The Thesis Committee for Pradeep Radhakrishnan
Certifies that this is the approved version of the following thesis:

A graph grammar scheme for representing and evaluating planar mechanisms

APPROVED BY
SUPERVISING COMMITTEE:

Supervisor:

Matthew I Campbell

S.V.Sreenivasan

A graph grammar scheme for representing and evaluating planar mechanisms

by

Pradeep Radhakrishnan B.E

Thesis

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Engineering

The University of Texas at Austin

May 2010

Dedication

I dedicate the work to my family

Acknowledgements

I would like to sincerely thank Dr. Matthew I Campbell for accepting me into his research group and assigning a challenging problem in the area of mechanism synthesis. He played a pivotal role in shaping up my thoughts and research process through constant support and encouragement. I also would like to thank Dr.S.V.Sreenivasan for agreeing to read my thesis and providing valuable suggestions. This material is based upon work supported by the National Science Foundation under grant 0448806 and I wish to thank them for the same. I should mention the help of Ms.Cindy Raman and Mr.David Justh, who were always there to help me out with matters concerning teaching and academic procedures. Lastly, without my parents - Papa and Amma and my sister Anu and Hari, I would not have achieved my dream of pursuing graduate education in the USA.

07 May 2010

Abstract

A graph grammar scheme for generating and evaluating planar mechanisms

Pradeep Radhakrishnan MSE

The University of Texas at Austin, 2010

Supervisor: Matthew I Campbell

There are different phases in any design activity, one of them being concept generation. Research in automating the conceptual design process in planar mechanisms is always challenging due to the existence of many different elements and their endless combinations. There may be instances where designers arrive at a concept without considering all the alternatives. Computational synthesis aims to arrive at a design by considering the entire space of valid designs. Different researchers have adopted various methods to automate the design process that includes existence of similar graph grammar approaches. But few methods replicate the way humans' design. An attempt is being made in the thesis in this direction and as a first step, we focus on representing and evaluating planar mechanisms designed using graph grammars. Graph grammars have been used to represent planar mechanisms but there are disadvantages in the methods

currently available. This is due to the lack of information in understanding the details of a mechanism represented by the graph since the graphs do not include information about the type of joints and components such as revolute links, prismatic blocks, gears and cams. In order to overcome drawbacks in the existing methods, a novel representation scheme has been developed. In this method, labels and x, y position information in the nodes are used to represent the different mechanism types. A set of sixteen grammar rules that construct different mechanisms from the basic seed is developed, which implicitly represents a tree of candidate solutions. The scheme is tested to determine its capability in capturing the entire set of feasible planar mechanisms of one degree of freedom including Stephenson and double butterfly linkages. In addition to the representation, another important consideration is the need for an accurate and generalized evaluator for kinematic analysis of mechanisms which, given the lack of information, may not be possible with current design automation schemes. The approach employed for analysis is purely kinematic and hence the instantaneous center of rotation method is employed in this research. The velocities of pivots and links are obtained using the instant center method. Once velocities are determined, the vector polygon approach is used to obtain accelerations and geometrical intersection to determine positions of pivots. The graph grammar based analysis module is implemented in an existing object-oriented grammar framework and the results have found this to be superior to or equivalent to existing commercial packages such as Working Model and SAM for topologies consisting of four-bar loop chain with single degree of freedom.

Table of Contents

List of Tables

List of Figures

Chapter 1: Overview of Automated Synthesis of Planar Mechanisms

A design process is comprised of various activities such as gathering user requirements, generating concepts, analyzing properties of those concepts, building prototype and finally testing and refining to get the final outcome. In all the aforementioned activities, each of which are equally important, the concept generation phase is one that brings out the unique feature of the design. A concept may refer to the form (physical design) or the technology used in the functioning of the product. The concept generation process is generally a manual and time-consuming process. Different researchers have attempted computerization of the concept generation process in various domains including the design of planar mechanisms [1-4]. In most researches on automated synthesis of planar mechanisms, the scope is restricted to four-bar or six-bar linkages with rotary joints [5]. There is a constant urge within the design community for a concept generation system that comprises of all the different elements.

The design process usually begins with an understanding of the mechanism function which may be a path through space [6] followed by choosing a standard mechanism, customizing it by adding links, determining kinematic properties using analysis tools and then iterating to determine if the mechanism is able to satisfy the user requirements. In the process, the designer also determines the degrees of freedom, F, using Gruebler's criterion [7], which states

$$F = 3(n-1) - 2 \times j_1 - j_2$$

where n is the number of links, j_1 refers to number of one degree-of-freedom joints and j_2 refers to the number of two degree-of-freedom joints in the mechanism. Since most mechanisms have only a single-degree-of-freedom (F = 1) and are comprised of links and joints, the equation is compelling as it easily describes what is, and what is not a valid

solution. Other than fairly simple geometry methods like three-position synthesis and its higher-order variations, there are no specific guidelines for the design of a mechanism.

The steps involved in the research is presented in Figure 1 that begins with the development of a new graph scheme for representing mechanisms followed by the creation of rules that generate the full language of planar mechanisms. In order to find an ideal solution for a particular problem, a method of generating candidate solutions, evaluating their worth, and guiding the process to better solutions is required. These three steps comprise a search process for finding mechanisms automatically and thereby reducing the design time of creating the mechanism by hand. The report focuses on representation and evaluation for a graph grammar based design synthesis.

Defining a representation scheme is one of the most important aspects of the process. The ease with which the rest of the principles such as rules, generation and evaluation are built clearly depends on the representation scheme. The scheme developed in this paper combines different elements commonly associated with planar mechanisms such as links, revolute joints, sliding blocks, pin-in-slots, etc. and in the future is extensible to other elements such as gears, cam-followers and so on. Combined with the representation are the rules that are invoked at the time of mechanism generation. An important aspect considered at the time of representation and rule formulation was that of the tree-based design generation process wherein successive invocation of the rules generates the different states (shown as circles in Figure 2) on the tree. The generation can be based on a tree-search technique such as breadth first search, depth first search or other similar techniques [8]. The rules were developed by reviewing traditional mechanism design literature [9],[10] wherein links are added to existing structures thereby building a more complex mechanism (Figure 3).

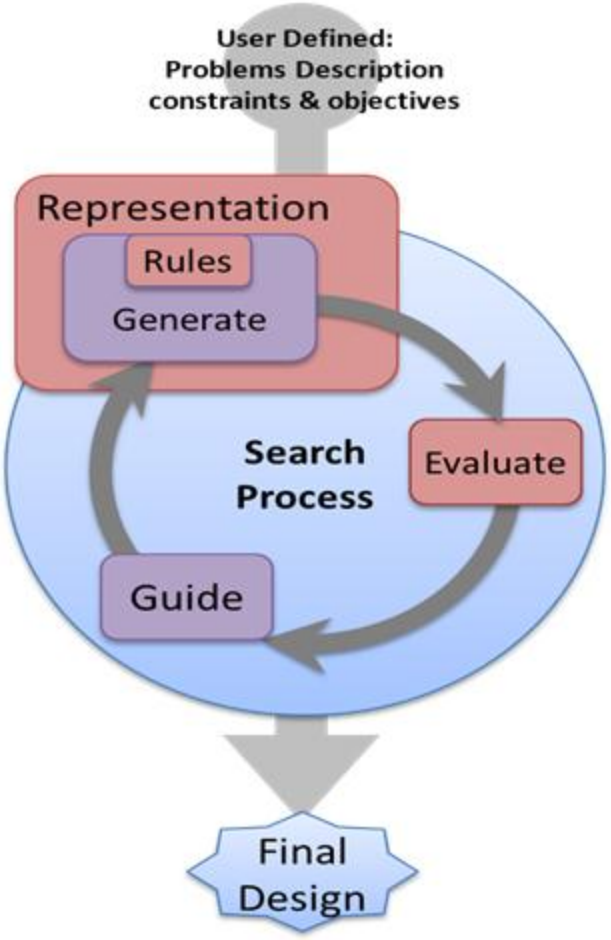


Figure 1: The five step design automation process

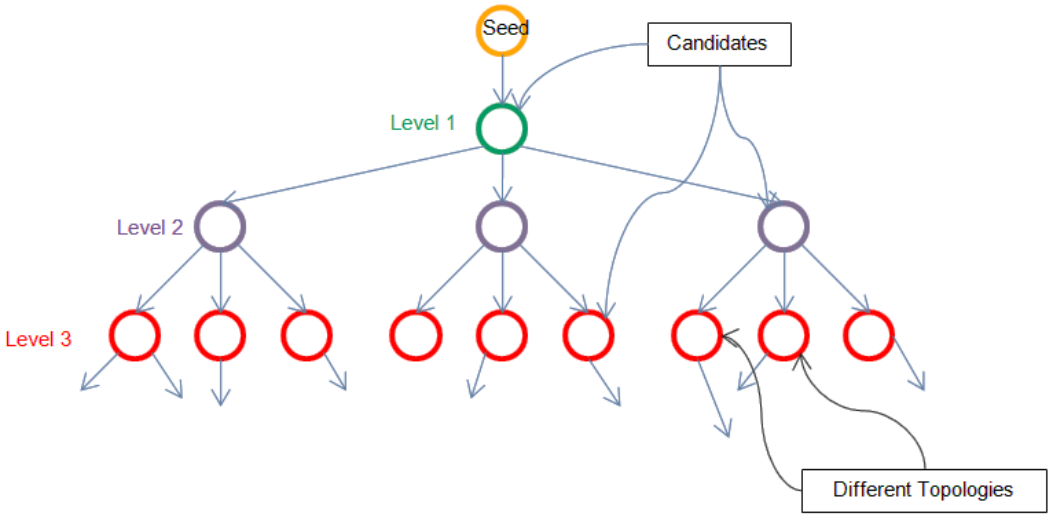


Figure 2: Tree-based generation of candidates

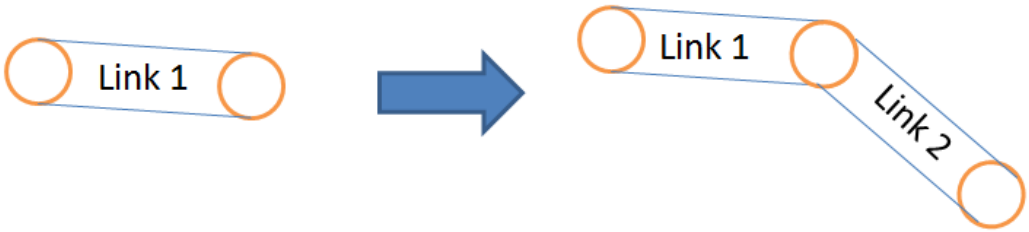


Figure 3: A sample rule of adding links

Once the rules have been developed and mechanisms generated, it is important to evaluate the kinematic properties such as velocity and acceleration of the generated mechanism to conform to user requirements. Kinematic analysis through analytical methods is widely documented for four-bar mechanisms and computational tools that

perform path optimization (like SAM [11] and WATT [12]) extend this functionality to the analysis of six-bar mechanisms as well. (Commercially available applications such as Working Model [13] and ADAMS [14] support kinematic analysis of not just planar mechanisms but also other machine elements such as gears, belts and chain drives). The basis of these applications is not known and they do not include automated concept generations (WATT does include design generator but is restricted to four and six-bar mechanisms). Also, most applications do not provide convenient means to link concept generator results with their analysis routine (as is done in linking KIDDS software to the synthesis routine mentioned in [15]) for kinematic or dynamic analysis. The analytical kinematics routine developed is applicable for any topology with a single-degree of freedom consisting of a four-bar chain. The limitation with regard to the four-bar chain is expected to be overcome in future development. Also the program currently works only for single-input systems with constant input angular velocity.

After a mechanism is kinematically analyzed, its motion is compared with user specifications and the resulting error (i.e. difference in paths) is stored as an objective function for further optimization. If the error cannot be effectively minimized, the particular design is discarded and another is chosen. Optimization is under active research and is presented as an overarching principle.

BACKGROUND

Graph grammar based design synthesis has been around for many years that began with the formalism proposed by Freundenstein [16]. Later on, investigations have dealt-with building representation and synthesis schemes including a computational concept generator by Kota [17]. The Systematic method developed by Tsai [18] is quite popular based on which the epicyclic gear-train design tool was developed by Schmidt [19] and Li [20]. Since the representation is different in all these cases for linkages, gears,

etc., the very idea of a concept generating tool encompassing all is a daunting task if not impossible. Also, sufficient information about the mechanism is not conveyed by the approaches that are currently used since they are not descriptive enough to operate on a variety of different elements within the same framework. The representation and the rule-based generation process described in this report has been used by researchers of the same laboratory to automate the design of sheet metal parts [21] and gear trains [22] among other applications and have shown promise over traditional approaches.

An engineering concept generator is useful only if the results are valid, for which an evaluation scheme is necessary. The same argument can be extended to the synthesis of planar mechanisms computationally. There are different techniques that have been developed for kinematic analysis of planar mechanisms such as the graphical vector polygon method, instantaneous center of rotation method and analytical equation method [7]. Each technique has certain limitations, for example, instant center method can only be used to determine velocity while the analytical equation method requires any n-bar mechanism to be subdivided into simpler chains for analysis. These methods are robust in the determination of velocity, acceleration and new position since they are analytical in nature and have been largely successful for standard four-bar and six-bar mechanisms. Numerous analysis tools have been developed in different research laboratories as well as available commercially. Some of them that have been used as a reference in this paper are SAM, WATT, Working Model and ADAMS. Though the basis of these applications is not exactly known (they could be completely analytical or combined analytical and numerical), they are typically for fixed topologies, hence cannot be adopted for generic planar mechanism design automation problems with results emanating from a concept generator. The non-availability of an analysis engine that is generic is a major issue in the automation of planar mechanism design process and this prevents generalization of the

concept. Taking these drawbacks into account, an evaluation scheme has been implemented based on the instant center method within the GraphSynth framework [23].

Chapter 2: Representation of Planar Mechanisms

Representation is an important aspect in computational design synthesis since rules, evaluation and optimization are all based off it. The approach employed here is to use a skeletal graph for representation. The representation developed by Tsai [18] and others using graph grammars emphasize on the relation between links and pivots. In doing so, the understanding of the relationship (adjacency) matrix becomes necessary to identify the elements within the mechanism. Also, the method is incapable of integrating a variety of elements within a single representation framework. In order to overcome this constraint and to provide a clear scheme, a representation scheme has been developed that distinguishes different elements such as pivots and links by means of labels stored in each node. It should be noted that though the method of manipulating labels in nodes has been in existence, it has not been used in the field of planar mechanisms as effectively as described in this report. The representation developed also incorporates coordinate information that aids the user in visualizing the mechanism without requiring to work from the adjacency matrix to construct the generated topology. Therefore, no additional information indicating the type of mechanism represented by the candidate is required in this method. This type of representation also helps in formulating analysis (evaluation) schemes that directly operate on the graph objects (nodes and arcs).

REPRESENTATION SCHEME

The representation scheme developed in this research is illustrated by means of an example four-bar mechanism (Figure 4). The four-bar mechanism shown in Figure 4(a) is depicted as a graph as seen in the screenshot of Figure 4(b). From Figure 4(b) it could be seen that links (rectangles on the graph) and pivots (circles) are represented using nodes; and arcs connect the pivot and link nodes to create the mechanism. Each node is identified by a name with associated labels in parentheses. There are different labels

associated with every node and are listed in Table 1. On a closer observation of the graph and labels, the graph consists of the "Ground" node with labels *ground* and *link* referring to the ground link (frame in Figure 4 (a)) in Figure 4 (b). The "Ground" node is connected to two pivots namely "Input" node and the other ground node "IP2" whose labels are also given in the table. The input pivot is connected to the output by means of a link (identified by green color to differentiate input link and for better understanding of the example), which in turn is connected to the other pivot "IP" on the coupler link and "IP" is finally connected to the ground pivot "IP2". The directional arcs (with a specific direction as indicated by the arrowhead) without labels are indicative of the energy flow in the system. This representation helps in recreating the structure of the mechanism from the graph itself.

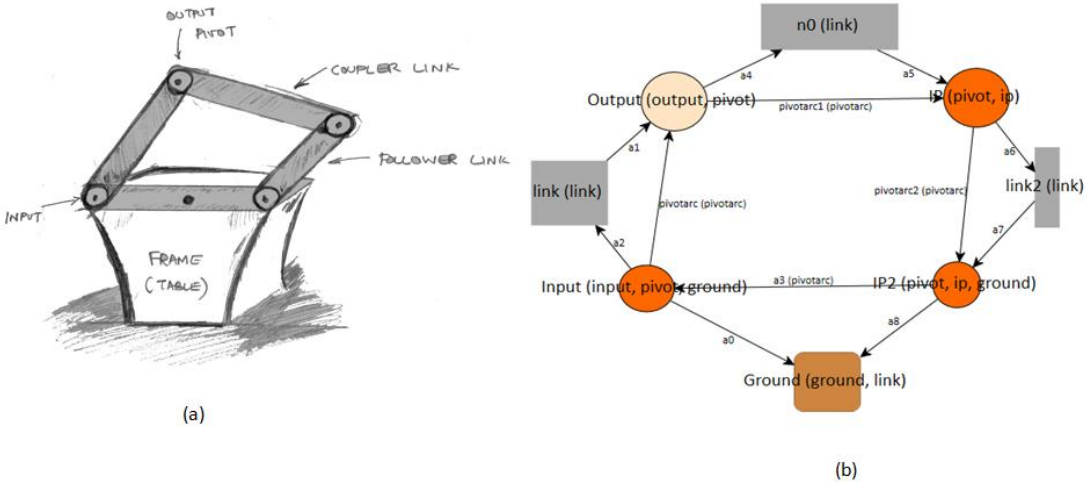


Figure 4: Four-bar Mechanism (a) 2d Representation (b) Graph Grammar Representation

Table 1: Node labels associated with the graph representation of a four-bar mechanism

Node/Arc Name	Type: Link / Pivot	Node Labels
Ground	Link	ground, link
Input	Link	input, pivot, ground
Output	Pivot	output, pivot
IP (coupler)	Pivot	pivot, ip
IP2 (connected to ground)	Pivot	pivot, ip, ground
link	Link	link
link1	Link	link
link2	Link	link

The different nodes and arcs have labels associated with him. These labels indicate the function of the particular node or arc. For example, the labels such as *ground* and *pivot* attached to a node indicate that the particular node is connected to the frame of the mechanism. An *output* label associated with a node indicates that the output characteristics of the node correspond to the required result. In this research, the main area of focus is on path generating mechanisms and hence the *output* label corresponds to the path required to transverse by the mechanism. In addition to the labels that are associated with nodes, there is a label that is associated with each directional arc namely *pivotarc*. The label indicates that the arc connects two pivots only. The directional arcs help in representing the design space in a concise manner. There are also arcs that do not contain labels, and these arcs are used to connect a link to its end pivots. The representation also includes schemes for sliding blocks and pin-in-slots as illustrated in the example of a quick return mechanism shown in Figure 5. The illustration in Figure 5 (a) is shown as a graph in our formalism in Figure 5(b). Note that the copper-colored node in the lower right hand corner of Figure 5(b) indicates the ground reference plane. It connects to three pivots: the input pivot, the rocker link's pivot with ground, and the sliding block's reference. Also, the sliding block is actually composed of two pivots: one

to represent the prismatic sliding, and another to represent the rotation with the coupler link. On a closer inspection, these pivots carry some additional labels to improve their interpretation. The prismatic sliding joint node contains labels: *sliderh*, *pivot* and *ip*; and the rotary pivot node has labels: *slider_conn*, *ip*, *and pivot*. Thus two pivot nodes and a link are used to represent the sliding block. Similarly the pin-in-slot is constructed from a single pivot node with labels *ip*, *pivot* and *pis* (an *output* label is present in this case, but is subject to change depending on the design requirements) but the links to which the pin is connected carry an additional label *pis_conn* indicating that those links are part of the pin-in-slot structure. The additional labels that are part of the quick return mechanism are given in Table 2**.**

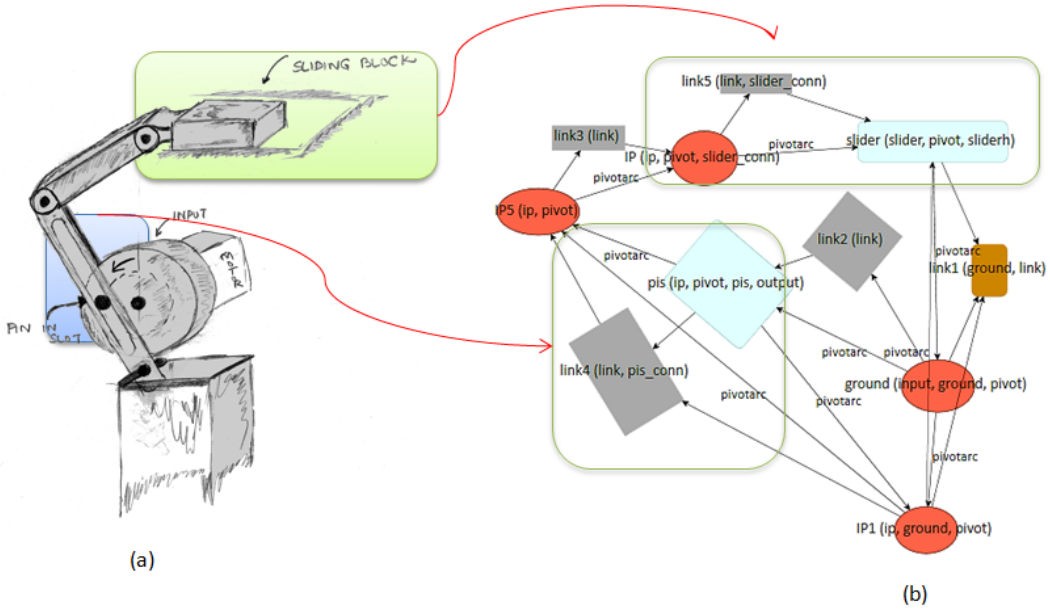


Figure 5: Quick Return Mechanism (a) 2D representation (b) Graph Grammar representation

Table 2: Labels associated with the quick return mechanism

Node Name	Link/Pivot	Node Labels
PIS (pin-in-slot)	Pivot	pivot, ip, pis
Slider (prismatic)	Pivot	sliderh , pivot, ip
Slider (rotary)	Pivot	slider_conn, ip, pivot
link (connected to slider)	Link	slider _conn, link
link (connected to PIS)	Link	pis_conn, link

As shown in the figure, this method is able to represent features such as sliding blocks and pin-in-slots more effectively than other graph representation schemes such as the Systematic method. Another example is that of the Stephenson mechanism representation (shown in Figure 6(b)**)** which has an additional plate node (link consisting of three pivots) with label *add_plate*. It is now possible to incorporate a variety of mechanism elements by adopting this method of explicitly defining the element by means of labels. Another difference here is that links are represented by nodes and not arcs. Figure 7 shows the comparison between the two schemes in representing a four-bar mechanism, which clearly points to the fact that the Systematic method does not distinguish the type of mechanism represented by it to the user at the graph level as clearly as the method described in this paper. Another advantage of the representation technique presented in this paper is the ability to have connections between multiple links and pivots. Also, the use of labels such as *pivot* and *link* enables identification of the pivots and links respectively at the start of the evaluation process for computing the number of instant centers and degree of freedom of the mechanism. Additionally, the use of shapes and coordinate information available in the graph renders a realistic construction of the mechanism which is otherwise unavailable. Due to many advantages of this method over existing ones, it is possible to expand the scope of automated design synthesis by including many different elements.

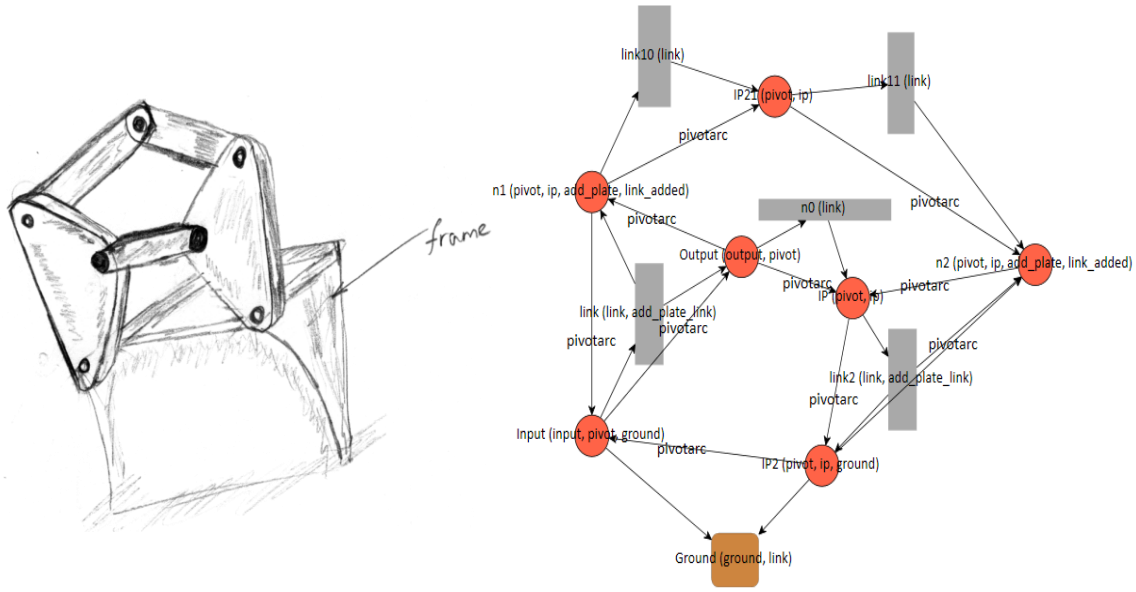


Figure 6: Stephenson Mechanism (a - left) 2D representation (b - right) Graph Grammar representation

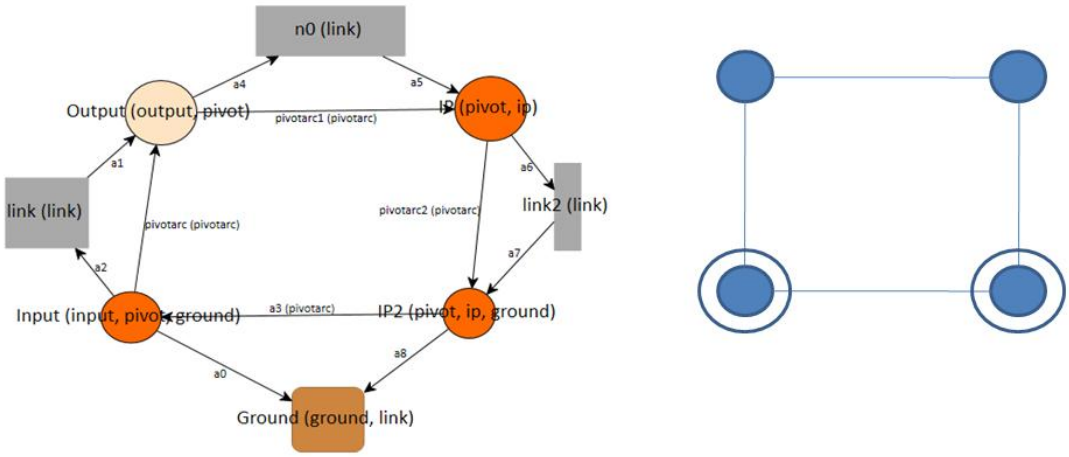


Figure 7: Difference in representation (a) new method (b) Systematic Method

SEED AND RULES

Once the representation scheme is in place, grammar rules are formulated such that they are generic and can build standard as well as unique mechanisms, which may not be captured in normal design practice. The rules are essentially building blocks for planar mechanisms arrived by reviewing configurations from standard mechanism handbooks. There are three different rules, that have been developed namely the generation rules, inversion rules and modularization rules. Generation rules are the main design rules. Inversion rules are used to create inversions of valid mechanisms and modularization rules are essentially helpful in segregating mechanisms based on the elements contained in the mechanism. All the three different rule types are explained in the later part of this chapter.

The design generation rules require a starting point which is termed as 'seed'. The seed is responsible for the way the rules are formulated and also influences the degree of generalization. The seed used in this research is shown in Figure 8, which indicates that the process begins with the knowledge of some output pivot and the reference frame information represented by the ground link. The locations of the output pivot and the ground link are based on the task specified by the user. For instance if the generated mechanism is required to trace a path, then the output pivot represents one of the points on the path and ground link represents the bounding box within which the mechanism has to be contained. Both these nodes are integral to any mechanism and thus form the seed of the generation process.

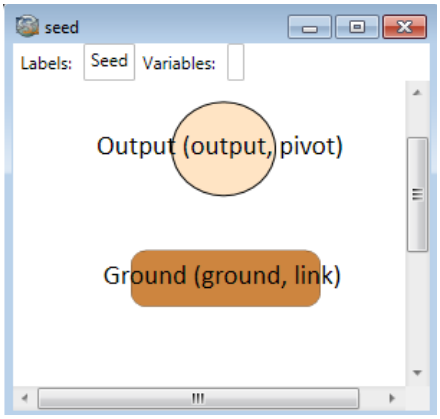
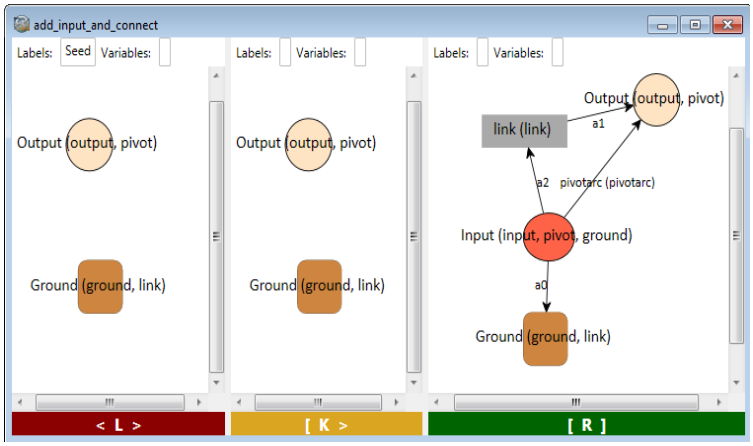
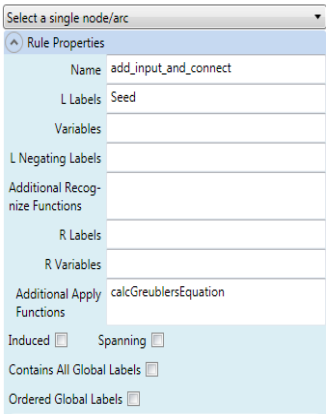


Figure 8: Seed for design generation



Grammar Rule



Properties of a Grammar Rule

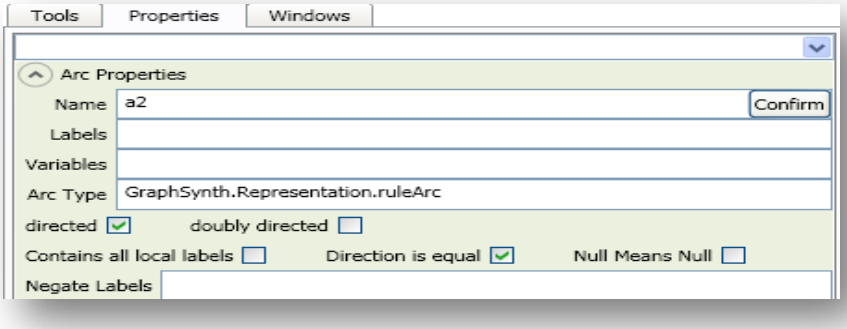


Figure 9: Grammar rule shown with associated properties

The rules are created once the seed has been formalized. Figure 9 shows a sample rule with associated properties. As can be seen in the figure**,** each grammar rule can be subdivided into three sections namely: L, K, R, which are in fact intersecting graphs and may be indicated as < L [K > R] . The left-hand side as indicated by elements between < and > serve as the recognition section, wherein a candidate on the tree is checked if it contains the nodes and arcs as given in this section of the rule. The [K > represents common elements between < L > and [R] and [R] is the desired change to the candidate. Also shown in the figure are the properties of one of the arcs for the rule to be correctly recognized within a host graph. For this particular rule, the arc 'a2' is directed thus indicating that the respective arc in the host must also be directed. There are some properties connected to the display of the arcs on the screen and are not discussed here. There are other properties that the candidate under consideration must satisfy. The labels and variables on the candidate are initially checked to see if they match with those given in the rule. In this rule, there are no specific label and variable requirements. After this step, if there are additional recognition functions prescribed, they are also checked. In this example, there are no specific recognition functions to be applied on the graph. Additionally, there are functions called "Induced", "Spanning", and "Contains All Global Labels" that could be applied on the candidate to determine the suitability of the rule for that particular candidate. In this example, there are no labels or variables, but there is an additional function that computes Gruebler's equation and stores the resulting degree of freedom of the topology as a variable. In addition, properties such as "Contains all local labels", "strict degree match" and "negating labels" that a node of the candidate must satisfy are all checked. "Contains all local labels" requires the labels available in the rule correspond to that of the candidate. "Strict Degree Match" requires the nodes in the host graph to have the same number of arcs connecting to it. "Negating labels" is a feature that

helps to avoid nodes that contain certain labels. For instance, if a candidate containing nodes with labels "slider" should not have this rule applied on those nodes, then the negating labels feature helps in achieving that objective. It is not mandatory for all rules to take these Booleans into consideration. Further details on representation using the graph grammar technique are available in [24]. The figure below gives an illustration of the rule-recognition and application processes.

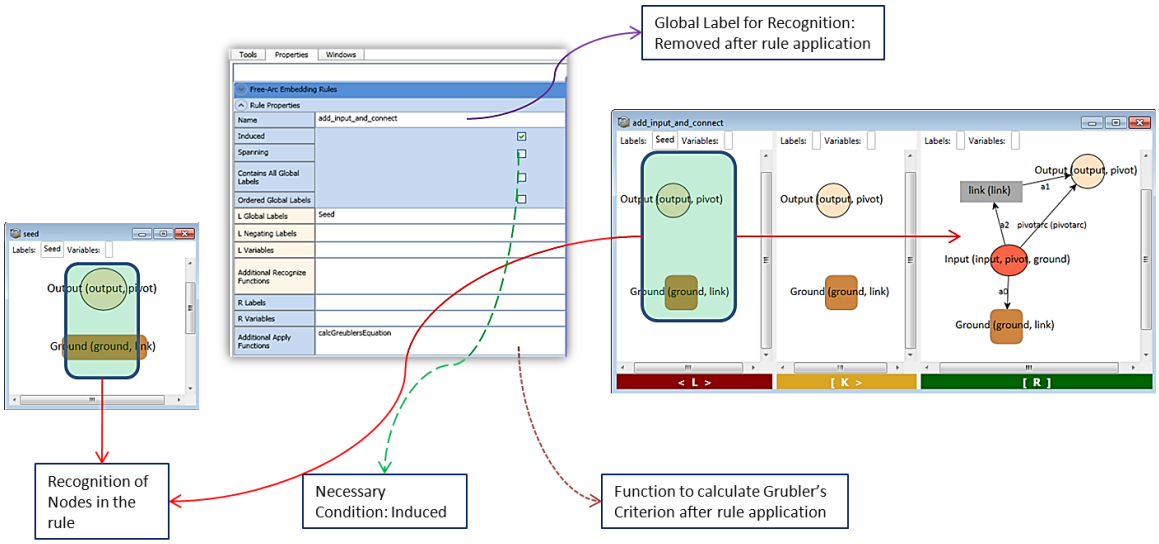


Figure 10: Illustration of the rule recognition and application processes

Figure 10 shows a rule that is applied primarily at the start of the generation process wherein a link is added to connect the output pivot to an input drive. In doing so, the recognition process also checks if the graph satisfies global labels criterion (explained in the previous paragraph). In this case, "seed" is the global label. Figures 11 and 12 show a couple of generation rules that were created. Figure 11 adds a link and a ground pivot creating a four-bar loop and Figure 12 creates a ternary link from a binary link. It is very important to strive for fewer rules so that generalization is not affected and would lead to a targeted result. There are 16 rules developed which are classified as given in

Table 3**.** It is clear that the majority of rules are additive in nature with a few that replace or expand in-between existing members.

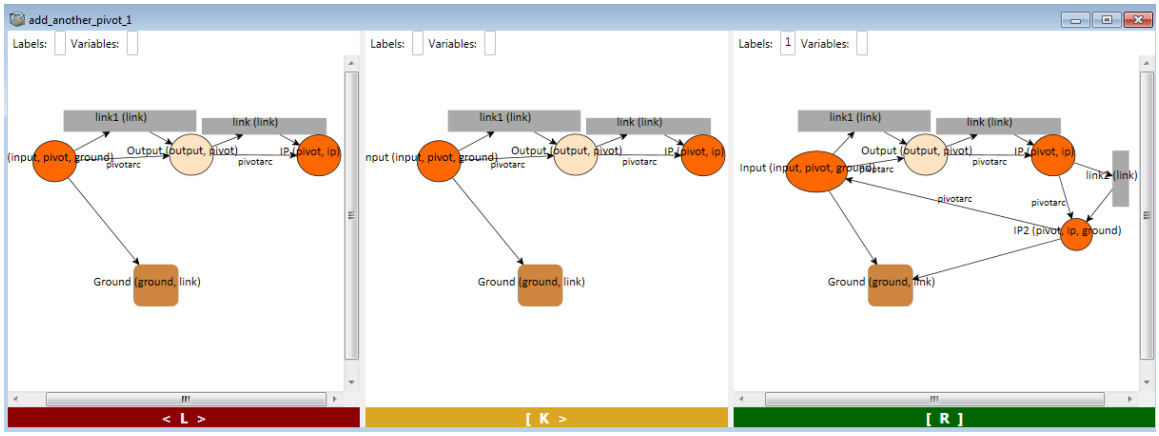


Figure 11: Grammar rule that creates a four-bar loop

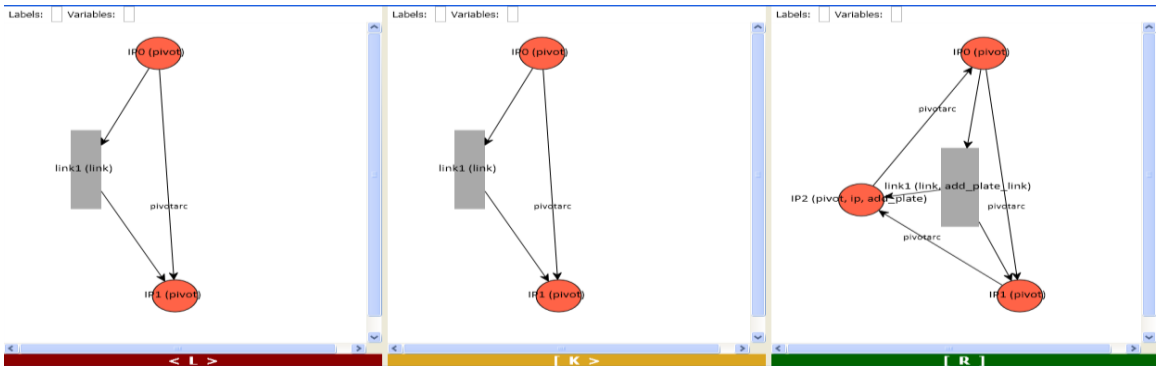


Figure 12: Grammar rule that creates a link with three pivots

These rules are grouped into a rule-set, which is associated with a function to calculate Gruebler's criterion. This function is invoked with every rule and the degree of freedom is stored in the resulting graph. In addition to these rules, there are a few rules that are applied at the time of topology optimization such as varying the location of the output pivot, introducing vertical sliding blocks in place of horizontal and other similar rules and are not part of the main set of rules governing mechanism design. The complete set of rules is available at http://www.graphsynth.com/mechsynth.

Table 3: Classification of the 16 generation rules

	Type	Categories			Total
		Add Links (to frame)	Replace an Existing link	Expand in-between links	
1	Connect to Input Motor	Rule #1	-	-	1
2	Create Four-bar loop	Rule #2, #3	-	-	2
3	Increase number of links	Rules #6, #7	-	Rules #8,#9,#16	5
4	Add Slider blocks	Rules #4,#15	-	-	2
5	Add Pin-in-slot	Rules #12,#13	Rule #14	-	3
6	Add Ternary and Quaternary Links	-	Rules #5,#11	Rule #10	3
	Total	***8***	***3***	***5***	***16***

In addition to the design generation rules, there are three additional rule-sets created to widen the design space as well as to segregate the candidates. The first two rule-sets comprise of rules that create inversions of mechanisms. Creating inversions of a particular candidate is useful since it displays the different possibilities within one candidate without requiring additional design generation rules. The two rule-sets that create inversions are shown in Figure 13. This process is illustrated in Figures 14-16 where an inversion of a slider-crank mechanism is generated. As shown in Figure 14, the arc directions are first eliminated from the candidate. This rule is applied multiple times on the resulting candidate to generate the graph shown as "Result" in the figure. The resulting graph enables structural manipulation without considering energy flow directions. Once the arcs are eliminated, ground labels are removed from pivots and links and input label from pivot. Now the candidate is just composed of links and pivots with

no ground or input information as shown in Figure 15. The next rule that is applied assigns a ground label to one of the links and its end pivots (two or more). After assigning ground labels, the input label is assigned to one of the ground pivots, thereby generating one of the inversion solutions of the candidate (shown in Figure 16). There could be multiple inversion candidates (in the case of the slider crank mechanism presented in Figure 17, there are four) possible for a particular candidate. It may be suggested that inversions are best carried out at the time of generating candidates using design rules. But it is prudent to generate inversions of only valid mechanisms and not all candidates of the design space, hence this approach was adopted.

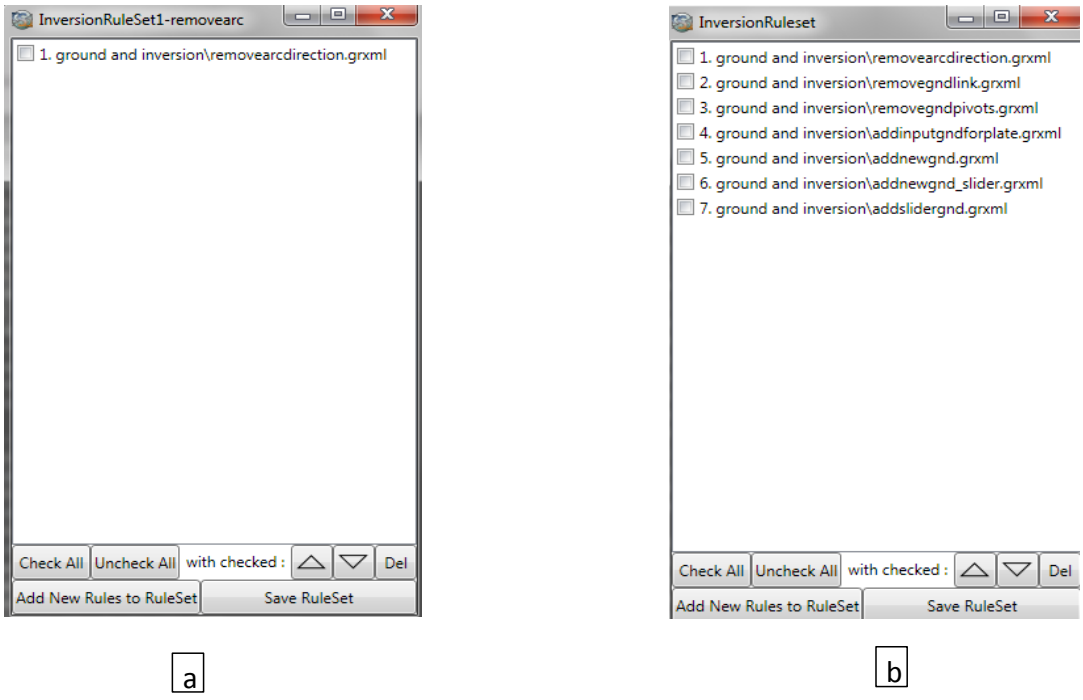


Figure 13: Rule-sets to create mechanism inversions

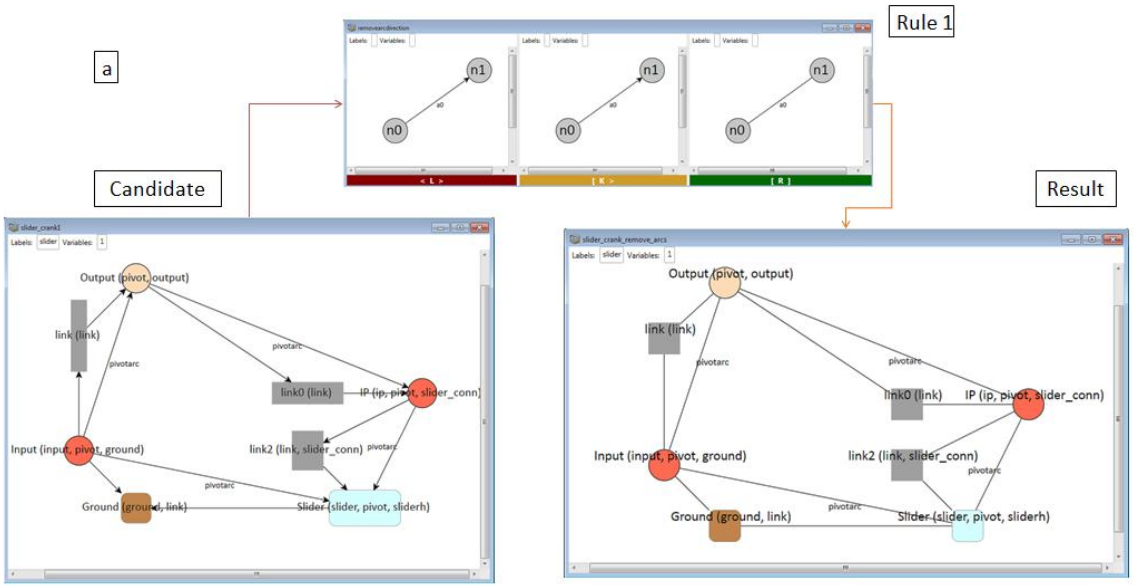


Figure 14: Creating mechanism inversion (a) removing arc directions

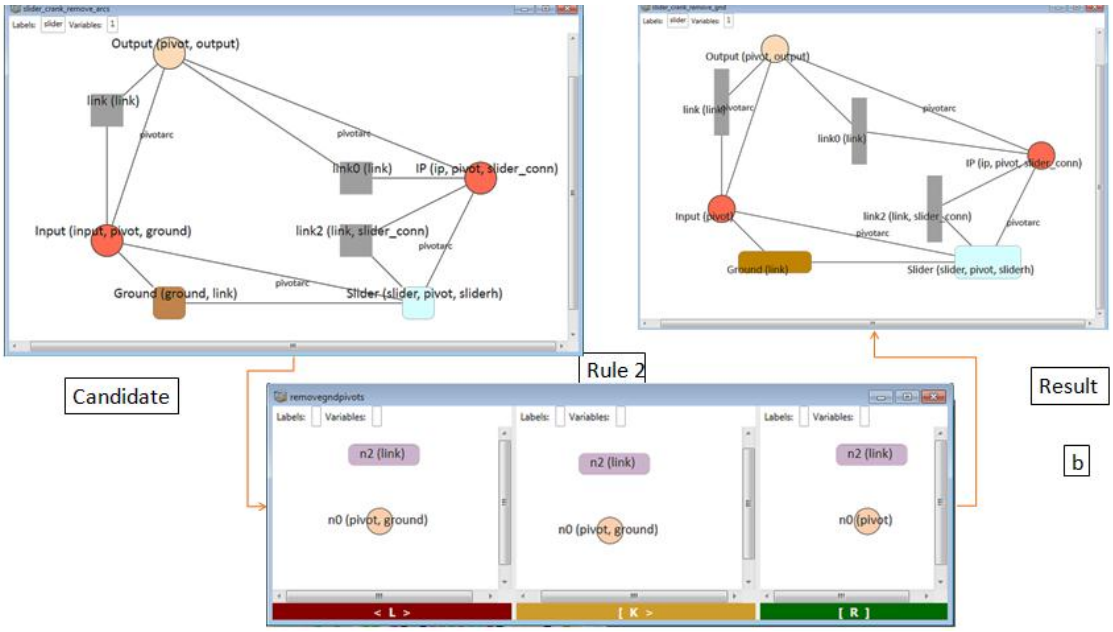


Figure 15: Creating mechanism inversion (b) removing ground and input labels from nodes

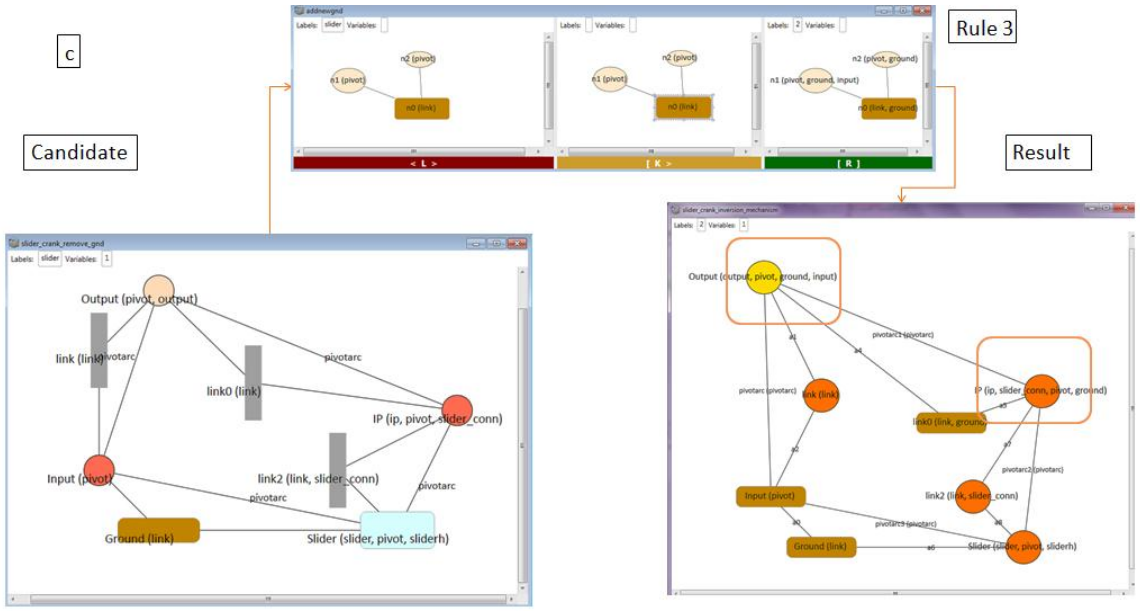


Figure 16: Creating mechanism inversion (c) assigning new input and ground nodes

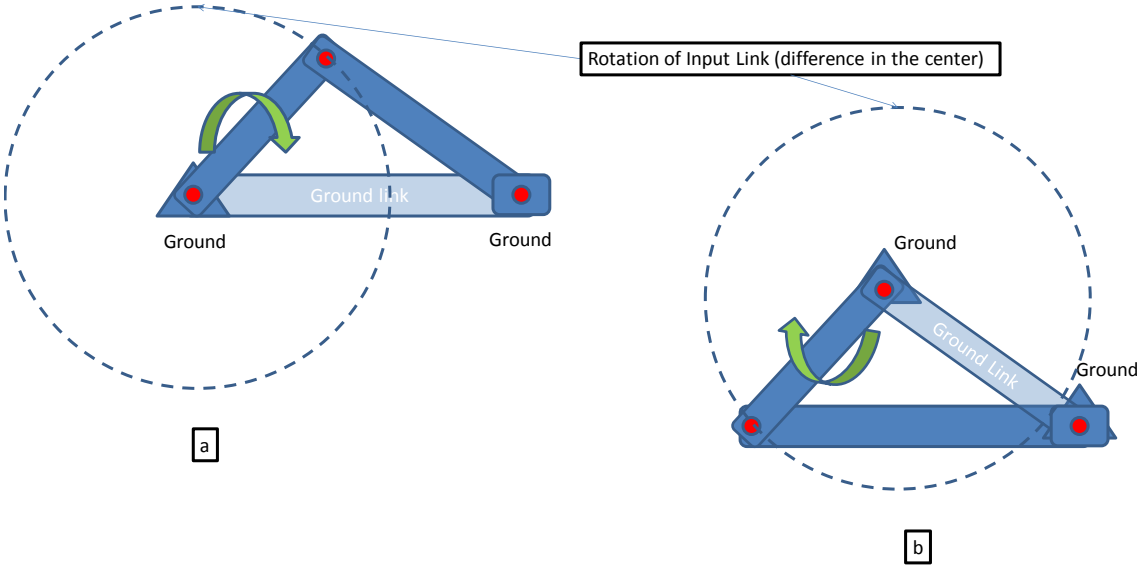


Figure 17: Slider crank mechanism and its inversion

The third rule-set is comprised of rules that identify common mechanisms or loop structures within a candidate. This allows us to modularize each candidate obtained from the generation process, which in turn aids in the categorization of the design space

accordingly. For instance, consider Figure 18 wherein a rule applied to the six-bar mechanism replaces the four-bar with a single node identified by the label *fourbar*. The different modularization rules are grouped into a rule-set as shown in Figure 19. Note that this rule set is a growing list with frequent additions.

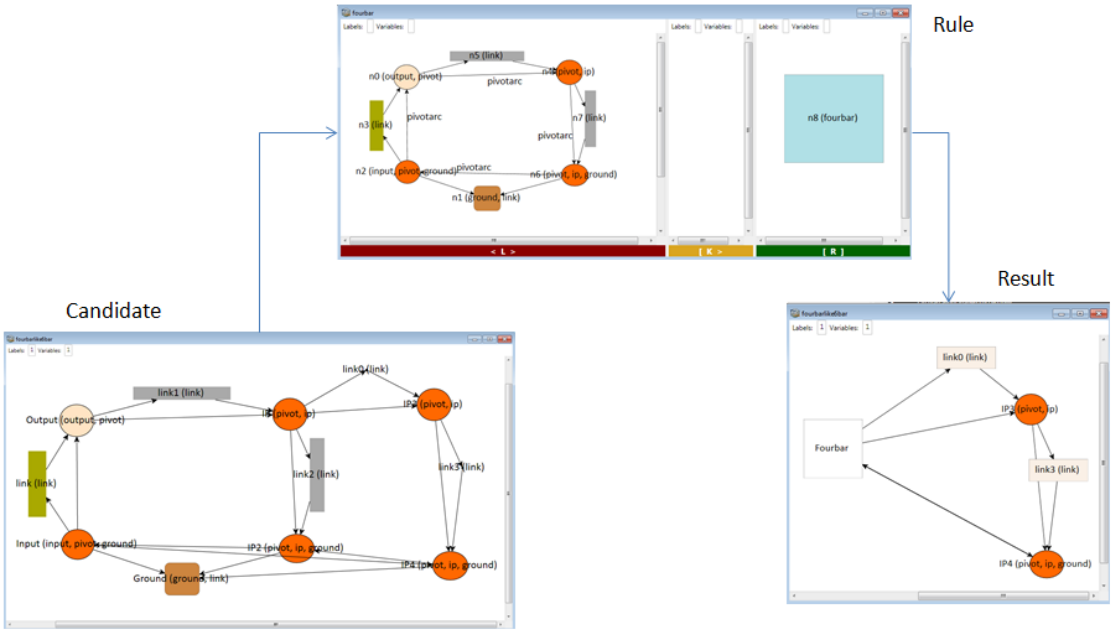


Figure 18: Modularization rule example

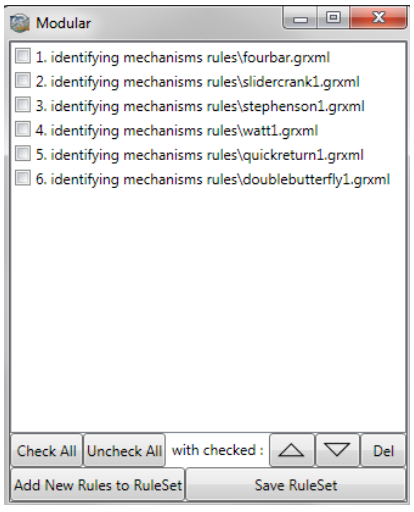


Figure 19: Modularization rule-set

DESIGN GENERATION RESULTS

The different candidates that are generated by applying the design generation rules are shown in Figures 20-22**.** Figure 20 demonstrates the generation of a four-bar mechanism [25], Figure 21 shows a Stephenson-I mechanism [25] and Figure 22 displays a double-butterfly linkage [26] generated in GraphSynth**.** The candidates that are generated and those used for illustration in the earlier sections are a testimony to the capability of the representation and rules reported in this project.

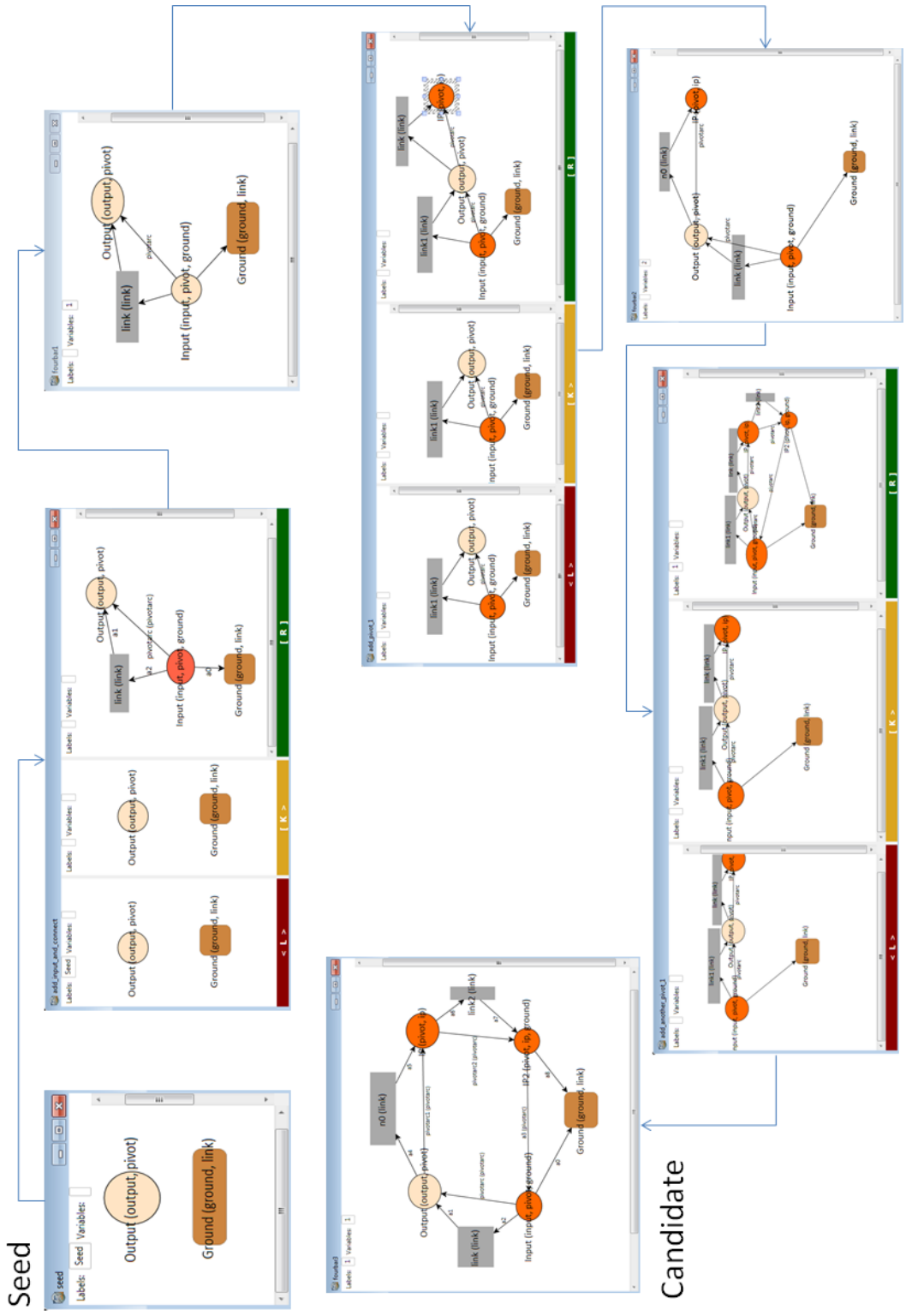


Figure 20: Generation of a four-bar mechanism

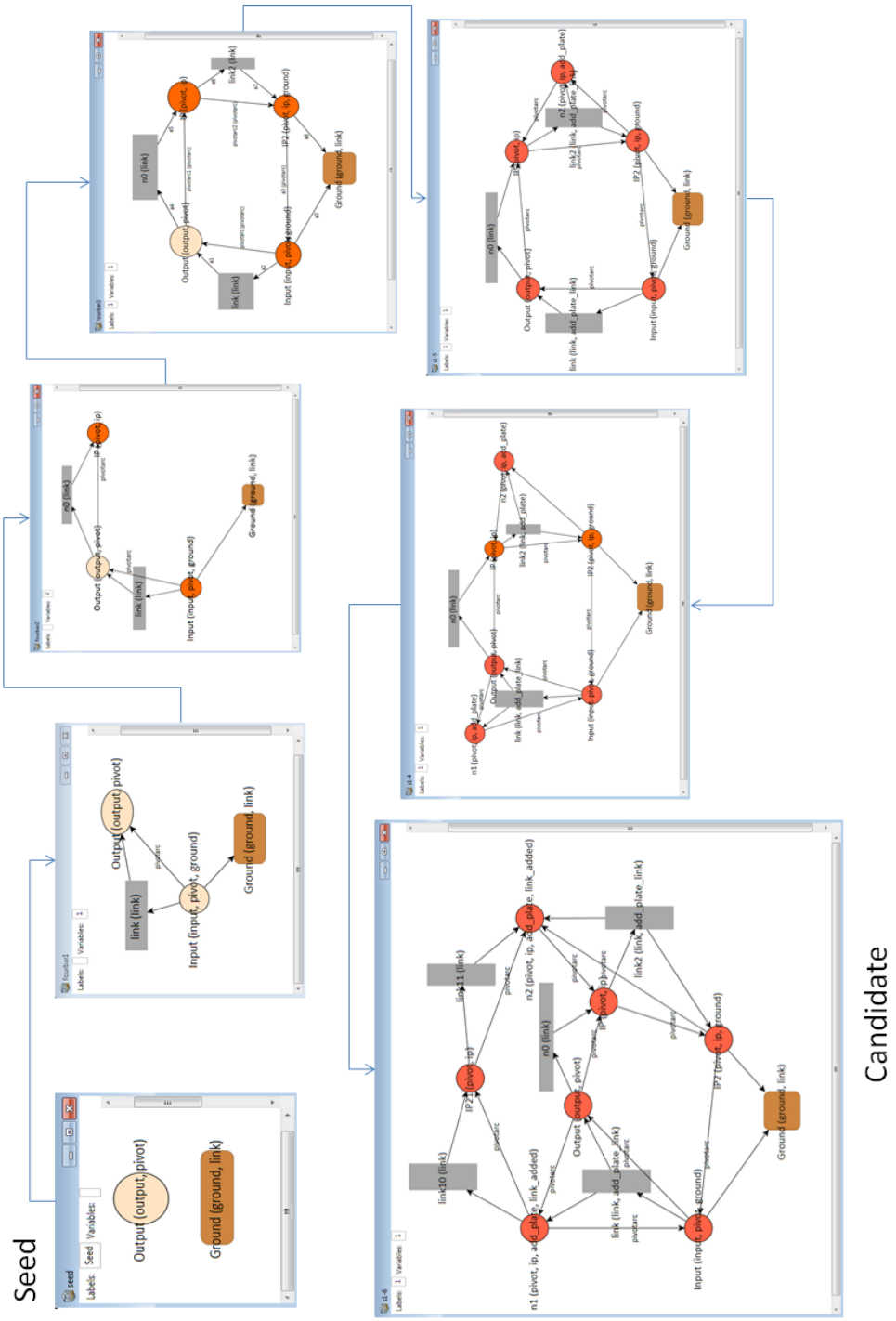


Figure 21: Generation of Stephenson-I mechanism

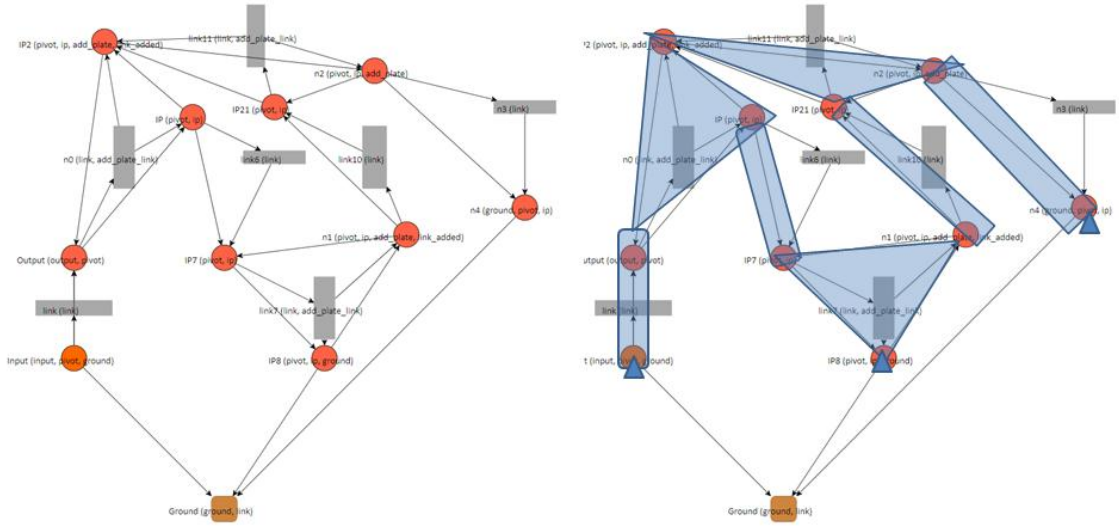


Figure 22: A double-butterfly linkage mechanism

Chapter 3: Evaluation of Planar Mechanisms

Evaluation helps in verifying whether the generated topology satisfies the user's requirements or not. Generally any evaluation involves three stages namely preprocessing, simulation and post-processing and commercial programs such as ADAMS, Working Model and SAM are built this way. The requirements for an evaluation tool are the ability to construct the topology, simulate the working conditions and make conclusions depending on the results. It is also important that the three stages of evaluation are quick and accurate. In this research, the preprocessing stage requires handling topologies from the concept generator. Commercial programs that are currently available for kinematic analysis do not permit customizable inputs, requiring the user to manually code the preprocessing stage. In order to carry out the different functions such as accepting the generated topology, determining the kinematics and providing feedback regarding usability with good accuracy and speed, an evaluation tool was built as part of this research. The evaluation tool is generic and can operate on mechanisms that are of single-degree of freedom and contain a four-bar chain. The kinematic analysis developed (Figure 23**)** has three important sections namely velocity, acceleration and position. This section is explained using the example of a four-bar mechanism shown in (Figure 24**)**.

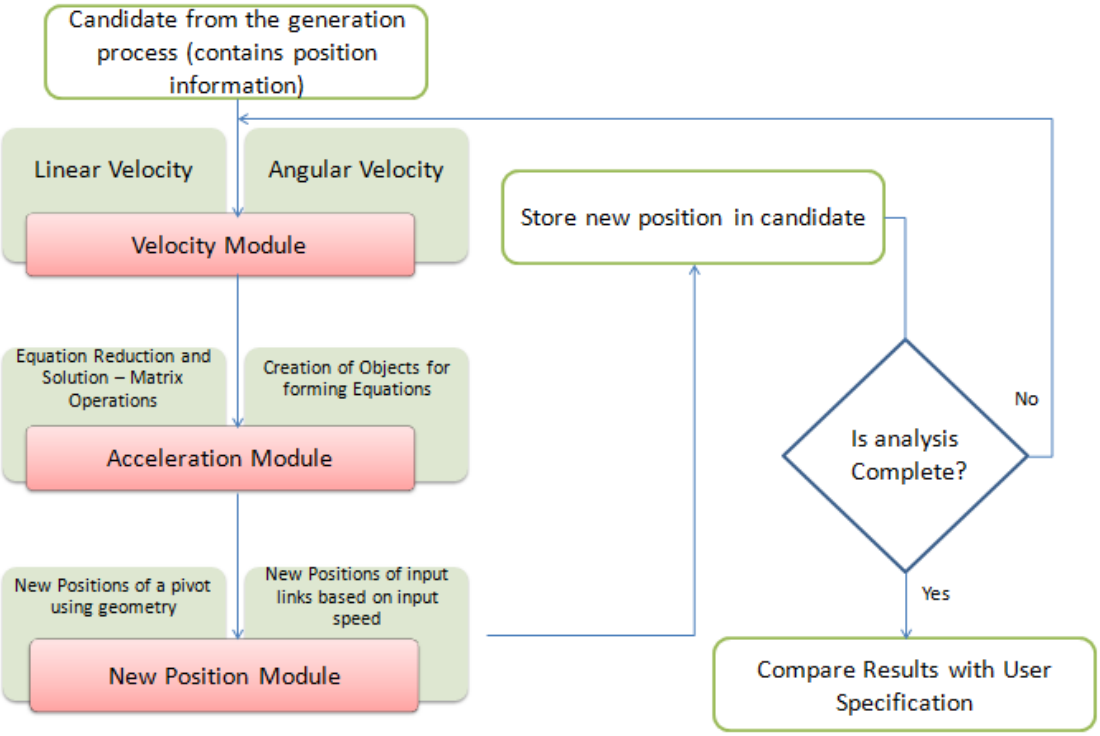


Figure 23: Overview of the Evaluation Process

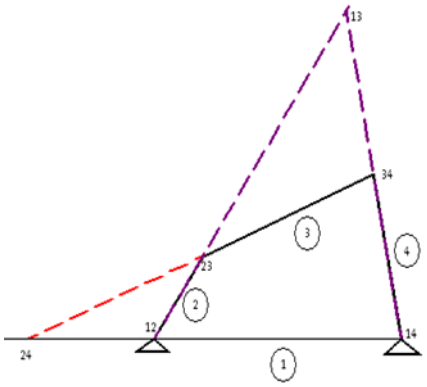


Figure 24: Four-bar mechanism used in evaluation

VELOCITY

The determination of velocity involves comparing the instant centers between every link and every other link. These instant centers can be classified as primary and

secondary and are determined using the Kennedy-Aronhold theorem [7]. This theorem states that the primary instant centers are those defined between connected links and are located at shared pivots. Each secondary instant center is located at the intersection of two lines (the end points of each line being instant centers), which are managed using the circle diagram method (Figure 25**)**. The instant center technique is chosen for velocity determination since it exhibits algorithm logic that could be generalized to any topology and is completely analytical. An illustration of the instant center technique is shown in Figure 24**.**

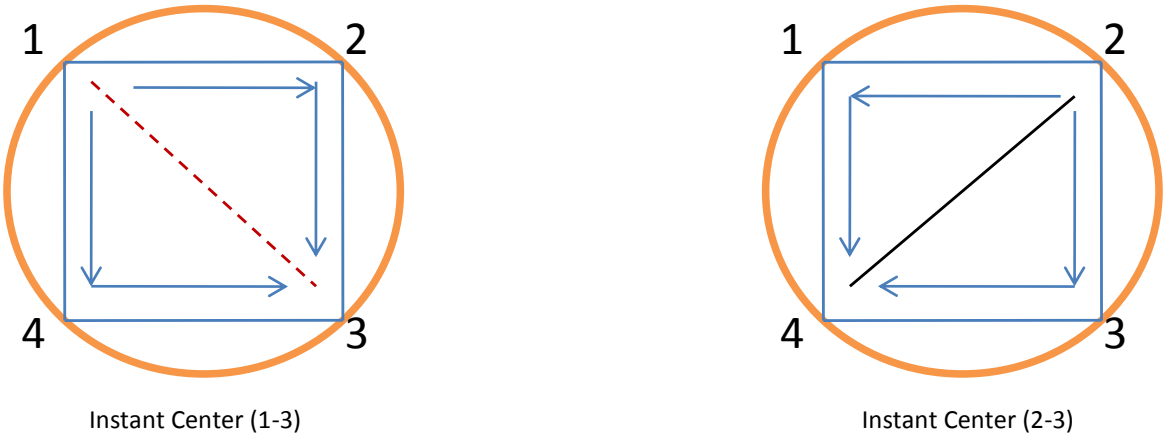


Figure 25: Circle diagram method for instant centers shown with respective paths

The basis of solving the instant center method computationally is to create a list of objects of type ϕ, for each pair of links, where:

$$\phi = \{x, y, \omega, link_i, link_j, pivot\}$$

(x,y – Instant center coordinates; ω - link angular velocity; pivot – common pivot to the links if primary instant center)

The list of ϕ's has [n * (n-1) / 2] members that correspond to the number of unique instant centers in the entire mechanism. During the first pass of the program,

primary instant centers are determined since their information is available from the candidate topology and those details are filled in the corresponding ϕ. Once information on all primary instant centers are available, secondary instant centers are obtained using an innovative methodology that replicates the circle-diagram approach. In the given example, primary instant centers are located in Figure 24 (secondary instant centers are indicated by dotted lines). The two secondary instant centers in the four-bar mechanism are determined as follows:

1. Instant Center (1-3): Intersection of line containing instant centers (1-2) and (2-3) and another line containing instant centers (1-4) and (4-3).
2. Instant Center (2-4): Intersection of line containing instant centers (1-2) and (2-4) and another line containing instant centers (2-3) and (3-4).

Each instant center is denoted by the indices of the links (say (1-3) denotes instant center between links 1 and 3_i and object ϕ denote links 1 and 3 respectively). The algorithm below indicates the method to determine primary and secondary instant centers.

do
{
For each object ϕ
#1 let i=$link_i$ and j=$link_j$
#2 Create two new instances of ϕ
#3 ϕ_1 =function(determine common instant centers pertaining to i)
#4 ϕ_2 =function(determine common instant centers pertaining to j)
#5 matrix k=function(determine the instant center paths (as in 12-23) from ϕ_1 & ϕ_2)
#6 ϕ(x,y) = function(determine new secondary instant center)
}while (ϕ(x,y) is determined for each instant center)

Function (determine common instant centers pertaining to i)

//this function will determine all common instant centers that have already been established

{

foreach(ϕ)

if (ϕ(x,y) ≠ NaN & (ϕ(linki)=i or ϕ(linki)=j))

add ϕ to ϕ_1

}

Function (determine the instant center paths (as in 12-23) from ϕ_1 & ϕ_2)

//this function will determine the instant centers paths from the common instant centers from ϕ_1 & ϕ_2

{

foreach(ϕ_1)

if (ϕ_1(linki) = = i)

node = ϕ_1(linkj)

else

node = ϕ_1(linki)

foreach(ϕ_2)

if (node = = ϕ_2(linkj) || node = = ϕ_2(linki))

add to matrix k

}

Function (determine new secondary instant center)

//this function will determine the instant center since path information is already available in the matrix k

//the matrix k is of the form [a b; c d]

{

determine each line from the two coordinates of the instant center

determine the intersection of two lines which gives the new instant center

}

During the execution of do-while loop, there could be instances where the mandatory two paths (in the circle diagram) are not obtained. So the loop would continue to the next instant center and revisit during the next cycle of the loop. The use of do-while-loop makes the process generic since, until all instant centers are determined, the process repeats. In case, during one complete pass, new instant centers are not determined due to an infeasible topology, the program exits. It should be noted that the instant center methodology works only if the mechanism consists of a four-bar chain. (This limitation is to be overcome in future research in the area). These built-in checks are some of the unique features of the analysis methodology presented in this paper. There are a few special cases built in to the tool for pin-in-slots and slider blocks (since the method of determining instant centers vary for such elements). These are incorporated in such a way that the generic architecture of the program is unaffected. Once all instant centers are obtained, the computation of angular velocities and linear velocities are carried out using a standard procedure as explained below for one of the links and pivots.

$$\omega_3 = \frac{\omega_2 \times (I_{1-2} - I_{2-3})}{(I_{1-3} - I_{2-3})} \text{ rad/s}$$

$$V_3 = \omega_2 \times (I_{1-3} - C) \quad \text{unit/s}$$

where ω_3 denotes angular velocity of link 3 and V_3 denotes the linear velocity of pivot C. Given a single known input angular velocity, other angular and linear velocities can be easily determined once the instant centers are obtained.

ACCELERATION

Once all velocity parameters are determined, angular and linear accelerations are computed. Acceleration of different pivots is based on the following equations (for the four bar mechanism in Figure 24):

$$a_A = a_D + 2\,v_A \times \omega_1 + r_{A/D} \times (\omega_1 \times \omega_1) + a_{Slip-A} + \propto_1 \times r_{A/D} \quad \text{1}$$

$$a_B = a_A + 2\,v_B \times \omega_2 + r_{B/A} \times (\omega_2 \times \omega_2) + a_{Slip-B} + \propto_2 \times r_{B/A} \quad \text{2}$$

$$a_C = a_B + 2\,v_C \times \omega_3 + r_{C/B} \times (\omega_3 \times \omega_3) + a_{Slip-C} + \propto_3 \times r_{C/B} \quad \text{3}$$

$$a_D = a_C + 2\,v_D \times \omega_4 + r_{D/C} \times (\omega_4 \times \omega_4) + a_{Slip-D} + \propto_4 \times r_{D/C} \quad \text{4}$$

where a refers to the absolute acceleration; $2\,v \times \omega$ corresponds to the Coriolis acceleration; $r \times (\omega \times \omega)$ is the radial acceleration and $\alpha \times r$ corresponds to the tangential acceleration. There are many unknowns namely a, $\propto$ and a_{slip} for the various moving links. Additionally, there are terms that are clearly zero such as ω_1 in the example since it is a ground link. Similar to the creation of the ϕ objects for instant centers, another object is created for developing acceleration equations. Once the equations are formulated, they are rewritten as shown in Figure 26 and reduced to satisfy linearity by eliminating terms that do not bear any significance (like ω_1 and $\propto_1$) as shown in Figure 27. This formulation and reduction is done automatically for each candidate topology and the order of the matrices is different in each case. Once linearity is achieved, the equations are solved using a matrix inversion technique. Cramer's rule is not practical in this case, since the method is extremely slow for matrices with more than six rows when solved on a typical desktop computer. Likewise, the Gauss-Elimination and Gauss Seidel techniques require dominant diagonals, which are not guaranteed in this automated method for generic topologies. Therefore, the LU Decomposition technique is

chosen wherein the existing matrix is subject to a reordering to ensure non-zero diagonals. The inversion technique that is implemented gives appreciable results with errors on the order of 10^{-9}.

$$\begin{pmatrix} 2\,v_A\times\omega_1+r_{A/D}\times(\omega_1\times\omega_1)_x \\ 2\,v_A\times\omega_1+r_{A/D}\times(\omega_1\times\omega_1)_y \\ 2\,v_B\times\omega_2+r_{B/A}\times(\omega_2\times\omega_2)_x \\ 2\,v_B\times\omega_2+r_{B/A}\times(\omega_2\times\omega_2)_y \\ 2\,v_C\times\omega_3+r_{C/B}\times(\omega_3\times\omega_3)_x \\ 2\,v_C\times\omega_3+r_{C/B}\times(\omega_3\times\omega_3)_y \\ 2\,v_D\times\omega_4+r_{D/C}\times(\omega_4\times\omega_4)_x \\ 2\,v_D\times\omega_4+r_{D/C}\times(\omega_4\times\omega_4)_y \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -r_{A/D} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -r_{A/D} & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -r_{B/A} & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -r_{B/A} & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -r_{C/B} & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -r_{C/B} & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -r_{D/C} \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -r_{D/C} \end{pmatrix} \begin{pmatrix} a_{A_x} \\ a_{A_y} \\ a_{B_x} \\ a_{B_y} \\ a_{C_x} \\ a_{C_y} \\ a_{D_x} \\ a_{D_y} \\ a_{Slip\text{-}A_x} \\ a_{Slip\text{-}A_y} \\ a_{Slip\text{-}B_x} \\ a_{Slip\text{-}B_y} \\ a_{Slip\text{-}C_x} \\ a_{Slip\text{-}C_y} \\ a_{Slip\text{-}D_x} \\ a_{Slip\text{-}D_y} \\ \propto_1 \\ \propto_2 \\ \propto_3 \\ \propto_4 \end{pmatrix}$$

Figure 26: Matrix Formulation of Acceleration terms

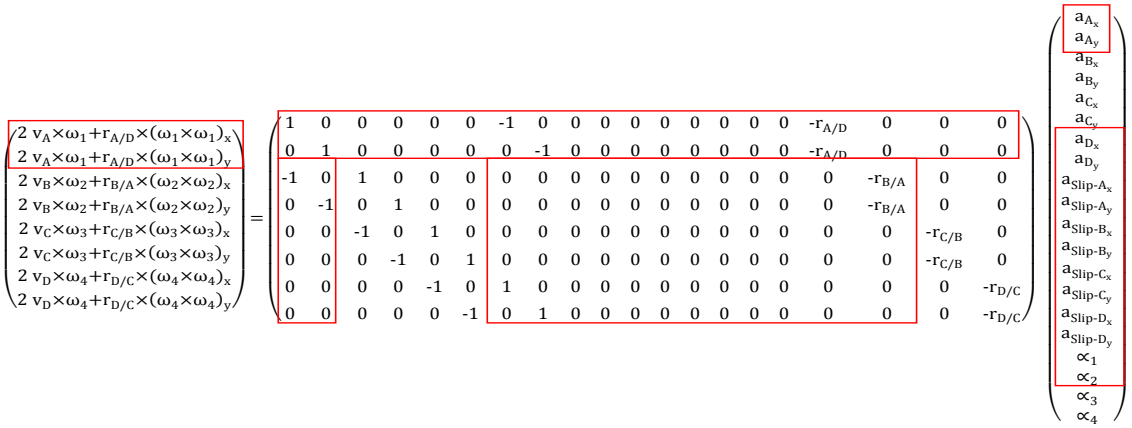


Figure 27: Acceleration Matrix Reduction

POSITION

Once velocities and accelerations of different links and pivots are obtained, the new positions of links and pivots are determined. A Taylor series approximation of position is first employed from the velocity and acceleration values followed by an analytical method to perfectly eliminate numerical error using the geometric circle

intersection technique. As can be seen in Figure 28, pivot positions can be found by geometrically determining the intersections of two circles whose radii are the fixed link lengths. In the example, a four-bar mechanism is shown where the input crank on the left moves at a constant velocity and thus its position can be analytically determined at any time step. The rocker pivot position must therefore be at one of two points given the fixed link lengths. The following algorithm gives the overall methodology for determining position.

#determine time step and angle increment depending on input omega

#determine lengths between pivots (link lengths)

#Determine new input position of pivots connected to the input

#Assign *Not-A-Number* (NaN) to all pivots

#do

{

#Take two pivots whose new positions are known which are connected to the third pivot whose new position is to be determined

#Find the lengths between the chosen pivots

#Determine new position of third pivot through intersection of two circles technique

}while (each pivot (x,y) ≠ NaN)

As in velocity determination, there are cases where pin-in-slots and sliders require slightly different computation (like circle-line intersection), which is adapted into the program structure without affecting generality.

The position module is also generic since the do-while loop operates in the same way as explained during instant center determination and continues until all pivots are

assigned new positions. The position module also has the built-in benefit to check when links can no longer rotate. This occurs when the circles for the mating links do not intersect as in the given example since this produces an imaginary number by taking the square-root of a negative sum.

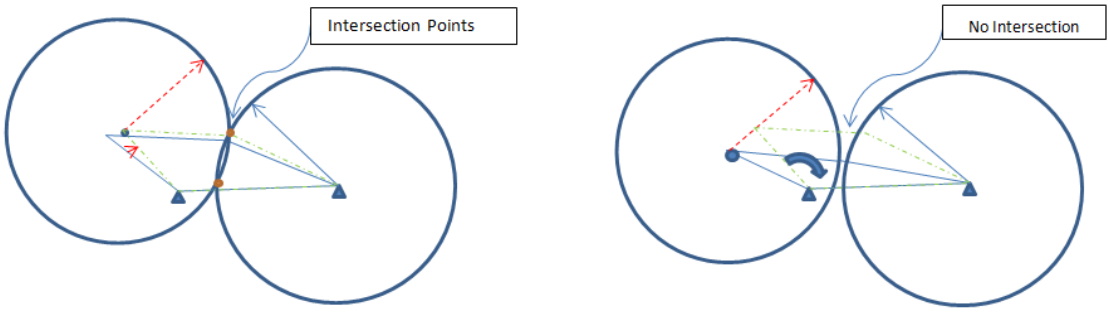


Figure 28: Circle Intersection method for determining position. a) Indicates that in nominal cases there are two possible positions, b) The method also detects when rotation is no longer feasible.

Once the kinematic properties are determined, the generated path, function or motion is compared with the user specifications and the necessary action (whether to accept the design or change parameters or topology) is considered. The capability of the program to generate kinematic properties of different topologies is shown in the following figures with examples of a four bar mechanism (Figure 29) and a quick return mechanism (Figure 30).

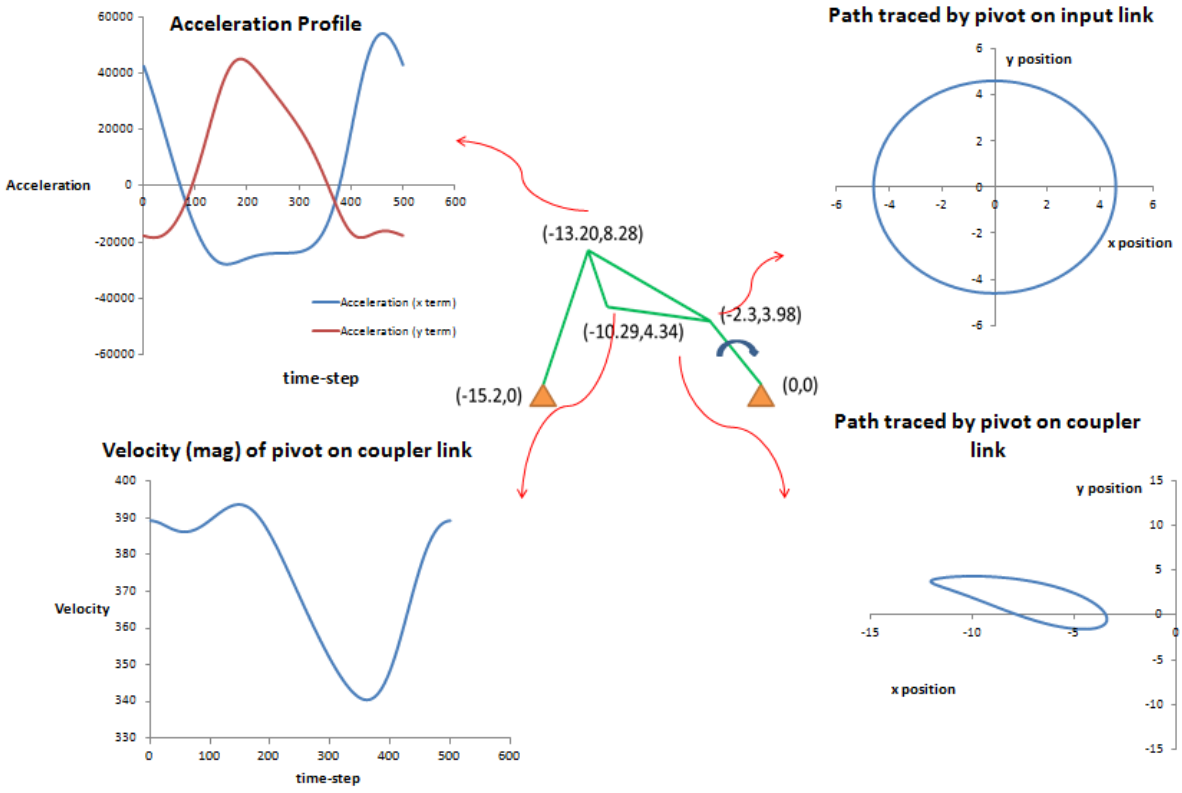


Figure 29: Kinematics of a four-bar mechanism

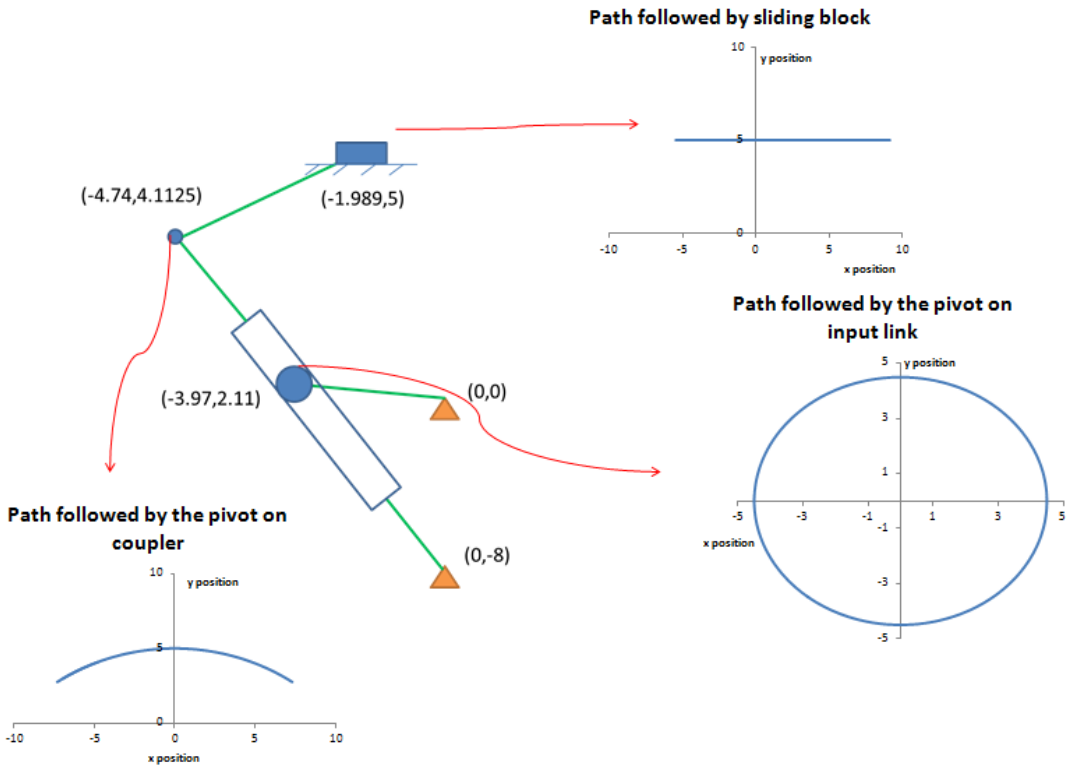


Figure 30: Position Kinematics of a quick return mechanism

Chapter 3: Discussion

Representation and evaluation are important aspects in automated design synthesis research. In planar mechanism synthesis, it assumes greater significance due to the requirement of a method that encompasses the different element types such as rotary links, sliding blocks and pin-in-slots. Though the final intended outcome is a generalized and automated designer of planar mechanisms, the report focuses on the representation and evaluation methods that aids in achieving the final objective. The method developed in this research is unique since it combines some of the best practices of graph grammar methodology along with the traditional design methods to create rules that operate in an iterative manner to generate the results.

REPRESENTATION

Since representation is the starting point, formulation of a scheme should ease the remaining challenges at generation, evaluation and optimization. Representing mechanism elements such as links and pivots as nodes and their connections as arcs presents an alternative approach when compared to existing methods like the Systematic Method**,** which require designers to have prior understanding of the transformation between the graph representation and actual mechanism. This prior understanding is not required in this case. The different labels associated with the links and pivots and their coordinate information are able to convey the type of mechanism represented by the graph without requiring any additional arbitration. Thereby any design generated can be easily visualized by the user who can continue with the post-processing activities using the generated design. The rules that are concerned with the design generation are few and are capable of generating varied designs as illustrated earlier. The rules also consist of several graph-centered Booleans that are used to specify when a rule is valid and this is crucial in carefully controlling the space of generated mechanisms. This space of

generated candidates is compared to the other approaches in the literature that attempt to enumerate the space of planar mechanisms. This is the first step towards proving the correctness of the rules and the utility of method.

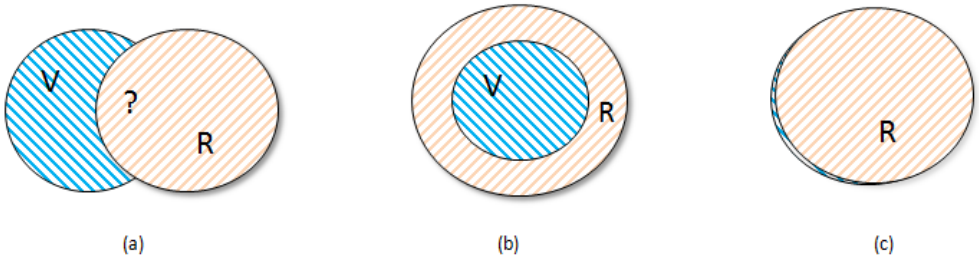


Figure 31: Rule Validation a) The intersection of the valid set, V, with the set created by rules, R, is desired. b) Through a depth-limited depth-first search of the tree, it is found that R encompasses all of V. c) Additionally, no solutions are found in R that violate Gruebler's equation, hence the difference is minimal.

In order to show that the 16 rules are valid, it is necessary to take a set theory approach. Consider the abstract set of all valid planar mechanisms, V. For the set of all candidates created through a rule-set, R; we gauge the validity of R by determining the amount of intersection with V (Figure 31a). There are several approaches that could be taken to measure this. For example, one could directly take the intersection of V and R ($V \cap R$) only if V is well understood. However, in many design problems the valid space, V, cannot be enumerated, but one can easily check when a solution is within the set.

The approach used here is to see if R encapsulates all the elements of V ($V \subset R$; Figure 31b) and then check whether there are any solutions in R that are not in V ($R - V$; Figure 31c**)**. If the latter is the null set ($R - V = \{\ \}$), then it can be proved that no infeasible solutions are found, and taken with the former, that the two spaces are

equivalent. This is challenging in comparing graph languages that result from a set of grammar rules since graphs are difficult to compare [27] and the search tress are large and perhaps intractable.

A review of the literature [7] presents the list of unique variations for different n-bar mechanisms of 1-DOF. Since the list is compiled considering only revolute joints without assignment of ground and input links, it is not appropriate to compare the capability presented here on a one-to-one basis. Nevertheless, to test the validity of the rules we conducted a depth-first search of the tree (limited to a certain level) and found that the candidates generated can be classified as listed in the literature. This is a first step at proving that R includes all of the solutions found in V for 1-DOF revolute joint mechanisms. In order to show that R does not include any infeasible solutions, each solution found in the tree-search is checked against the Gruebler's equation and none had a calculated degree-of-freedom with a value other than 1. Hence, there does not seem to be solutions in R that are not in V.

Since part of the research deals with a methodology to represent mechanisms using graph grammar for synthesis purposes, isomorphism and confluence become important issues to address. Isomorphism refers to the structural equivalence of topologies and researchers have developed different methods to identify and deal with isomorphic solutions as stated in the review by Mruthyunjaya [28]. While a particular degree of freedom system is desired by the user, there are usually constraints on kinematics that are not considered in isomorphism. Since the goal is synthesis wherein topologies generated by a search process will be evaluated, one could take isomorphism into advantage to reduce computation. However, the time associated with detecting isomorphic graphs does not justify the additional and considerable computational resources required in comparison to running another kinematic analysis. But having said

so, it is imperative to develop rules that reduce the occurrence of structurally equivalent topologies. Thus, the 16 rules developed result in distinct topologies and not their isomorphic variations. Also, the rich set of labels that are associated with every node and arc used in this research prevent the ambiguity in dealing with isomorphism. Without labels, it is not possible to easily identify a single mechanism as is the case in other research. The topological variations, as depicted in (Figure 32**)** where the topology is the same but the desired output pivot's locations are different, are achieved by another set of rules that are applied on a particular topology to determine if the user's criterion is satisfied.

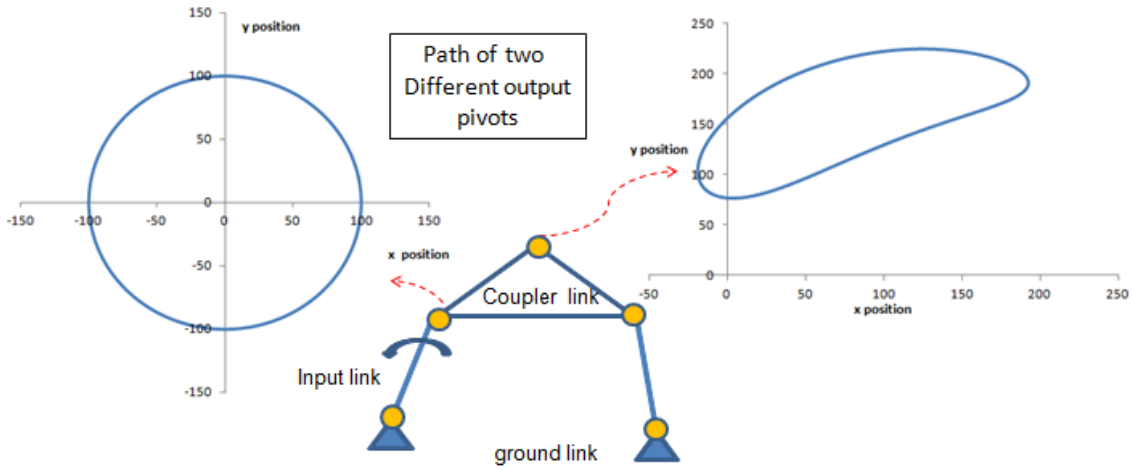


Figure 32: Same topology but different output characteristics

Referring back to Figure 2, it is important to note that search trees may unavoidably include repeat states. This indicates that there may be multiple paths to the same configuration as indicated in Figure 33, where the Stephenson–I mechanism is found using a different path than that shown in Figure 21. The difference is highlighted in the box outline. This is an issue in graph rewriting systems known as confluence, wherein identical topologies at different locations in the tree can be traced to a common parent.

The variations of the Stephenson–I mechanism may be obtained at different levels on the tree and is an ideal example of confluence. Through other generative grammar research, we can note that it is sometimes impossible to create a set that eliminates all confluence between rules. This topic is an important area of research to reduce the search space by avoiding repetitive states.

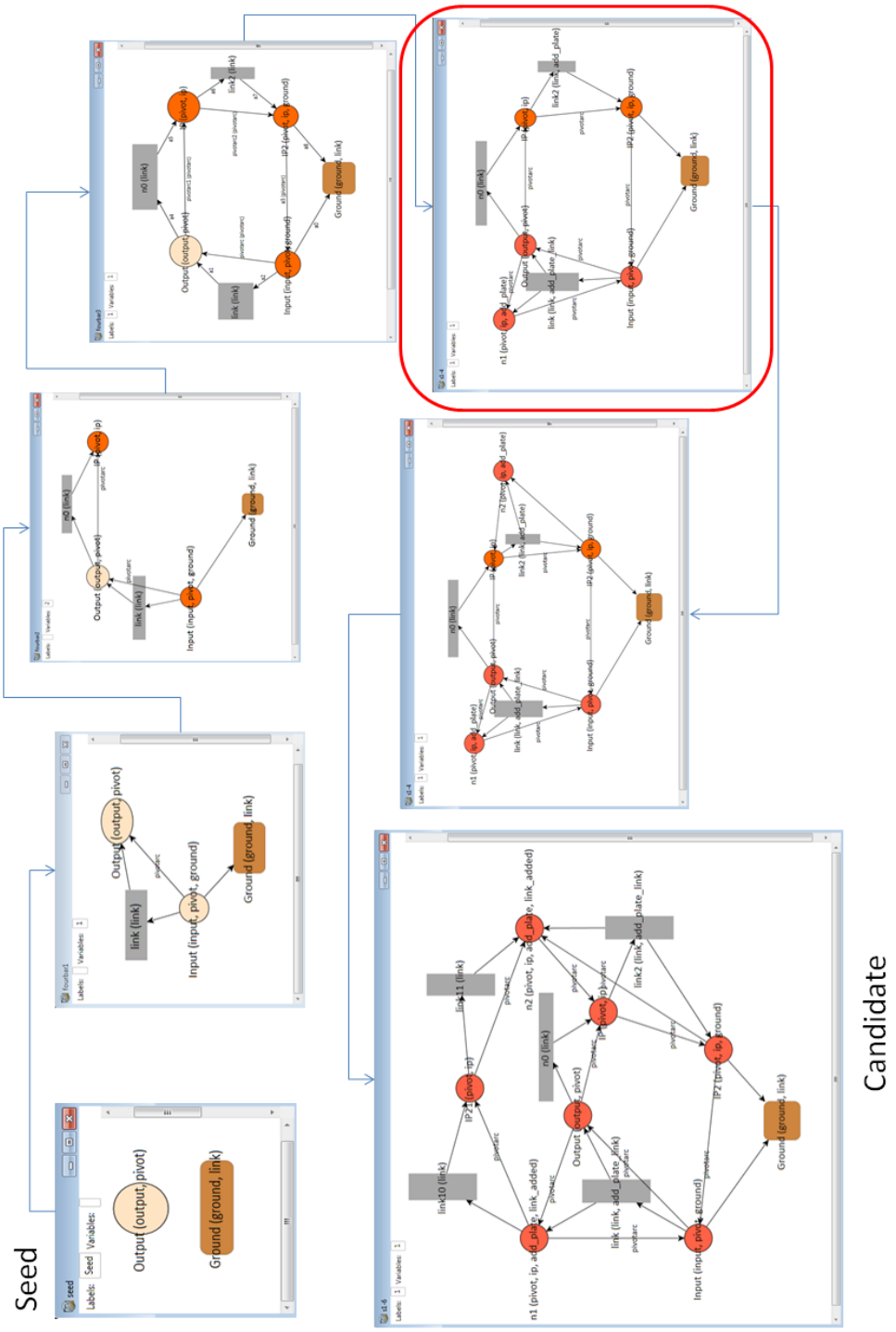


Figure 33: Stephenson -I mechanism generated with a different rule receipe

EVALUATION

The evaluation module operates on candidates generated by the concept generator with the limiting factors being the four-bar chain and single degree of the mechanism. The kinematics obtained is analytical and accurate. In order to test the accuracy of the computed values, a four-bar mechanism is chosen since it provides a good reference since analytical equations exist for the motion and thus the path can be determined with no numerical approximations. The baseline analytical result is compared with this method along with results from Working Model and SAM. Figure 34 shows how the ratio of the coupler link's length changes to the actual link length (a link is rigid, and should not change length throughout the simulation) changes with time. It may noted from the figure that the analytical equation method and the method programmed in the report are accurate. The results of SAM are very close to the actual value but Working Model results greatly vary compared to all other methods. Figure 35 gives the variation in position and velocity between Working Model and the method developed in the report. The main idea behind this comparison is to prove the utility of developing a method for graph-based analysis systems. The developed evaluation framework can also be adapted to any research activity on kinematics of planar mechanisms. Another important issue in such methods is computational speed and the evaluation method is comparable in speeds with the commercial packages. There are no commercially available kinematic analysis tools that have a built-in concept generator other than the WATT Mechanism Suite but even there it is restricted to a few topologies.

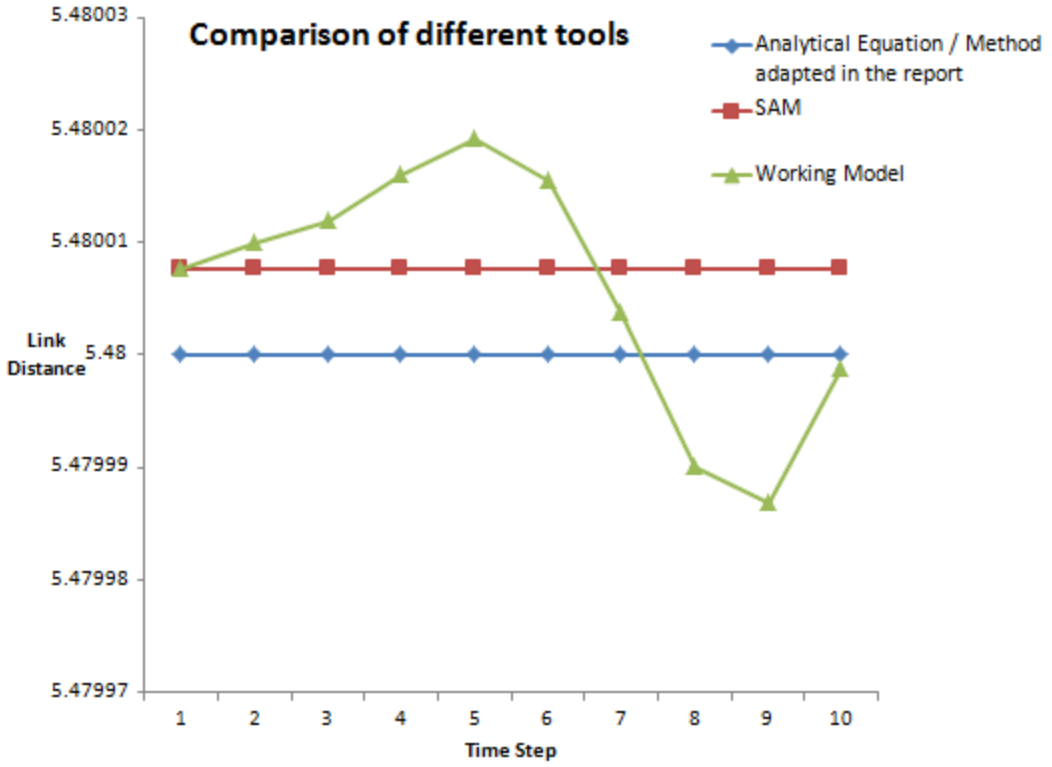


Figure 34: Variation in link length of a four-bar mechanism using different tools

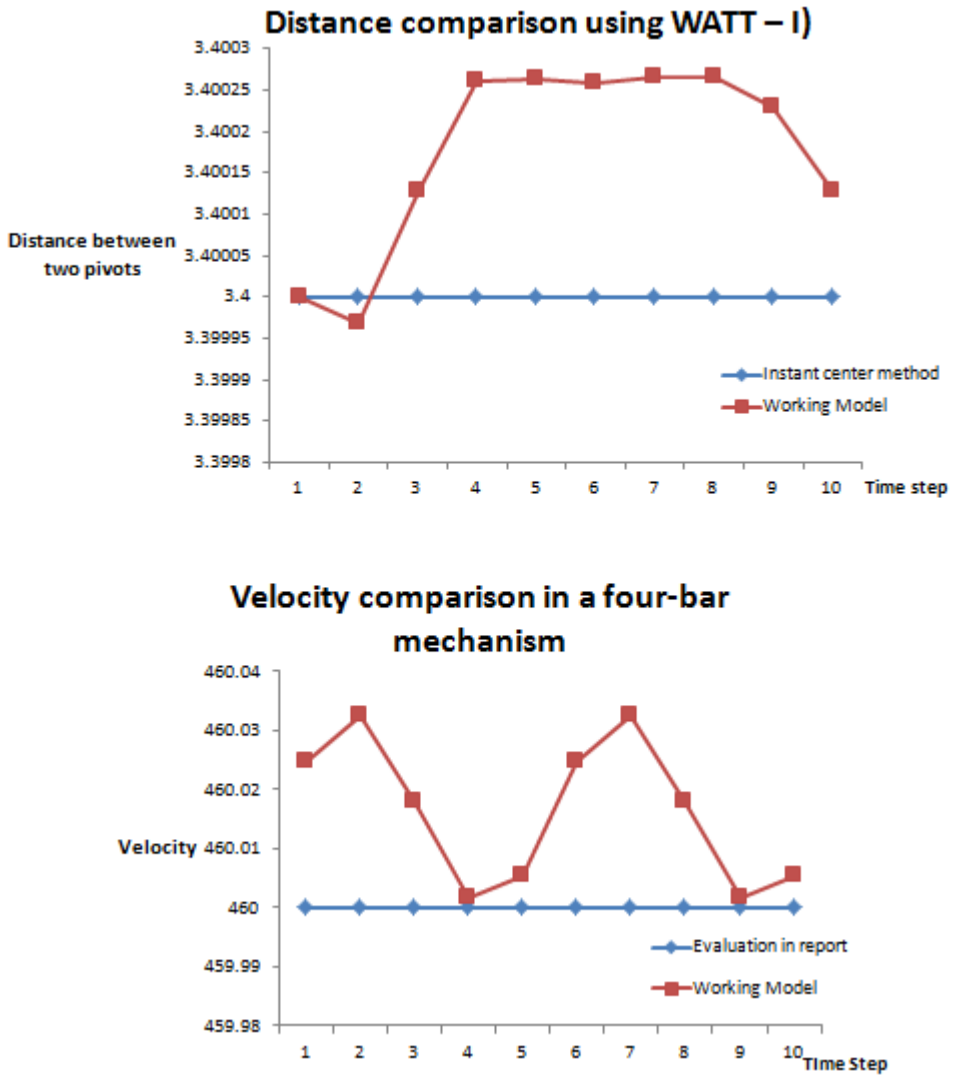


Figure 35: Comparison of different parameters of different mechanisms between Working Model and the method developed in this report

It was also decided to check the results with recent research that have dealt with evaluation methodologies for automated planar mechanism synthesis. In this regard, the method presented here is tested with the position kinematics obtained from [29] and [30] for the mechanism shown in Figure 36. In this example, the four-bar starts at the pivot locations shown in the figure. The goal is to properly predict the position of the output

rocker for a constant rotation of the input crank. For such a simple topology, the analytical equations of the positions can be found, and the third column shows the exact position for one particular location published by the other authors. The method presented in this report is compared to this analytically result as well as the results obtained by these authors. From Table 4, it is clear that the new method is superior in terms of positional accuracy and also in computing speed. It also finds the position in one-tenth the time (2 s.) compared to 22.9 s. and 24.1 s. of the energy methods illustrated in Gea [29] and Chen [30] respectively. This result was obtained on a standard desktop computer (1.73 GHz; 2 GB RAM) where 200 time-steps were common to all approaches. In addition, the complete kinematics (velocity, acceleration and position) of the four-bar shown is obtained in 18 seconds while the competitive methods are only determining position.

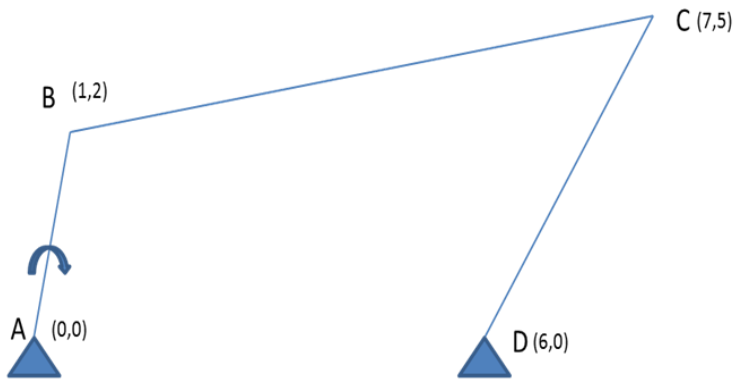


Figure 36: Four-bar mechanism used for comparison with new analysis methods

Table 4: Comparison of position with other techniques

Pivot	Initial Coordinates	Analytical (Exact Values)	New Method	Accuracy	Total Potential Energy(TPE) [5]	Accuracy	Constrained SuperPosition Method (CSM) [13]	Accuracy
B	X=1	0.2557	0.2557	0.00%	0.2630	2.87%	0.2595	1.51%
	Y=2	2.2214	2.2214	0.00%	2.2214	0.00%	2.2210	0.02%
C	X=7	6.3193	6.3193	0.00%	6.3296	0.16%	6.3237	0.07%
	Y=5	5.0890	5.0890	0.00%	5.0872	0.04%	5.0890	0.00%
Computational Time			2 sec		22.9 sec		24.1 sec	

The examples and figures from the previous section clearly indicate the capability of the method to analyze the kinematics of different topologies. At the same time, it must be noted that the results obtained are completely analytical and superior in accuracy and comparable in computation speed when compared to SAM and Working Model. These results have been verified with the analytical methods available in Norton [7]. The program also has in-built capability to determine infeasibilities such as a truss-type mechanism (no motion capability) by virtue of the geometric method employed for position determination and avoids further computation and saves resources. Additionally, advances in object-oriented programming have helped in generalizing the determination of instant centers and this is one of the only software tools available for the same (there are no commercial tools available for instant centers). Overall, the method presented here takes advantage of advances in computational techniques to arrive at a completely analytical kinematics package for n-bar planar mechanism. Future research is to synthesize planar mechanisms for which this analysis package is to be integrated with a concept generator. The lack of topology synthesis in existing software tools and their inability to support external applications (concept generators) has been an important motivation for the development of this analytical kinematics program. In order to fully integrate with automated concept generation methods, it will be necessary to develop

path, motion or function error determination schemes to validate designs with user specifications.

Chapter 4: Conclusion

The human approach to conceptual design is difficult to comprehend due to its complex nature and the uncertainties involved. In order to automate this process, it is very important to recreate the entire design space so that all probable candidates are evaluated. This is done using different design generation rules as the computer is not intelligent enough to determine the starting point. This logic applies to the design of planar mechanisms, which are equally complex. But nonetheless there is a simple equation that can be invoked to determine the validity of the generated mechanism. The research reported here is focused on the representation and evaluation schemes that are integral to the overall process of automated synthesis of planar mechanisms. One of the key improvements to the representation schemes already available is that nodes represent both links and pivots and a rich label set is used to identify the type of link and pivot. The representation leads to design rules that replicate the iterative design process that we humans usually follow. With fewer rules that are organized into different rule-sets, we are able to recreate many generic planar mechanisms and their inversions. The generated topology graph has various included data such as coordinate information and degree of freedom that aid in better manipulation of the same during evaluation and optimization processes.

The evaluation module is an important development since the concepts are tested for their kinematics accurately without comprising on time. The instantaneous center of rotation method and the vector polygon approach are used to analytically determine the kinematic properties of the generated mechanisms. Due to this generality, the method can be easily adapted to the automation of planar mechanism design process and can accept candidate designs as input directly from a search process. This saves considerable resources compared to commercial software packages like SAM and Working Model that

require a designer to manually construct the designs for analysis. The method also determines infeasible configurations for a particular mechanism at two stages (the instant center stage and the new position determination stage) thus eliminating unnecessary computation. The drawback with this method is that it is limited to mechanisms with four-bar loops and single degree freedom systems, and is expected to be overcome in future research though mechanisms are only theoretically derived and no practical device falls within this category. Future work required for realizing the automated concept generator is to adapt suitable search and optimization techniques to generate valid solutions for a problem.

The creation of a valid representation (16 grammar rules) that is able to generate the entire design space by accommodating rotary and prismatic joints and the evaluation that is quick and accurate show promise for the future automated synthesis of planar mechanisms.

CONTRIBUTIONS

My contribution in this research has been the development of the design representation scheme that provides a generic scheme to represent a variety of planar mechanism elements. I have demonstrated rotary joints, sliding blocks and pin-in-slots. Supplemental to the design representation scheme are the different rules (design, inversion and modularization) that capture the entire set of valid designs. Finally, the evaluation tool determines the kinematics of the generated mechanism and compares it to user specifications. The major components in the automated synthesis of planar mechanisms have been developed and presented in this report.

Appendix A: List of Design Generation Rules

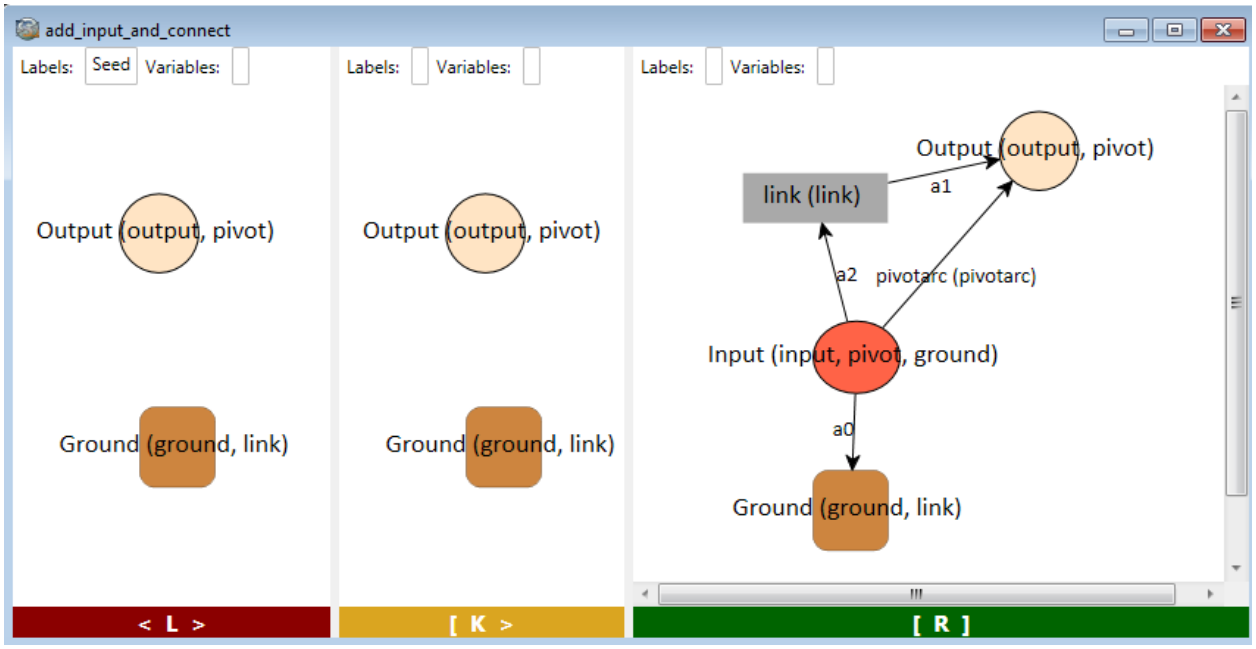


Figure A1: Rule adds an input pivot and a link to connect the input with the output pivot. The input pivot is also connected to the ground link. The left hand of the rule consists of a global label "Seed"

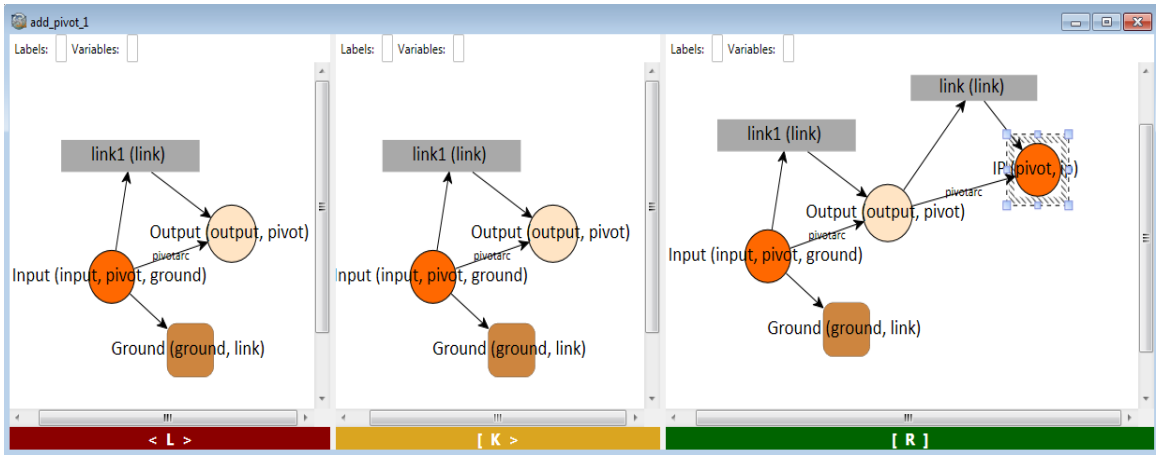


Figure A2: Rule adds a link node and a pivot node to the candidate in < L >

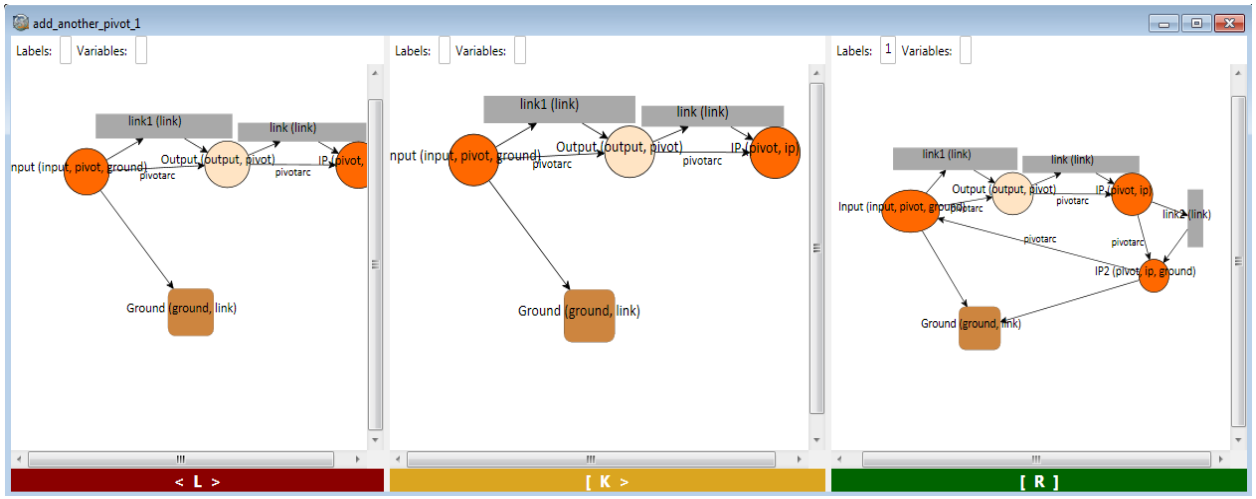


Figure A3: Rule adds a link node and a pivot node to create a four-bar mechanism. The new pivot node is connected to the ground link.

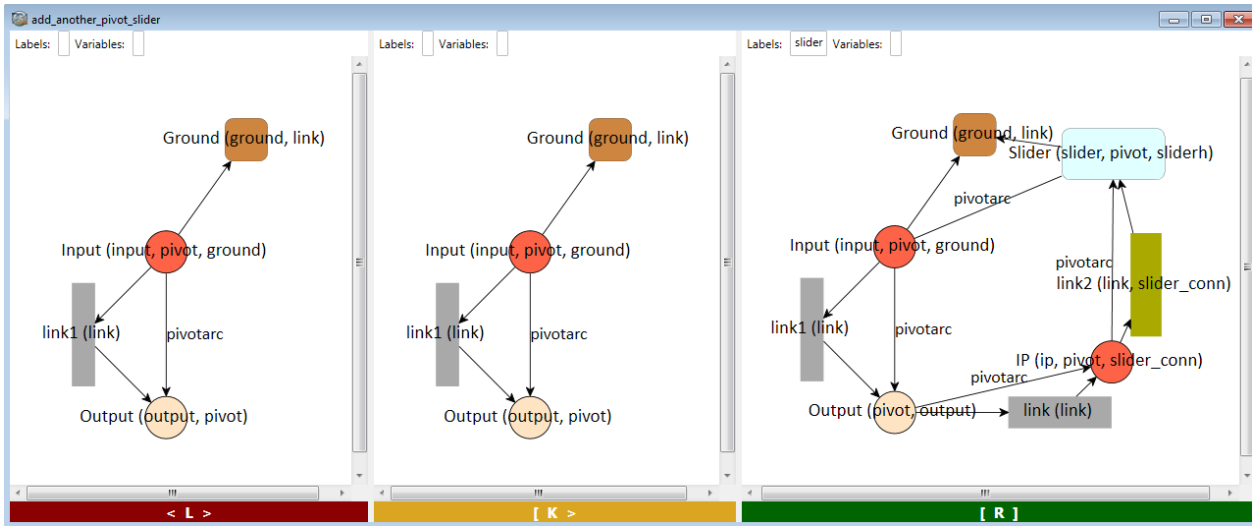


Figure A4: Rule adds a horizontal sliding block (with label “sliderh”) to the graph in <L>. This rule works only to create a four-bar mechanism

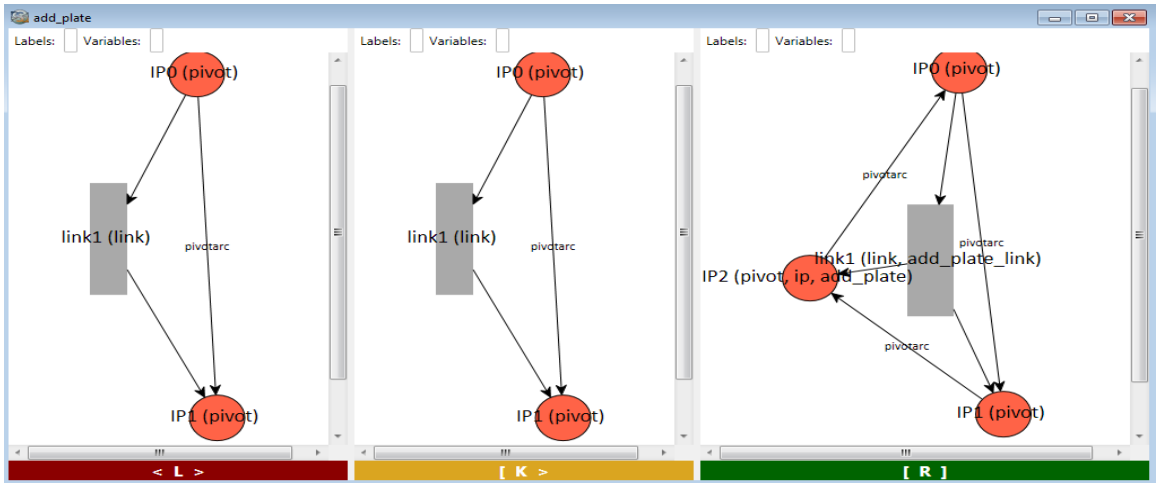


Figure A5: Rule adds a new pivot to an existing link with two pivots making a ternary link

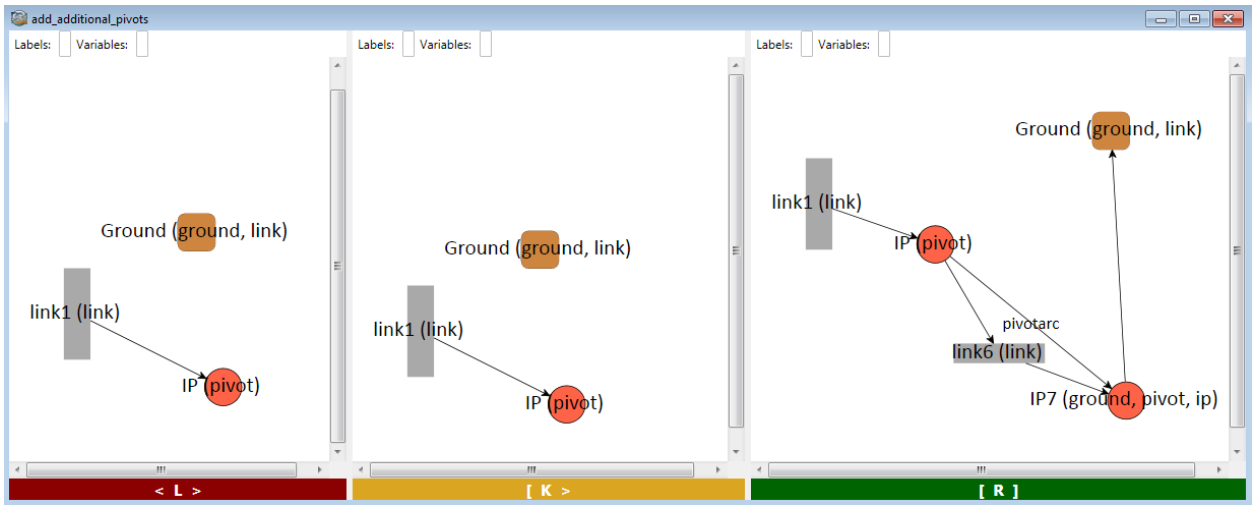


Figure A6: Rule adds a link node and a pivot node (with ground label) to any pivot that is not shared by two links

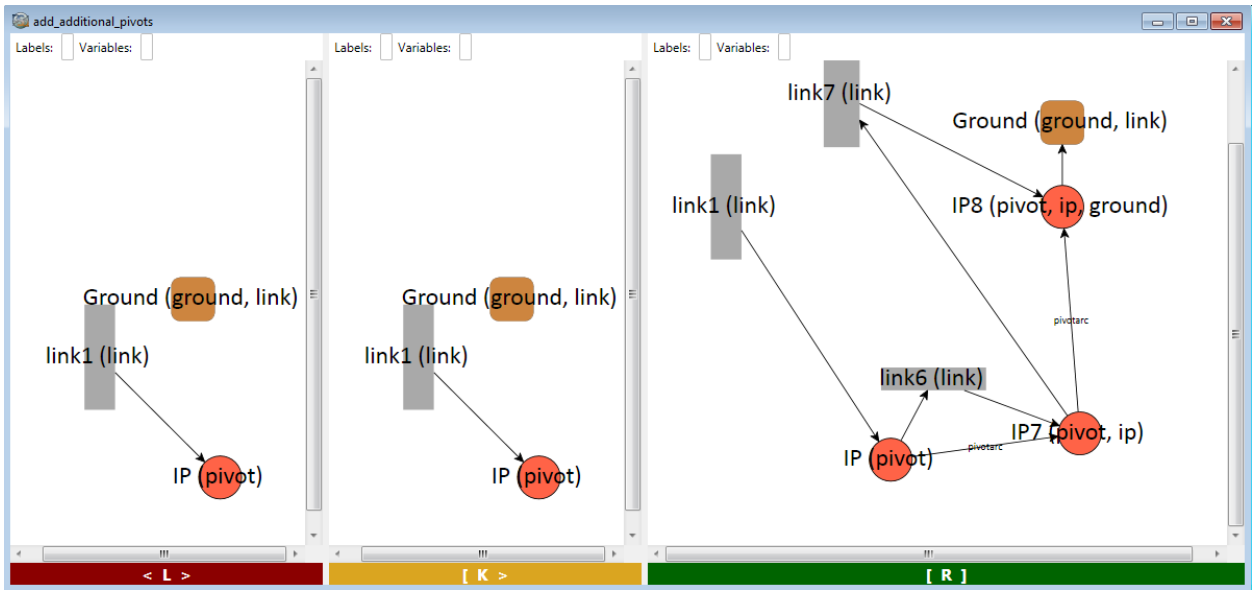


Figure A7: Rule connects a pivot node to the ground by means of two links. Note that the pivot node on which the rule is applied is connected to one link only.

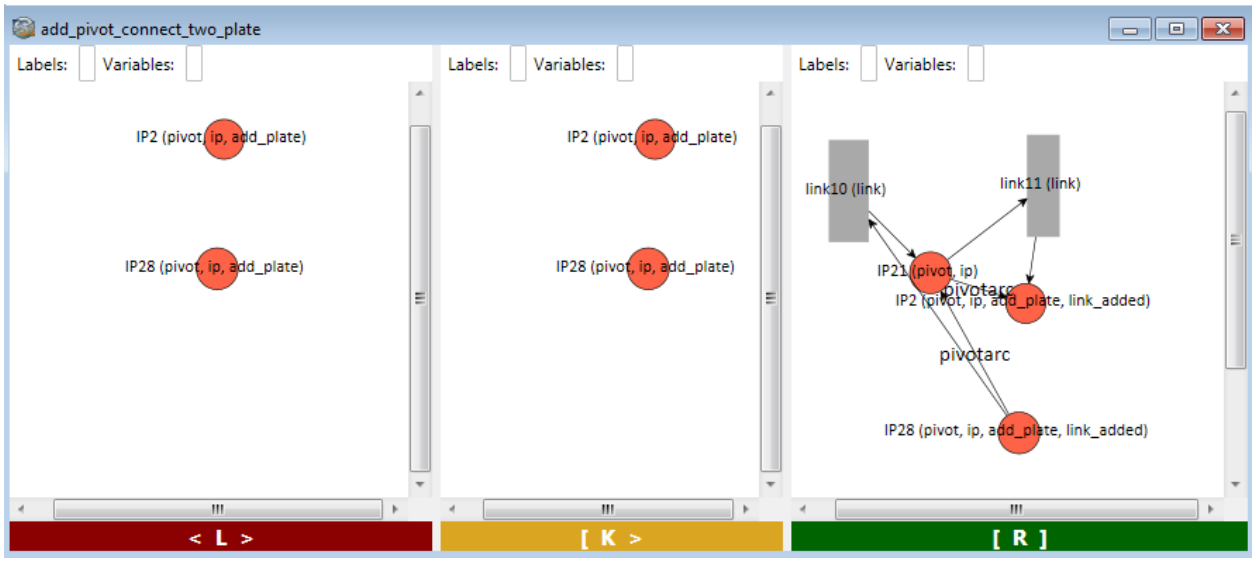


Figure A8: Rule connects pivots (that are not shared by more than one link) of two ternary links by two links with a pivot shared between the new links

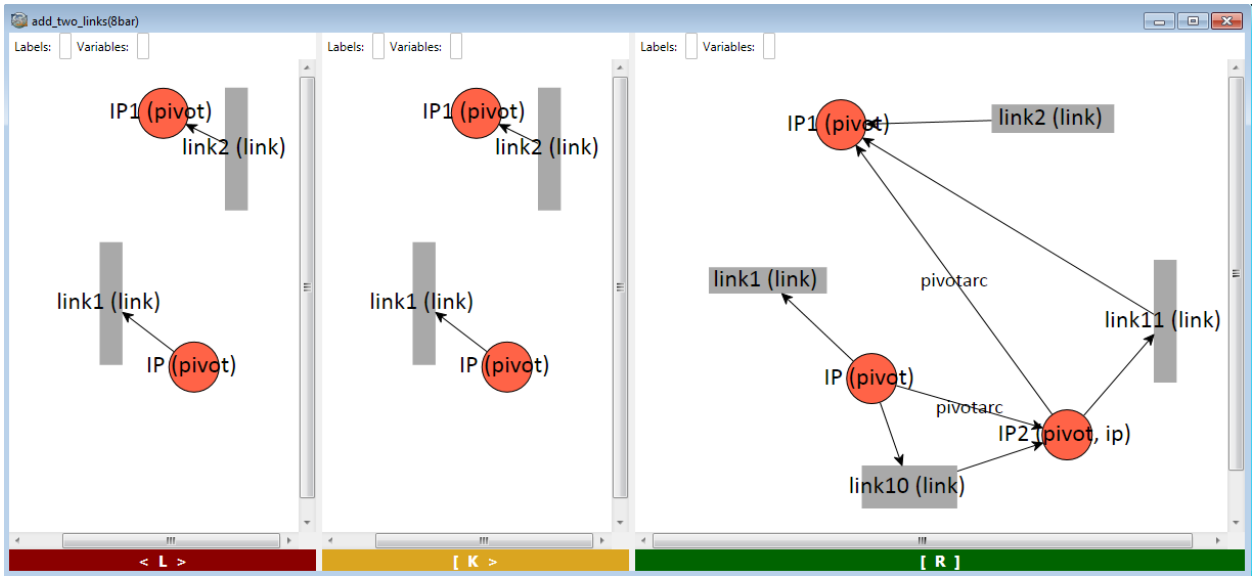


Figure A9: Rule connects two pivots by means to two links and a pivot shared between them

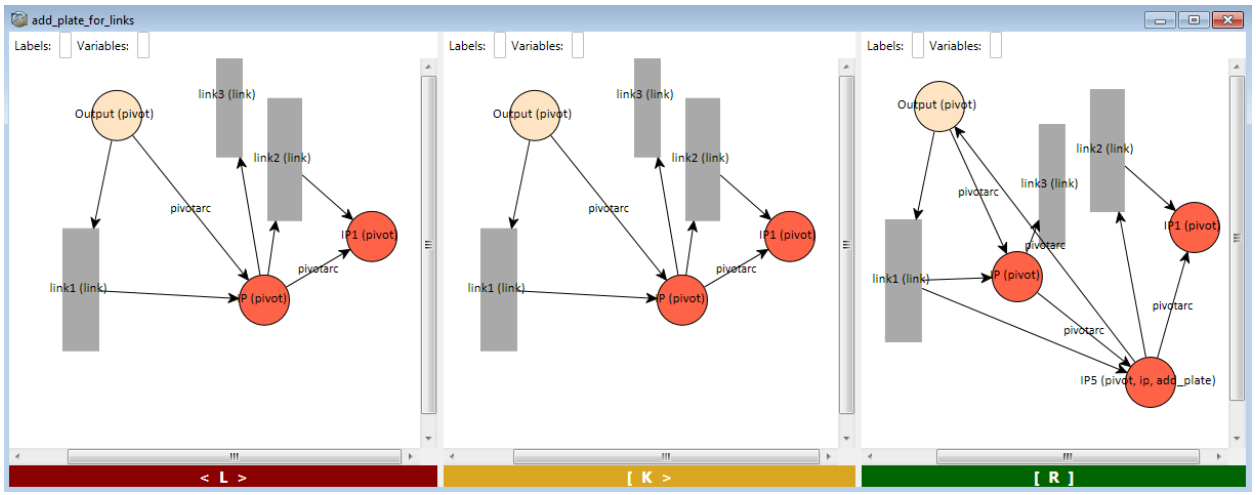


Figure A10: Rule replaces an existing binary link with a ternary link. This rule specifically works for a WATT Mechanism

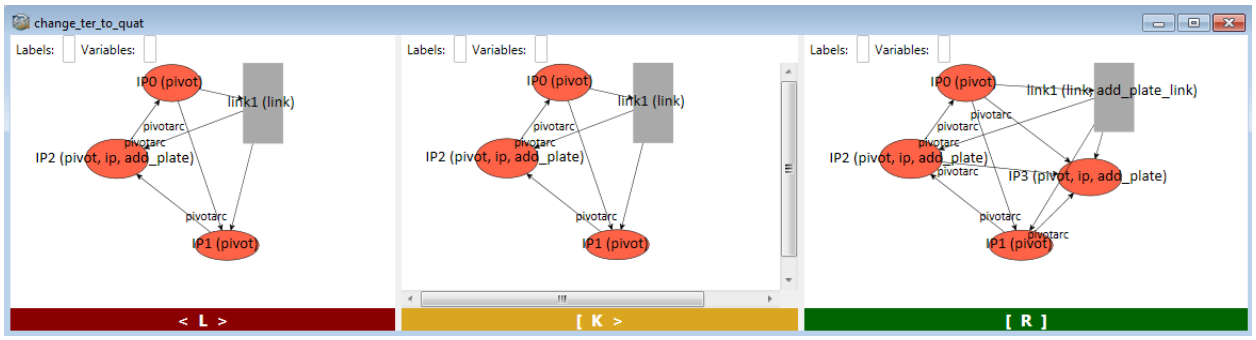


Figure A11: Rule adds a new pivot to a ternary link

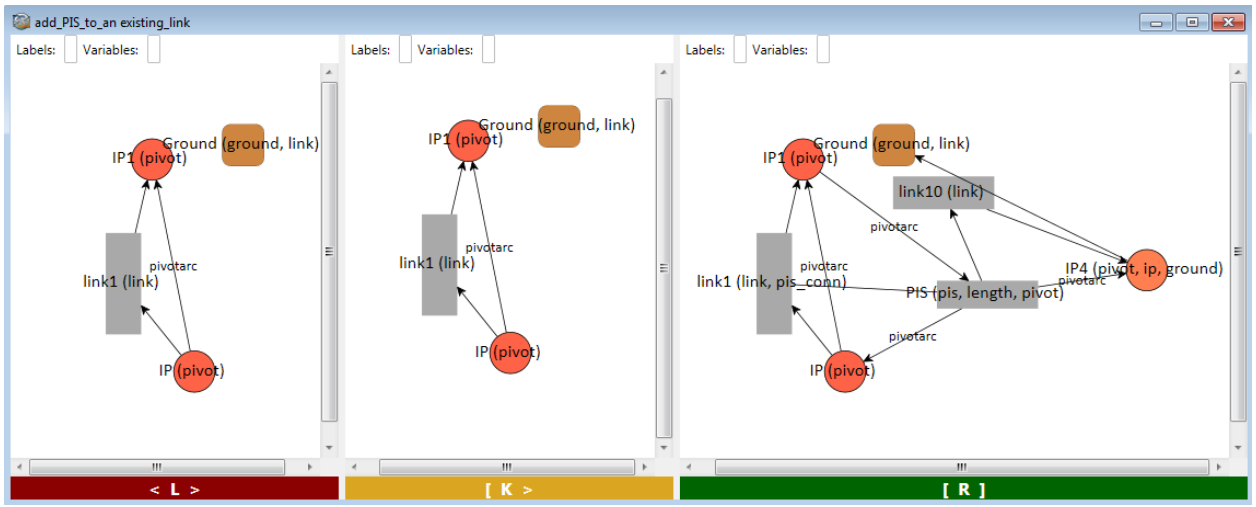


Figure A12: Rule connects a pin-in-slot element to an existing link. The other end of the pin-in-slot is connected to the ground link using a ground pivot.

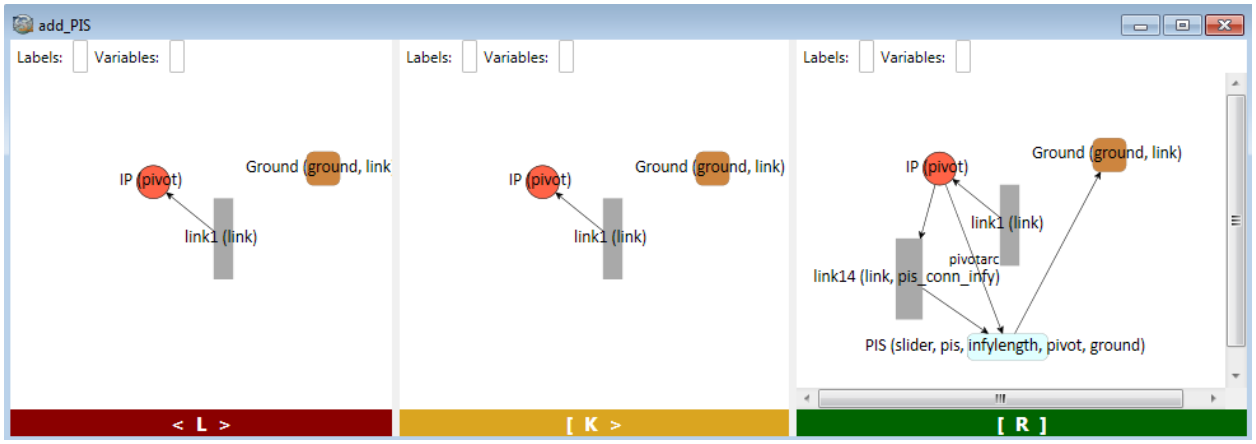


Figure A13: Rule adds a pin-in-slot element to a link that is connected to only one link

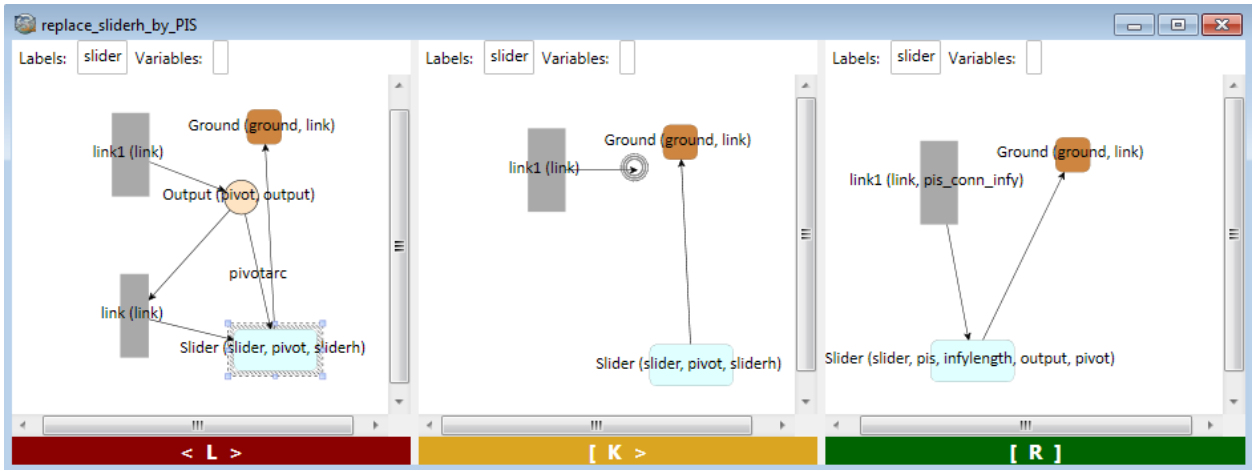


Figure A14: Rule replaces a horizontal sliding block with a pin-in-slot element

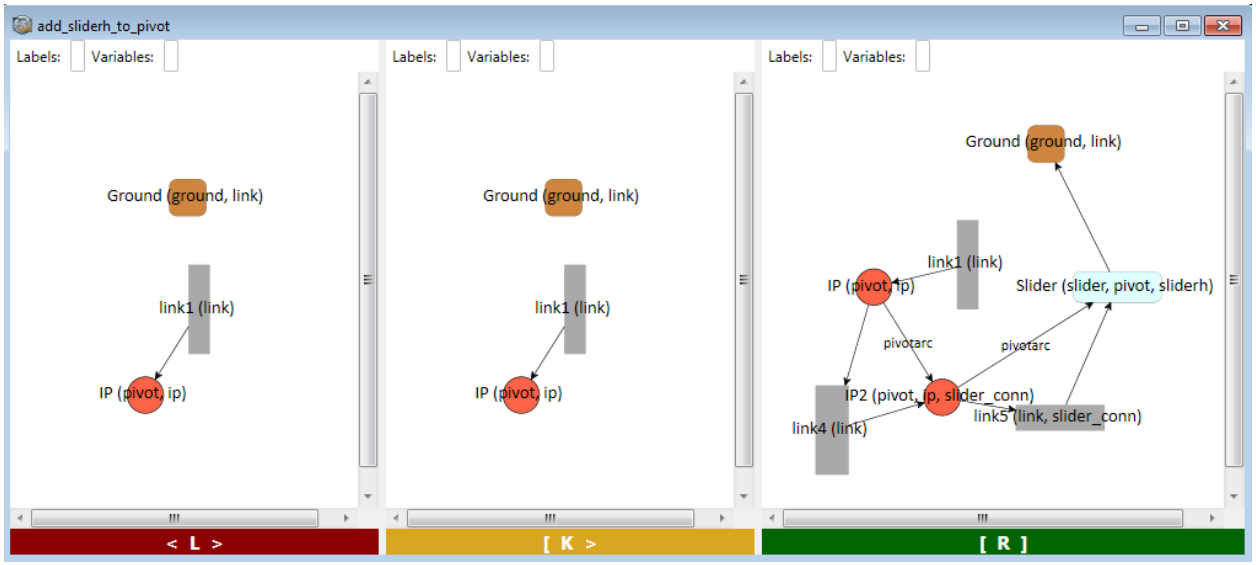


Figure A15: Rule adds a horizontal sliding block to a pivot. This rule works with topologies with more than four links

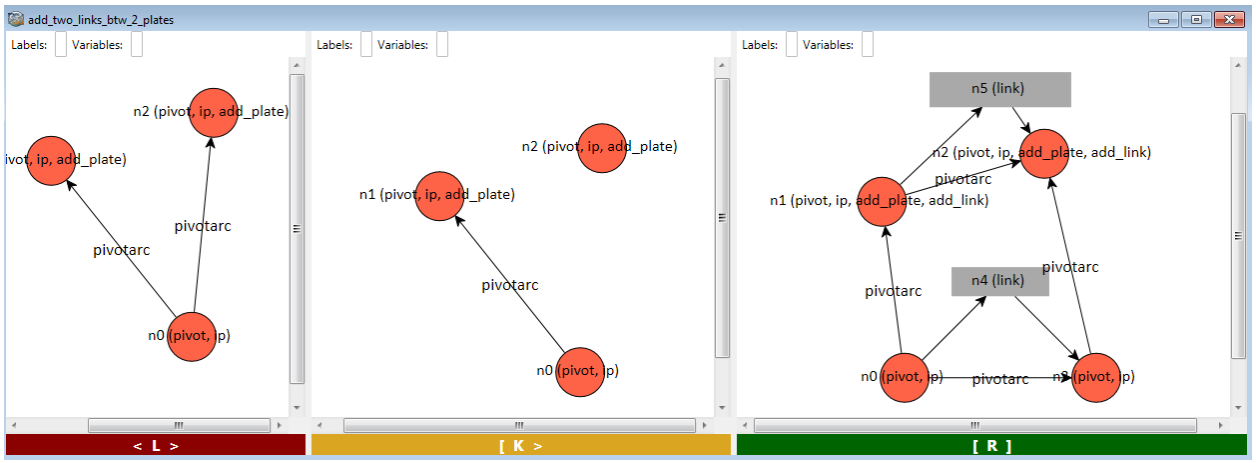


Figure A16: Rule adds two links between two pivots that are on two ternary plates

References

[1] P.J. Martin, K. Russell, and R.S. Sodhi, "On mechanism design optimization for motion generation," *Mechanism and Machine Theory*, vol. 42, Oct. 2007, pp. 1251-1263.

[2] A.A. Smaili, N.A. Diab, and N.A. Atallah, "Optimum Synthesis of Mechanisms Using Tabu-Gradient Search Algorithm," *Journal of Mechanical Design*, vol. 127, 2005, p. 917.

[3] N. Diab and A. Smaili, "Optimum exact/approximate point synthesis of planar mechanisms," *Mechanism and Machine Theory*, vol. 43, 2008, pp. 1610–1624.

[4] M. Stolpe and A. Kawamoto, "Design of planar articulated mechanisms using branch and bound," *Mathematical Programming*, vol. 103, 2005, pp. 357–397.

[5] E.K. Antonsson and J. Cagan, *Formal engineering design synthesis*, Cambridge University Press, 2001.

[6] K.J. Waldron and G.L. Kinzel, *Kinematics, dynamics, and design of machinery*, Wiley, 2004.

[7] R.L. Norton, *Design of machinery*, McGraw-Hill Professional, 2004.

[8] T.H. Cormen, *Introduction to algorithms*, MIT Press, 2001.

[9] A.G. Erdman and G.N. Sandor, *Mechanism design*, Prentice-Hall, 1997.

[10] N. Sclater and N.P. Chironis, *Mechanisms and mechanical devices sourcebook*, McGraw-Hill, 2001.

[11] "ARTAS - Engineering Software," *http://www.artas.nl/*.

[12] "WATT Mechanism Suite," *http://www.heron-technologies.com/watt/*.

[13] "Working Model 2D - Home," *http://www.design-simulation.com/WM2D/index.php*.

[14] "Adams - Overview," *http://www.mscsoftware.com/products/adams.cfm*.

[15] Y. Liu and J. McPhee, "Automated Kinematic Synthesis of Planar Mechanisms with Revolute Joints," *Mechanics Based Design of Structures and Machines*, vol. 35, 2007, pp. 405-445.

[16] F. Freudenstein and E.R. Maki, "The creation of mechanisms according to kinematic structure and function," *Environment and Planning B: Planning and Design*, vol. 6, 1979, pp. 375-391.

[17] S. Kota and S.J. Chiou, "Conceptual design of mechanisms based on computational synthesis and simulation of kinematic building blocks," *Research in Engineering Design*, vol. 4, 1992, pp. 75–87.

[18] L. Tsai, *Mechanism design*, CRC Press, 2001.

[19] L.C. Schmidt, H. Shetty, and S.C. Chase, "A Graph Grammar Approach for Structure Synthesis of Mechanisms," *Journal of Mechanical Design*, vol. 122, Dec. 2000, pp. 371-376.

[20] X. Li and L. Schmidt, "Grammar-Based Designer Assistance Tool for Epicyclic Gear Trains," *Journal of Mechanical Design*, vol. 126, 2004, p. 895.

[21] J. Patel and M.I. Campbell, “An Approach to Automate Concept Generation of Sheet Metal Parts Based on Manufacturing Operations,” *Volume 1: 34th Design Automation Conference, Parts A and B*, Brooklyn, New York, USA: 2008, pp. 133-142.
[22] A. Swantner and M. Campbell, “Automated Synthesis and Optimization of Gear Train Topologies,” *ASME Design Engineering Technical Conference DETC2009/DTM*, San Diego, CA: ASME, .
[23] “Official Website for GraphSynth - UT Austin - Automated Design Lab,” *http://www.me.utexas.edu/~adl/graphsynth/*.
[24] M. Campbell, “A Graph Grammar Methodology for Generative Systems.”
[25] R.L. Norton, *Machine design*, Pearson Prentice Hall, 2006.
[26] D.E. Foster and G.R. Pennock, “A Graphical Method to Find the Secondary Instantaneous Centers of Zero Velocity for the Double Butterfly Linkage,” *Journal of Mechanical Design*, vol. 125, 2003, p. 268.
[27] V. Arvind and P.P. Kurur, “Graph Isomorphism is in SPP,” *Information and Computation*, vol. 204, May. 2006, pp. 835-852.
[28] T.S. Mruthyunjaya, “Kinematic structure of mechanisms revisited,” *Mechanism and Machine Theory*, vol. 38, Apr. 2003, pp. 279-320.
[29] H.C. Gea and J. Kwon, “Topological Synthesis for Linkage Mechanism Design Using the Minimum Potential Energy Principle,” *Volume 2: 31st Design Automation Conference, Parts A and B*, Long Beach, California, USA: 2005, pp. 931-937.
[30] W. Chen, C. Chang, and H.C. Gea, “Topology and Dimensional Synthesis of Linkage Mechanism Based on the Constrained Superposition Method,” *Volume 1: 34th Design Automation Conference, Parts A and B*, Brooklyn, New York, USA: 2008, pp. 789-797.

Vita

Pradeep Radhakrishnan was born in Coimbatore, India. After graduating from GRG Matriculation Higher Secondary School in 2002, he enrolled for Bachelor of Engineering in Mechanical at PSG College of Technology. During the time, he worked on various design and manufacturing projects, both within the department and in industries. He graduated in 2006 after which he worked at M/s. TVS Motor Company Ltd. Hosur as a Member, Production Engineering – Engine Assembly. In January, 2008, he entered the Mechanical Engineering Graduate Program at The University of Texas at Austin. He is currently a member of ASME (American Society of Mechanical Engineers) and SAE (Society of Automotive Engineers).

Permanent address: Pradeep Radhakrishnan

115, G.V.Residency, Sowripalayam, Coimbatore – 641 028 India

Email: rkprad@yahoo.com

This thesis was typed by the author.