THE INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN, ICED11
15 - 18 AUGUST 2011, TECHNICAL UNIVERSITY OF DENMARK

UNDERSTANDING ENGINEERING SYSTEMS THROUGH THE ENGINEERING KNOWLEDGE GENOME: STRUCTURAL GENES OF SYSTEMS TOPOLOGIES

Offer Shai[1] and Yoram Reich[1]
(1) Tel Aviv University

ABSTRACT
The design of contemporary products requires knowledge from diverse disciplines. Presently, there is very little common denominator among engineering disciplines. This state of affairs is hurting practice, potentially leading to failures. Similar to the knowledge genome in biology, we briefly present the concept of the Interdisciplinary Engineering Knowledge Genome (IEKG) as a foundation for integrating engineering disciplines. We review the concepts of genes as abstract entities that provide the same functionality in different disciplines. In addition, we present a new type of gene – structural system gene – that serves as a building block from which well-constrained systems in diverse disciplines could be, and are constructed. We illustrate how these genes are used to compose all possible determinate trusses; how they are used to support the analysis of linkages; the assessment of trusses rigidity and the construction of sketches from well-constrained graphs. Each of these illustrations is a contribution in its own right in its particular discipline, yet they are derived from the same genes. These results demonstrate the power of the IEKG, as a theoretical construct, to engineering design. Further benefits could be obtained by incorporating additional disciplines into the IEKG, discovery of new genes, and enrichment of systems and method genes in a bootstrapping fashion. Other approaches that draw upon, or are applied to, multiple disciplines such as graph grammar and Bond graphs could also benefit from the advent of the IEKG.

Keywords: knowledge genome, interdisciplinary design, mechanism, truss, sketching

1 INTRODUCTION

Most things around us were designed and created by humans. They are artificial systems whose conception required disparate knowledge sources from diverse engineering disciplines. The diversification of disciplines has been a response to knowledge explosion but has also caused communication problems between people as presently, there is very little common denominator among engineering disciplines. This state of affairs is hurting practice, potentially leading to failures (e.g., [1], [2]). Can we do better[1] or are we doomed to further diversify and disintegrate?

In biology and some related disciplines, the situation was similar until about 100 years ago when the term gene was first used [3]. Fifty years later, the DNA was discovered as the carrier of genetic information and shortly after, its structure was identified. These discoveries revolutionized biology and its related disciplines [4] providing a single basis for future discoveries and a foundation for the multi-national human genome project with its unprecedented benefits.

Our vision is to integrate engineering disciplines in a way that circumvents communication difficulties and leads to tremendous benefits for engineering. We propose that there exist fundamental entities in engineering systems that provide similar benefits as the biological genome does for biology. Reich and Shai [5] proposed the concept of the *Interdisciplinary Engineering Knowledge Genome* (*IEKG*), focusing on two types of genes: *system genes* that are basic elements or systems that provide some concise functionality such as rectifying a signal; and *method genes* that are basic methods that could be used to analyze or synthesize systems. For example, the method gene *cutset* on graph representations could be implemented in structural mechanics as the known displacement method [6],

[1] We clearly provide an answer to [2] in that the different theoretical domains depicted in their figures 1 and 2 are found to be corresponding through our proposed interdisciplinary engineering knowledge genome.

in electrical circuits as the node method [8], and in economics, for analyzing flows of commodities [9].

The fact that something is a system or method gene means that it is a conceptual entity, shared by multiple disciplines with supposedly different theoretical basis; furthermore, a gene could be instantiated into systems or methods in these diverse disciplines. This corresponds to biological genes that could appear in different organisms providing the same function in each (e.g., the creation of the same protein).

In this paper, we demonstrate that the concept of a genome could be further expanded in engineering. We show that systems from different domains, although seem to be constructed differently and their functionality is very different, are completely composed from the same building blocks and derived in a systematic way. We call such a building block *structural system gene* (*s-gene*) to denote basic systems from which other systems could be built. For example, consider genes that could tell us how to construct all the well-constrained trusses in the world. In this interpretation, we expand the concept of a gene to mean a *language* for defining systems that has meaning beyond a particular discipline. With this language, we show that we can decompose well-structured systems in a way that supports analysis in mechanisms and structures and synthesis (construction) in sketches. Similar benefits are expected to be found in any discipline connected to the IEKG due to its properties. In addition, we foresee that a different perspective on products and systems in multiple disciplines could uncover other existing genes thus, further expanding the basis for integrating multiple disciplines.

There have been other approaches or ideas that might be viewed as similar although are quite different. This work has a potential to contribute to their further development. Among these approaches are graph grammars and bond graphs. *Graph grammar* is an example of a method for creating languages for deriving products. These grammars have been defined specifically for different types of products or product style [10]. With the advent of the IEKG, graph grammars can now transcend products within a single discipline and transcend disciplines. For example, the genes we discuss in this paper support the generation of all well-constrained trusses. This generation could be described by a graph grammar. In fact, the particular grammar could be considered as a method gene since *it will have the same effect in multiple disciplines*.

Bond graphs have been developed to support analysis of multidisciplinary systems **Error! Reference source not found.**. Since they could be represented by the IEKG,[2] they could benefit from its advantages including being applied to disciplines for which they have not been implemented before. These examples show that the IEKG could serve as an integrative foundation even to other attempts to integrate diverse engineering knowledge and to benefit from all towards providing better system analysis and synthesis methods for engineers. More specifically, Infused Design (ID) [12][13][14], a framework for conceptual creative design, uses combinatorial representations to model engineering systems and solutions to design problems. The concept of the IEKG provides a richer foundation for its operation. With the results of the present study, ID could be used more effectively and on a wider scope.

In what follows, we briefly review the main concepts of the IEKG (Section 2), describe the new s-gene (Section 3), and present a map of all these genes (Section 4). We discuss how s-genes are used to compose and decompose well-structured systems (Section 5). *Each of the examples presented is a novel contribution in its own discipline*. In this paper, we briefly describe them as support for the IEKG concept. We summarize and conclude with future research in Section 6.

2. PRELIMINARIES

Synopsis of the Interdisciplinary Engineering Knowledge Genome (IEKG)

In biology, genes are considered the carrier of hereditary information. They form the blueprint for generating proteins and the whole body and its functioning. While in biology, the concept of a gene is continuously evolving [3], a significant property of genes or their derivative proteins is that they could be found in multiple organisms performing similar functions. By understanding the nature of genes and their interactions, insight could be obtained about multiple organisms. One way to gain such understanding is studying simple organisms and transferring this knowledge to others.

[2] It is beyond the scope of the paper to demonstrate this.

Similar to this concept, we find that if we represent systems from diverse disciplines by combinatorial representations (e.g., graphs or matroids), we find that they could be related to each other through a web of relations shown in Figure 1 [5]. The "peanuts" in the figure are different combinatorial representations. A peanut is composed of two connected balls: the system genes part (large) and the method genes part (small). The peanuts are connected with different mathematical relations such as duality, equivalence, and generality. The large ellipse below is the level of the different engineering disciplines. Each discipline is depicted as a box and connected to its appropriate representation(s). We can construct small systems with concise functionality such as rectifying a signal and represent them in the graphs. Such graph is called a *system gene* and it can be transformed into all the disciplines through the web of connections to create a system in that discipline that rectify the corresponding signal. By assembling such genes into a larger system, we can create systems with the required functionality that is solely determined by the genes and their interactions. This directly mimics the situation in biology.
There is another benefit afforded by the combinatorial representation, which we call *method genes*. Methods genes are methods that operate on graph structures such as the cutset method. Transforming this method to different disciplines yields a method with the same capability. Hence, methods are also determined by some genes. There is no simple correspondence in biology to method genes. The collection of all genes in engineering and their properties is referred to as the IEKG.

Types of combinatorial representations:

MR – matroid representation
RGR - resistance graph representation
PGR – potential graph representation
FGR – flow graph representation
LGR – line graph representation
PLGR – potential line graph representation
FLGR – flow line graph representation

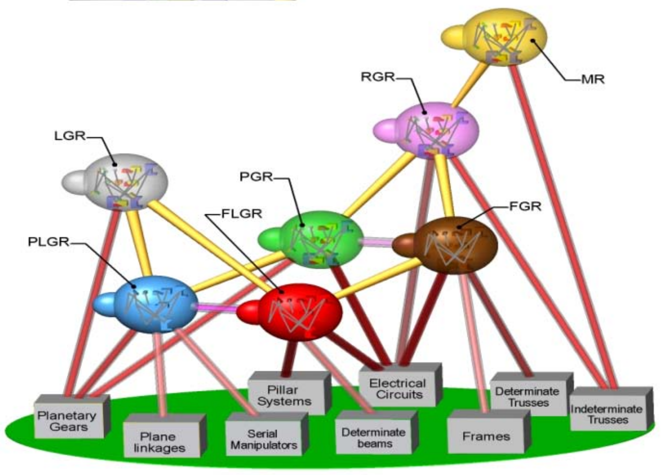


Figure 1: The network of combinatorial relations and their relations to particular disciplines.

Systems topologies in view of the IKEG

All systems are composed of simple building blocks. Systems genes provide a way to transfer knowledge between disciplines. However, can we find a way that will help us create large systems from simple building blocks? Can this way be made general enough that it could also transcend disciplines? Modeled in combinatorial representation, systems topologies could be classified into three classes: over, under and well-constrained systems. Preliminary results indicate that all the over and under-constrained graphs can be derived from the well-constrained graphs by applying the following operations: contracting, adding, and deleting edges or splitting vertices. Accordingly, the well-constrained systems can be considered the core from which all system topologies could be derived. This capability holds in abstract spaces of any dimension. The aforementioned question therefore becomes "can we find a general method for composing well-constrained systems that transcends disciplines?"
As we show here, it is possible to derive formally all the well-constrained systems in 2D and most of the topologies in 3D. The well-constrained systems, referred in this paper also as *the well-constrained*

graphs, are composed of *genes*, which are special graphs with special properties termed *Assur Graphs*. Thus, every well-constrained system is composed of genes and therefore, all the topologies of all the systems, over or under, are derivation of structures composed of genes. Exactly the same genes appear in different disciplines affording the same composition/ decomposition capabilities. This paper concentrates on these types of genes that provide new analysis and synthesis capabilities for engineers. The methods that support the composition and decomposition of systems to these s-genes are considered method genes as they effect is the same in all disciplines.

3. STRUCTURAL SYSTEM GENES AND THEIR SPECIAL TOPOLOGY PROPERTIES

The definition of the structural system genes, called for simplicity *s-genes*, is done recursively as follows:

> G is an *s-gene* iff it is a well-constrained graph and removing any set of vertices does not result in an s-gene.

This definition has several interpretations, each for a different engineering domain. In statics, for example, a system G is an s-gene iff it is a rigid determinate truss but does not contain any sub-structure that is a rigid determinate truss. Rigidity in statics refers to a structure that for almost any configuration related to the given topology, the structure could sustain any external force applied at any joint.

The structure, appearing in Figure 2(a) is an s-gene since it is a determinate truss and all its sub-structures are not rigid. For example, the system in Figure 2(b) is obtained from the s-gene (a) by deleting vertex C and all its incident rods resulting in a linkage. The system in Figure 2(c) is obtained by deleting rod 7 and is also a linkage. In contrast, the structure in Figure 3(a) is not an s-gene since deleting vertex C results is an s-gene, shown in Figure 3(b).

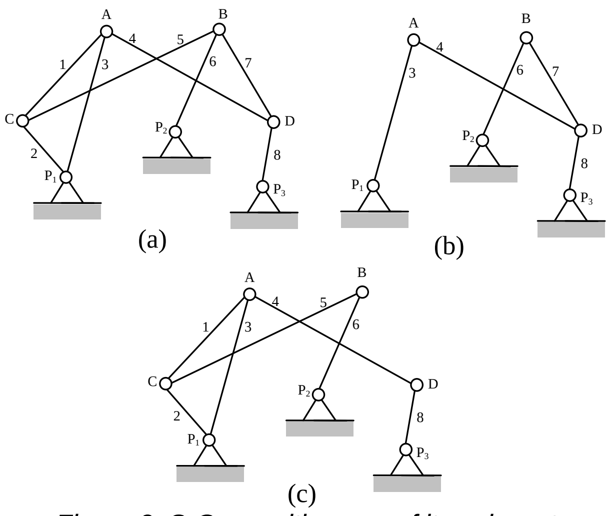


Figure 2: S-Gene with some of its sub systems

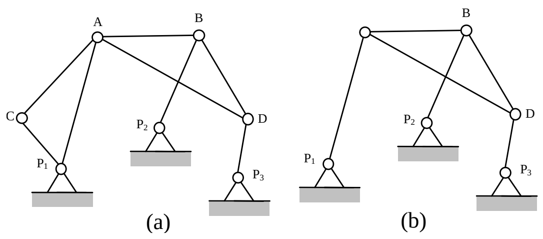


Figure 3: Example of a system that is not an s-gene

In geometric constraint problems, used to power the sketching facility in CAD systems, an s-gene is a constrained graph representing well-defined geometry objects, where removing any set of vertices results in a graph that does not relate to a geometric object. We further define:

> A system is a *floating s-gene* if there is at least one element that after grounding it, the system becomes an s-gene.

For example, consider the geometric object in Figure 4(a) whose constraint graph appears in (b). The graph is a floating s-gene since after grounding edge (2,C) the resulting graph, appearing in Figure 4(c), is the well known s-gene, called Triad. Sometimes an s-gene is better recognized after rearranging its graph as shown in (d) and (e), including splitting a support as done from (c) to (d).

4. THE MAP OF ALL S-GENES IN 2D

The map of all 2D s-genes has been developed and appears in Figure 13. Genes are arranged in a table with infinite rows and infinite columns; they are all derived from one basic s-gene, called *dyad*, Figure

12(a), by applying two types of extensions. The first extension, termed *fundamental extension*, produces all the s-genes in the first row, called also the *fundamental s-genes*. This operation is done by replacing a ground edge by a triangle and two new ground edges, as shown in Figure 5(a,a1). The fundamental s-genes can also be related as representatives, since from each one of them it is possible to derive an infinite number of different s-genes. This is done by applying a second extension, termed *regular extension*, that divides one of the edges (*x*,*y*) by a new vertex *z* and adding a new edge (*z*,*t*) for some vertex $t \neq x, y$, as shown in Figure 5(b,b1).

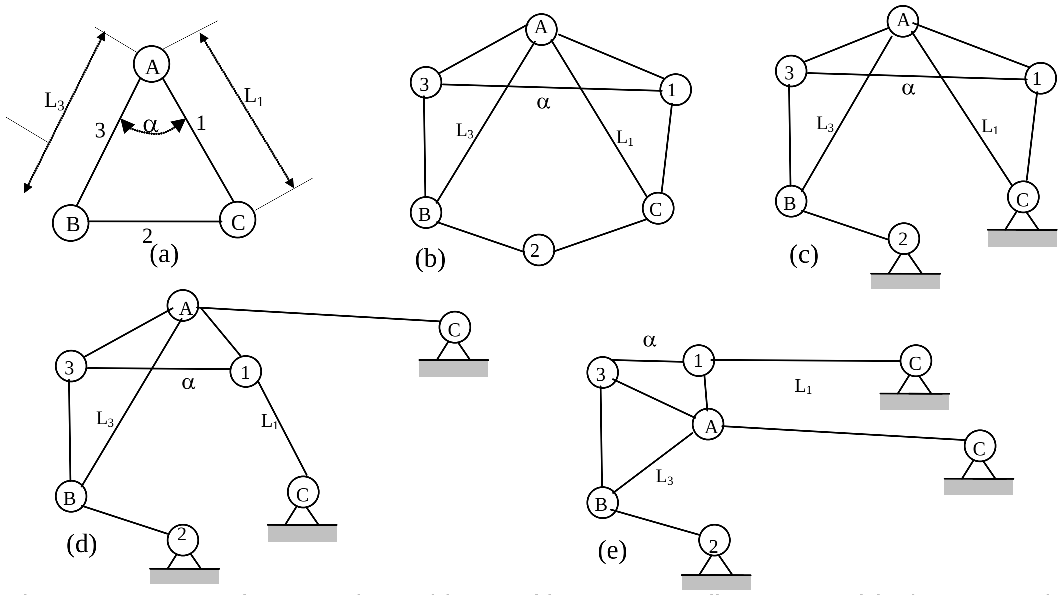


Figure 4: A geometric constraint problem and its corresponding s-gene. (a) The geometric object. (b) The corresponding floating s-gene. (c) The s-gene after grounding edge (2,C). (d) The same s-gene with support C split. (e) The s-gene in (d) with graph reorganized

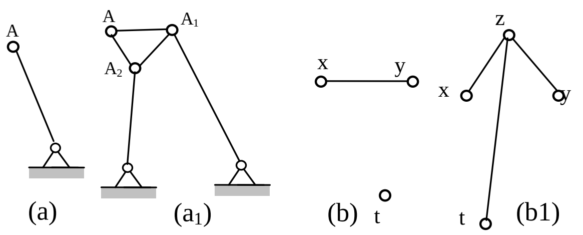


Figure 5: The resultant of applying fundamental and regular extensions (a1,b1) on ground and inner edges (a,b), respectively

Figure 12[3] depicts the process of constructing the s-genes from the basic s-gene, the dyad (a). The first row presents the fundamental s-genes, called also the representatives, all derived from the basic s-dyad through applying the fundamental extensions. For example, the fundamental s-gene in (b) known also as Triad, is obtained by replacing the ground edge (A,O_2) with the triangle <A,B,C> with the two new ground edges (C,O_2) and (B,O_3). All the other infinite fundamental s-genes are obtained in the same way. Now, that we can generate all the fundamental s-genes, each one of them defines an infinite number of new s-genes, all derived by applying successive regular extensions. For example, the s-gene in (b_1) is derived from the fundamental s-gene, the triad, by applying the regular extension on edge (A,B) by adding vertex D and adding the additional edge – the ground edge – (D,O_4). In the subsequent extension to (b_2), the additional edge is an inner edge (E,B).
Applying the two types of extensions infinite number of times, enables deriving the infinite map of all the s-genes in 2D as shown in Figure 13[4]. This result is based on theorems and methods developed in rigidity theory [15] where it is mathematically proved that all the rigidity circuits in 2D can be derived

[3] Figure 12 appears at the end of the paper.
[4] Figure 13 appears at the end of the paper.

from the complete graph with four vertices, called K4. Based on a new relation between rigidity circuits and Assur Graphs [16][17][18] it can be proved that the presented map of s-genes is both complete and sound, i.e., all the s-genes can be found and every graph produced by the two extensions is an s-gene.

5. THE DECOMPOSITION OF WELL-CONSTRAINED SYSTEMS INTO S-GENES

Now, that we have all the s-genes, it is possible to define any well constrained system through the s-genes, i.e.,

Theorem: A system is *well constrained* if it can be decomposed into s-genes.

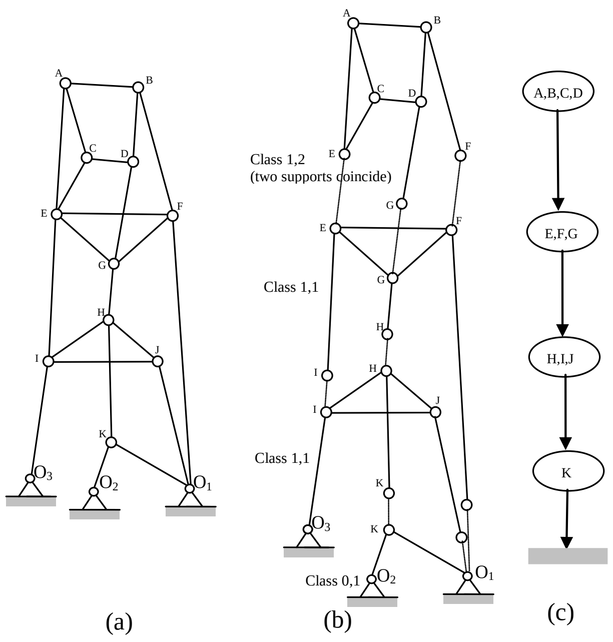
The decomposition can be automated by using the pebble game algorithm [19]. By implementing these ideas in different disciplines, this immediately paves new ways for checking the rigidity of trusses; decomposing huge trusses and linkages into small component for analysis purposes; and more in other disciplines by relying on the IEKG. These capabilities support design in different stages. We illustrate some of these implementations and capabilities here.

Decomposing well constrained trusses

In structures for example, it follows that *a truss is determinate iff it can be decomposed into s-genes*. The idea of decomposing a determinate truss into s-genes is demonstrated through example appearing in Figure 6. The truss in the figure is shown to be rigid and determinate since it is decomposed into s-genes as shown in (b), where the doted lines indicate the connections between the s-genes. The type of s-gene from Figure 13 appears on the left of each s-gene. To clarify the decomposition order, another graph is used called decomposition graph, given (c), where the s-genes are presented by vertices and edges define the decomposition order. The directed edge <x,y> indicates that in order to decompose s-gene corresponding to vertex y there is a need first to decompose the s-gene corresponding to vertex 'x'. This graph can be used also for the composition order when it is treated in a bottom-up order.

Synthesizing geometric objects from well constrained sketches

In Floating Well Constrained Systems, there is a unique decomposition only after grounding one of the elements; thus for each grounding, there is a different decomposition. Hence, for each floating s-gene there are more options for decomposition or composition in case there are several edges that can be grounded. For example, in the geometric problem given in Figure 4(a) whose graph appears in Figure 4(b) when edge (2,C) is grounded, Figure 4(c), the system is decomposed into one s-gene, the Triad, which is Class 1,3 in Figure 13 after rearranging it as shown in Figure 4(d,e). In the terminology of geometry, all the equations are to be solved simultaneously. On the other hand, when edge (3,A) is grounded, Figure 7(a), the system is decomposed into 4 s-genes – all dyads; thus, the solution of the sketch is straight forward.



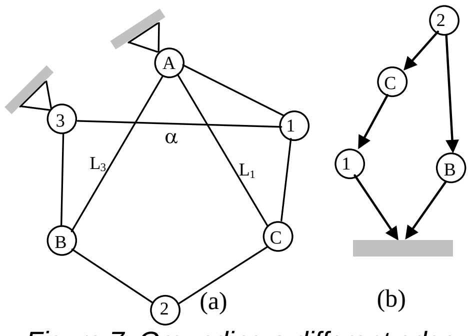


Figure 7: Grounding a different edge of the geometric constraint problem. (a) The graph after grounding edge (A,3). (b) The decomposition graph.

Figure 6: Decomposing a determinate truss into s-genes.
(a) The truss. (b) Scheme illustrating the decomposition.
(c) The decomposition graph.

The (de)composition graph in Figure 7(b) gives the needed information for constructing the well-constrained geometric system. In this example, grounding edge (A,3) indicates that line 3 is incident to point A, Figure 8(a). Then, two s-genes can be constructed, dyad 1 or dyad B. Choosing, arbitrarily, s-gene dyad 1 leads to adding line 1 with angle α related to line 3, shown in (b). Now dyad B or dyad C are available, choosing arbitrarily dyad B, indicates adding point B at a distance L_3 from point A, shown in (c). Now, only dyad C should be composed, meaning that point C is added to line 1 at a distance L_1 from point A. The last s-gene 2 adds line 2 between the points B and C, Figure 8(e).

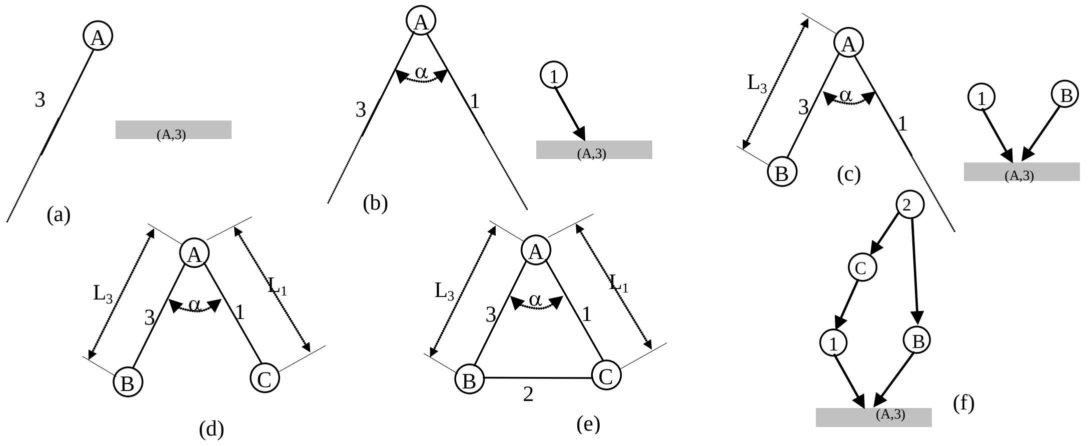


Figure 8: The composition steps of the geometric constraint problem following the order of the composition order of the composition graph.

Structural genes of linkages

Linkages can be derived from the system topology of determinate trusses by adding a ground edge to one of the ground vertices. This edge is called the *driving link*. Thus, the s-genes of linkages are the same as those of determinate trusses. For each determinate s-gene there are several corresponding linkages since there are several options to add the driving link. This idea relies on the work of Assur [5]. For example, for the s-gene appearing in Figure 2(a) and reproduced in Figure 9(a) (Class 1,2 in Figure 13) there are three corresponding linkages that appear in Figure 9 (b,c,d).

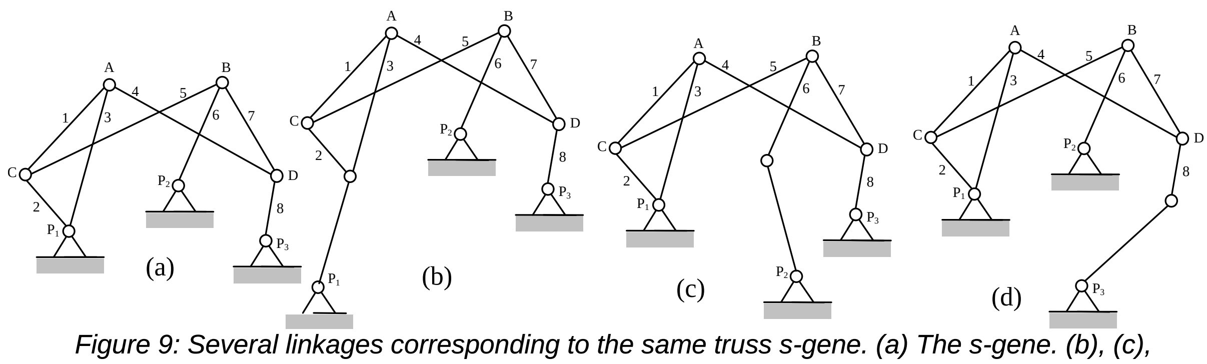


Figure 9: Several linkages corresponding to the same truss s-gene. (a) The s-gene. (b), (c), (d) The corresponding linkages.

With these s-genes, any linkage topology could be generated in a similar way to the determinate truss topology. In addition, complex linkages could be decomposed into s-gene linkages thus supporting quick analysis and providing other benefits to linkage designers.

Employing S-Genes for the analysis of mechanisms

S-genes can be used for analyzing mechanisms. It is mathematically proven that every kinematical

system has a unique decomposition into s-genes [16]. To clarify the decomposition we use a structural scheme (Figure 10) in which the driving link is deleted and replaced by a ground vertex and each joint connects only two links. In the current example, link 1 is deleted and vertex A is grounded and links 2,5 and 7 are replaced with triangle rigid bodies . Then, the system is decomposed into three s-genes, a tetrad (Class 1,2), triad (Class 1,1) and a dyad (Class 0,1). The order of the decomposition is important. If an inner joint of an Assur Graph, G1 becomes a ground vertex in Assur Graph G2, then G1 should precede G2.
The order of analysis of the s-genes is the composition order, i.e., the composition follows the opposite operations in a bottom up manner. The unique order of decomposition as appears in Figure 10 is: First analyze the tetrad {B,D,C,J}, and then analyze the triad {G,H,I} and dyad {F}, simultaneously.

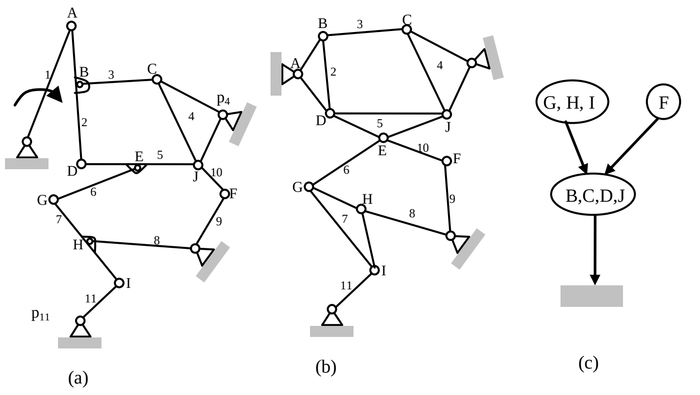


Figure 10: Decomposition of a linkage into s-genes. (a) The linkage. (b) The structural scheme. (c) The decomposition graph.

The analysis process of the above linkage appears in Figure 11. The analysis uses the following property of s-genes: For each s-gene, if the velocities of the outer vertices (in the map they are marked as ground vertices) are known, then the velocities of the inner vertices can be calculated using known methods in kinematics. For example, in Figure 11(a), in the first s-gene in the composition order, tetrad {B,C,D,J}, the outer velocities, A and p4, are known; the first is part of the driving link and the second is zero since it is a ground vertex. Thus, the velocities of the inner vertices {B,C,J,E,D} can be calculated and since they are connected to other s-genes, some of them become outer vertices in other s-genes as can be seen in the decomposition graph. For example, vertex J becomes outer vertex in the dyad {F} as shown in Figure 11(b), and vertex E becomes outer vertex in the s-gene triad {G,H,I} as shown in Figure 11(c).

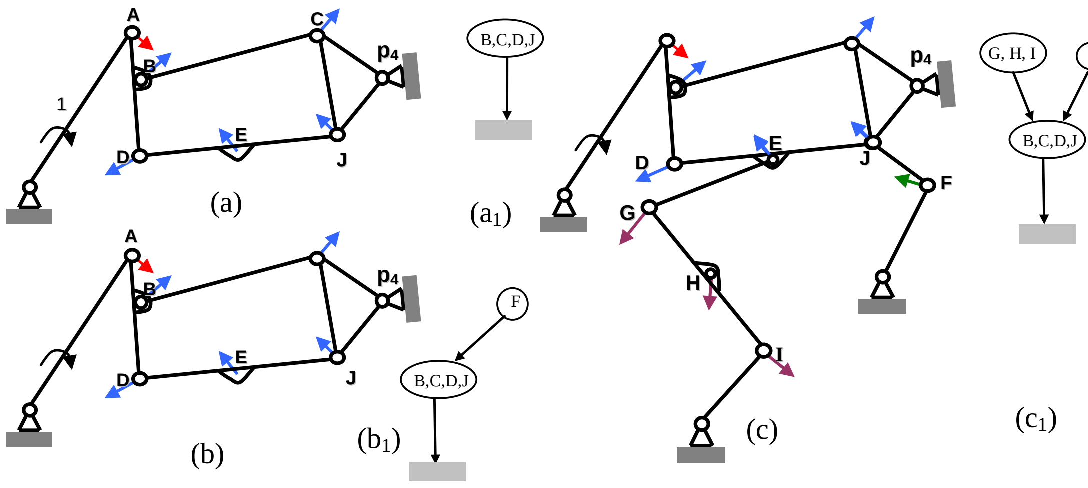


Figure 11: Analysis process of a mechanism

6. SUMMARY AND FURTHER RESEARCH

We briefly reviewed the concept of the IEKG with systems and method genes. The IEKG is a theoretical concept that allows viewing different engineering disciplines from the same foundation. In this paper, we introduced a new type of genes that also transcends disciplines. We demonstrated the utility of the s-genes in three domains: trusses, linkages, and sketches. In each of these disciplines, the concept of the s-gene leads to new results related to design. In trusses, we have a new algorithm for checking the rigidity of 2D trusses, and the ability to describe all the possible trusses. For linkages, we have a new method for analyzing linkages and describing all the 2D linkages. For sketches, we have a new method for checking whether a sketch is well constrained, solve it easily, and obtain more uses than conventional methods provide. Although not demonstrated due to space limitations, these new algorithms can solve synthesis problems such as synthesizing foldable tensegrity structures, for which no other method exists. The use of these algorithms led to developing and implementing new types of robots [21].

Similar benefits could be obtained by using s-genes in other disciplines for analyzing or synthesizing large systems – a benefit afforded by implementing them in all domains connected to the IEKG. Following these examples in 2D, we continue to work on s-genes in 3D and intend subsequently to extend this work to any dimension.

Our experience with the IEKG has shown that we should expect to obtain bootstrapping effects by collaborative work on multiple disciplines [22][23]. This should provide additional motivation to engage in research on the IEKG. Since methods such as bond graphs, graph grammars, and TRIZ cross disciplines, studying them with the IEKG might yield additional promising results. Finally, we foresee that new types of genes could be discovered, providing yet additional benefit to engaging with the IEKG.

REFERENCES

[1] Subrahmanian, E. Monarch I., Konda S. L., Granger H., Collins M., Milliken R., Westerberg A. W., Boundary Objects and Prototypes at the Interfaces of Engineering Design, Computer Supported Co-operative Work, 12(2):185-203, 2003.

[2] Tomiyama T., D'Amelio V., Urbanic J., and ElMaraghy W., Complexity of Multi-Disciplinary Design, CIRP Annals - Manufacturing Technology, 56(1):185-188, 2007.

[3] Gerstein M. B., Bruce C., Rozowsky J. S., Zheng D., Du J, Korbel J. O., Emanuelsson O., Zhang Z.D., Weissman S., and Snyder M., What is a gene, post-ENCODE? History and updated definition, Genome Research, 17:669-681, 2007.

[4] Berg P. and Singer M., Dealing with Genes: The Language of Heredity, University Science Books, Mill Valley, CA, 1992.

[5] Reich Y., Shai O., The Interdisciplinary Engineering Knowledge Genome, Research in Engineering Design, in press, 2011.

[6] Fenves S. J. and Branin F. H., Network topological formulation of structural analysis, Journal of the Structural Division, ASCE, 89(ST4):483-514, 1963.

[7] Shai O., Deriving structural theorems and methods using Tellegen’s theorem and combinatorial representations, International Journal of Solids and Structures, 38:8037-8052, 2001b.

[8] Balabanian N. and Bickart T.A., Electrical Network Theory, John Wiley & Sons, New York, 1969.

[9] Ford L. R., and Fulkerson D. R., Flows in Networks, Princeton University Press, NJ, 1962.

[10] Schmidt L.C. and Cagan J., GGREADA: A graph grammar-based machine design algorithm, Research in Engineering Design, 9(4):195-213, 1997.

[11] Borutzky W., Bond graph modelling and simulation of multidisciplinary systems - An introduction, Simulation Modelling Practice and Theory, 17(1):3-21, 2009.

[12] Shai O. and Reich Y., Infused design. I Theory, Research in Engineering Design, 15(2):93-107, 2004a.

[13] Shai O. and Reich Y., Infused design. II Practice, Research in Engineering Design, 15(2):108-121, 2004b.

[14] Shai O., Reich Y., and Rubin D., Creative conceptual design: Extending the scope by infused design, Computer-Aided Design, 41(3): 117-135, 2009a.

[15] Berg A. R. and Jordan T., A proof of Connelly's conjecture on 3-connected circuits of the rigidity matroid, Journal of Combinatorial Theory Series B, 88(1):77-97, 2003.

[16] Servatius B., Shai O. and Whiteley W., Combinatorial Characterization of the Assur Graphs from

Engineering, *European Journal of Combinatorics*, 31(4):1091-1104, 2010.

[17] Shai O., Assur graphs, Recent Progress in Rigidity Theory, July 12, Banff International research Station – Canada, 2008

[18] Shai O., The Canonical Form of All Planar Linkage topologies, ASME Design Engineering Technical Conferences, San Diego, California, 2009.

[19] Jacobs D. and Hendrickson B., An algorithm for two-dimensional rigidity percolation: the pebble game, Journal of Computational Physics 137:346–365, 1997.

[20] Assur L.V., Issledovanie ploskih sterzhnevyh mehanizmov s nizkimi parami s tochki zrenija ih struktury i klassifikacii. Akad. Nauk SSSR, Edited by I.I. Arotobolevskii, 1952.

[21] Shai O., Tehori I., Bronfeld A., Slavutin M. and Ben-Hanan U., Adjustable Tensegrity Robot based on Assur Graph Principle, *ASME International Mechanical Engineering Congress and Exposition*, November 13-19, Lake Buena Vista, Florida, 2009, USA.

[22] Reich Y., Shai O., Subrahmanian E., Hatchuel A., Le Masson P., The interplay between design and mathematics: Introduction to bootstrapping effects, Proceedings of the 9th Biennial ASME Conference on Engineering Systems Design and Analysis ESDA2008, Haifa, Israel, 2008.

[23] Shai O., Reich Y., Hatchuel A., and Subrahmanian E., Creativity theories and scientific discovery: A study of C-K theory and Infused Design, In CD Proceedings of the International Conference on Engineering Design, ICED'09, Stanford, CA, 2009. Outstanding paper award.

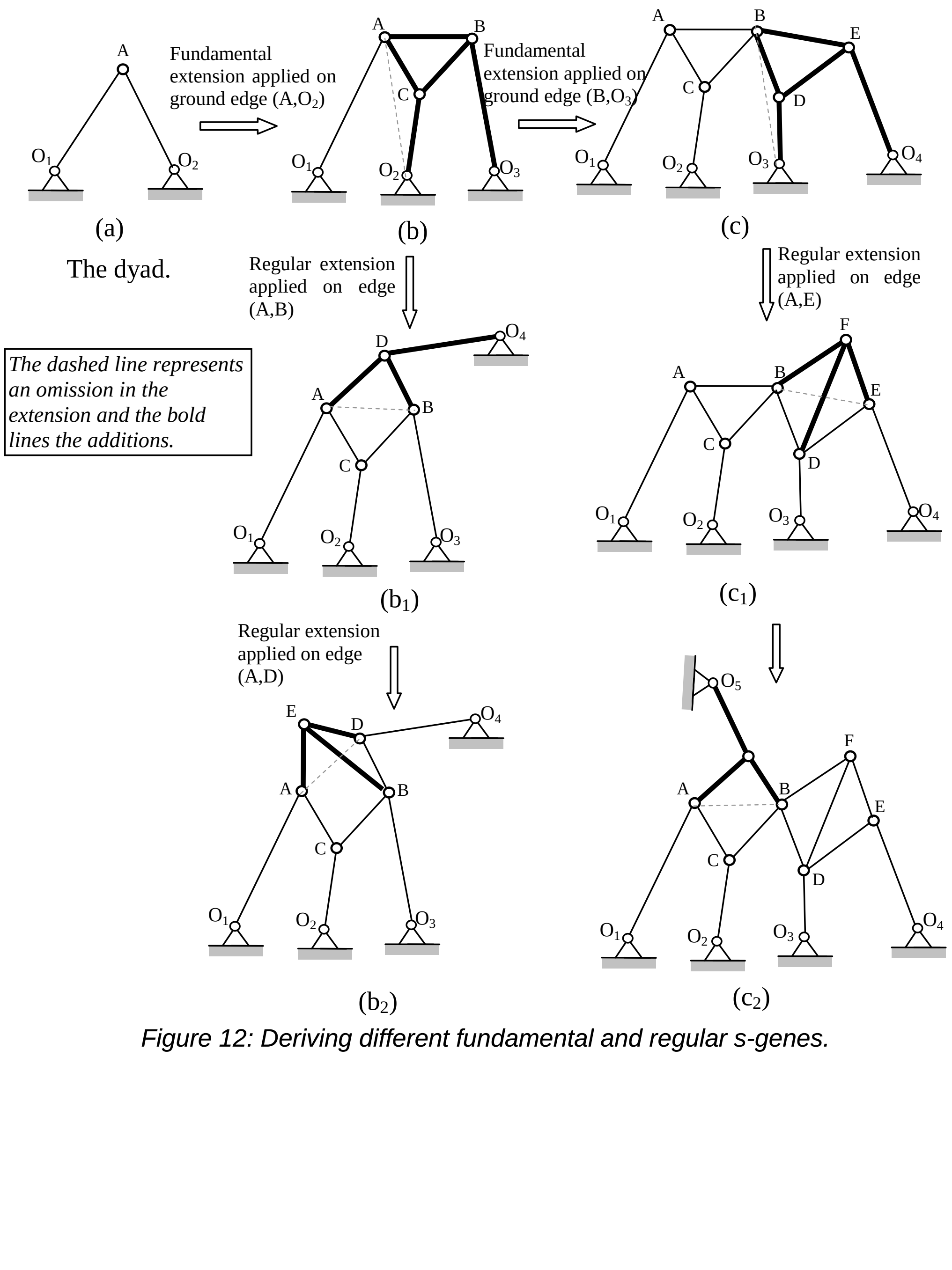

A
O1
O2
(a)
The dyad.
Fundamental extension applied on ground edge (A,O2)
A
B
C
O1
O2
O3
(b)
Fundamental extension applied on ground edge (B,O3)
A
B
E
C
D
O1
O2
O3
O4
(c)
Regular extension applied on edge (A,B)
The dashed line represents an omission in the extension and the bold lines the additions.
D
O4
A
B
C
O1
O2
O3
(b1)
Regular extension applied on edge (A,E)
F
A
B
E
C
D
O1
O2
O3
O4
(c1)
Regular extension applied on edge (A,D)
E
D
O4
A
B
C
O1
O2
O3
(b2)
O5
F
A
B
E
C
D
O1
O2
O3
O4
(c2)

Figure 12: Deriving different fundamental and regular s-genes.

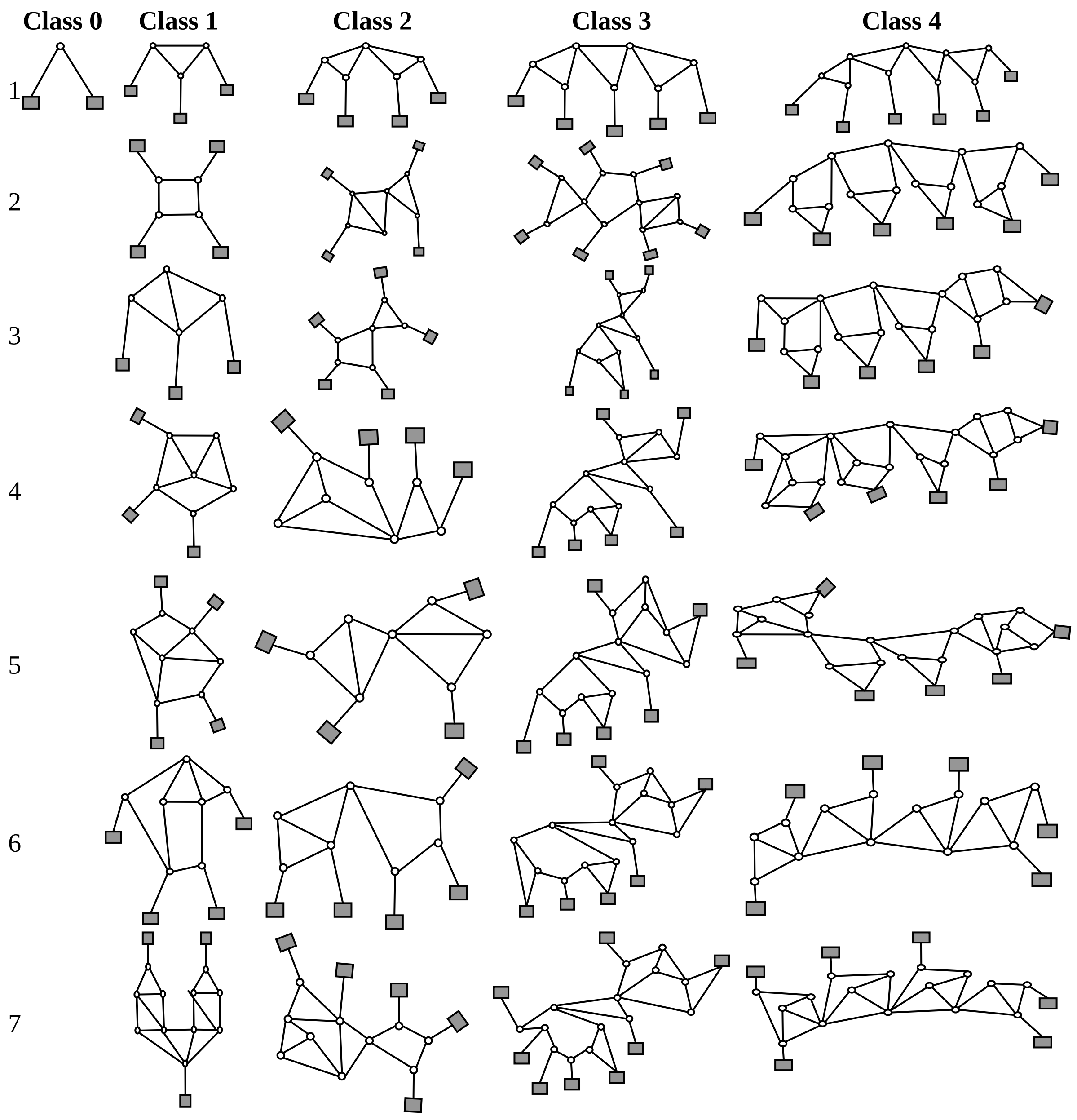


Figure 13: The infinite map of all the genes in 2D.

Contact: Offer Shai
School of Mechanical Engineering
Tel Aviv University
Tel Aviv 69978
Israel
Tel: +972 3 6406710
Fax: +972 3 6407617
Email: shai@eng.tau.ac.il
URL: http://www.eng.tau.ac.il/~shai

Offer Shai is a senior lecturer at the Faculty of Engineering, Tel Aviv University. His main research interest is developing combinatorial engineering representations and applying/using them in the following fields: mechanisms, design theory and rigidity theory.

Yoram Reich is a processor at the Faculty of Engineering, Tel Aviv University. He is the Editor-in-Chief of *Research in Engineering Design* and serves on the editorial board of 5 other journals; a co-chair of the design theory special interest group of the design society, and a member of the advisory board of the society. He recently founded and is the chairman of Israel Institute for Innovation Technology