a step-wise manner. The gray-coloured circles show the chemical interactions and black 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.

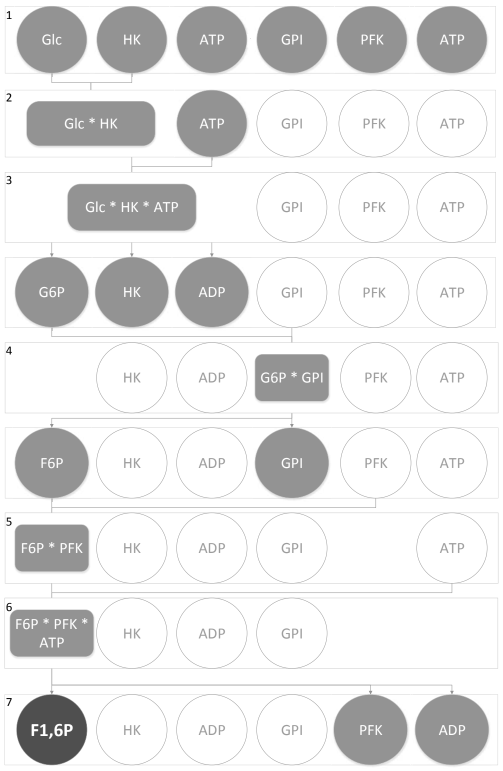

Fig. 1. Reaction Representing the Formulation of F1,6P

As part of this work, we also developed a simplifier Z_SYNTAX_SIMP [1] that simplifies the proof with a single iteration of the function z_deduction_recur and works very efficiently with the proofs involving our functions. The 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 users can observe the status of the reaction at every iteration, which is a very useful feature to get an insight of what is happening inside a reaction. Each application of `Z_SYNTAX_SIMP` on the reaction, depicted in Figure 1, would result in moving from a state n to $n+1$.

The verification time required for each iteration step is given in Table 4. HOL4 was running on a linux based machine (Intel Core i5, 4GB RAM). The iteration time depends on the total number of molecules (elements of list) present at a given iteration. Low number of molecules translate to less number of possible combinations, which in turn leads to less time required to move to the next iteration.

Table 4. Runtime per Iteration

Iteration	**Duration (Seconds)**
$1 \rightarrow 2$	11.996
$2 \rightarrow 3$	7.376
$3 \rightarrow 4$	12.964
$4 \rightarrow 5$	12.756
$5 \rightarrow 6$	9.240
$6 \rightarrow 7$	0.048

Our HOL4 proof script is available for download [1], and thus can be used for further developments and analysis of different molecular pathways. It is important to note that formalizing Zsyntax and then verifying its properties was a very tedious effort. However, it took only 10 lines of code to define and verify the theorem related to the above case study in HOL4, which clearly illustrates the usefulness of our foundational work.

We have shown that our formalization is capable of modeling molecular reactions using Zsyntax inference rules, i.e., given a set of possible EVFs, our formalism can derive a final aggregate **B** from an initial aggregate **A** automatically. In case of a failure to deduce **B**, the proposed method still provides the biologist with all the intermediate steps so that one can examine the reaction in detail and figure out the possible cause of failure.

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 as the theorems are being verified in a formal way using a sound theorem prover. Thus, there is no risk of human error or wrong application of EVFs. Finally, due to the computer-based analysis, the proposed approach is much more scalable than the paper-and-pencil based analysis presented in [4].

6 Conclusion

Most of the existing formal verification research related to molecular biology has been focussed on using model checking. As a complementary approach, the primary focus of the current paper is on using a theorem prover for reasoning about molecular pathways. The main strength of this approach, compared to existing model checking related work, is that the underlying methods and deduction rules can also be formally verified besides the verification of a particular molecular pathway case. Leveraging upon this strength, we formally verified two key behavioral properties of molecular pathways based on the Zsyntax language, which presents a deduction style formalism for molecular biology in the most biologist-centered way. Besides ensuring the correctness of our formalization of the Zsyntax operators and inference rules, the formally verified properties also play a vital role in reasoning about molecular pathways in the sound core of a theorem prover. The practical utilization and effectiveness of the proposed development has been shown by presenting the automatic analysis of Glycolytic pathway leading to Fructose-1,6-bisphosphate.

The proposed work opens the doors to many new directions of research. Firstly, we are developing a GUI to add more biologist friendly features in it. Moreover, we are also targeting some larger case studies, such as Dysregulation of the cell cycle pathway during tumor progression [16] and Fanconi Anemia/Breast Cancer (FA/BRCA) pathway [26]. Another interesting future direction is to leverage the high expressiveness of higher-order-logic and utilize calculus and differential theoretic reasoning to add reaction kinetics support in our formalism.

References

1. S. Ahmad. Formal Reasoning about Molecular Pathways - HOL Proof Script. `http://save.seecs.nust.edu.pk/projects/holsyntax/holzsyntax.html`, 2014.
2. S. Ahmad, O. Hasan, and U. Siddique. Towards Formal Reasoning about Molecular Pathways in HOL. In *International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 378–383. IEEE, 2014.
3. C. Baier and J. Katoen. *Principles of Model Checking*. MIT Press, 2008.
4. 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*, 5(3):e9511–1–e9511–12, 2010.
5. A. Church. A Formulation of the Simple Theory of Types. *Journal of Symbolic Logic*, 5(2):56–68, 1940.
6. V. Danos and C. Laneve. Formal Molecular Biology. *Theoretical Computer Science*, 325(1):69–110, 2004.
7. N.H. Hunt et al. Immunopathogenesis of Cerebral Malaria. *International Journal for Parasitology*, 36(5):569–582, 2006.
8. L. Trilling Fab. Corblin, E. Fanchon. Applications of a Formal Approach to Decipher Discrete Genetic Networks. *BMC Bioinformatics*, 11(1):385, 2010.
9. W. Fontana. Systems Biology, Models, and Concurrency. *SIGPLAN Notices*, 43(1):1–2, January 2008.
10. F. Cassez et al. G.Bernot. Semantics of Biological Regulatory Networks. *Electronic Notes Theoretical Computer Science*, 180(3):3–14, 2007.
11. Peter J. E. Goss and J. Peccoud. Quantitative Modeling of Stochastic Systems in Molecular Biology by using Stochastic Petri Nets. *Proceedings of the National Academy of Sciences*, 95(12):6750–6755, 1998.

12. J. Harrison. *Handbook of Practical Logic and Automated Reasoning.* Cambridge University Press, 2009.
13. K. Hirayama. Genetic Factors Associated with Development of Cerebral Malaria and Fibrotic Schistosomiasis. *Korean J. Parasitol*, 40(4):165–172, Dec 2002.
14. M. Magnin L. Paulevé and O. Roux. Abstract Interpretation of Dynamics of Biological Regulatory Networks. *Electronic Notes Theoretical Computer Science*, 272(0):43–56, 2011.
15. C.J. Langmead. Generalized Queries and Bayesian Statistical Model Checking in Dynamic Bayesian Networks: Application to Personalized Medicine. In *Proc. International Conference on Computational Systems Bioinformatics*, pages 201–212, 2009.
16. 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*, 12(1):608, 2012.
17. R. Milner. A Theory of Type Polymorphism in Programming. *Journal of Computer and System Sciences*, 17(3):348–375, 1977.
18. D. Nelson. *Lehninger Principles of Biochemistry.* W.H. Freeman, New York, 2008.
19. L.C. Paulson. *ML for the Working Programmer.* Cambridge University Press, 1996.
20. R. Pelánek. Fighting State Space Explosion: Review and Evaluation. In *Formal Methods for Industrial Critical Systems*, volume 5596 of *Lecture Notes in Computer Science*, pages 37–52. Springer, 2008.
21. J. Pospchal and V. Kvasnika. Reaction Graphs and a Construction of Reaction Networks. *Theoretica Chimica Acta*, 76(6):423–435, 1990.
22. A. Regev and E. Shapiro. Cells as Computation. *Nature*, 419:343, 2002.
23. A. Regev and E. Shapiro. The π-Calculus as an Abstraction for Biomolecular Systems. In *Modelling in Molecular Biology*, Natural Computing Series, pages 219–266. Springer, 2004.
24. A. Regev, W. Silverman, and E. Y. Shapiro. Representation and Simulation of Biochemical Processes Using the pi-Calculus Process Algebra. In *Pacific Symposium on Biocomputing*, pages 459–470, 2001.
25. M. Rizzotti and A. Zanardo. Axiomatization of Genetics. 1. Biological Meaning. *Journal of Theoretical Biology*, 118(1):61–71, 1986.
26. A. Rodríguez, D. Sosa, L. Torres, B. Molina, S. Frías, and L. Mendoza. A Boolean Network Model of the FA/BRCA Pathway. *Bioinformatics*, 28(6):858–866, 2012.
27. L. Thomas and R. d' Ari. *Biological Feedback.* CRC Press, USA, 1990.
28. R. Thomas. *Kinetic Logic: A Boolean Approach to the Analysis of Complex Regulatory Systems*, volume 29 Lecture Notes in Biomathematics. Springer-Verlag, 1979.
29. J.J Tyson, C.A. Nagy, and B. Novak. The Dynamics of Cell Cycle Regulation. *Bioessays*, 24(12):1095–1109, 2002.
30. O. Wolkenhauer, D. Shibata, and M.D. Mesarovic. The Role of Theorem Proving in Systems Biology. *Journal of Theoretical Biology*, 300(0):57–61, 2012.
31. J.H. Woodger, A. Tarski, and W.F. Floyd. *The Axiomatic Method in Biology.* The University Press, 1937.
32. A. Zanardo and M. Rizzotti. Axiomatization of Genetics 2. Formal Development. *Journal of Theoretical Biology*, 118(2):145–152, 1986.