Towards the Application of Formal Methods in Process Engineering

Sidi Mohamed Beillahi, Umair Siddique and Sofiène Tahar

Department of Electrical and Computer Engineering,
Concordia University, Montreal, Quebec, Canada
{beillahi, muh_sidd,tahar}@ece.concordia.ca

Abstract. Process engineering deals with the design, operation and optimization of different physical processes such as chemical reactions, biological mechanisms and petroleum processing. Recent technological advancements and short time to market in almost every industry bring new challenges and hence the complexity in the process engineering systems. One of the main techniques to analyze such models is to represent underlying systems using signal-flow-graphs which provide a systematic procedure to evaluate the system performance in the form of transfer functions. Traditionally, the analysis of signal-flow-graphs based models has been carried out by the paper-and-pencil based proofs and numerical techniques. In this paper, we present an overview of using theorem proving to formally analyze process engineering models. The main motivation of this work is twofold: First, the application of formal methods in a new domain to improve the analysis accuracy; Second, an effort to reduce the gap between formal methods and different engineering communities such as mechanical, chemical and engineering management.

1 Motivation and Background

International competition, demand for quality, and new technologies have heightened the need to develop high quality products, which can compete in the global marketplace. As a result of this increased competition, the pace of product development has quickened, forcing manufacturers into an era in which continuous quality improvement is a matter of survival, and not simply a competitive advantage. In general, process engineering is composed of different interrelated tasks which can be described in the form of a block diagram or a signal-flow-graph (SFG) [10,9]. In this paper, we address how to formally model engineering design strategies and methods in a theorem prover based on the formalization of signal-flow-graph theory. Consequently, many features can be predicted such as success probabilities and lead time, which is the time required to procure or manufacture an item of industrial design process. Our formalization of SFG is generic and allows the modeling of dynamically changing design conditions in a systematic way. Moreover, we can compute the distribution of project durations and easily predict important project metrics such as the expected mean and variance of lead time [11,12,8]. Finally, the information regarding the iterative structure of the project, and sensitivity of the lead time, which describes the

impact of varying one of the model variables i.e., probability or task duration on the model lead time, can also be evaluated.

The signal-flow-graph is composed of a network of directed branches which are connected by nodes (as shown in Figure 1). The branches represent the design activities, and the nodes represent states of the process and may involve a probabilistic choice as to which subsequent branch to follow. A special type of branch is one which goes back to a previous state of the process, i.e., the design iteration. The simplest form of design iteration is the direct repetition of an activity, i.e., return to the same state (as shown in Figure 1). If the duration and/or probability are different for the subsequent iterations and/or we wish to limit the possible number of iterations, another approach to model the process must be used, i.e., adding new nodes and branches that represent subsequent iterations. The branch jk depicts the activity when going from node j to node k. Each branch is associated with a quantity known as the branch transmission, i.e., $P_{jk} = p_{jk}z^{t_{jk}}$ where p_{jk} is the probability associated with the branch, and t_{jk} is the time taken to traverse the branch. The parameter z is the transform variable used to connect the physical system (time domain) to the quantities used in the analysis (transform domain). Also, z is used to separate the time and probabilities when branch transmissions are multiplied, so we will be able to extract the lead time of the process and the probability of each branch.

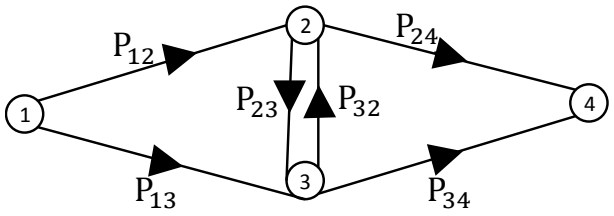

Fig. 1. Signal-Flow-Graph Example

In order to analyze the model behavior and to ensure that a given model satisfies the system properties and specifications, we need first to obtain the system transfer function which relates the system input and output. The main benefit of using SFG is the Mason's gain formula (MGF) [9] to compute the system transfer function, which is indeed applicable to any linear signal-flow-graph. The formula is given as follows :

$$G = \frac{y_{out}}{y_{in}} = \sum_{k=1}^{N} \frac{G_k \Delta_k}{\Delta} \tag{1}$$

$\Delta = 1 - \sum L_i + \sum L_i L_j - \sum L_i L_j L_k + \ldots + (-1)^m \sum \ldots + \ldots$
where

– Δ: The determinant of the graph

– y_{in}: The input-node variable
– y_{out}: The output-node variable
– G: Complete gain between y_{in} and y_{out}
– N: Total number of forward paths between y_{in} and y_{out}
– G_k: Gain of k^{th} forward path between y_{in} and y_{out}
– L_i: Loop gain of each closed loop in the system
– L_iL_j: Product of the loop gain of any two non-touching loops
– $L_iL_jL_k$: Product of the loop gain of any three non-touching loops
– Δ_k: The cofactor value of Δ for the k^{th} forward path, with the loops touching the k^{th} forward path removed.

The MGF provides an efficient way to compute the transfer function by avoiding the linear algebraic computations of simultaneous equations and the overall problem reduces to the extraction of the forward paths and feedback loops. We use the HOL Light theorem prover [6] to formalize the underlying theories of signal-flow-graph. This work is a part of larger projects on the formal analysis of signal processing and optics[1][2].

Figure 2 outlines the main idea to formally model process engineering using signal-flow-graphs, compute the system transfer function using Mason's gain formula, and prove that the given process engineering model satisfies the specification.

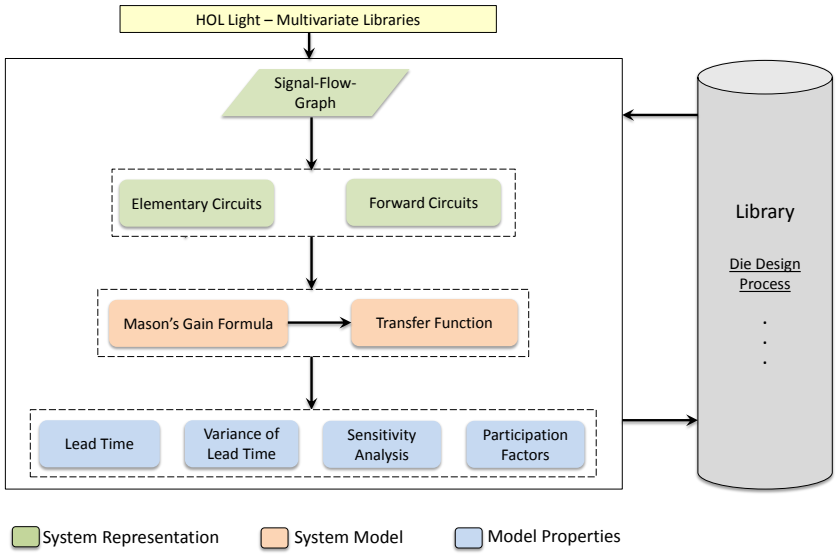

Fig. 2. Formal Analysis Framework

[1] http://hvg.ece.concordia.ca/projects/optics/
[2] http://hvg.ece.concordia.ca/projects/signal-processing/

The whole framework can be decomposed into three major parts which are depicted by different colors as shown in Figure 2. First, the formalization of signal-flow-graph theory which consists of the formalization of the two algorithms, i.e., *Elementary Circuits* and *Forward Circuits* which compute the forward paths and feedback loops, respectively. Consequently, these feedback loops and forward paths can be used to compute the transfer function using Mason's gain formula; Second, the formalization of Mason's gain formula to compute the transfer function; Third, the modeling of system properties, which are the model lead time and its expected value and variance along with the model sensitivity. Moreover, modeled systems will then also be available in a library for future use either independently or as a part of a larger process, which can be used by engineering managers to gain insight into the process through lead time computation and sensitivity analysis. The verified systems also serve to characterize the industrial processes, and to design and optimize them for the best use of time, effort and cost.

The rest of the paper is organized as follows: Section 2 presents some highlights of our SFG formalization. We give the formal analysis of engineering process properties in Section 3. In Section 4 we present the analysis of the die design process as an illustrative practical application Finally, Section 5 concludes the paper and provides hints for some future directions.

2 Signal-Flow-Graph Formalization

We model a single branch as a triplet (i, P_{ij}, j), where i, P_{ij} and j represent the start node, the transmittance and the end node, respectively. Therefore, a path can be modeled as a list of branches and further an SFG can be described as a composition of a list of all branches in the graph along with the information about the total number of nodes in the graph (size of the graph) and the sink node (output node) at which we want to compute the output of the model. We define three type abbreviations in HOL Light[1], i.e., branch, path and signal-flow-graph as follows:

Definition 1 (Branch, Path and SFG).

```
new_type_abbrev ("branch", ‘:ℕ × ℂ × ℕ‘)
new_type_abbrev ("path", ‘:(branch)list‘)
new_type_abbrev ("sfg", ‘:(branch)list × ℕ × ℕ‘)
```

where the `sfg` represent the size and the output node of a signal-flow-graph, respectively.

Next, in order to apply the MGF to a given SFG, we need to compute all the forward paths (forward circuit) and feedback loops (elementary circuit) from the source node to the node given by the user as the output node. We implemented a procedure to extract this information which is mainly inspired from the method

[1] Note that throughout this paper, we used minimal HOL Light syntax in the presentation of definitions and theorems to improve the readability and for the better understanding without prior experience of HOL Light notations.

proposed in [13]. Briefly, we take an SFG and generate a square matrix in which nodes are arranged in the first column and each row represents the nodes from which there is at least a branch that starts from the node under consideration. The matrix has the same size as the graph and the nodes numbers are arranged as the first elements of each row and the rest of the row is zeros. For the extraction of elementary circuits, we start the process by the first node of the graph and go through all possible paths which start from the node under consideration and test for each path whether it is a loop or not, i.e., the start and the end node of the path are the same. In the next iteration, we go to the next node of the graph and repeat the same process, only that we do not consider the nodes which have been considered in the previous node. For example when we search for loops which start from the node 2 we do not consider the paths which pass by the node 1 in the search process. The main process stops when we reach the last node of the SFG. For forward circuits (forward paths) extraction, we repeat a similar process, but we only consider the paths starting from the source node rather than exploring all the nodes, also instead of testing the node whether it is a loop or not we test if it is a forward path, the end node of the path is the output node of the model. For the sake of clarity, we give the following two main definitions of our formalization and more details can be found in [1].

Definition 2 (Elementary Circuits).

$\vdash \forall$ (system : sfg).
EC system = if (fst_of_trpl system = [ ]) then [ ]
else all_loops (EC_MAIN system) system

where the function EC_MAIN accepts an SFG, i.e., ((branch)list $\times \mathbb{N} \times \mathbb{N}$) and returns the list of loops in which each loop is represented a list of nodes only, and all_loops takes the result of EC_MAIN and the SFG (system) under consideration and returns the list of loops in the standard format, in which each loop is a list of branches and each branch represents a triplet. Finally, the main function, EC returns an empty list if the system has no branches otherwise it gives the list of all loops in the system found in the search.

Definition 3 (Forward Circuits).

$\vdash \forall$ (system : sfg).
FC system = if (fst_of_trpl system = [ ]) then [ ]
else forward_paths (FC_MAIN system) system

where the function FC_MAIN accepts an SFG (i.e., system) and returns the list of forward paths in which each forward path is considered as a list of nodes. Then the function forward_paths takes the result of FC_MAIN and system and returns the list of forward paths, such that each forward path is a list of branches.

We use Definitions 2 and 3 to extract the elementary and forward circuits of the SFG given in Figure 1. We consider node ① as the input (source) and node ④ as the output (sink), the graph size is 4 (number of nodes in the graph). First, we model the signal-flow-graph as follows :

Definition 4 (Model of SFG in Figure 1).

$\vdash \forall\ P_{12}\ P_{24}\ P_{13}\ P_{23}\ P_{32}\ P_{34} \in \mathbb{R}$.

$\texttt{Sfg\_model}\ \texttt{P}_{12}\ \texttt{P}_{24}\ \texttt{P}_{13}\ \texttt{P}_{23}\ \texttt{P}_{32}\ \texttt{P}_{34} =$
$[(1, \texttt{P}_{12}, 2); (2, \texttt{P}_{23}, 3); (2, \texttt{P}_{24}, 4); (1, \texttt{P}_{13}, 3); (3, \texttt{P}_{32}, 2); (3, \texttt{P}_{34}, 4)], 4, 4$

Second, we apply the Definitions 2 and 3 on the above model to extract the forward and elementary circuits:

Theorem 1 (Forward Circuits in SFG of Figure 1).

$\vdash \forall\ \texttt{P}_{12}\ \texttt{P}_{24}\ \texttt{P}_{13}\ \texttt{P}_{23}\ \texttt{P}_{32}\ \texttt{P}_{34}\ \in \mathbb{R}.$
$\texttt{FC}\ (\texttt{Sfg\_model}\ \texttt{P}_{12}\ \texttt{P}_{24}\ \texttt{P}_{13}\ \texttt{P}_{23}\ \texttt{P}_{32}\ \texttt{P}_{34}) =$
$[[(1, \texttt{P}_{12}, 2); (2, \texttt{P}_{24}, 4)]; [(1, \texttt{P}_{13}, 3); (3, \texttt{P}_{34}, 4)]]$

Theorem 2 (Elementary Circuits in SFG of Figure 1).

$\vdash \forall\ \texttt{P}_{12}\ \texttt{P}_{24}\ \texttt{P}_{13}\ \texttt{P}_{23}\ \texttt{P}_{32}\ \texttt{P}_{34}\ \in \mathbb{R}.$
$\texttt{EC}\ (\texttt{Sfg\_model}\ \texttt{P}_{12}\ \texttt{P}_{24}\ \texttt{P}_{13}\ \texttt{P}_{23}\ \texttt{P}_{32}\ \texttt{P}_{34}) = [[(2, \texttt{P}_{23}, 3); (3, \texttt{P}_{32}, 2)]]$

Note that the proofs of Theorem 1 and 2 are mainly based on some simplifiers [1] developed for the forward and elementary circuits extraction, respectively. By inspection of the Figure 1, we note that there are two forward paths; ①-②-④ and ①-③-④, and one feedback loop; ②-③-②. The same result as in Theorems 1 and 2.

Next, we present the formalization of two important functions. The first takes the list of loops and gives the determinant of the model, by computing the gain of all loops (which is the product of all loop branches transmittances), and the gain of no touching loops. The second calculates the numerator of the MGF by taking the lists of forward paths and feedback loops, and computes $\sum_{k \in system} G_k \Delta_k$, where G_k and Δ_k represent, respectively, the gain of the k^{th} forward path and the determinant of the k^{th} forward path considering the elimination of all loops touching the k^{th} forward path.

Definition 5 (MGF Denominator).

$\vdash \forall\ (\texttt{t} : (\texttt{path})\texttt{list}).$
$\texttt{determinant t} = 1 + (\texttt{delta t}\ (\texttt{touching\_loop t}))$

where the function `determinant` accepts the list of loops (t) in the graph and calculates the determinant of Mason's gain formula by first, calling `touching_loop` which determines all the touching loops for each loop of the graph.

Definition 6 (MGF Numerator).

$\vdash \forall\ (\texttt{t}_1 : (\texttt{path})\texttt{list})\ (\texttt{t}_2 : (\texttt{path})\texttt{list}).$
$\texttt{product\_gain\_det}\ \texttt{t}_1\ \texttt{t}_2 = \texttt{if}\ (\texttt{t}_1 = [\ ])\ \texttt{then 0 else}$
$((\texttt{gain}\ \texttt{t}_1[1]) * (1 + (\texttt{forward\_delta}\ \texttt{t}_1[1]\ \texttt{t}_2))) +$
$(\texttt{product\_gain\_det}\ (\texttt{t}_1/\texttt{t}_1[1])\ \texttt{t}_2)$

where the function `product_gain_det` accepts the lists of loops (t_2) and forward paths (t_1) in the graph and returns zero if there is no forward path in the graph otherwise it computes in each iteration the product of the gain of forward path under consideration (G_k, which is computed by `gain`) and its determinant (Δ_k,

which is calculated by `forward_delta`) until we go through all the given forward paths. Here $t_1[1]$ is first element of the list t_1.

Finally, we utilize above described functions to formalize the Mason's gain formula given in Equation 1, as follows:

Definition 7 (Mason's Gain Formula).
$\vdash \forall$ (`system` : `sfg`).

$$\texttt{Mason\_Gain system} = \frac{\texttt{product\_gain\_det (EC system) (FC system)}}{\texttt{determinant (EC system)}}$$

where the function `Mason_Gain` accepts an SFG (i.e., `system`) and computes the transfer function as in Mason's gain formula [10,9]. The model transfer function is computed using the function `transfer_function`, which is defined as follows:

Definition 8 (System Model Transfer Function).
$\vdash \forall$ (`system` : `sfg`).

$$\texttt{transfer\_function system} = \texttt{Mason\_Gain } (\lambda \texttt{z. system z})$$

where the function `transfer_function` accepts a `system` which has type $\mathbb{C} \rightarrow sfg$ and returns a complex ($\mathbb{C}$) quantity which represents the transfer function of the design process model (i.e., `system`). The detailed HOL Light formalization can be found in [1].

3 Formalization of the System Properties

Reasoning about lead time is a very critical task because it evaluates the performance and survival of an industrial project. The uncertainty of lead time can have huge effects on the overall business of the company as described in Table 1. A firm usually prefers to have a long and more reliable lead time, than a shorter, more variable lead time [5]. The expected value and variance of the lead time

Lead time	**Outcomes**
Lead time is as expected	Ideal situation
Lead time is shorter than expected	Unused stock when the delivery arrives early
Lead time is longer than expected	Shortages when running out before the delivery arrives

Table 1: Uncertain Lead Time Effects

(or the standard deviation) of the design process are directly dependent on the probabilities of iteration and the task times. The expected value of lead time is used to compute a safety stock based on a target service level [7]. The variance of the lead time is used in service-level computation of safety stock when it is zero, that means the lead time has no uncertainty and it should not be greater than the lead time [7].

Definition 9 (Expected Value of Lead Time).

$\vdash \forall$ (system : sfg).

$$\texttt{expected\_LT system} = \frac{\texttt{d}}{\texttt{dz}}(\lambda \texttt{z. transfer\_function (system z)}) \mid_{\texttt{z=1}}$$

where expected_LT accepts a model as an SFG, i.e., (system : path, $\mathbb{N}$, $\mathbb{N}$) and returns the expected lead time. The function $\frac{d}{dz}f(z)$ represents the complex derivative of function f with respect to z. As each term of the transfer function is of the from $p_i z^{t_i}$, where t_i is the time and p_i is the probability, for $z = 1$ we get $p_i t_i$ and the sum is the expected value of lead time of the process.

Definition 10 (Variance of Lead Time).

$\vdash \forall$ (system : sfg).

$$\texttt{variance\_LT system} = \frac{\texttt{d}}{\texttt{dz}}(\lambda \texttt{z. z} * \frac{\texttt{d}}{\texttt{dz}}(\lambda \texttt{z. transfer\_function (system z)})) \mid_{\texttt{z=1}} - \frac{\texttt{d}}{\texttt{dz}}((\lambda \texttt{z. transfer\_function (system z)}) \mid_{\texttt{z=1}})^2$$

where variance_LT accepts a system and computes the variance of the lead time (or standard deviation).

The sensitivity of the expected value and variance of lead time to each parameter can be calculated as the change in value of the quantity in response to a small change in the value of the parameter (task duration or the probability).

Definition 11 (Sensitivity).

$\vdash \forall$ (system : sfg).

$$\texttt{sensitivity system k} = (\frac{\texttt{d}}{\texttt{dx}}(\lambda \texttt{x. } \frac{\texttt{d}}{\texttt{dz}}(\lambda \texttt{z. transfer\_function (system x z)}) \mid_{\texttt{z=1}}) \mid_{\texttt{x=k}}) * (\frac{\texttt{k}}{\frac{\texttt{d}}{\texttt{dz}}(\lambda \texttt{z. transfer\_function (system x z)}) \mid_{\texttt{z=1,x=k}}})$$

where sensitivity accepts a system and a parameter k in the system (probability p_{jk} or task times t_{jk}), and returns the sensitivity of lead time with respect to change in parameter k. In order to demonstrate the utility of our proposed framework, we formally analyze the die design process in the next section.

4 Application : Die Design Process

Die design process is a manufacturing process which involves the production of geometrically complex and reusable dies (also called molds). This is a critical industrial process as mentioned in [2,4,3] because dies are designed primarily to reduce the toolmaker's costs, to reduce production times, and to improve the product quality. Thus the precision in the die design process is required to avoid problems in the production process. Figure 3 shows the signal-flow-graph of an example of the process of die design [11], where the task duration (the exponents of the parameter z in the branch's transmittance) are in scaled

work days required for a certain range of panel complexity. The signal-flow-graph shows the tasks of manufacturability evaluation and die design replicated since the rework probabilities (the real value in the branch's transmittance) and task times change after the first iteration. By inspection of the graph, we find that there are 10 nodes (each node represents a task in the process) and the output node is node (10). So by modelling the SFG of die design to the type of abbreviation we gave in Definition 1, we define it in HOL Light as follows:

Definition 12 (Die Design Model).
$\vdash \forall$ `z` $\in \mathbb{R}$.
`Die_model z` =
$[(1, \mathtt{z}^3, 2); (2, \mathtt{z}^3, 3); (3, 0.75\mathtt{z}^2, 4); (3, 0.25\mathtt{z}^2, 7); (4, \mathtt{z}^2, 5); (5, 0.25\mathtt{z}, 6); (6, \mathtt{z}, 5);$
$(5, 0.75\mathtt{z}^2, 7); (7, 0.1\mathtt{z}, 6); (7, 0.9\mathtt{z}^3, 8); (8, \mathtt{z}^7, 9); (9, 0.5\mathtt{z}, 5); (9, 0.5, 10)], 10, 10$

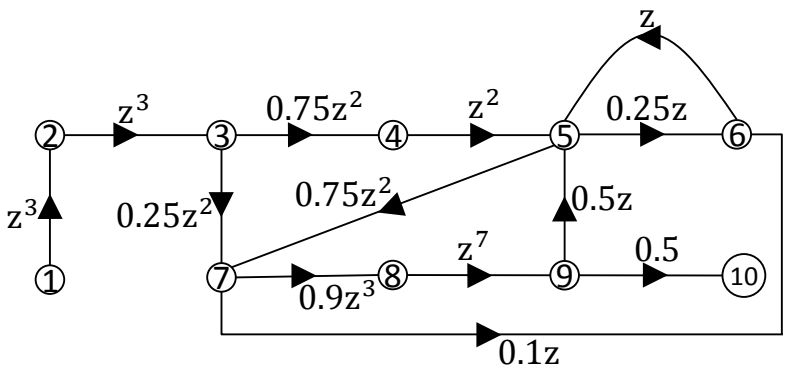

Fig. 3. Signal-Flow-Graph of Die Design Process

Next, we utilize our formalization to verify the transfer function of the above mentioned die design process [11] as follows:

Theorem 3 (Transfer Function of Die Design Process).
$\vdash \forall$ `z` $\in \mathbb{R}$.

$$\texttt{transfer\_function (Die\_model z)} = \frac{0.1125\mathtt{z}^{18} - 0.028125\mathtt{z}^{20} + 0.253125\mathtt{z}^{22}}{1 - 0.25\mathtt{z}^2 - 0.75\mathtt{z}^4 - 0.3375\mathtt{z}^{13}}$$

The proof of this theorem is mainly based on the extraction of forward paths and feedback loops in the graph and then using the Mason's gain formula. Note that we used the simplifiers developed for the elementary and forward circuits extraction, and Mason's gain formula [1] which reduce the interaction with HOL Light during the proof. Based on the obtained transfer function we can verify the process's most likely lead time, the shortest lead time, the expected lead time, the standard deviation of lead time (variance of lead time), and the sensitivity of the process.

This completes our formal analysis of the die design process. So far, we have developed a core formalization of the signal-flow-graphs theory. Moreover, we developed a generalized procedure for the formal stability and resonance analysis of engineering systems (details can be found in [1]). Finally, we applied our signal-flow-graphs formalization in process engineering by developing a procedure to model a design process and verify the corresponding transfer function. Based on the transfer function of the die design process, our next step is to verify the properties of this model using our formalization. The source code of the entire formalization is available at [1]. The formalization of signal-flow-graphs took approximately 1500 lines of HOL Light code and about 250 man-hours. However, the utilization of higher-order-logic theorem proving in industrial settings (particularly, engineering process) is always questionable due to the huge amount of time required to formalize the underlying theories. Another important factor is the gap between theorem proving and the engineering communities, which limits its usage in industry. For example, it is hard to find industrial engineers with theorem proving background and vice-versa.

5 Conclusion

Signal-flow-graphs provide a powerful, flexible design and modeling tool for the purpose of analyzing engineering systems such as the product development processes. In this paper, we reported a new application of formal methods in the domain of process engineering. We presented a formal analysis framework based on higher-order logic which provides the required expressiveness and soundness to formally model and verify the properties of design process models. In particular, we formalize the signal-flow-graph theory along with the Mason's gain formula and transfer functions. Consequently, we presented the formalization of the properties of the design process models. Finally, we described the formal analysis of a die design process. We believe that the reported work is a first step towards an ultimate goal of building a comprehensive framework to analyze process engineering systems using formal methods.

References

1. S. M. Beillahi and U. Siddique. Towards the Application of Formal Methods in Process Engineering. `http://hvg.ece.concordia.ca/projects/signal-processing/sfg.html`, 2014.
2. D. Tang et. al. Re-Engineering of the Design Process for Concurrent Engineering. *Computer δ Industrial Engineering*, 38:479–491, 2000.
3. H. Zein et. al. Effect of Die Design Parameters on Thinning of Sheet Metal in the Deep Drawing Process . *American Journal of Mechanical Engineering*, 1(2):20–29, 2013.
4. S. H. Kong et. al. Internet-Based Collaboration System: Press-Die Design Process for Automobile Manufacturer. *The International Jouranl of Advanced Manufaturing Technology*, 20:701–708, 2002.

5. G. Gallego. Variability of Leadtime Demand and the Production Period, University of Columbia, USA, December 2000. http://www.columbia.edu/~gmg2/4000/pdfold/lect_11.pdf.
6. J. Harrison. HOL light: A Tutorial Introduction. In *Formal Methods in Computer-Aided Design*, volume 1166 of *LNCS*, pages 265–269, 1996.
7. IBM. Production Lead Time, 2000. http://pic.dhe.ibm.com/infocenter/scappinf/v1r0/index.jsp?topic=%2Fcom.ibm.scapps.lnp.doc%2FReports%2FWarehouseInventoryReport.html.
8. O. Isaksson, S. Keski-Seppala and S. D. Eppinger. Evaluation of Design Process Alternatives Using Signal Flow Graphs. *ASME Design Engineering Technical Conferences*, 11(3):211–224, 2000.
9. S. J. Mson. Feedback Theory, Further Properties of Signal Flow Graphs. In *Proceedings of the Institute of Radio Engineers*, volume 44, pages 920–926, July 1956.
10. S. J. Mson. Feedback Theory, Some Properties of Signal Flow Graphs. In *Proceedings of the Institute of Radio Engineers*, volume 41, pages 1144–1156, September 1953.
11. S. D. Eppinger, M. V. Nukala and D. E. Whitney. Generalized Models of Design Iteration Using Signal Flow Graphs. *Reseach in Engineering Design*, 9(2):112–123, 1997.
12. J. Andersson, J. Pohl and S. D. Eppinger. A Design Process Modeling Approach Incorporating Nonlinear Elements. *Proceedings of ASME Design Engineering Technical Conferences*, 1998.
13. J. C. Tiernan. An Efficient Search Algorithm to Find the Elementary Circuits of a Graph. *Communnications of the ACM*, 13(12):722–726, 1970.